TWINFUZZ: Dual-Model Fuzzing for Robustness Generalization in Deep Learning

Enze Dai[1], Wentao Mo[1], Kun Hu[2], Xiaogang Zhu[3*], Xi Xiao[1*], Sheng Wen[4], Shaohua Wang[5], Yang Xiang[4]

[1]Shenzhen International Graduate School, Tsinghua University
[2]Edith Cowan University
[3]Adelaide University
[4]Swinburne University of Technology
[5]Central University of Finance and Economics
{dez23, mow10}@mails.tsinghua.edu.cn, k.hu@ecu.edu.au, xiaogang.zhu@adelaide.edu.au, xiaox@sz.tsinghua.edu.cn, {swen, yxiang}@swin.edu.au, davidshwang@ieee.org

Abstract

Deep learning (DL) models are increasingly deployed in safety-critical applications such as face recognition, autonomous driving, and medical diagnosis. Despite their impressive accuracy, they remain vulnerable to adversarial examples - subtle perturbations that can cause incorrect predictions, i.e., the robustness issues. While adversarial training improves robustness against known attacks, it often fails to generalize to unseen or stronger threats, revealing a critical gap in robustness generalization. In this work, we propose a dual-model fuzzing framework to enhance generalized robustness in DL models. Central to our method is a lightweight metric, the Lagrangian Information Bottleneck ($\mathcal{L}$IB), which guides entropy-based mutation toward semantically meaningful and high-risk regions of the input space. The executor uses a resistant model and a more error-prone vulnerable model; their prediction consistency forms the basis of *agreement mining*, a label-free oracle for isolating decision-boundary samples. To ensure fuzzing effectiveness, we further introduce a task-driven seed selection strategy (e.g., SSIM for vision) that filters out low-quality inputs. We implement a prototype, TWINFUZZ, and evaluate it on six benchmark datasets and nine DL models. Compared with state-of-the-art testing approaches, TWINFUZZ achieves superior improvements in both training-specific and generalized robustness.

Code — https://github.com/nuwaLab/TwinFuzz

Introduction

Deep learning (DL) underpins a wide range of safety-critical applications (Du et al. 2022; Weng et al. 2023; Chen et al. 2024; Yue et al. 2024; Zheng et al. 2023). Their ubiquity and integration into high-stakes scenarios raise serious concerns about reliability and security. Despite achieving remarkable performance on standard benchmarks, SOTA models remain surprisingly vulnerable to adversarial examples - subtle, human-imperceptible perturbations that can cause misclassification or misbehavior (Madry et al. 2018; Cao et al. 2023; Duan et al. 2021). These adversarial attacks not only expose systemic blind spots in model generalization but also undermine trustworthiