

Hierarchical Distribution-Aware Testing of Deep Learning

WEI HUANG, Purple Mountain Laboratories, China, University of Liverpool, U.K.

XINGYU ZHAO, WMG, University of Warwick, U.K.

ALEC BANKS, Defence Science and Technology Laboratory, U.K.

VICTORIA COX, Defence Science and Technology Laboratory, U.K.

XIAOWEI HUANG, University of Liverpool, U.K.

With its growing use in safety/security-critical applications, Deep Learning (DL) has raised increasing concerns regarding its dependability. In particular, DL has a notorious problem of lacking robustness. Input added with adversarial perturbations, i.e. Adversarial Examples (AEs) are easily mis-predicted by the DL model. Despite recent efforts made in detecting AEs via state-of-the-art attack and testing methods, they are normally input distribution agnostic and/or disregard the perceptual quality of adversarial perturbations. Consequently, the detected AEs are irrelevant inputs in the application context or unrealistic that can be easily noticed by humans. This may lead to a limited effect on improving the DL model's dependability, as the testing budget is likely to be wasted on detecting AEs that are encountered very rarely in its real-life operations.

In this paper, we propose a new robustness testing approach for detecting AEs that considers both the feature level distribution and the pixel level distribution, capturing the perceptual quality of adversarial perturbations. The two considerations are encoded by a novel hierarchical mechanism. First, we select test seeds based on the density of feature level distribution and the vulnerability of adversarial robustness. The vulnerability of test seeds are indicated by the auxiliary information, that are highly correlated with local robustness. Given a test seed, we then develop a novel genetic algorithm based local test case generation method, in which two fitness functions work alternatively to control the perceptual quality of detected AEs. Finally, extensive experiments confirm that our holistic approach considering hierarchical distributions is superior to the state-of-the-arts that either disregard any input distribution or only consider a single (non-hierarchical) distribution, in terms of not only detecting imperceptible AEs but also improving the overall robustness of the DL model under testing.

CCS Concepts: • **Software and its engineering** → **Software testing and debugging**; **Software reliability**; • **Computing methodologies** → **Machine learning**.

Additional Key Words and Phrases: Deep learning robustness, adversarial examples detection, natural perturbations, distribution-aware testing, robustness growth, safe AI

ACM Reference Format:

Wei Huang, Xingyu Zhao, Alec Banks, Victoria Cox, and Xiaowei Huang. 2023. Hierarchical Distribution-Aware Testing of Deep Learning. *ACM Trans. Softw. Eng. Methodol.* 0, 0, Article X (September 2023), 34 pages. <https://doi.org/XXXXXXXX.XXXXXXX>

Authors' addresses: Wei Huang, Purple Mountain Laboratories, China, and University of Liverpool, U.K., w.huang23@liverpool.ac.uk; Xingyu Zhao, WMG, University of Warwick, U.K., Xingyu.Zhao@warwick.ac.uk; Alec Banks, Defence Science and Technology Laboratory, U.K., abanks@dstl.gov.uk; Victoria Cox, Defence Science and Technology Laboratory, U.K., vcox@dstl.gov.uk; Xiaowei Huang, University of Liverpool, U.K., Xiaowei.Huang@liverpool.ac.uk.

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 ACM 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.

© 2023 Association for Computing Machinery.

1049-331X/2023/9-ARTX \$15.00

<https://doi.org/XXXXXXXX.XXXXXXX>

1 INTRODUCTION

Deep Learning (DL) is being explored to provide transformational capabilities to many industrial sectors including automotive, healthcare and finance. The reality that DL is not as dependable as required now becomes a major impediment. For instance, key industrial foresight reviews identified that the biggest obstacle to gaining benefits of DL is its dependability [27]. There is an urgent need to develop methods to enable the dependable use of DL, for which great efforts have been made in recent years in the field of DL Verification and Validation (V&V) [22, 60].

DL **robustness** is arguably *the* property in the limelight. Informally, robustness requires that the decision of the DL model is invariant against small perturbations on inputs. That is, all inputs in a small input region (e.g., a norm ball defined in some L_p -norm distance) should share the same prediction label by the DL model. Inside that region, if an input is predicted differently to the given label, then this input is normally called an **Adversarial Example (AE)**. Most V&V methods designed for DL robustness are essentially about detecting AEs, e.g., adversarial attack based methods [16, 33] and coverage-guided testing [12, 20, 32, 35, 41, 55].

As recently noticed by the software engineering community, emerging studies on systematically evaluating AEs detected by aforementioned state-of-the-arts have two major drawbacks: (i) they do not take the *input data distribution* into consideration, therefore it is hard to judge whether the identified AEs are meaningful to the DL application [5, 10]; (ii) most detected AEs are of *poor perceptual quality* that are too unnatural/unrealistic [17] to be seen in real-life operations. That said, not all AEs are equal nor significantly contribute to the robustness improvement, given limited resources. A wise strategy is to detect those AEs that are both being “distribution-aware” and with natural/realistic pixel-level perturbations, which motivates this work.

Prior to this work, a few notable attempts at distribution-aware testing for DL have been made. Broadly speaking, the field has developed two types of approaches: *Out-Of-Distribution (OOD) detector* based [4, 10] and *feature-only* based [7, 44]. The former can only detect anomalies/outliers, rather than being “fully-aware” of the distribution. While the latter is indeed generating new test cases according to the learnt distribution (in a latent space), it ignores the pixel-level information due to the compression nature of generative models used [65]. To this end, our approach is advancing in this direction with the following novelties and contributions:

a) We provide a “divide and conquer” solution—Hierarchical Distribution-Aware (HDA) testing—by decomposing the input distribution into two levels (named as *global* and *local*) capturing how the feature-wise and pixel-wise information are distributed, respectively. At the global level, isolated problems of estimating the feature distribution and selecting best test seeds can be solved by dedicated techniques. At the local level where features are fixed, the clear objective is to precisely generate test cases considering perceptual quality¹. Our extensive experiments show that such *hierarchical* consideration is more effective to detect high-quality AEs than state-of-the-art that either disregards any data distribution or only considers a single (non-hierarchical) distribution. Consequently, we also show the DL model under testing exhibits higher robustness after “fixing” the high-quality AEs detected.

b) At the global level, we propose novel methods to select test seeds based on the *approximated feature distribution* of the training data and *predictive robustness indicators*, so that the norm balls of the selected seeds are both from the high-density area of the distribution and relatively unrobust (thus more cost-effective to detect AEs in later stages). Notably, state-of-the-art DL testing methods

¹While determining perceptual quality typically involves subjective assessments from human observers, objective metrics such as peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM) can also be used as measures of perceptual quality [49]. Throughout the paper, perceptual quality is defined as the metrics in Sec. 2.4 that are commonly used in computer vision.

normally select test seeds *randomly* from the training dataset without any principled rules. Thus, from a software engineering perspective, our test seed selection is more practically useful in the given application context.

c) Given a carefully selected test seed, we propose a novel two-step Genetic Algorithm (GA) to generate test cases locally (i.e. within a norm ball) to control the *perceptual quality* of detected AEs. At this local level, the perceptual quality distribution of data-points inside a norm ball requires pixel-level information that cannot be sufficiently obtained from the training data alone. Thus, we innovatively use common perceptual metrics that quantify image quality as an approximation of such local distribution. Our experiments confirm that the proposed GA is not only effective after being integrated into HDA (as a holistic testing framework), but also outperforms other pixel level AE detectors in terms of perception quality when applied separately.

d) We investigate black-box (to the DL model under testing) methods for the main tasks at both levels. Thus, to the best of our knowledge, our HDA approach provides an *end-to-end, black-box* solution, which is the first of its kind and more versatile in software engineering practice.

e) A publicly accessible tool of our HDA testing framework with all source code, datasets, DL models and experimental results.

2 PRELIMINARIES AND RELATED WORK

In this section, we first introduce preliminaries and related work on DL robustness, together with formal definitions of concepts adopted in our HDA approach. Then existing works on distribution-aware testing are discussed. Since our HDA testing also considers the naturalness of detected AEs, some common perception quality metrics are introduced. In summary, we present Fig. 1 to show the stark contrast of our proposed HDA testing (the green route) to other related works (the red and amber routes).

2.1 DL Robustness and Adversarial Examples

We denote the prediction output of DL model as the vector $f(x)$ with size equal to the total number of labels. The predicted label $\hat{f}(x) = \arg \max_i f_i(x)$ where $f_i(x)$ is the i^{th} attribute of vector $f(x)$.

DL robustness requires that the decision of the DL model $\hat{f}(x)$ is invariant against small perturbations on input x . That is, all inputs in an input region η have the same prediction label, where η is usually a small norm ball (defined with some L_p -norm distance²) around an input x . If an input x' inside η is predicted differently to x by the DL model, then x' is called an *Adversarial Example* (AE).

DL robustness V&V can be based on formal methods [23, 39] or statistical approaches [50, 53], and normally aims at detecting AEs. In general, we may classify two types of methods (the two branches in the red route of Fig. 1) depends on how the test cases are generated: (i) Adversarial attack based methods normally optimise the DL prediction loss to find AEs, which include white-box attack methods like Fast Gradient Sign Method (FGSM) [16] and Projected Gradient Descent (PGD) [33], as well as black-box attacks [2, 54] using GA with gradient-free optimisation. (ii) Coverage-guided testing optimises the certain coverage metrics on the DL model's internal structure, which is inspired by the coverage testing for traditional software. Several popular test metrics, like neuron coverage [32, 35], modified condition/decision coverage [41] for CNNs and temporal coverage [12, 20] for RNNs are proposed. While it is argued that coverage metrics are not strongly correlated with DL robustness [17, 56], they are seen as providing insights into the internal behaviours of DL models and hence may guide test selection to find more diverse AEs [20].

Without loss of generality, we reuse the formal definition of DL robustness in [50, 52] in this work:

² $p = 0, 1, 2$ and ∞ . L_∞ norm is more commonly used.

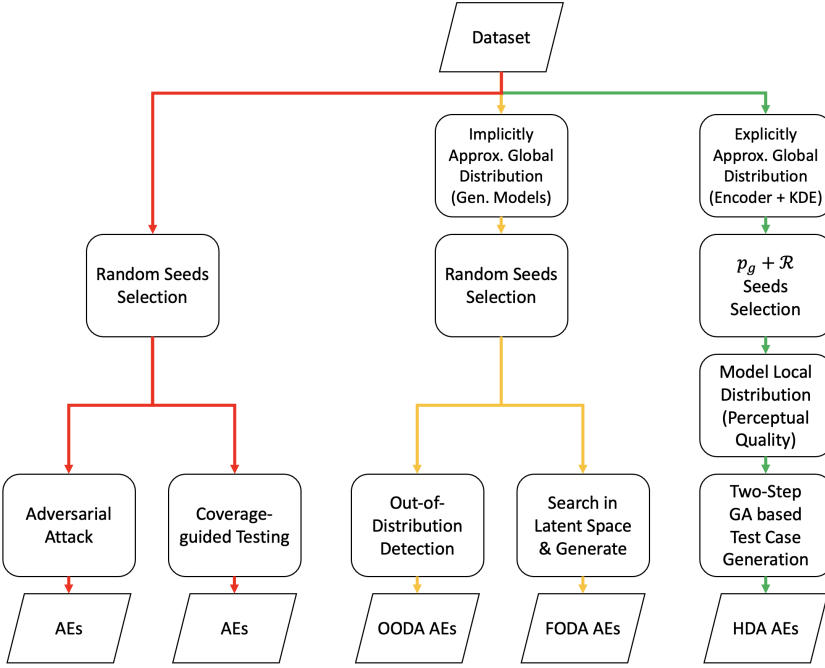


Fig. 1. Comparison between our proposed Hierarchical Distribution-Aware (HDA) testing and related works.

DEFINITION 1 (LOCAL ROBUSTNESS). *The local robustness of the DL model $f(x)$, w.r.t. a local region η and a target label y , is:*

$$\mathcal{R}_l(\eta, y) := \int_{x \in \eta} I(x) p_l(x | x \in \eta) dx \quad (1)$$

where $p_l(x | x \in \eta)$ is the local distribution of region η which is precisely the “input model” used by both [50, 52]. $I(x)$ is an indicator function, and $I(x) = 1$ when $\hat{f}(x) = y$, $I(x) = 0$ otherwise.

To detect as many AEs as possible, normally the first question is—which local region shall we search for those AEs? I.e. how to select test seeds? To be cost-effective, we want to explore unrobust regions, rather than regions where AEs are relatively rare. This requires the local robustness of a region to be known *a priori*, which may imply a paradox (cf. Remark 4 later). In this regard, we can only *predict* the local robustness of some regions before doing the actual testing in those regions. We define:

DEFINITION 2 (LOCAL ROBUSTNESS INDICATOR). *Auxiliary information that strongly correlated with $\mathcal{R}_l(\eta, y)$ (thus can be leveraged in its prediction) is named as a local robustness indicator.*

We later seek for such indicators (and empirically show their correlation with the local robustness), which forms one of the two key factors considered in selecting test seeds in our method.

Given a test seed, we search for AEs in a local region η (around the test seed) that produce different label from the test seed. This involves the question on what size of η should be, for which we later utilise the property of:

REMARK 1 (r -SEPARATION). *For real-world image datasets, any data-points with different ground truth labels are at least distance $2r$ apart in the input (pixel) space, where r is estimated case by case and depending on the dataset.*

The r -separation property was first observed by [57]: intuitively it says, there is a minimum distance between two real-world objects of different labels.

Finally, not all AEs are equal in terms of the “strength of being adversarial” (stronger AEs may lead to greater robustness improvement in, e.g., adversarial training [48]), for which we define:

DEFINITION 3 (PREDICTION LOSS). *Given a test seed x with label y , the prediction loss of an input x' , which is obtained by adding perturbations to x , is defined as:*

$$\mathcal{J}(f(x'), y) = \max_{i \neq y} (f_i(x') - f_y(x')) \quad (2)$$

where $f_i(x')$ returns the probability of label i after input x' being processed by the DL model f .

Note, $\mathcal{J} \geq 0$ implies $\arg \max_i f_i(x) \neq y$ and thus x' is an AE of x .

Next, to measure the DL models’ overall robustness across the whole input domain, we introduce a notion of global robustness. Being different to some existing definitions where robustness of local regions are treated equally when calculating global robustness over several regions [45, 46], ours is essentially a “weighted sum” of the robustness of local regions where each weight is the probability of the associated region on the input data distribution. Defining global robustness in such a “distribution-aware” manner aligns with our motivation—as revealed later by empirically estimated global robustness, our HDA appears to be more effective in supporting the growth of the overall robustness after “fixing” those distribution-aware AEs.

DEFINITION 4 (GLOBAL ROBUSTNESS). *The global robustness of the DL model $f(x)$ is defined as:*

$$\mathcal{R}_g := \sum_{\eta \in \mathcal{X}} p_g(x \in \eta) \mathcal{R}_l(\eta, y) \quad (3)$$

where $p_g(x | x \in \eta)$ is the global distribution of region η (i.e., a pooled probability of all inputs in the region η_z) and $\mathcal{R}_l(\eta, y)$ is the local robustness of region η to the label y .

The estimation of \mathcal{R}_g , unfortunately, is very expensive that requires to compute the local robustness \mathcal{R}_l of a large number of regions over the input domain \mathcal{X} . Thus, from a practical standpoint, we adopt an empirical definition of the global robustness in our later experiments, which has been commonly used for DL robustness evaluation in the adversarial training [33, 47, 48, 59].

DEFINITION 5 (EMPIRICAL GLOBAL ROBUSTNESS). *Given a DL model f and a validation dataset D_v , we define the empirical global robustness as $\hat{\mathcal{R}}_g : (f, D_v, T) \rightarrow [0, 1]$ where T denotes a given type of AE detection method and $\hat{\mathcal{R}}_g$ is the weighted accuracy on AEs obtained by conducting T on $\langle f, D_v \rangle$.*

To be “distribution-aware”, the synthesis of D_v should conform to the global distribution. D_v can be sampled from the train/test data according to global distribution. The norm ball around each input data x in D_v represents a region η . Each region η is explicitly assigned a weight to indicate the density on global distribution. For each region η , we calculate the prediction accuracy on AEs, detect by T according to local distribution, to approximate the local robustness \mathcal{R}_l . Consequently, the set of AEs for D_v may represent the input distribution and the weighted accuracy on these AEs approximate the global robustness.

2.2 Distribution-Aware Testing for DL

There are increasing amount of DL testing works developed towards being distribution-aware (as summarised in the amber route of Fig. 1). Deep generative models, such as Variational Auto-Encoders (VAE) and Generative Adversarial Networks (GAN), are applied to approximate the training data distribution, since the inputs (like images) to Deep Neural Network (DNN) are usually in a high dimensional space. Previous works heavily rely on OOD detection [4, 10] or synthesising

new test cases directly from latent spaces [6, 13, 14, 25, 44]. The former does not comprehensively consider the whole distribution, rather flags outliers, thus a more pertinent name of it should be *out-of-distribution-aware* (OODA) testing. While for both types of methods, another problem arises that the distribution encoded by generative models only contain the *feature-wise* information and easily filter out the *pixel-wise* perturbations [65]. The images added with *pixel-wise* perturbations should still fall into the local region η , which is decided by r -separation property (ref. Remark 1). Although, Dunn et al. [13] propose to generate the fine-grained perturbations by perturbing the output of last layers of GAN's generator, the fine-grained perturbations cannot be guaranteed to follow r -separation property and thus may change the class of the image. Consequently, directly searching and generating test cases from the latent space of generative models may *only perturb features*, thus called *Feature-Only Distribution-Aware* (FODA) in this paper (while also named as *semantic* AEs in some literature [19, 64]). As an example in later experiment results, i.e. Table 9, FODA will produce AEs, the perturbations of which exceed the r -separation limit. Our approach, the green route in Fig. 1, differs from aforementioned works by 1) considering both the global (feature level) distribution in latent spaces and the local (pixel level) perceptual quality distribution in the input space; 2) leveraging the robustness indicator \mathcal{R} to select test seeds which is more error-prone and thus easier for detecting AEs; 3) proposing novel GA based test case generation to detect AEs with high perceptual quality. Therefore, when HDA is experimentally compared to the existing works, the selected test seeds of HDA have higher probability density and lower robustness. The detected AEs by HDA also have higher perceptual quality (in terms of those metrics encoded by GA) than others.

2.3 Test Input Prioritisation and Generation

When testing DL based systems, in order to save computation and reduce the cost on labelling data, test input prioritisation strategies are adopted. [51] experimentally confirms that DeepGini outperforms various types of surprise and neuron coverage metrics in terms of the capability to detect misclassifications. [3] propose the unsafe set selection algorithm, leveraging the density-based clustering of error-inducing images. Both test selection methods aim at detecting misclassified images for retraining and improving generalisation performance of DNNs.

There are also some test input generation methods, the main idea of which is to promote diversity of test cases in order to cover more faults of DL systems. [37] develops a search-based tool to generate frontier inputs for DL systems. DEEPMETIS [36] augment the test set for DL by increasing the mutation scores.

The above test input generation/prioritisation methods have different goals from our HDA testing. They focus on detecting or generating more misclassified test cases within certain test budget, while our HDA testing targets at generating on-distribution AEs to improve operational robustness of DL systems. Feature distribution learnt from VAE as well as the predicted robustness indicator facilitate the generation of those on-distribution AEs.

2.4 Perceptual Quality of Images

Locally, data-points (around the selected seed) sharing the same feature information may exhibit perceptual difference from the selected seed. To capture such distribution, some *perceptual quality* metric can be utilised to compare the perceptual difference between the original image x and perturbed image x' . Some *common* metrics for perceptual quality include:

- **Mean Squared Error (MSE)**: measures the mean squared difference between the original image x and perturbed image x' ,

$$MSE = \frac{1}{n} \sum_{i=1}^n (x_i - x'_i)^2 \quad (4)$$

where n is the image size, and x_i is the value of image pixel i .

- **Peak Signal-to-Noise Ratio (PSNR)** [15]: measures the quality of a signal's representation after being subject to noise during transmission or processing. PSNR is expressed in decibels (dB) and is calculated by comparing the maximum possible power of the original signal to the power of the noise that affects its fidelity,

$$PSNR = 20 * \log_{10} \frac{MAX}{\sqrt{MSE}} \quad (5)$$

where MAX is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255.

- **Structural Similarity Index Measure (SSIM)** [49]: measures the similarity between two images based on the idea that the human visual system is highly sensitive to changes in structural information, such as edges, textures, and patterns, rather than just changes in pixel values. It works by comparing the luminance (l), contrast (c), and structural information (s) of two images, and producing a score between 0 (completely dissimilar) and 1 (identical) that indicates their similarity,

$$SSIM(x, x') = l(x, x')^\alpha \cdot c(x, x')^\beta \cdot s(x, x')^\gamma \quad (6)$$

$$l(x, x') = \frac{2\mu_x\mu_{x'} + c_1}{\mu_x^2 + \mu_{x'}^2 + c_1}, \quad c(x, x') = \frac{2\sigma_x\sigma_{x'} + c_2}{\sigma_x^2 + \sigma_{x'}^2 + c_2}, \quad s(x, x') = \frac{\sigma_{xx'} + c_3}{\sigma_x\sigma_{x'} + c_3}$$

where μ_x and $\mu_{x'}$ are the mean values of x and x' , respectively. σ_x and $\sigma_{x'}$ are the standard deviations of x and x' , respectively. $\sigma_{xx'}$ is the covariance between x and x' . c_1 , c_2 , and c_3 are constants that prevent division by zero. The constants α , β , and γ are typically set to 1, 1, and 1, respectively, although different values may be used depending on the application. Unlike other similarity measures, such as MSE and PSNR, SSIM is able to account for perceptual differences in image quality, and is often considered a more accurate measure of image quality.

- **Fréchet Inception Distance (FID)** [18]: compares the distribution between a set of original images and a set of perturbed images. It is calculated by first using a pre-trained Inception-v3 neural network [43] to extract features from both sets of images. Then, the mean and covariance of these features are calculated for each set, and the distance between these mean and covariance matrices is calculated using the Fréchet distance,

$$FID = \|\mu - \mu'\|_2^2 + Tr(\Sigma + \Sigma' - 2(\Sigma^{1/2} \cdot \Sigma' \cdot \Sigma^{1/2})^{1/2}) \quad (7)$$

where μ and μ' are the mean feature vectors of the original and perturbed image sets, respectively. Σ and Σ' are the covariance matrices of the original and perturbed image sets, respectively. $Tr()$ denotes the trace of a matrix. Lower FID scores indicate that the perturbed images are closer in terms of perception to the original images, while higher FID scores indicate greater differences between the perturbed and original images.

Notably, all these metrics are current standards for assessing the quality of images, as widely used in the experiments of aforementioned related works.

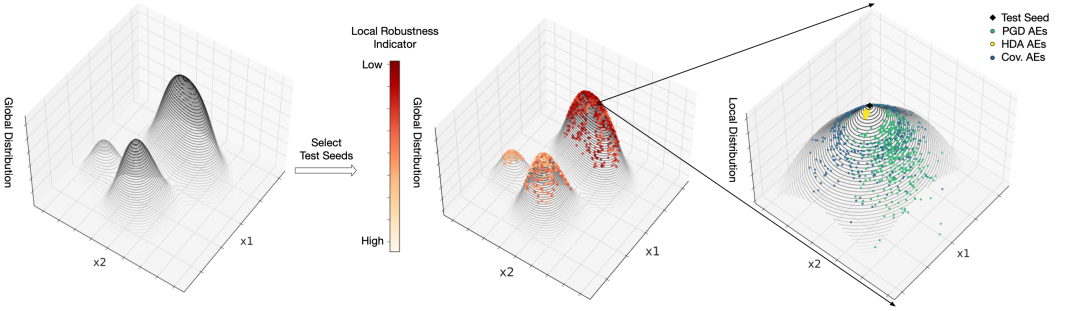


Fig. 2. An example of Hierarchical Distribution Aware Testing

3 THE PROPOSED METHOD

We first present an overview of our HDA testing, cf. the green route in Fig. 1, and then dive into details of how we implement each stage by referring to an illustrative example in Fig. 2.

3.1 Overview of HDA Testing

The core of HDA testing is the hierarchical structure of two distributions. We formally define the following two levels of distributions:

DEFINITION 6 (GLOBAL DISTRIBUTION). *The global distribution captures how **feature** level information is distributed in some (low-dimensional) latent space after data compression.*

DEFINITION 7 (LOCAL DISTRIBUTION). *Given a data-point sampled from the latent space, we consider its norm ball in the input **pixel** space. The local distribution is a conditional distribution capturing the perceptual quality of all data-points within the norm ball.*

Due to the sparsity of data over the high dimensional input space, it is hard to estimate the input distribution. Therefore, we turn to estimate the feature level (global) distribution in latent space, which is the representation of compressed data, in which data points with similar features are closer to each other [30]. DNNs, e.g. encoder of VAEs, map any data points in the high dimensional input space to the low dimensional latent space. It infers that the input space can be partitioned into well-defined regions, where each region corresponds to a particular data point in the latent space. latent space is an abstract, multidimensional space where each dimension represents a feature or characteristic of the data [30]. The regions in the input space are determined based on the values of these features and how they relate to each other. In other words, we can identify distinct patterns or clusters of data in the input space, which share the same set of features, and maps them to corresponding point in the latent space. By fitting a global distribution in the latent space, we actually model the distribution of distinct regions over the input space. The local distribution is defined as a conditional distribution within each region. Thus, we propose the following remark.

REMARK 2 (DECOMPOSE ONE DISTRIBUTION INTO TWO LEVELS). *Given the definitions of global and local distributions, denoted as p_g and p_l respectively, we may decompose a single distribution over the entire input domain X as:*

$$p(x) = \int p_l(x|x \in \eta_z)p_g(x \in \eta_z) dz \quad (8)$$

where variable z represents a set of features while η_z represents a region in the input space that “maps” to the z point in the latent space.

Intuitively, compared to modelling a single distribution, our hierarchical structure of distributions is superior in that the global distribution guides for which regions of the input space to test, while the local distribution can be leveraged to precisely control the quality of test cases. Given the definition of distribution, the goal of HDA testing can be formalised as below:

REMARK 3 (GOAL OF HDA TESTING). *The goal of HDA testing is to detect AE x' , which is around the local region of test seed x , such that x' lies within the high probability density region of distribution $p(x')$, and x and x' are classified differently.*

To detect such on distribution AEs, HDA testing has the following process, which can be divided into three stages, depicted as the green route in Fig. 1:

Stage 1: Explicitly Approximate the Global Distribution. We first extract *feature-level* information from the given dataset by using data compression techniques—the encoder of VAEs in our case, and then explicitly approximate the global distribution in the latent-feature space, using Kernel Density Estimator (KDE).

Stage 2: Select Test Seeds Based on the Global Distribution and Local Robustness Indicators. Given the limited testing budget, we want to test in those local input regions that are both more error-prone and representative of the input distribution. Thus, when selecting test seeds, we consider two factors—the local robustness indicators (cf. Definition 2) and the global distribution. For the former, we propose several auxiliary information with empirical studies showing their correlation with the local robustness, while the latter has already been quantified in the first stage via KDE.

Stage 3: Generate Test Cases Around Test Seeds Considering the Local Distribution and Prediction Loss of AEs. When searching for AEs locally around a test seed given by the 2nd stage, we develop a two-step GA in which the objective function is defined as a *fusion* of the prediction loss (cf. Definition 3) and the local distribution (modelled by common perceptual quality metrics). Such fusion of two fitness functions allows the trade-off between the “strength of being adversarial” and the perceptual quality of the detected AEs. The optimisation is subject to the constraint of only exploring in a norm ball whose central point is the test seed and with a radius smaller than the r -separation distance (cf. Remark 1).

While there are some alternatives that may also suffice for the purpose of each stage, our chosen technical solutions are the most effective and popular in consideration of the distribution awareness.

3.2 Approximation of the Global Distribution

Given the training dataset \mathcal{D} , the task of approximating the input distribution is equivalent to estimating a Probability Density Function (PDF) over the input domain \mathcal{X} given \mathcal{D} . Despite this is a common problem with many established solutions, it is hard to accurately approximate the distribution due to the relatively sparse data of \mathcal{D} , compared to the high dimensionality of the input domain \mathcal{X} . So the practical solution is to do dimensionality reduction and then estimate the global distribution, which indeed is the first step of all existing methods of distribution-aware DL testing.

Specifically, we choose VAE-Encoder+KDE³ for their simplicity and popularity. To effectively train the VAE model, we use a combination of a reconstruction loss and a KL divergence loss to optimise the model. The reconstruction loss measures the difference between the input data and the output of the decoder, while the KL divergence loss measures the difference between the learned latent distribution and the prior distribution. During training, the reconstruction loss and

³We only use the encoder of VAEs for feature extraction, rather than generate new data from the decoder, which is different to other methods mentioned in Section 2.2.

KL divergence loss are calculated and minimised simultaneously. Regularisation techniques such as dropout and weight decay can be used to prevent overfitting.

Assume \mathcal{D} contains n samples and each $x_i \in \mathcal{D}$ is encoded by VAE-Encoder as a Gaussian distribution z_i in the latent space, we can estimate the PDF of z (denoted as $Pr(z)$) based on the encoded \mathcal{D} . The $Pr(z)$ conforms to the mixture of Gaussian distributions, i.e., $z \sim \mathcal{N}(\mu_{z_i}, \sigma_{z_i})$. Notably, this mixture of Gaussian distributions nicely aligns with the gist of adaptive KDE [31], which uses the following estimator:

$$p_g(x \in \eta_z) \propto Pr(z) \approx \frac{1}{n} \sum_{i=1}^n K_{h_i}(z - \mu_{z_i}) \quad (9)$$

That is, when choosing a Gaussian kernel for K in Eqn. (9) and adaptively setting the bandwidth parameter $h_i = \sigma_{z_i}$ (i.e., the standard deviation of the Gaussian distribution representing the compressed sample z_i), the VAE-Encoder and KDE are combined “seamlessly”. Finally, our global distribution $p_g(x \in \eta_z)$ (a pooled probability of all inputs in the region η_z that corresponds to a point z in the latent space) is proportional to the approximated distribution of z with the PDF $Pr(z)$.

Running Example: The left diagram in Fig. 2 depicts the global distribution learnt by KDE, after projected to a two-dimensional space for visualisation. The peaks⁴ are evaluated with highest probability density over the latent space by KDE.

3.3 Test Seeds Selection

Selecting test seeds is actually about choosing which norm balls (around the test seeds) to test for AEs. To be cost-effective, we want to test those with higher global probabilities and lower local robustness at the same time. For the latter requirement, there is potentially a paradox:

REMARK 4 (A PARADOX OF SELECTING UNROBUST NORM BALLS). *To be informative on which norm balls to test for AEs, we need to estimate the local robustness of candidate norm balls (by invoking robustness estimators to quantify $\mathcal{R}_l(\eta, y)$, e.g., [50]). However, local robustness evaluation itself is usually about sampling for AEs (then fed into statistical estimators) that consumes the testing resources.*

To this end, instead of *directly evaluating* the local robustness of a norm ball, we can only *indirectly predict* it (i.e., without testing/searching for AEs) via auxiliary information that we call local robustness indicators (cf. Definition 2). In doing so, we save all the testing budget for the later stage when generating local test cases.

Given a test seed x with label y , we propose two robustness indicators (both relate to the vulnerability of the test seed to adversarial attacks)—the *prediction gradient based score* (denoted as S_{grad}) and the *score based on separation distance of the output-layer activation* (denoted as S_{sep}):

$$\begin{aligned} S_{grad} &= \|\nabla_x \mathcal{J}(f(x), y)\|_\infty \\ S_{sep} &= \min_{\hat{x}} \|f(x) - f(\hat{x})\|_\infty \quad \text{s.t. } y \neq \hat{y} \end{aligned} \quad (10)$$

These allow prediction of a whole norm ball’s local robustness by the limited information of its central point (the test seed). The gradient of a DNN’s prediction with respect to the input is a white-box metric, that is widely used in adversarial attacks, such as FGSM [16] and PGD [33] attacks. A greater gradient calculated at a test seed implies that AEs are more likely to be found around it. The activation separation distance is regarded as a black-box metric and refers to the minimum L_∞ norm between the output activations of the test seed and any other data with different labels. Intuitively, a smaller separation distance implies a greater vulnerability of the seed to adversarial

⁴Most training data lie in this region or gather around the region.

attacks. We later show empirically that indeed these two indicators are highly correlated with the local robustness.

After quantifying the two required factors, we combine them in a way that was inspired by [62]. In [62], the DL reliability metric is formalised as a weighted sum of local robustness where the weights are operational probabilities of local regions. To align with that reliability metric, we do the following **steps to select test seeds**:

(i) For each data-point x_i in the test set, we calculate its global probability (i.e., $Pr(z_i)$ where z_i is its compressed point in the VAE latent space) and one of the local robustness indicators (either white-box or black-box, depending on the available information).

(ii) Normalise both quantities to the same scale.

(iii) Rank all data-points by the product of their global probability and local robustness indicator.

(iv) Finally we select top- k data-points as our test seeds, and k depends on the testing budget.

Running Example: In the middle diagram of Fig. 2, we add in the local robustness indicator results of the training data which are represented by a scale of colours—darker means lower predicted local robustness while lighter means higher predicated local robustness. By our method, test seeds selected are both from the highest peak (high probability density area of the global distribution) and relatively darker ones (lower predicated local robustness).

3.4 Local Test Cases Generation

Not all AEs are equal in terms of the “strength of being adversarial”, and stronger AEs are associated with higher prediction loss (cf. Definition 3). Detecting AEs with higher prediction loss may benefit more when considering the future “debugging” step, e.g., by adversarial retraining [48]. Thus, at this stage, we want to search for AEs that exhibit a high degree of adversarial behaviour while also conform to the local distribution. That is, the local test case generation can be formulated as the following optimisation given a seed (x, y) :

$$\begin{aligned} \max_{x'} \mathcal{J}(f(x'), y) + \alpha \cdot p_l(x'|x' \in \eta_{z_x}) \\ \text{s.t. } \|x - x'\|_\infty \leq r \end{aligned} \quad (11)$$

where \mathcal{J} is the prediction loss, $p_l(x'|x' \in \eta_{z_x})$ is the local distribution (note, z_x represents the latent features of test seed x), r is the r -separation distance, and α is a coefficient to balance the two terms. As what follows, we note two points on Eqn. (11): why we need the constraint and how we quantify the local distribution.

The constraint in Eqn. (11) determines the right *locality* of local robustness—the “neighbours” that should have the same ground truth label y as the test seed. We notice the r -separation property of real-world image datasets (cf. Remark 1) provides a sound basis to the question. Thus, it is formalised as a constraint that the optimiser can only search in a norm ball with a radius smaller than r , to guarantee the detected AEs are indeed “adversarial” to label y .

While the feature level information is captured by the global distribution over a latent space, we only consider how the pixel level information is locally distributed in terms of *perceptual quality*. Three common quantitative metrics—MSE, PSNR and SSIM introduced in Section 2.4—are investigated. We note, those three metrics by no means are the true local distribution representing perceptual quality, rather quantifiable indicators from different aspects. Thus, in the optimisation problem of Eqn. (11), replacing the local distribution term with them would suffice our purpose. So, we redefine the optimisation problem as:

$$\max_{x'} \mathcal{J}(f(x'), y) + \alpha \cdot \mathcal{L}(x, x'), \quad \text{s.t. } \|x - x'\|_\infty \leq r \quad (12)$$

where $\mathcal{L}(x, x')$ represents those perceptual quality metrics correlated with the local distribution of the seed x . Certainly, implementing $\mathcal{L}(x, x')$ requires some preprocessing, e.g., normalisation and negation, depending on which metric is adopted.

Considering that the second term of the objective function in Eqn. (12) may not be differentiable and/or the DL model's parameters are not always accessible, Non-dominated Sorting Genetic Algorithm II (NSGA-II) [9] may be adopted here to solve the multi-objective optimisation. NSGA-II is designed to search for a diverse set of solutions along the Pareto-optimal front, rather than just a single solution. This means that NSGA-II produces a set of solutions that span the entire Pareto-optimal front, providing decision-makers with a range of options to choose from. Therefore, NSGA-II is usually computationally intensive and requires a huge amount of queries of DL model's prediction. Since we are only concern about test cases which have high perceptual quality and conditioned on AEs (prediction loss greater than 0), it is not necessary to produce a range of options for the trades-off between perceptual quality and prediction loss, which may waste a lot of computational resource on generating high perceptual quality but no adversarial examples. For this reason, we propose to scalarize the vector of objectives into one objective by averaging the objectives with weight vector and reduce the weight dependency with alternation mechanism. That is, we develop a GA with two fitness functions to effectively and efficiently detect AEs, as shown in Algorithm 1.

Algorithm 1 Two-Step GA Based Local Test Cases Generation

Input: Test seed (x, y) , neural network function $f(x)$, local perceptual quality metric $\mathcal{L}(x, x')$, population size N , maximum iterations T , norm ball radius r , weight parameter α , number of generated test cases m .

Output: A set of m test cases \mathcal{T}

```

1:  $F_1 = \mathcal{J}(f(x'), y), F_2 = \mathcal{J}(f(x'), y) + \alpha \cdot \mathcal{L}(x, x')$ 
2: for  $i = 1, \dots, N$  do
3:    $\mathcal{T}[i] = x + \text{uniform}(-r, +r)$ 
4: end for
5: while  $t < T$  or  $\max(\text{fit\_list}_2)$  does not converge do
6:    $\text{fit\_list}_1 = \text{cal\_fitness}(F_1, \mathcal{T})$ 
7:    $\text{fit\_list}_2 = \text{cal\_fitness}(F_2, \mathcal{T})$ 
8:   if  $\text{majority}(\text{fit\_list}_1 < 0)$  then
9:      $\text{parents} = \text{selection}(\text{fit\_list}_1, \mathcal{T})$ 
10:  else
11:     $\text{parents} = \text{selection}(\text{fit\_list}_2, \mathcal{T})$ 
12:  end if
13:   $\mathcal{T} = \text{crossover}(\text{parents}, N)$ 
14:   $\mathcal{T} = \text{mutation}(\mathcal{T}) \cup \text{parents}$ 
15:   $t = t + 1$ 
16: end while
17:  $\text{fit\_list}_2 = \text{cal\_fitness}(F_2, \mathcal{T})$ 
18:  $\text{idx} = \arg \max(\text{fit\_list}_2)[ : m]$ 
19:  $\mathcal{T} = \mathcal{T}[\text{idx}]$ 
20: return test set  $\mathcal{T}$ 

```

Algorithm 1 presents the **process of generating a set of m test cases \mathcal{T}** from a given seed x with label y (denoted as (x, y)). At line 1, we define two fitness functions (the reason behind it

will be discussed later). GA based test case generation consists of 4 steps: initialisation, selection, crossover, and mutation, the last three of which are repeated until the convergence of fitness values or the maximum iterations are reached.

Initialisation. The initialisation of population is crucial to the quick convergence. Diversity of initial population could promise approximate global optimal [26]. We initialise the population by adding uniform noise in range $(-r, +r)$ to the test seed, at line 2-4.

Selection. The fitness function is defined to select fitted individuals as parents for the latter operations. We use the fitness proportionate selection [28] for operator *selection()*.

$$p_i = \frac{\mathcal{F}_i}{\sum_{i=1}^n \mathcal{F}_i}, \mathcal{F}_i \in fit_list \tag{13}$$

The fitness value is used to associate a probability of selection p_i for each individuals to maintaining good diversity of population and avoid premature convergence. The fitness function is the objective function to be optimised. At line 6-12, we calculate two defined fitness function values on the population and select individuals based on one of the fitness values according to some judgement (the reason behind it will be discussed later).

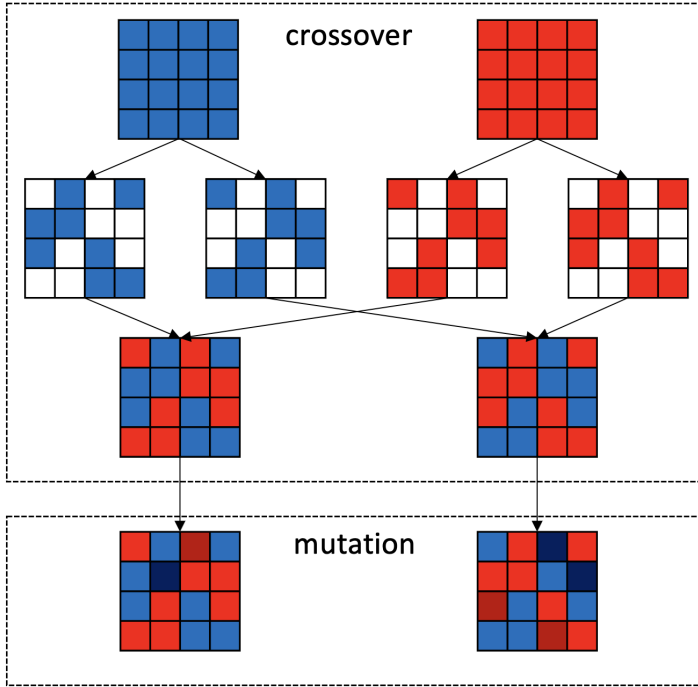


Fig. 3. Illustration of crossover and mutation in Two-Step GA Based Local Test Cases Generation

Crossover. At line 13, the crossover operator will combine a pair of parents from last step to generate a pair of children, which share many of the characteristics from the parents. The half elements of parents are randomly exchanged as an example shown in Fig. 3.

Mutation. At line 14, some elements of children are randomly altered to add variance in the evolution. It should be noticed that the mutated samples should still fall into the local region η around test seed x . Finally, the children and parents will be the individuals for the next generation.

Termination. At line 5, the termination condition of GA is either maximum number of iterations is reached or the highest ranking of fitness reaches a plateau such that successive iterations no longer produce better results.

As seen above, the main difference between our approach and common GA test case generation is that we use two fitness functions, which work alternatively to guide the selection of parents. The reason why we propose two fitness functions is because, we notice that there is a trade-off between the two objectives \mathcal{J} and \mathcal{L} in the optimisation. Prediction loss \mathcal{J} is related to the adversarial strength, while \mathcal{L} indicates the local distribution. Intuitively, generating the test cases with high local probability tends to add small amount of perturbations to the seed, while a greater perturbation is more likely to induce high prediction loss. To avoid the competition between the two terms that may finally leads to a failure of detecting AEs, we define two fitness functions to precisely control the preference at different stages:

$$F_1 = \mathcal{J}(f(x'), y), \quad F_2 = \mathcal{J}(f(x'), y) + \alpha \cdot \mathcal{L}(x, x') \quad (14)$$

At early stage, F_1 is optimised to quickly direct the generation of AEs, with $F_1 > 0$ meaning the detection of an AE. When most individuals in the population are AEs, i.e., $\text{majority}(\text{fit_list}_1 \geq 0)$, the optimisation moves to the second stage, in which F_1 is replaced by F_2 to optimise the local distribution indicator as well as the prediction loss. It is possible⁵ that the prediction loss of most individuals again becomes negative, then the optimisation will go back to the first stage. With such a mechanism of alternatively using two fitness functions in the optimisation, the proportion of AEs in the population is effectively prevented from decreasing.

Algorithm 1 describes the process for generating m local test cases given a single test seed. Suppose n test seeds are selected earlier and in total M local test cases are affordable, we can allocate, for each test seed x_i , the number of local test cases m_i , according to the n (re-normalised) global probabilities, which emphasises more on the role of distribution in our detected set of AEs.

Running Example: The right diagram in Fig. 2 plots the local distribution using MSE as its indicator, and visualises the detected AEs by different testing methods. Unsurprisingly, all AEs detected by our proposed HDA testing are located at the high density regions (and very close to the central test seed), given it considers the perceptual quality metric as one of the optimisation objectives in the two-step GA based test case generation. In contrast, other methods (PGD and coverage-guided testing) are less effective.

4 EVALUATION

We evaluate the proposed HDA testing method by performing extensive experiments to address the following research questions (RQs):

RQ1 (Effectiveness): How effective are the methods adopted in the three main stages of HDA? Namely, we conduct experiments to *i)* examine the accuracy of combining VAE-Encoder+KDE to approximate the global distribution; *ii)* check the correlation significance of the two proposed local robustness indicators with the local robustness; *iii)* investigate the effectiveness of our two-step GA for local test cases generation.

⁵Especially when a large α is used, i.e., with preference on detecting AEs with high local probability than with high adversarial strength, cf. the aforementioned trade-off.

RQ2 (AE Quality): How is the quality of AEs detected by HDA? Comparing to conventional adversarial attack and coverage-guided testing methods and more recent distribution-aware testing methods, such as OODA and FODA, we introduce a comprehensive set of metrics to evaluate the quality of AEs detected by HDA and others.

RQ3 (Sensitivity): How sensitive is HDA to the DL models under testing? We carry out experiments to assess the capability of HDA applied on DL models (adversarially trained) with different levels of robustness.

RQ4 (Robustness Growth): How useful is HDA to support robustness growth of the DL model under testing? We examine the global robustness of DL models after “fixing” the AEs detected by various testing methods.

4.1 Experiment Setup

We consider five popular benchmark datasets and five diverse model architectures for evaluation. In order to obtain statistical results, we train 10 models for each benchmark dataset, initialising their weights according to the normal distribution. We use Adam optimiser with learning rate 10^{-2} and weight decay 10^{-5} , and train 100 epochs. Details of the datasets and average accuracy (Mean \pm SD) of trained DL models under testing are listed in Table 1. The norm ball radius r is calculated based on the r -separation distance (cf. Remark 1) of each dataset. When comparing different AE detection approaches on 10 models, the evaluation results are dependent on individual model and the difference approximately follow the normal distribution by visualisation and normality test⁶. Therefore, we perform *paired two-sample T-test* and discuss the statistical significance in the experiments. The null hypothesis is that different AE detection approaches have identical average values for evaluation metrics. The calculated p -value > 0.05 indicates that the null hypothesis is true, otherwise it is false.

In **RQ1**, we validate the accuracy and effectiveness of HDA on detecting high quality AEs from high probability density region and decide the hyper-parameter settings for the following experiments. That is, for **RQ2-RQ4**, we use the activation separation distance score S_{sep} as local robustness indicator and MSE as perceptual quality metric for images. In **RQ2** we compare the quality of AEs detected by HDA and others. In **RQ3**, we add the comparison on DL models, enhanced by PGD-based adversarial training, for sensitivity analysis. Table 1 also records the accuracy of these adversarially trained models. Adversarial training trades the generalisation accuracy for the robustness as expected (thus a noticeable decrement of the training and testing accuracy) [59]. In **RQ4**, we firstly sample 10000 data points from the global distribution as validation set and detect AEs around them by different methods. Then, we fine-tune the normally trained models with training dataset augmented by these AEs. 10 epochs are taken along with ‘slow start, fast decay’ learning rate schedule [24] to reduce the computational cost while improving the accuracy-drop and robustness. To empirically estimate the global robustness on validation set, we find another set of AEs according to local distribution, different from the fine-tuning data. These AEs, crafted from validation datasets, are miss-classified by normally trained models. Thus, empirical global robustness of normally trained models is set to 0 as the baseline.

When training VAE models, the loss function is a combination of reconstruction loss and KL divergence loss. Two losses are weighted equally and simultaneously optimised. We also use Adam optimiser with learning rate 10^{-2} and weight decay 10^{-5} , and train 100 epochs to obtain VAE models. To avoid the posterior collapse that VAE often converges to a degenerated local optimum,

⁶Some statistical tests for AE Prop. and % of Valid AEs will output “NaN” due to the identical evaluation results between different approaches.

we add Batch Normalisation [66] before the output of encoder. The latent dimensions and the reconstruction loss are listed in Table 3.

For readers’ convenience, all the metrics used in **RQ2**, **RQ3** and **RQ4** for comparisons are listed in Table 2. The metrics are introduced to comprehensively evaluate the quality of detected AEs and the DL models from different aspects. When comparing HDA testing with PGD attack, coverage guided testing, OODA and FODA, we have the following settings for the tools. Specifically, we use PGD attack with 10 steps for gradient ascent, and step size $2/255$; HDA testing with population size $N = 1000$, maximum iterations $T = 500$, and weight parameter $\alpha = 1$; neuron coverage [42] with Gaussian noise $mean = 0$, $std = 1$ to generate perturbed data, and 100 iterations to increase the coverage rate by fuzzing; OODA testing [10] with 10 steps for gradient ascent, step size $2/255$, default hyperparameter for balancing two goals and default reconstruction probability threshold used in the released code ⁷; FODA testing [7] with the same latent space encoded by VAE used in HDA, random sampling to search for AEs in the latent space. To achieve the fair comparison, we set the same perturbation radius r for PGD attack, HDA testing and coverage guided testing, since they have the restrictions for the validity of test cases. In addition, we set the same number of steps and step size for PGD attack and OODA testing, and utilise the same latent space across HDA and FODA.

Table 1. Details of the datasets and DL models under testing.

Dataset	Image Size	r	DL Model	Normal Training		Adversarial Training	
				Avg. Train Acc.	Avg. Test Acc.	Avg. Train Acc.	Avg. Test Acc.
MNIST	$1 \times 32 \times 32$	0.1	LeNet5	$99.88\% \pm 0.06\%$	$98.73\% \pm 0.12\%$	$99.77\% \pm 0.01\%$	$98.84\% \pm 0.01\%$
Fashion-MNIST	$1 \times 32 \times 32$	0.08	AlexNet	$95.12\% \pm 0.85\%$	$90.70\% \pm 0.25\%$	$86.23\% \pm 0.05\%$	$85.11\% \pm 0.05\%$
SVHN	$3 \times 32 \times 32$	0.03	VGG11	$96.12\% \pm 0.45\%$	$94.86\% \pm 0.21\%$	$89.89\% \pm 0.69\%$	$90.32\% \pm 0.81\%$
CIFAR-10	$3 \times 32 \times 32$	0.03	ResNet20	$97.81\% \pm 0.58\%$	$88.28\% \pm 0.63\%$	$79.48\% \pm 0.72\%$	$76.42\% \pm 1.01\%$
CelebA	$3 \times 64 \times 64$	0.05	MobileNetV1	$94.19\% \pm 1.63\%$	$90.79\% \pm 0.49\%$	$77.74\% \pm 0.22\%$	$79.72\% \pm 0.22\%$

Table 2. Evaluation metrics for the quality of detected AEs and DL models

Metrics	Meanings
AE Prop.	Proportion of test seeds from which AEs can be detected over the total number of test seeds
Pred. Loss	Adversarial strength of AEs as formally defined by Definition 3
p_g	Normalised global probability density of test-seeds/AEs
\mathcal{R}_l	Local robustness to the correct classification label, as formally defined by Definition 1
$\hat{\mathcal{R}}_g$	Empirical global robustness of DL models over input domain as defined in Definition 5
FID	Distribution difference between original images (test seeds) and perturbed images (AEs)
ϵ	Average perturbation distance between test seeds and AEs
% of Valid AEs	Percentage of “in-distribution” AEs in all detected AEs

All experiments were run on a machine of Ubuntu 18.04.5 LTS x86_64 with Nvidia A100 GPU and 40G RAM. The source code, DL models, datasets and all experiment results are publicly available at <https://github.com/havelhuang/HDA-Testing>.

4.2 Evaluation Results and Discussions

4.2.1 RQ1. There are 3 sets of experiments in **RQ1** to examine the accuracy of technical solutions in our tool-chain, corresponding to the 3 main stages respectively.

⁷<https://github.com/swa112003/DistributionAwareDNNTesting>.

First, to approximate the global distribution, we essentially proceed in two steps—dimensionality reduction and PDF fitting, for which we adopt the VAE-Encoder+KDE solution. Notably, the VAE trained in this step is for data-compression only (not for generating new data). To reflect the effectiveness of both aforementioned steps, we (i) compare VAE-Encoder with the Principal Component Analysis (PCA), and (ii) measure the FID between the training dataset and a set of random samples drawn from the fitted global distribution by KDE.

PCA is a common approach for dimensionality reduction. We use scikit-learn [34] to implement the PCA with 'auto' solver for applying Singular Value Decomposition (SVD). That is, if the input data is larger than 500x500 and the number of components to extract (latent dimensions) is lower than 80% of the smallest dimension of the data, then the more efficient 'randomised' method [34] is enabled. Otherwise the exact full SVD is computed and optionally truncated afterwards. To achieve the fair comparison, the latent dimensions of PCA and VAE-Encoder are set to be the same. We compare the performance of VAE-Encoder and PCA from the following two perspectives. The *quality of latent representation* can be measured by the *clustering* and *reconstruction accuracy*.

To learn the feature level (global) distribution from latent data, we require that latent representations should group together data points that share similar semantic features. To evaluate this clustering ability, we apply K-means clustering to the latent data, which partitions the data points into clusters based on their similarity. Then, we calculate the Completeness Score (CS), Homogeneity Score (HS) and V-measure Score (VS) [38]. These scores provide a measure of how well the resulting clusters group together data points that share similar semantic features. Specifically, the CS measures how well the clustering captures all data points that belong to the same true class in a single cluster, while the HS measures how well the clustering captures all data points within a cluster that belong to the same true class. The VS is a harmonic mean of the CS and HS, providing an overall measure of the quality of the clustering.

In addition to ensuring that latent representations group together similar data points, we also require that the latent representations can be decoded to reconstruct the original images with minimal information loss. The reconstruction loss is calculated based on the MSE. As is shown in Table 3, VAE-Encoder achieves higher CS, HS, VS scores and less reconstruction loss than PCA. In other words, the latent representations encoded by VAE-Encoder is better in terms of capturing feature information than that of PCA.

Table 3. Quality of Latent Representation in PCA & VAE-Encoder

Dataset	Latent Dimensions	PCA			Recon. Loss	VAE-Encoder			
		Clustering				Clustering			Recon. Loss
		CS	HS	VS	CS	HS	VS		
MNIST	8	0.505	0.508	0.507	44.09	0.564	0.566	0.565	27.13
F.-MNIST	4	0.497	0.520	0.508	55.56	0.586	0.601	0.594	23.72
SVHN	4	0.007	0.007	0.007	65.75	0.013	0.011	0.015	62.38
CIFAR-10	8	0.084	0.085	0.085	188.22	0.105	0.105	0.105	168.44
CelebA	32	0.112	0.092	0.101	764.94	0.185	0.150	0.166	590.54

To evaluate the accuracy of using KDE to fit the global distribution, we calculate the FID between a new dataset (with 1000 samples) based on the fitted global distribution by KDE and the training dataset. The new dataset is sampled from the fitted global distribution over latent space and decoded by VAE decoder into images. The FID scores are shown in Table 4. As a baseline, we also present the results of using a uniform distribution over the latent space. As expected, we observe that all FID scores based on approximated distributions are significantly smaller (better). We further decode the newly generated images for visualisation in Fig. 4, from which we can see that generated images by KDE keep high fidelity while the uniformly sampled images are more difficult to be recognised.

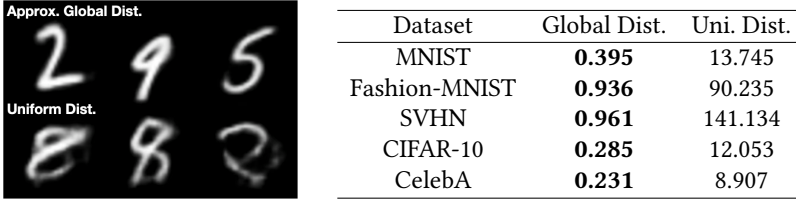


Fig. 4 & Table 4. Samples drawn from the approximated global distribution by KDE and a uniform distribution over the latent feature space (Figure); and FID to the ground truth based on 1000 samples (Table).

Answer to **RQ1** on HDA stage 1: The combination of VAE-Encoder+KDE may accurately approximate the global distribution, since the new sampled data from approximated distribution keep high fidelity.

Move on to stage 2, we study the correlations between a norm ball's local robustness and its two indicators proposed earlier—the prediction gradient based score and the score based on separation distance of output-layer activation (cf. Eq. 10).

Table 5. Pearson correlation coefficients (in absolute values) between the local robustness & its two indicators.

Dataset	S_{grad}	S_{sep}	$T(S_{grad}, S_{sep})$	
			t	$p - value$
MNIST	0.631 ± 0.025	0.564 ± 0.019	5.488	5.813×10^{-4}
Fashion-MNIST	0.717 ± 0.109	0.789 ± 0.056	-1.793	0.107
SVHN	0.747 ± 0.039	0.745 ± 0.030	0.150	0.884
CIFAR-10	0.603 ± 0.038	0.668 ± 0.065	-5.466	3.975×10^{-4}
CelebA	0.639 ± 0.073	0.728 ± 0.077	-3.872	0.004

We invoke the tool [50] for estimating the local robustness \mathcal{R}_l defined in Definition 1. Based on 1000 randomly selected data-points from the test set as the central point of 1000 norm balls, we calculate the local robustness of each norm ball⁸ as well as the two proposed indicators. Then, we do the scatter plots (in log-log scale⁹), as shown in Fig. 5. Apparently, for all 5 datasets, the indicator based on activation separation distance is negatively correlated (1st row), while the gradient based indicator is positively correlated with the estimated local robustness (2nd row). We further quantify the correlation by calculating the Pearson correlation coefficients, as recorded in Table 5. There is a rule of thumb that Pearson correlation coefficients greater than 0.6 indicate strong correlations [1]. We observe, both indicators are highly correlated with the local robustness, while the separation distance based indicator is slightly better. The statistical test shows that the separation distance based indicator is significantly better than gradient based indicator in CIFAR10 and CelebA datasets. Therefore, for the latter experiment in RQ2-RQ4, we choose separation distance based indicator S_{sep} to guide the selection of test seeds when comparing HDA testing with other AE detection methods.

⁸Radius r is usually small by definition (cf. Remark 1), yielding very small $\log(1 - \mathcal{R}_l)$.

⁹There are dots collapsed on the vertical line of $\log(1 - R) = -70$, due to a limitation of the estimator [50]—it terminates with the specified threshold when the estimation is lower than that value. Note, the correlation calculated with such noise is not undermining our conclusion, rather the real correlation would be even higher.

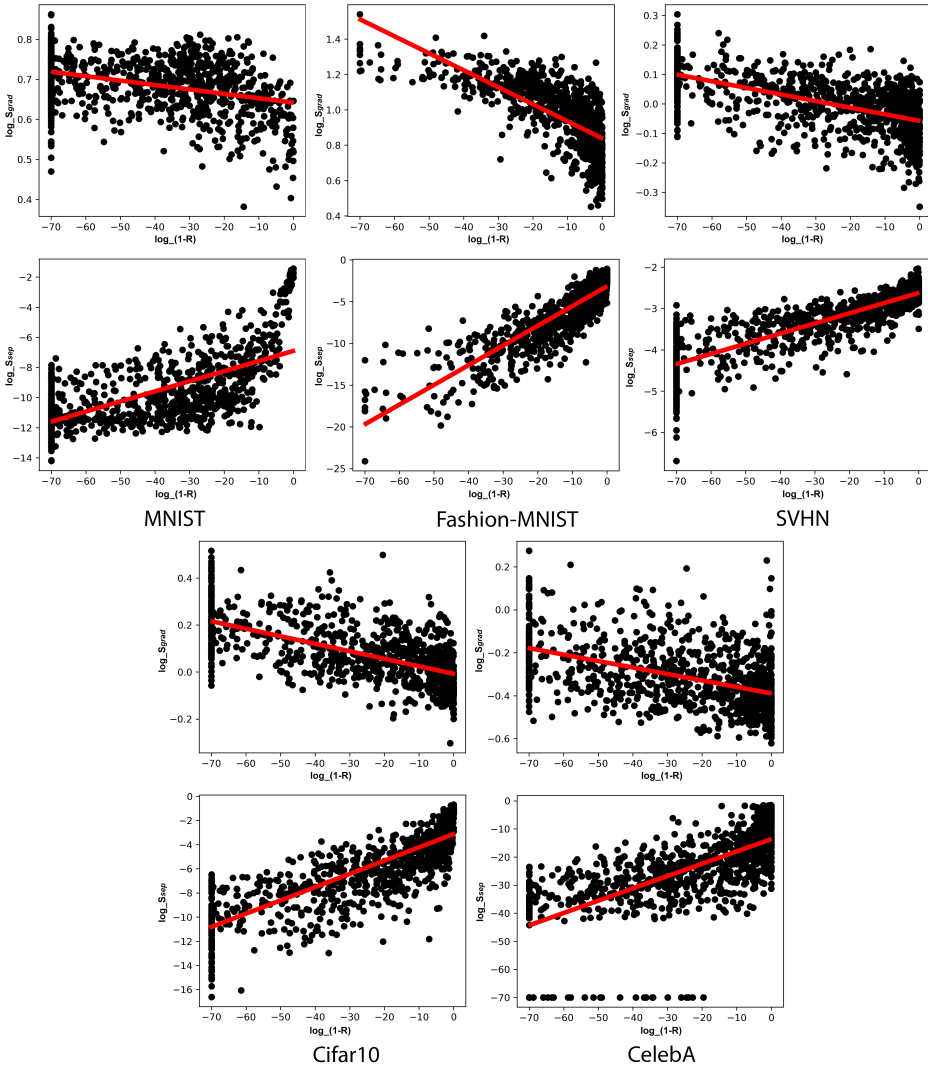


Fig. 5. Scatter plots of the local robustness evaluation vs. its two indicators, based on 1000 random norm balls.

Answer to **RQ1** on HDA stage 2: The two proposed local robustness indicators are significantly correlated with the local robustness.

For the local test case generation in stage 3, by configuring the parameter α in our two-step GA, we may do trade-off between the “strength of being adversarial” (measured by prediction loss \mathcal{J}) and the local distribution (measured by a specific perceptual quality metric \mathcal{L} , they are MSE, PSNR and SSIM), so that the quality of detected AEs can be optimised.

In Fig. 6, we visualise the changes of the two fitness values as the iterations of the GA. As shown in the first plot, only the prediction loss \mathcal{J} is taken as the fitness function (i.e., $\alpha = 0$) during the whole iteration process. The GA can effectively find AEs with maximised adversarial strength, as

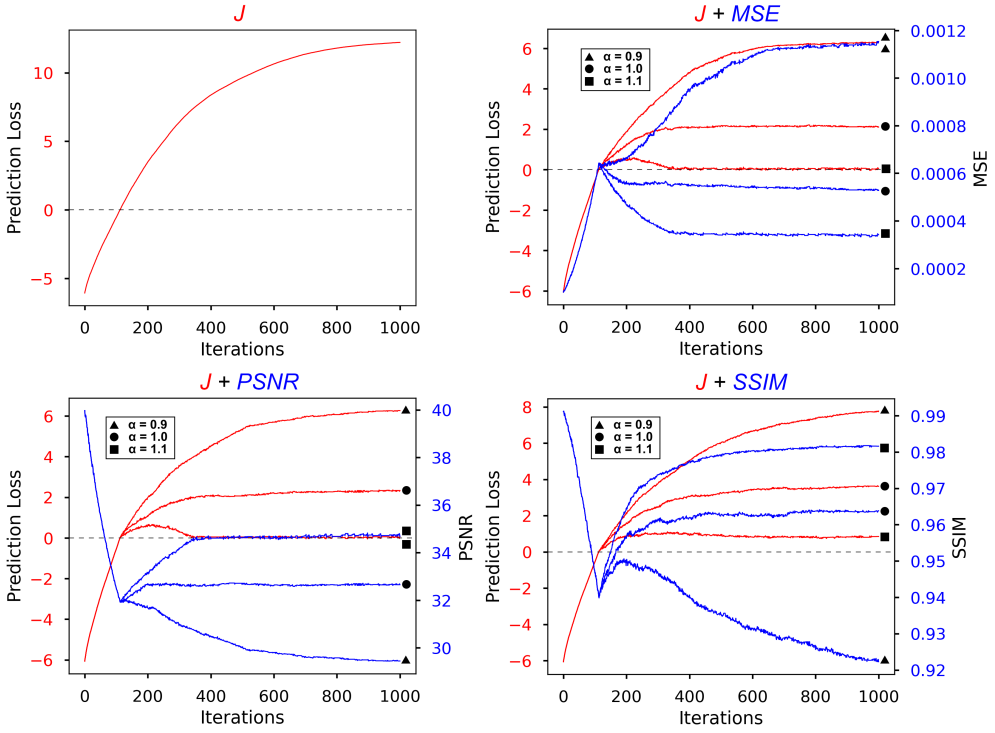


Fig. 6. The prediction loss (red) and the three quantified local distribution indicators (blue) of the best fitted test case during the iterations of our two-step GA based local test case generation.

observed by the convergence of the prediction loss of the best-fitted test case in the population after hundreds of iterations. From the second to the last plot, the fitness function consists not only of prediction loss \mathcal{J} , but also of a perceptual quality metric \mathcal{L} , representing the local distribution information (i.e., $\alpha > 0$). Intuitively, a smaller MSE or greater PSNR and SSIM implies higher local probability density.

Thanks to the two-step setting of the fitness functions, the prediction loss \mathcal{J} of best-fitted test case goes over 0 quickly in less than 200 iterations, which means it detects a first AE in the population. The \mathcal{J} of the best fitted test case is always quite close to the rest in the population, thus we may confidently claim that many AEs are efficiently detected by the population not long after the first AE was detected. Then, the optimisation goes to the second stage, in which the quantified local distribution indicator \mathcal{L} is pursued. The \mathcal{J} and \mathcal{L} finally converge and achieve a balance between them. If we configure the coefficient α , the balance point will change correspondingly. A greater α (e.g., $\alpha = 1.1$ in the plots) detects less perceptible AEs (i.e., with higher local probability density), and the price paid is that the detected AEs are with weaker adversarial strength (i.e., with smaller but still *positive* prediction loss).

We further investigate the advantages of our 2-step GA over the regular GA (using F_2 as the objective function). In Fig. 7, as α increases, the proportion of AEs in the population exhibits a sharp drop to 0 when using the regular GA. In contrast, the two-step GA prevents such decreasing of the AE proportion while preserving it at a high-level of 0.6, even when α is quite large. Moreover, larger α represents the situations when the AEs are less perceptible in terms of perceptual quality

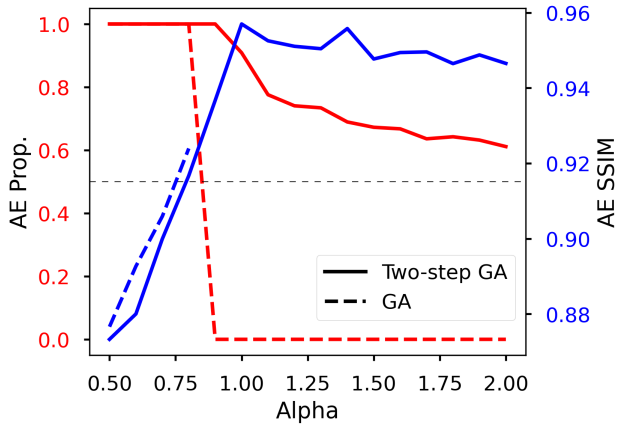


Fig. 7. Comparison between regular GA and two-step GA.

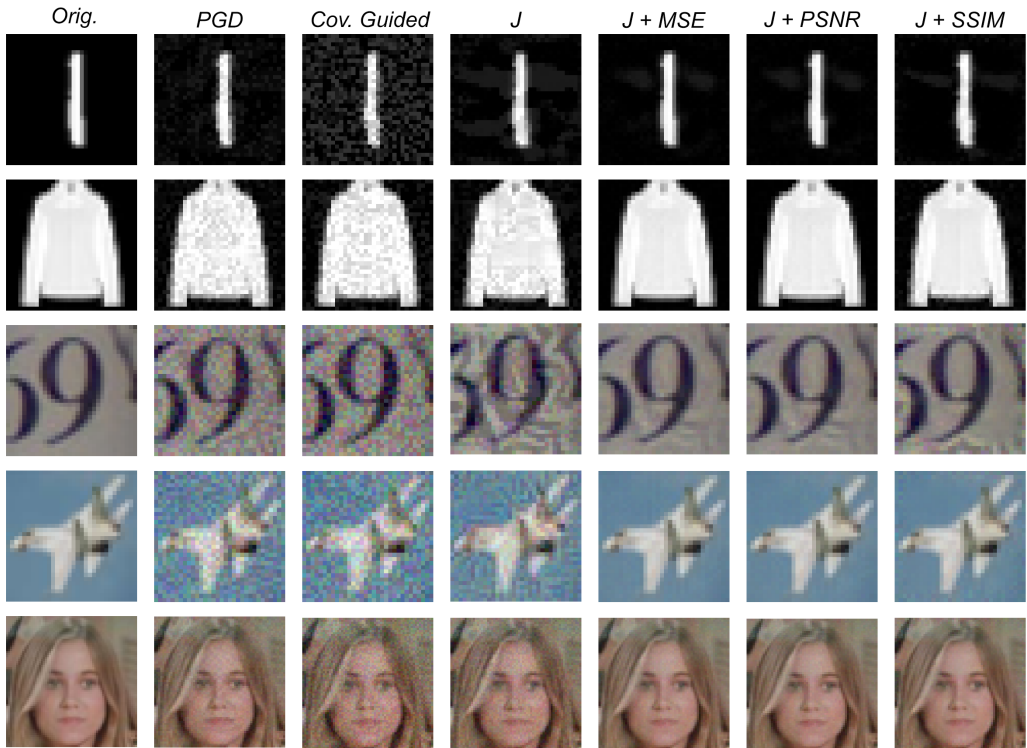


Fig. 8. AEs detected by our two-step GA (last 3 columns) & other methods

metrics—as shown by the blue curves¹⁰, the imperceptibility (measured by SSIM in this case) is only sufficiently high when α is big enough. Thus, compared to the regular GA, we may claim our

¹⁰The blue dashed line stops earlier as there is no AEs in the population when α is big.

novel 2-step GA is more robust (in detecting AEs) to the choices of α and more suitable in our framework for detecting AEs with high perceptual quality.

Table 6. Perceptual quality measured by FID between a set of original images and a set of AEs detected by two-step GA with different fitness functions.

Dataset	J	$J + MSE$	$J + PSNR$	$J + SSIM$
MNIST	1.185 ± 0.157	0.667 ± 0.202	0.668 ± 0.203	0.875 ± 0.184
Fashion-MNIST	2.736 ± 0.680	0.138 ± 0.106	0.131 ± 0.100	0.679 ± 0.182
SVHN	120.172 ± 7.529	105.399 ± 9.009	104.147 ± 9.934	111.139 ± 6.754
CIFAR-10	96.449 ± 5.906	65.727 ± 5.672	67.426 ± 7.398	76.293 ± 6.493
CelebA	85.658 ± 3.334	65.981 ± 4.976	66.599 ± 3.312	71.878 ± 3.153

(a) Results

Dataset	$T(J + MSE, J)$		$T(J + MSE, J + PSNR)$		$T(J + MSE, J + SSIM)$	
	t	$p - value$	t	$p - value$	t	$p - value$
MNIST	-13.631	2.581×10^{-7}	-0.912	0.385	-10.279	2.844×10^{-6}
Fashion-MNIST	-13.362	3.065×10^{-7}	2.188	0.056	-11.957	7.934×10^{-7}
SVHN	-9.214	7.036×10^{-6}	1.271	0.235	-4.443	0.002
CIFAR-10	-17.32	3.215×10^{-8}	-1.363	0.206	-8.125	1.954×10^{-5}
CelebA	-23.491	2.187×10^{-9}	-0.511	0.621	-3.931	0.003

(b) Statistical Test

Fig. 8 displays some selected AEs from the five datasets. Same as the PGD attack and the coverage-guided testing, if we only use the prediction loss \mathcal{J} as the objective function in the GA, the perturbations added to the images can be easily recognised. In stark contrast, AEs generated by our two-step GA (with the 3 perceptual quality metrics in the last 3 columns) are of high quality and less distinguishable from the original images (first column). We further calculate the FID to quantify the perceptual quality of detected AEs in Table 6. Since the comparison with PGD attack, coverage guided testing, FODA, and OODA are given in the subsequent experiments, i.e. Table 8 and Table 9, we focus on comparing the performance of 3 perceptual quality metrics here. Results show that MSE is significantly better than J and SSIM to guide the generation of high fidelity AEs for the experiment dataset. While MSE has similar performance with PSNR. Therefore, we decide to utilise the J+MSE in the following experiments for the comparison with the state-of-the-art.

Answer to **RQ1** on HDA stage 3: Two-step GA based local test case generation can effectively detect AEs with high perception quality (in terms of those metrics encoded by GA).

4.2.2 RQ2. We compare HDA with the state-of-the-art AE detection methods in two sets of experiments. In the first set, we focus on comparing with the adversarial attack and coverage-guided testing (i.e., the typical PGD attack and neuron coverage metric for brevity, while the conclusion can be generalised to other attacks and coverage metrics, since they all lack the consideration of distribution when detecting AEs). Then in the second set of experiments, we show the advantages of our HDA testing over other distribution-aware testing methods.

In fact, both PGD attack and coverage-guided testing do not contribute to test seeds selection. They simply use randomly sampled data from the test set as test seeds, by default. We also notice that a large amount of test seeds prioritisation metrics are proposed, the typical ones among which are Distance-Based Surprise Adequacy (DSA) [51], DeepGini [51]. Thus, we compare the randomly selected test seeds, DSA guided test seeds, DeepGini guided test seeds with our “global

distribution probability¹¹ plus local robustness indicated” test seeds, shown as “ $p_g + \mathcal{R}_l$ ” in Table 7. Specifically, for each test seed, we calculate two metrics—the local robustness \mathcal{R}_l of its norm ball and its corresponding global probability p_g . We invoke the estimator of [50] to calculate the former ($\log(1 - \mathcal{R}_l)$, to be exact). To reduce the sampling noise, we repeat the test seed selection 100 times and present the averaged results in Table 7.

Table 7. Comparison between randomly selected test seeds, DSA guided test seeds, DeepGini guided test seeds and our “ $p_g + \mathcal{R}_l$ indicated” test seeds (based on 100 test seeds).

Dataset	Random Test Seeds		DSA Test Seeds		DeepGini Test Seeds		$p_g + \mathcal{R}_l$ Test Seeds	
	$\log(1 - \mathcal{R}_l)$	p_g	$\log(1 - \mathcal{R}_l)$	p_g	$\log(1 - \mathcal{R}_l)$	p_g	$\log(1 - \mathcal{R}_l)$	p_g
MNIST	-64.67 ± 1.27	0.0034 ± 0.0005	-64.59 ± 1.15	0.0016 ± 0.0003	-65.16 ± 1.31	0.0018 ± 0.0002	-22.12 ± 3.01	0.0293 ± 0.0072
Fashion-MNIST	-14.98 ± 4.50	0.0033 ± 0.0009	-15.74 ± 4.82	0.0016 ± 0.0003	-15.29 ± 4.89	0.0017 ± 0.0003	-1.24 ± 0.48	0.0226 ± 0.0088
SVHN	-53.34 ± 3.98	0.0032 ± 0.0002	-53.44 ± 2.10	0.0013 ± 0.0002	-53.62 ± 2.32	0.0008 ± 0.0001	-8.27 ± 2.43	0.0132 ± 0.0021
CIFAR-10	-13.77 ± 1.24	0.0033 ± 0.0009	-14.25 ± 2.26	0.0018 ± 0.0003	-14.82 ± 0.86	0.0018 ± 0.0003	-2.49 ± 0.88	0.0397 ± 0.0121
CelebA	-17.68 ± 5.00	0.0034 ± 0.0002	-17.27 ± 5.44	0.0010 ± 0.0001	-17.28 ± 5.37	0.0005 ± 0.0001	-1.59 ± 0.54	0.0118 ± 0.0005

(a) Results

Dataset	Metric	$T(p_g + \mathcal{R}_l, \text{Random})$		$T(p_g + \mathcal{R}_l, \text{DSA})$		$T(p_g + \mathcal{R}_l, \text{DeepGini})$	
		t	$p - \text{value}$	t	$p - \text{value}$	t	$p - \text{value}$
MNIST	$\log(1 - \mathcal{R}_l)$	41.186	2.890×10^{-19}	41.680	2.337×10^{-19}	41.461	2.568×10^{-19}
	p_g	11.348	1.234×10^{-9}	12.155	4.101×10^{-10}	12.073	4.574×10^{-10}
Fashion-MNIST	$\log(1 - \mathcal{R}_l)$	9.601	1.665×10^{-8}	9.466	2.063×10^{-8}	9.042	4.106×10^{-8}
	p_g	6.899	1.885×10^{-6}	7.542	5.616×10^{-7}	7.506	6.001×10^{-7}
SVHN	$\log(1 - \mathcal{R}_l)$	30.564	5.769×10^{-17}	44.475	7.345×10^{-20}	42.686	1.528×10^{-19}
	p_g	14.991	1.303×10^{-11}	17.839	6.867×10^{-13}	18.651	3.203×10^{-13}
CIFAR-10	$\log(1 - \mathcal{R}_l)$	23.459	6.033×10^{-15}	15.334	8.916×10^{-12}	31.688	3.045×10^{-17}
	p_g	9.487	1.997×10^{-8}	9.902	1.039×10^{-8}	9.902	1.039×10^{-8}
CelebA	$\log(1 - \mathcal{R}_l)$	10.117	7.461×10^{-9}	9.071	3.922×10^{-8}	9.193	3.207×10^{-8}
	p_g	49.326	1.156×10^{-20}	66.979	4.830×10^{-23}	70.079	2.145×10^{-23}

(b) Statistical Test

From Table 7, we observe: (i) test seeds selected by our method have much higher global probability density, meaning their norm balls are much more representative of the data distribution; (ii) the norm balls of our test seeds have worse local robustness, meaning it is more *cost-effective* to detect AEs in them. Both metrics of HDA are significantly different from those of other test seeds selection methods, as indicated by large t-scores and small p-values. This is unsurprising because we have explicitly considered the distribution and local robustness information in the test seed selection. (iii) DSA and DeepGini guided test seeds are even worse than random test seeds, since they target at prioritising the test seeds which are prone to be misclassified. These misclassified test seeds are usually out of distribution.

Finally, the overall evaluation on the generated test cases and the detected AEs by them are shown in Table 8. The results are presented in two dimensions—3 types of testing methods versus 2 ways of test seeds selection, yielding 6 combinations (although by default, PGD attack and coverage-guided methods are using random seeds, while our method is using the “ $p_g + \mathcal{R}_l$ ” seeds). For each combination, we study 4 metrics (cf. Table 2 for meanings behind them): (i) the AE proportion; (ii) the average prediction loss; (iii) the FID¹² of the test set quantifying the image quality; and (iv)

¹¹Refer to Section 3.2 for the calculation. The value of probability density is further normalised by training dataset for a better presentation.

¹²To show how close the perturbed test cases are to the test seeds in the latent space, we use the last convolutional layer of InceptionV3 to extract the latent representations of colour images for FID. InceptionV3 is a well-trained CNN and commonly used to show FID that captures the perturbation levels, e.g., in [18]. While InceptionV3 is used for colour images, VAE is used for grey-scale datasets MNIST and Fashion-MNIST.

the computational time (and an additional coverage rate for coverage-guided testing). We note the observations on these 4 metrics in the following paragraphs.

Table 8. Evaluation of the generated test cases and detected AEs by PGD Attack, coverage-guided testing and the proposed HDA testing (all results are averaged over 100 seeds). HDA can detect more high perception quality AEs by $p_g + \mathcal{R}_l$ seeds selection, which is evidenced by high AE Prop. and low FID.

AE Detection Method	Test Seeds	Metric	MNIST	F-MNIST	SVHN	CIFAR-10	CelebA
PGD Attack	Random Seeds	AE Prop.	0.422 ± 0.046	1.000 ± 0.000	0.951 ± 0.031	1.000 ± 0.000	0.989 ± 0.035
		Pred. Loss	6.362 ± 1.089	29.200 ± 15.754	6.911 ± 0.571	46.099 ± 4.037	46.502 ± 48.154
		FID	0.705 ± 0.047	2.938 ± 0.249	106.581 ± 1.275	96.149 ± 2.265	88.401 ± 1.459
	$p_g + \mathcal{R}_l$ Seeds	AE Prop.	0.038 ± 0.070	0.035 ± 0.013	9.632 ± 0.274	9.294 ± 0.532	8.998 ± 0.646
		Pred. Loss	0.785 ± 0.103	1.000 ± 0.000	0.989 ± 0.006	1.000 ± 0.000	1.000 ± 0.000
		FID	10.184 ± 1.135	36.139 ± 18.205	9.385 ± 0.763	44.816 ± 3.221	52.136 ± 58.991
Cov. Guided Testing	Random Seeds	Time(s)	0.696 ± 0.051	1.351 ± 0.208	103.269 ± 1.153	98.592 ± 1.187	83.709 ± 1.095
		Cov. Rate	0.980 ± 0.002	0.946 ± 0.023	0.966 ± 0.002	0.979 ± 0.004	0.985 ± 0.002
		AE Prop.	0.0002 ± 0.0006	0.153 ± 0.021	0.014 ± 0.005	0.150 ± 0.016	0.080 ± 0.033
		Pred. Loss	0.019 ± 0.001	2.404 ± 1.512	0.230 ± 0.045	3.732 ± 1.098	2.841 ± 4.311
		FID	0.802 ± 0.026	3.771 ± 0.357	101.034 ± 1.014	87.899 ± 1.807	85.918 ± 1.109
		Time(s)	248.902 ± 11.088	1189.87 ± 223.93	4416.95 ± 144.93	6678.35 ± 690.49	972.17 ± 83.57
	$p_g + \mathcal{R}_l$ Seeds	Cov. Rate	0.977 ± 0.003	0.915 ± 0.025	0.921 ± 0.008	0.978 ± 0.004	0.978 ± 0.002
		AE Prop.	0.030 ± 0.017	0.485 ± 0.068	0.138 ± 0.030	0.449 ± 0.049	0.286 ± 0.056
		Pred. Loss	1.184 ± 0.632	2.867 ± 1.244	0.198 ± 0.039	3.607 ± 0.459	2.137 ± 2.283
		FID	0.804 ± 0.043	1.816 ± 0.285	94.043 ± 1.786	90.491 ± 1.071	81.243 ± 1.162
		Time(s)	81.33 ± 7.09	216.59 ± 152.62	3073.97 ± 598.92	1822.64 ± 102.24	2065.70 ± 32.68
		AE Prop.	0.208 ± 0.042	0.999 ± 0.003	0.843 ± 0.046	1.000 ± 0.000	1.000 ± 0.000
HDA	Random Seeds	Pred. Loss	1.076 ± 0.341	5.214 ± 3.325	2.784 ± 0.198	32.251 ± 2.287	16.020 ± 15.306
		FID	0.094 ± 0.009	0.119 ± 0.043	91.431 ± 1.118	62.587 ± 2.144	59.162 ± 1.469
		Time(s)	74.211 ± 1.71	136.652 ± 22.71	153.956 ± 62.92	351.992 ± 74.62	191.313 ± 68.35
	$p_g + \mathcal{R}_l$ Seeds	AE Prop.	0.676 ± 0.127	1.000 ± 0.000	0.989 ± 0.006	1.000 ± 0.000	1.000 ± 0.000
		Pred. Loss	1.591 ± 0.306	10.020 ± 5.961	3.925 ± 0.433	33.361 ± 1.873	20.702 ± 21.941
		FID	0.042 ± 0.011	0.016 ± 0.004	91.102 ± 1.342	65.829 ± 2.154	55.478 ± 1.399
Time(s)	204.002 ± 85.96	61.793 ± 1.69	237.882 ± 98.26	708.832 ± 269.81	305.499 ± 82.47		

(a) Results

Dataset	Metric	Random Seeds				$p_g + \mathcal{R}_l$ Seeds			
		$T(\text{HDA, PGD})$		$T(\text{HDA, Cov.})$		$T(\text{HDA, PGD})$		$T(\text{HDA, Cov.})$	
		t	$p - \text{value}$	t	$p - \text{value}$	t	$p - \text{value}$	t	$p - \text{value}$
MNIST	AE Prop.	-10.864	2.458×10^{-9}	15.644	6.365×10^{-12}	-2.108	0.049	15.943	4.627×10^{-12}
	FID	-40.376	4.118×10^{-19}	-81.374	1.469×10^{-24}	-39.640	5.715×10^{-19}	-54.290	2.080×10^{-21}
Fashion-MNIST	AE Prop.	-1.054	0.306	126.114	5.595×10^{-28}	nan	nan	23.949	4.204×10^{-15}
	FID	-35.279	4.541×10^{-18}	-32.117	2.400×10^{-17}	-20.293	7.495×10^{-14}	-19.970	9.882×10^{-14}
SVHN	AE Prop.	-6.157	8.186×10^{-6}	56.656	9.695×10^{-22}	nan	nan	87.961	3.629×10^{-25}
	FID	-28.252	2.312×10^{-16}	-20.119	8.690×10^{-14}	-21.746	2.259×10^{-14}	-4.163	5.843×10^{-4}
CIFAR-10	AE Prop.	nan	nan	167.996	3.221×10^{-30}	nan	nan	35.559	3.945×10^{-18}
	FID	-34.030	8.609×10^{-18}	-28.547	1.925×10^{-16}	-42.126	1.933×10^{-19}	-32.419	2.033×10^{-17}
CelebA	AE Prop.	0.994	0.333	88.160	3.485×10^{-25}	nan	nan	40.319	4.223×10^{-19}
	FID	-44.658	6.825×10^{-20}	-45.968	4.073×10^{-20}	-50.251	8.295×10^{-21}	-44.801	6.449×10^{-20}

(b) Statistical Test

Regarding the AE proportion in the set of generated test cases, the default setting of our proposed approach ($p_g + \mathcal{R}_l$ Seeds with HDA) has comparable performance with PGD attack, as indicated by identical results and *nan* output by statistical test. Both our novel $p_g + \mathcal{R}_l$ test seed selection and two-step GA local test case generation methods contribute to the comparable performance. This is evident from the decreased AE proportion when using random seeds in our method, but still the result is relatively higher than most combinations. PGD attack, as a white-box approach using the gradient information, is definitely quite efficient in detecting AEs, especially when paired with our new test seed selection method. On the other hand, coverage-guided testing is comparatively less

effective in detecting AEs (even with high coverage rate), but our test seed selection method can improve it.

As per the results of prediction loss, PGD attack, as a gradient based attack method, unsurprisingly finds the AEs with the largest prediction loss. With better test seeds considering local robustness information by our method, the prediction loss of PGD attack can be even higher. Both coverage-guided testing and our HDA testing can detect AEs with relatively lower prediction loss, meaning the AEs are with “weaker adversarial strength”. The reason for the low prediction loss of AEs detected by our approach is that two-step GA makes the trade-off and sacrifices it for higher local probabilities (i.e., perceptual quality). This can be seen through the significantly smaller FID of test set generated by HDA, compared with PGD attack and coverage-guided testing. PGD attack has relatively high FID scores, as well as coverage-guided testing.

On the computational overheads, we observe PGD is the most efficient, given it is by nature a white-box approach using the gradient information. While, our approach is an end-to-end black-box approach (if without using the gradient based indicator when selecting test seeds) requiring less information and being more generic, at the price of being relatively less efficient. That said, the computational time of our approach is still acceptable and better than coverage-guided testing.

Answer to **RQ2** on comparing with adversarial attack and coverage-guided testing: HDA can select more significant test seeds, which are from high probability density region and lack of robustness. HDA can generate higher perception quality AEs, which are measured with smaller FID values.

Table 9. Evaluation of AEs detected by OODA, FODA and our HDA testing methods (based on 100 test seeds).

Dataset	AE Detection Method	p_g	% of Valid AEs	ϵ	FID
MNIST	OODA	0.0018 ± 0.0002	22.5 ± 8.3	0.917 ± 0.031	3.463 ± 0.729
	FODA	0.0032 ± 0.0006	98.5 ± 0.4	0.558 ± 0.009	0.158 ± 0.017
	HDA	0.0292 ± 0.0071	99.3 ± 0.5	0.076 ± 0.003	0.042 ± 0.011
SVHN	OODA	0.0026 ± 0.0004	11.3 ± 1.2	0.811 ± 0.009	125.126 ± 2.838
	FODA	0.0034 ± 0.0002	100 ± 0	0.252 ± 0.012	111.694 ± 1.461
	HDA	0.0132 ± 0.0021	100 ± 0	0.029 ± 0.001	91.102 ± 1.342

(a) Results

Dataset	Metric	$T(\text{HDA, OODA})$		$T(\text{HDA, FODA})$	
		t	$p\text{-value}$	t	$p\text{-value}$
MNIST	p_g	12.199	3.871×10^{-10}	11.539	9.456×10^{-10}
	% of Valid AEs	29.208	1.286×10^{-16}	3.951	9.368×10^{-4}
	ϵ	-85.391	6.183×10^{-25}	-160.667	7.186×10^{-30}
	FID	-14.838	1.546×10^{-11}	-18.116	5.275×10^{-13}
SVHN	p_g	15.680	6.124×10^{-12}	14.691	1.826×10^{-11}
	% of Valid AEs	233.745	8.456×10^{-33}	<i>nan</i>	<i>nan</i>
	ϵ	-273.086	5.146×10^{-34}	-58.563	5.359×10^{-22}
	FID	-34.273	7.588×10^{-18}	-32.825	1.632×10^{-17}

(b) Statistical Test

Next, we try to answer the difference between HDA testing and other distribution-aware testing as summarised earlier (the amber route of Fig. 1). We not only study the common evaluation metrics in earlier RQs, but also the input validation method in [10], which flags the validity of AEs according

to a user-defined reconstruction probability threshold. Overall, HDA is significantly better than OODA and FODA in four evaluation metrics, as observed from the results of statistical test.

As shown in Table 9, HDA can select test seeds from much higher density region on the global distribution and find more valid AEs than OODA. The reason behind this is that OODA aims at detecting outliers—only AEs with lower reconstruction probabilities (from the test seed) than the given threshold will be marked as invalid test cases. While, HDA explicitly explores the high density meanwhile error-prone regions by combining the global distribution and local robustness indicators. In other words, HDA performs priority ordering (according to the global distribution and local robustness) and then selects the best, while OODA rules out the worst. As expected, FODA performs similarly poorly as OODA in terms of p_g , since both use randomly selected seeds. However, FODA has high proportion of valid AEs since the test cases are directly sampled from the distribution in latent space.

Regarding the perceptual quality of detected AEs, HDA can always find AEs with small pixel-level perturbations (ϵ) in consideration of the r -separation constraint, and with small FID thanks to the use of perceptual quality metrics (MSE in this case) as objective functions. While OODA only utilises the reconstruction probability (from VAE) to choose AEs, and FODA directly samples test cases from VAE without any restrictions (thus may suffer from the oracle problem, cf. Remark 5 later). Due to the compression nature of generative models—they are good at extracting feature level information but ignore pixel level information [65], AEs detected by OODA and FODA are all distant to the original test seeds, yielding large ϵ and FID scores. Notably, the average distance ϵ between test seeds and AEs detected by OODA and FODA are much (7~28 times) greater than the r -separation constraints (cf. Table 1), leading to the potential oracle issues of those AEs, for which we have the following remark:

REMARK 5 (ORACLE ISSUES OF AEs DETECTED BY OODA AND FODA). *AEs detected by OODA and FODA are normally distant to the test seeds with a perturbation distance even greater than the r -separation constraint. Consequently, there is the risk that the perturbed image may not share the same ground truth label of the test seed, and thus hard to determine the ground truth label of the “AE”¹³.*

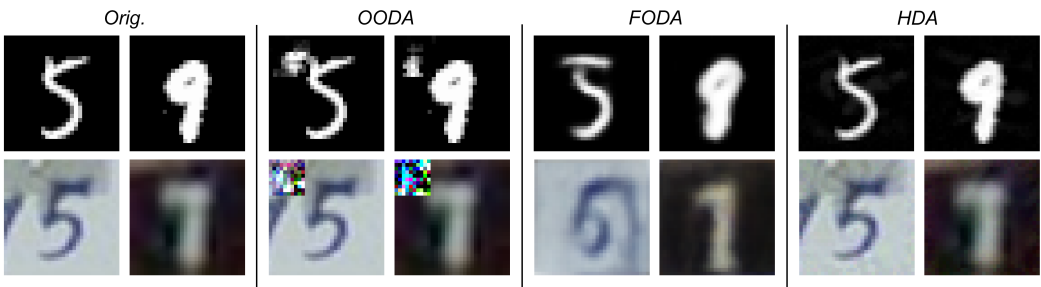


Fig. 9. Example AEs detected by different distribution-aware testing methods. AEs detected by our HDA are indistinguishable from the original images, while AEs detected by FODA and OODA are of low perceptual quality and subject to the oracle issues noted by Remark 5.

To visualise the difference between AEs detected by HDA, FODA and OODA, we present 4 examples in Fig. 9. We may observe the AEs detected by HDA are almost indistinguishable from

¹³In quotes, because the perturbed image could be a “benign example” with a *correct* predicted label (but different to the test seed).

the original images. Moreover, the AEs by FODA is a set of concrete evidence for Remark 5—it is actually quite hard to tell what is the ground truth label of some perturbed image (e.g., the bottom left one), while others appear to have a different label of the seed (e.g., the bottom right one should be with a label “1” instead of “7”).

Answer to **RQ2** on comparing with other distribution-aware testing: Compared to OODA and FODA, the proposed HDA testing can detect more valid AEs, free of oracle issues, with higher global probabilities and perception quality.

4.2.3 RQ3. In earlier RQs, we have varied the datasets and model architectures to check the effectiveness of HDA. In this **RQ3**, we are concern about HDA’s sensitivity to DL models with different levels of robustness. Adversarial training may greatly improve the robustness of DL models and is widely used as the defence to adversarial attack. To this end, we apply HDA on both normally and *adversarially trained* models (by [33] to be exact), and then compare with three most representative adversarial attack methods—the most classic FGSM, the most popular PGD, and the most advanced AutoAttack [8]. Experimental results are presented in Table 10.

As expected, after the adversarial training by [33], the robustness of all five DL models are greatly improved. This can be observed from the metric of AE Prop.: For all four methods, the proportion of AEs detected in the set of test case is sharply decreased for adversarially trained models. Nevertheless, the AE detection performance of HDA is less affected by the adversarial training. HDA testing can maintain significantly higher proportion of AEs in the test set for adversarially trained models, compared with the state of art adversarial attack. This is evident from the fact that there is an insignificant difference in AE Prop. between HDA and other adversarial attacks for normally trained models, while the difference is significant for adversarially trained models.

In terms of the probability density p_g and perception quality measured by FID on generated test cases, HDA significantly outperforms others for both normally and adversarially trained models, as observed from the large t-scores and p-values $\ll 0.05$. This is unsurprising, since the rationales behind the three adversarial attacks disregard the consideration of input data distribution and perception quality.

Finally, we find that the measured p_g on test cases detected by HDA changes due to the variations in local robustness before and after the adversarial training, yet it remained much higher than all other attack methods.

Answer to **RQ3**: HDA is shown to be capable and superior to common adversarial attacks when applied on DL models with different levels of robustness.

4.2.4 RQ4. The ultimate goal of developing HDA testing is to improve the global robustness of DL models. To this end, we refer to a validation set of 10000 test seeds. We fine-tune [24] the DL models with AEs detected for validation set from different methods. Then, we calculate the train accuracy, test accuracy and empirical global robustness before and after the adversarial fine-tuning. Empirical global robustness is measured on a new set of on-distribution AEs for validation set, different from the fine-tuning data. Results are presented in Table 11.

We first observe that adversarial fine-tuning is effective to improve the DL models’ empirical global robustness, measured by the prediction accuracy on AEs, detected from normally trained models (\mathcal{R}_g), while compromising the train/test accuracy as expected (in contrast to normal training in Table 1). In most cases, DL models enhanced by HDA testing suffer from the least drop of

Table 10. Evaluation of AEs generated by FGSM, PGD, AutoAttack and HDA on normally and adversarially trained DL models (all results are averaged over 100 test seeds).

Model	AE Detection Method	Eval. Metric	MNIST	F.-MNIST	SVHN	CIFAR-10	CelebA
Normally Trained	FGSM	p_g	0.0045 ± 0.0005	0.0034 ± 0.0008	0.0031 ± 0.0002	0.0028 ± 0.0004	0.0034 ± 0.0002
		AE Prop.	0.413 ± 0.093	0.797 ± 0.039	0.727 ± 0.069	0.893 ± 0.016	0.989 ± 0.035
		FID	1.022 ± 0.053	4.134 ± 0.322	114.256 ± 2.931	104.877 ± 1.832	88.401 ± 1.459
	PGD	p_g	0.0040 ± 0.0005	0.0034 ± 0.0008	0.0032 ± 0.0002	0.0028 ± 0.0004	0.0034 ± 0.0002
		AE Prop.	0.422 ± 0.046	1.000 ± 0.000	0.984 ± 0.011	1.000 ± 0.000	0.989 ± 0.035
		FID	0.705 ± 0.047	2.938 ± 0.248	107.370 ± 1.206	101.157 ± 1.774	88.401 ± 1.459
	AutoAttack	p_g	0.0042 ± 0.0004	0.0034 ± 0.0008	0.0032 ± 0.0002	0.0028 ± 0.0004	0.0035 ± 0.0002
		AE Prop.	0.787 ± 0.075	1.000 ± 0.000	0.984 ± 0.011	1.000 ± 0.000	1.000 ± 0.000
		FID	0.795 ± 0.099	4.962 ± 0.359	106.780 ± 1.297	101.173 ± 1.543	90.385 ± 1.305
	HDA	p_g	0.0292 ± 0.0071	0.0224 ± 0.0087	0.0132 ± 0.0021	0.0406 ± 0.012	0.0124 ± 0.001
		AE Prop.	0.676 ± 0.127	1.000 ± 0.000	0.989 ± 0.006	1.000 ± 0.000	1.000 ± 0.000
		FID	0.042 ± 0.011	0.016 ± 0.004	91.102 ± 1.342	65.829 ± 2.154	55.478 ± 1.399
Adversarially Trained	FGSM	p_g	0.0042 ± 0.0005	0.0033 ± 0.0004	0.0036 ± 0.0004	0.0028 ± 0.0005	0.0031 ± 0.0003
		AE Prop.	0.033 ± 0.021	0.208 ± 0.037	0.323 ± 0.061	0.455 ± 0.039	0.404 ± 0.045
		FID	1.064 ± 0.055	4.898 ± 0.503	135.730 ± 4.321	113.106 ± 3.989	108.070 ± 4.368
	PGD	p_g	0.0038 ± 0.0005	0.0032 ± 0.0004	0.0034 ± 0.0004	0.0028 ± 0.0005	0.0032 ± 0.0003
		AE Prop.	0.014 ± 0.017	0.201 ± 0.039	0.507 ± 0.049	0.455 ± 0.039	0.372 ± 0.036
		FID	0.762 ± 0.040	3.871 ± 0.407	124.388 ± 2.613	113.106 ± 3.989	111.931 ± 2.539
	AutoAttack	p_g	0.0031 ± 0.0004	0.0031 ± 0.0004	0.0034 ± 0.0004	0.0028 ± 0.0005	0.0031 ± 0.0003
		AE Prop.	0.049 ± 0.021	0.268 ± 0.038	0.505 ± 0.049	0.419 ± 0.039	0.405 ± 0.045
		FID	0.035 ± 0.009	0.128 ± 0.046	125.382 ± 2.315	117.257 ± 3.750	107.742 ± 4.169
	HDA	p_g	0.0214 ± 0.0075	0.0118 ± 0.0032	0.0154 ± 0.0021	0.0261 ± 0.0112	0.0103 ± 0.0007
		AE Prop.	0.368 ± 0.071	0.903 ± 0.021	0.960 ± 0.029	0.865 ± 0.051	0.968 ± 0.012
		FID	0.029 ± 0.003	0.058 ± 0.010	92.389 ± 1.152	62.804 ± 2.398	58.558 ± 0.895

(a) Results

Model	Dataset	Metric	$T(\text{HDA, FGSM})$		$T(\text{HDA, PGD})$		$T(\text{HDA, AutoAttack})$	
			t	p -value	t	p -value	t	p -value
Normally Trained	MNIST	p_g	10.97	2.10×10^{-9}	11.20	1.53×10^{-9}	11.12	1.71×10^{-9}
		AE Prop.	5.28	5.05×10^{-5}	5.95	1.26×10^{-5}	-2.38	0.03
		FID	-57.25	8.04×10^{-22}	-43.43	1.12×10^{-19}	-23.91	4.34×10^{-15}
	F.-MNIST	p_g	6.88	1.97×10^{-6}	6.88	1.97×10^{-6}	6.88	1.97×10^{-6}
		AE Prop.	16.46	2.70×10^{-12}	nan	nan	nan	nan
		FID	-40.44	4.01×10^{-19}	-37.25	1.72×10^{-18}	-43.56	1.06×10^{-19}
	SVHN	p_g	15.14	1.10×10^{-11}	14.99	1.30×10^{-11}	14.99	1.30×10^{-11}
		AE Prop.	11.96	5.31×10^{-10}	1.26	0.22	1.26	0.22
		FID	-22.71	1.05×10^{-14}	-28.51	1.97×10^{-16}	-26.56	6.84×10^{-16}
	CIFAR-10	p_g	9.96	9.56×10^{-9}	9.96	9.56×10^{-9}	9.96	9.56×10^{-9}
		AE Prop.	21.15	3.67×10^{-14}	nan	nan	nan	nan
		FID	-43.67	1.02×10^{-19}	-40.03	4.79×10^{-19}	-42.18	1.89×10^{-19}
CelebA	p_g	27.91	2.87×10^{-16}	27.91	2.87×10^{-16}	27.59	3.49×10^{-16}	
	AE Prop.	0.99	0.33	0.99	0.33	nan	nan	
	FID	-51.51	5.34×10^{-21}	-51.51	5.34×10^{-21}	-57.69	6.99×10^{-22}	
Adversarially Trained	MNIST	p_g	7.24	9.92×10^{-7}	7.40	7.24×10^{-7}	7.71	4.17×10^{-7}
		AE Prop.	14.31	2.83×10^{-11}	15.33	8.9×10^{-12}	13.62	6.37×10^{-11}
		FID	-59.42	4.12×10^{-22}	-57.79	6.81×10^{-22}	-2.00	0.06
	F.-MNIST	p_g	8.33	1.36×10^{-7}	8.43	1.15×10^{-7}	8.53	9.70×10^{-8}
		AE Prop.	51.66	5.06×10^{-21}	50.12	8.70×10^{-21}	46.25	3.65×10^{-20}
		FID	-30.42	6.26×10^{-17}	-29.62	1.06×10^{-16}	-4.70	1.78×10^{-4}
	SVHN	p_g	17.46	9.95×10^{-13}	17.75	7.47×10^{-13}	17.75	7.47×10^{-13}
		AE Prop.	29.82	8.90×10^{-17}	25.16	1.77×10^{-15}	25.27	1.64×10^{-15}
		FID	-30.65	5.50×10^{-17}	-35.43	4.19×10^{-18}	-40.35	4.17×10^{-19}
	CIFAR-10	p_g	6.57	3.57×10^{-6}	6.57	3.57×10^{-6}	6.57	3.57×10^{-6}
		AE Prop.	20.19	8.15×10^{-14}	20.19	8.15×10^{-14}	21.97	1.89×10^{-14}
		FID	-34.18	7.98×10^{-18}	-34.18	7.98×10^{-18}	-38.69	8.82×10^{-19}
CelebA	p_g	29.90	8.52×10^{-17}	29.48	1.09×10^{-16}	29.90	8.52×10^{-17}	
	AE Prop.	38.30	1.05×10^{-18}	49.67	1.02×10^{-20}	38.23	1.09×10^{-18}	
	FID	-35.12	4.93×10^{-18}	-62.69	1.58×10^{-22}	-36.48	2.51×10^{-18}	

(b) Statistical Test

Table 11. Evaluation of DL models' train accuracy, test accuracy, and empirical global robustness (based on 10000 on-distribution AEs) after adversarial fine-tuning, using different number (N) of test cases.

AE Detection Method	Metric	MNIST			SVHN		
		N = 500	N = 5000	N = 50000	N = 500	N = 5000	N = 50000
PGD Attack	Train Acc.	98.26% ± 0.05%	98.10% ± 0.07%	97.70% ± 0.05%	94.85% ± 0.43%	92.76% ± 0.39%	94.02% ± 0.40%
	Test Acc.	97.64% ± 0.11%	97.05% ± 0.09%	96.90% ± 0.08%	93.32% ± 0.21%	81.43% ± 0.31%	63.81% ± 0.28%
	\mathcal{R}_g	46.27% ± 0.02%	84.09% ± 0.05%	90.52% ± 0.05%	46.43% ± 0.11%	72.88% ± 0.10%	70.09% ± 0.12%
Cov. Guided Testing	Train Acc.	99.89% ± 0.05%	99.61% ± 0.07%	99.12% ± 0.05%	95.73% ± 0.44%	93.65% ± 0.42%	95.76% ± 0.43%
	Test Acc.	98.93% ± 0.08%	98.77% ± 0.08%	98.41% ± 0.07%	94.11% ± 0.19%	85.66% ± 0.19%	76.43% ± 0.19%
	\mathcal{R}_g	36.71% ± 0.04%	48.91% ± 0.04%	71.12% ± 0.05%	16.21% ± 0.11%	38.59% ± 0.16%	56.58% ± 0.12%
OODA	Train Acc.	98.45% ± 0.06%	98.01% ± 0.06%	97.66% ± 0.07%	94.12% ± 0.38%	92.11% ± 0.39%	93.21% ± 0.38%
	Test Acc.	98.12% ± 0.10%	97.87% ± 0.09%	97.12% ± 0.08%	94.75% ± 0.19%	80.23% ± 0.18%	72.19% ± 0.19%
	\mathcal{R}_g	40.21% ± 0.05%	45.69% ± 0.04%	51.12% ± 0.05%	11.21% ± 0.11%	16.23% ± 0.12%	18.21% ± 0.11%
FODA	Train Acc.	98.44% ± 0.05%	98.18% ± 0.05%	97.32% ± 0.06%	94.66% ± 0.42%	92.75% ± 0.41%	94.11% ± 0.41%
	Test Acc.	97.87% ± 0.10%	97.43% ± 0.10%	97.11% ± 0.11%	92.13% ± 0.22%	82.32% ± 0.21%	78.19% ± 0.22%
	\mathcal{R}_g	37.71% ± 0.04%	47.26% ± 0.05%	55.37% ± 0.05%	12.21% ± 0.12%	20.56% ± 0.12%	23.98% ± 0.11%
HDA	Train Acc.	99.56% ± 0.07%	99.10% ± 0.07%	98.89% ± 0.06%	95.00% ± 0.33%	92.83% ± 0.39%	94.40% ± 0.40%
	Test Acc.	98.52% ± 0.09%	98.42% ± 0.08%	98.21% ± 0.08%	93.86% ± 0.24%	88.67% ± 0.25%	80.60% ± 0.22%
	\mathcal{R}_g	89.67% ± 0.03%	96.71% ± 0.06%	99.12% ± 0.05%	51.15% ± 0.09%	86.88% ± 0.09%	91.26% ± 0.05%

(a) Results

Dataset	No. of Test Cases	Metric	T(HDA, PGD)		T(HDA, Cov.)		T(HDA, OODA)		T(HDA, FODA)		
			<i>t</i>	<i>p-value</i>	<i>t</i>	<i>p-value</i>	<i>t</i>	<i>p-value</i>	<i>t</i>	<i>p-value</i>	
MNIST	N=500	Train Acc.	47.79	2.04×10^{-20}	-12.13	4.24×10^{-10}	38.07	1.17×10^{-18}	41.17	2.91×10^{-19}	
		Test Acc.	19.58	1.39×10^{-13}	-10.77	2.83×10^{-9}	9.40	2.29×10^{-8}	15.28	9.47×10^{-12}	
		\mathcal{R}_g	3806.43	1.31×10^{-54}	3349.48	1.31×10^{-53}	2682.35	7.12×10^{-52}	3286.24	1.84×10^{-53}	
	N=5000	Train Acc.	31.94	2.64×10^{-17}	-16.29	3.21×10^{-12}	37.39	1.62×10^{-18}	33.82	9.61×10^{-18}	
		Test Acc.	35.98	3.20×10^{-18}	-9.78	1.25×10^{-8}	14.44	2.42×10^{-11}	24.45	2.94×10^{-15}	
		\mathcal{R}_g	510.97	6.52×10^{-39}	2096.17	6.03×10^{-50}	2237.38	1.86×10^{-50}	2002.17	1.38×10^{-49}	
	N=50000	Train Acc.	48.18	1.76×10^{-20}	-9.31	2.64×10^{-8}	42.19	1.88×10^{-19}	58.51	5.45×10^{-22}	
		Test Acc.	36.62	2.35×10^{-18}	-5.95	1.25×10^{-5}	30.47	6.10×10^{-17}	25.57	1.33×10^{-15}	
		\mathcal{R}_g	384.60	1.08×10^{-36}	1252.19	6.42×10^{-46}	2146.63	3.93×10^{-50}	1956.56	2.08×10^{-49}	
	SVHN	N=500	Train Acc.	0.88	0.39	-4.19	5.42×10^{-4}	5.53	2.99×10^{-5}	2.01	0.06
			Test Acc.	5.35	4.34×10^{-5}	-2.58	0.02	-9.19	3.20×10^{-8}	16.80	1.90×10^{-12}
			\mathcal{R}_g	105.02	1.50×10^{-26}	777.40	3.42×10^{-42}	888.65	3.08×10^{-43}	820.93	1.28×10^{-42}
N=5000		Train Acc.	0.40	0.69	-4.52	2.63×10^{-4}	4.13	6.31×10^{-4}	0.45	0.66	
		Test Acc.	57.49	7.46×10^{-22}	30.31	6.67×10^{-17}	86.64	4.77×10^{-25}	61.50	2.23×10^{-22}	
		\mathcal{R}_g	329.07	1.79×10^{-35}	831.84	1.01×10^{-42}	1489.43	2.83×10^{-47}	1398.15	8.83×10^{-47}	
N=50000		Train Acc.	2.12	0.05	-7.32	8.43×10^{-7}	6.82	2.19×10^{-6}	1.60	0.13	
		Test Acc.	149.10	2.75×10^{-29}	45.36	5.16×10^{-20}	91.49	1.79×10^{-25}	24.50	1.83×10^{-15}	
		\mathcal{R}_g	514.96	5.67×10^{-39}	843.60	7.86×10^{-43}	1911.81	3.16×10^{-49}	1760.80	1.39×10^{-48}	

(b) Statistical Test

generalisation, compared with PGD attack, OODA and FODA. This can be seen from significant difference of Test Acc. between HDA and others. The reason behind this is that HDA testing targets at AEs from high density regions on distributions, usually with small prediction loss, shown in Fig. 6. Thus, eliminating AEs detected by HDA testing requires relatively minor adjustment to DL's models, the generalisation of which can be easily tampered during the fine-tuning with new samples.

In terms of empirical global robustness, HDA testing detects AEs around test seeds from the high global distribution region, which are more significant to the global robustness improvement. When comparing HDA with other methods, the p-values of \mathcal{R}_g is way smaller than 0.05, indicating HDA contributes more to the global robustness improvement than others. It also can be seen that with 5000 test cases generated by utilising 1000 test seeds, the HDA testing can improve empirical global robustness to nearly or over 90%, very closed to the fine-tuning with 50000 test cases from 10000 test seeds. This means the distribution-based test seeds selection is more efficient than random test seeds selection. Moreover, even fine-tuning with 50000 test cases, leveraging all the test seeds in the validation set, HDA is still better than others, due to the consideration of local distributions

(approximated by perceptual quality metrics). We notice that PGD-based adversarial fine-tuning minimises the maximum prediction loss within the local region, which is also effective to eliminate the high perceptual quality AEs, but sacrificing more train/test accuracy. DL models fined-tuned with HDA testing achieve the best balance between the generalisation and global robustness.

Answer to **RQ4**: Compared with adversarial attack and coverage-guide testing, HDA contributes more to the growth of global robustness, while mitigating the drop of train/test accuracy during adversarial fine-tuning.

5 THREATS TO VALIDITY

5.1 Internal Validity

Threats may arise due to bias in establishing cause-effect relationships, simplifications and assumptions made in our experiments. In what follows, we list the main threats of each research question and discuss how we mitigate them.

5.1.1 Threats from HDA Techniques. In **RQ1**, both the performance of the VAE-Encoder and KDE are threats. For the former, it is mitigated by using four established quality metrics (in Table 3) on evaluating dimensionality reduction techniques and compared to the common PCA method. It is known that KDE performs poorly with high-dimensional data and works well when the data dimension is modest [29, 40]. The data dimensions in our experiments are relatively low given the datasets have been compressed by VAE-Encoder, which mitigates the second threat. When studying the local robustness indicators, quantifying both the indicators and the local robustness may subject to errors, for which we reduce them by carefully inspecting the correctness of the script on calculating the indicators and invoking a reliable local robustness estimator [50] with fine-tuned hyper-parameters. For using two-step GA to generate local test cases, a threat arises by the calculation of norm ball radius, which has been mitigated by r -separation distance presented in the paper [57]. Also, the threat related to estimating the local distribution is mitigated by quantifying its three indicators (MSE, PSNR and SSIM) that are typically used in representing image-quality by human-perception.

5.1.2 Threats from AEs' Quality Measurement. A threat for **RQ1**, **RQ2** and **RQ3** (when examining how effective our method models the global distribution and local distribution respectively) is the use of FID as a metric, quantifying how "similar" two image datasets are. Given FID is currently the standard metric for this purpose, this threat is sufficiently mitigated now and can be further mitigated with new metrics in future. **RQ2** includes the method of validating AEs developed in [10], which utilises generative models and OOD techniques to flag valid AEs with reconstruction probabilities greater than a threshold. The determination of this threshold is critical, thus poses a threat to **RQ2**. To mitigate it, we use same settings across all the experiments for fair comparisons.

5.1.3 Threats from Adversarial Training and Fine-Tuning. In **RQ3** and **RQ4**, the first threat rises from the fact that adversarial training and adversarial fine-tuning will sacrifice the DL model's generalisation for robustness. Since the training process is data-driven and of black-box nature, it is hard to know how the predication of a single data-point will be affected, while it is meaningless to study the robustness of an incorrectly predicted seed. To mitigate this threat when we compare the robustness before and after adversarial training/fine-tuning, we select enough number of seeds and check the prediction of each selected seed (filtering out incorrect ones if necessary) to make sure test seeds are always predicted correctly. For the global robustness computation in **RQ4**, we refer to a validation dataset, where a threat may arise if the empirical result based on the validation dataset

cannot represent the global robustness. To mitigate it, we synthesise the validation set with enough data—10000 inputs sampled from global distribution. We further attack the validation dataset to find an AE per seed according to the local distribution. Thus, DL models’ prediction accuracy on this dataset empirically represents the global robustness as defined. For the training/fine-tuning to be effective, we need a sufficient number of AEs to augment the training dataset. A threat may arise due to a small proportion of AEs in the augmented training dataset (the DL model will be dominated by the original training data during the training/fine-tuning). To mitigate such a threat, we generate a large proportion of AEs in our experiments.

5.2 External Validity

Threats might challenge the generalisability of our findings, e.g. the number of models and datasets considered for experimentation; thus we mitigate these threats as follows. All our experiments are conducted on 5 popular benchmark datasets, covering 5 typical types of DL models, cf. Table 1. Experimental results on the effectiveness of each stage in our framework are all based on averaging a large number of samples, reducing the random noise in the experiments. In two-step GA based test case generation, a wide range of the α parameter has been studied showing converging trends. Finally, we enable replication by making all experimental results publicly available/reproducible on our project website to further mitigate the threat.

6 CONCLUSION & FUTURE WORK

In this paper, we propose a HDA testing approach for detecting AEs that considers both the data distribution (thus with higher operational impact assuming the training data statistically representing the future inputs) and perceptual quality (thus looks natural and realistic to humans). The key novelty lies in the hierarchical consideration of two levels of distributions. To the best of our knowledge, it is the first DL testing approach that explicitly and collectively models both (i) the feature-level information when selecting test seeds and (ii) pixel-level information when generating local test cases. To this end, we have developed a tool chain that provides technical solutions for each stage of our HDA testing. Our experiments not only show the effectiveness of each testing stage, but also the overall advantages of HDA testing over state-of-the-arts. From a software engineering’s perspective, HDA is cost-effective (by focusing on practically meaningful AEs), flexible (with end-to-end, black-box technical solutions) and may effectively contribute to the robustness growth of the DL software under testing.

The purpose of detecting AEs is to fix them. Although existing DL retraining/repairing techniques (e.g. [24] used in **RQ4** and [48, 58]) may satisfy the purpose to some extent, *bespoke* “debugging” methods with more emphasise on the feature-distribution and perceptual quality can be integrated into our framework in a more efficient way. To this end, our important future work is to close the loop of “detect-fix-assess” as depicted in [63] and then organise all generated evidence as safety cases [11, 61]. Finally, same as other distribution-aware testing methods, we assume the input data distribution is same as the training data distribution. To relax this assumption, we plan to take distribution-shift into consideration in future versions of HDA. Distribution-aware testing for systematically detecting explanation AEs [21] will also be explored.

ACKNOWLEDGMENTS

This work is supported by the U.K. DSTL (through the project of Safety Argument for Learning-enabled Autonomous Underwater Vehicles) and the U.K. EPSRC (through End-to-End Conceptual Guarding of Neural Architectures [EP/T026995/1]). Xingyu Zhao and Alec Banks’ contribution to the work is partially supported through Fellowships at the Assuring Autonomy International Programme. This project has received funding from the European Union’s Horizon 2020 research

and innovation programme under grant agreement No 956123. We thank all three anonymous reviewers whose comments helped improve the paper.

This document is an overview of U.K. MOD (part) sponsored research and is released for informational purposes only. The contents of this document should not be interpreted as representing the views of the U.K. MOD, nor should it be assumed that they reflect any current or future U.K. MOD policy. The information contained in this document cannot supersede any statutory or contractual requirements or liabilities and is offered without prejudice or commitment.

REFERENCES

- [1] Haldun Akoglu. 2018. User’s guide to correlation coefficients. *Turkish journal of emergency medicine* 18, 3 (2018), 91–93.
- [2] Moustafa Alzantot, Yash Sharma, Supriyo Chakraborty, Huan Zhang, Cho-Jui Hsieh, and Mani B Srivastava. 2019. Genattack: Practical black-box attacks with gradient-free optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1111–1119.
- [3] Mohammed Attaoui, Hazem Fahmy, Fabrizio Pastore, and Lionel Briand. 2023. Black-Box Safety Analysis and Retraining of DNNs Based on Feature Extraction and Clustering. *ACM Trans. Softw. Eng. Methodol.* 32, 3, Article 79 (2023), 40 pages.
- [4] David Berend. 2021. Distribution Awareness for AI System Testing. In *43rd IEEE/ACM Int. Conf. on Software Engineering: Companion Proceedings, ICSE Companion 2021, Madrid, Spain, May 25-28, 2021*. IEEE, 96–98.
- [5] 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 *Proc. of the 35th IEEE/ACM Int. Conference on Automated Software Engineering (ASE’20)*. ACM, New York, NY, USA, 1041–1052. <https://doi.org/10.1145/3324884.3416609>
- [6] Taejoon Byun and Sanjai Rayadurgam. 2020. Manifold-based Test Generation for Image Classifiers. In *ICSE ’20: 42nd Int. Conference on Software Engineering, Workshops*. ACM, 221. <https://doi.org/10.1145/3387940.3391460>
- [7] Taejoon Byun, Abhishek Vijayakumar, Sanjai Rayadurgam, and Darren Cofer. 2020. Manifold-based Test Generation for Image Classifiers. In *Int. Conf. On Artificial Intelligence Testing (AITest)*. IEEE, Oxford, UK, 15–22.
- [8] Francesco Croce and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proc. of the 37th Int. Conf. on Machine Learning (ICML’20)*, Vol. 119. PMLR, 2206–2216.
- [9] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.
- [10] Swaroopa Dola, Matthew B. Dwyer, and Mary Lou Soffa. 2021. Distribution-Aware Testing of Neural Networks Using Generative Models. In *IEEE/ACM 43rd Int. Conference on Software Engineering (ICSE’21)*. IEEE, Madrid, Spain, 226–237.
- [11] Yi Dong, Wei Huang, Vibhav Bharti, Victoria Cox, Alec Banks, Sen Wang, Xingyu Zhao, Sven Schewe, and Xiaowei Huang. 2023. Reliability Assessment and Safety Arguments for Machine Learning Components in System Assurance. *ACM Trans. Embed. Comput. Syst.* 22, 3 (2023).
- [12] Xiaoning Du, Xiaofei Xie, Yi Li, Lei Ma, Yang Liu, and Jianjun Zhao. 2019. Deepstellar: Model-based quantitative analysis of stateful deep learning systems. In *Proc. of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 477–487.
- [13] Isaac Dunn, Laura Hanu, Hadrien Pouget, Daniel Kroening, and Tom Melham. 2020. Evaluating robustness to context-sensitive feature perturbations of different granularities. *arXiv preprint arXiv:2001.11055* (2020).
- [14] Isaac Dunn, Hadrien Pouget, Daniel Kroening, and Tom Melham. 2021. Exposing Previously Undetectable Faults in Deep Neural Networks. In *ACM SIGSOFT Int. Symposium on Software Testing and Analysis (ISSTA’21)*. in press.
- [15] Rafael C. Gonzalez and Richard E. Woods. 1992. *Digital image processing*. 793 pages.
- [16] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *3rd Int. Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, Conference Track Proceedings*.
- [17] 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 *Proc. of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM, 851–862.
- [18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. 6626–6637.
- [19] Hossein Hosseini and Radha Poovendran. 2018. Semantic Adversarial Examples. In *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018*. 1614–1619.
- [20] Wei Huang, Youcheng Sun, Xingyu Zhao, James Sharp, Wenjie Ruan, Jie Meng, and Xiaowei Huang. 2022. Coverage-Guided Testing for Recurrent Neural Networks. *IEEE Transactions on Reliability* 71, 3 (2022), 1191–1206.

- [21] Wei Huang, Xingyu Zhao, Gaojie Jin, and Xiaowei Huang. 2023. SAFARI: Versatile and Efficient Evaluations for Robustness of Interpretability. In *IEEE/CVF Int. Conf. on Computer Vision (ICCV'23)*.
- [22] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, and et al. 2020. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review* 37 (2020), 100270.
- [23] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety verification of deep neural networks. In *Computer Aided Verification (LNCS, Vol. 10426)*. Springer International Publishing, Cham, 3–29.
- [24] Ahmadreza Jeddi, Mohammad Javad Shafiee, and Alexander Wong. 2021. A simple fine-tuning is all you need: Towards robust deep learning via adversarial fine-tuning. In *Workshop on Adversarial Machine Learning in Real-World Computer Vision Systems and Online Challenges (AML-CV) @ CVPR'21*. 1–5.
- [25] Sungmin Kang, Robert Feldt, and Shin Yoo. 2020. Sinvad: Search-based image space navigation for dnn image classifier test input generation. In *Proc. of the IEEE/ACM 42nd Int. Conf. on Software Engineering Workshops*. 521–528.
- [26] Abdullah Konak, David W Coit, and Alice E Smith. 2006. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability engineering & system safety* 91, 9 (2006), 992–1007.
- [27] David Lane, David Bisset, Rob Buckingham, Geoff Pegman, and Tony Prescott. 2016. *New foresight review on robotics and autonomous systems*. Technical Report No. 2016.1. LRF. 65 pages.
- [28] Adam Lipowski and Dorota Lipowska. 2012. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications* 391, 6 (2012), 2193–2196.
- [29] Han Liu, John Lafferty, and Larry Wasserman. 2007. Sparse nonparametric density estimation in high dimensions using the rodeo. In *Artificial Intelligence and Statistics*. PMLR, 283–290.
- [30] Yang Liu, Eunice Jun, Qisheng Li, and Jeffrey Heer. 2019. Latent space cartography: Visual analysis of vector space embeddings. In *Computer graphics forum*, Vol. 38. Wiley Online Library, 67–78.
- [31] S. H. Lokerse, L. P. J. Veelenturf, and J. G. Beltman. 1995. Density Estimation Using SOFM and Adaptive Kernels. In *Neural Networks: Artificial Intelligence and Industrial Applications - Proceedings of the Third Annual SNN Symposium on Neural Networks, Nijmegen, The Netherlands, September 14-15, 1995*. Springer, 203–206.
- [32] 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 *Proc. of the 33rd ACM/IEEE Int. Conference on Automated Software Engineering (ASE'18)*. 120–131.
- [33] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th Int. Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.
- [34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
- [35] 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*. ACM, 1–18.
- [36] Vincenzo Riccio, Nargiz Humbatova, Gunel Jahangirova, and Paolo Tonella. 2021. Deepmetis: Augmenting a deep learning test set to increase its mutation score. In *36th IEEE/ACM Int. Conf. on Automated Software Engineering*. 355–367.
- [37] 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.
- [38] Andrew Rosenberg and Julia Hirschberg. 2007. V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure. In *EMNLP-CoNLL 2007, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, June 28-30, 2007, Prague, Czech Republic*. ACL, 410–420.
- [39] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. 2018. Reachability Analysis of Deep Neural Networks with Provable Guarantees. In *Proceedings of the Twenty-Seventh Int. Joint Conference on Artificial Intelligence, IJCAI-18*. Int. Joint Conferences on Artificial Intelligence Organization, 2651–2659.
- [40] David W Scott. 1991. Feasibility of multivariate density estimates. *Biometrika* 78, 1 (1991), 197–205.
- [41] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. 2019. DeepConcolic: testing and debugging deep neural networks. In *Proceedings of the 41st Int. Conference on Software Engineering: Companion Proceedings, ICSE 2019, Montreal, QC, Canada, May 25-31, 2019*. IEEE / ACM, 111–114.
- [42] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. 2018. Concolic testing for deep neural networks. In *Proc. of the 33rd ACM/IEEE Int. Conf. on Automated Software Engineering*. 109–119.
- [43] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proc. of the IEEE conference on computer vision and pattern recognition*. 2818–2826.
- [44] Felipe Toledo, David Shriver, Sebastian Elbaum, and Matthew B Dwyer. 2021. Distribution Models for Falsification and Verification of DNNs. In *IEEE/ACM Int. Conf. on Automated Software Engineering (ASE'21)*.

- [45] Benjie Wang, Stefan Webb, and Tom Rainforth. 2021. Statistically robust neural network classification. In *Uncertainty in Artificial Intelligence*. PMLR, 1735–1745.
- [46] 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 *IEEE/ACM 43rd International Conference on Software Engineering*. 300–311.
- [47] 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 *IEEE/ACM 43rd Int. Conf. on Softw. Engineering*. 300–311.
- [48] Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. 2019. On the Convergence and Robustness of Adversarial Training. In *Proc. of the 36th Int. Conf. on Machine Learning*, Vol. 97. PMLR, 6586–6595.
- [49] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. Image Process.* 13, 4 (2004), 600–612.
- [50] Stefan Webb, Tom Rainforth, Yee Whye Teh, and M. Pawan Kumar. 2019. A statistical approach to assessing neural network robustness. In *ICLR’19*. New Orleans, LA, USA.
- [51] Michael Weiss and Paolo Tonella. 2022. Simple techniques work surprisingly well for neural network test prioritization and active learning (replicability study). In *Proc. of the 31st ACM SIGSOFT Int. Symp. on Software Testing and Analysis*. 139–150.
- [52] Lily Weng, Pin-Yu Chen, Lam Nguyen, Mark Squillante, Akhilan Boopathy, Ivan Oseledets, and Luca Daniel. 2019. PROVEN: Verifying Robustness of Neural Networks with a Probabilistic Approach. In *International Conference on Machine Learning (ICML’19)*, Vol. 97. PMLR, 6727–6736.
- [53] Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. 2018. Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach. In *6th Int. Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.
- [54] Chenwang Wu, Wenjian Luo, Nan Zhou, Peilan Xu, and Tao Zhu. 2021. Genetic Algorithm with Multiple Fitness Functions for Generating Adversarial Examples. In *IEEE Congress on Evolutionary Computation, CEC 2021, Kraków, Poland, June 28 - July 1, 2021*. IEEE, 1792–1799.
- [55] Xiaofei Xie, Tianlin Li, Jian Wang, Lei Ma, Qing Guo, Felix Juefei-Xu, and Yang Liu. 2022. NPC: Neuron Path Coverage via Characterizing Decision Logic of Deep Neural Networks. *ACM Trans. Softw. Eng. Methodol.* 31, 3 (2022).
- [56] Shenao Yan, Guan hong Tao, Xuwei Liu, Juan Zhai, Shiqing Ma, Lei Xu, and Xiangyu Zhang. 2020. Correlations between Deep Neural Network Model Coverage Criteria and Model Quality. In *The 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2020)*. ACM, 775–787.
- [57] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. 2020. A Closer Look at Accuracy vs. Robustness. In *Advances in Neural Information Processing Systems (NeurIPS’20, Vol. 33)*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.). Curran Associates, Inc., 8588–8601.
- [58] Bing Yu, Hua Qi, Qing Guo, Felix Juefei-Xu, Xiaofei Xie, Lei Ma, and Jianjun Zhao. 2022. DeepRepair: Style-Guided Repairing for Deep Neural Networks in the Real-World Operational Environment. *IEEE Transactions on Reliability* 71, 4 (2022), 1401–1416.
- [59] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. Theoretically Principled Trade-off between Robustness and Accuracy. In *Proceedings of the 36th International Conference on Machine Learning*, Vol. 97. PMLR, 7472–7482.
- [60] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2022. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering* 48, 1 (2022), 1–36.
- [61] Xingyu Zhao, Alec Banks, James Sharp, Valentin Robu, David Flynn, Michael Fisher, and Xiaowei Huang. 2020. A Safety Framework for Critical Systems Utilising Deep Neural Networks. In *Computer Safety, Reliability, and Security (LNCS, Vol. 12234)*. Springer Int. Publishing, Cham, 244–259.
- [62] Xingyu Zhao, Wei Huang, Alec Banks, Victoria Cox, David Flynn, Sven Schewe, and Xiaowei Huang. 2021. Assessing the Reliability of Deep Learning Classifiers Through Robustness Evaluation and Operational Profiles. In *AI Safety’21 Workshop at IJCAI’21*, Vol. 2916.
- [63] Xingyu Zhao, Wei Huang, Sven Schewe, Yi Dong, and Xiaowei Huang. 2021. Detecting Operational Adversarial Examples for Reliable Deep Learning. In *51th Annual IEEE-IFIP Int. Conf. on Dependable Systems and Networks (DSN’21)*, Vol. Fast Abstract.
- [64] Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating Natural Adversarial Examples. In *6th Int. Conference on Learning Representations, ICLR 2018, Conference Track Proceedings*. OpenReview.net.
- [65] Yue Zhong, Lizhuang Liu, Dan Zhao, and Hongyang Li. 2020. A generative adversarial network for image denoising. *Multimedia Tools and Applications* 79, 23 (2020), 16517–16529.
- [66] Qile Zhu, Wei Bi, Xiaojiang Liu, Xiyao Ma, Xiaolin Li, and Dapeng Wu. 2020. A Batch Normalized Inference Network Keeps the KL Vanishing Away. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2636–2649.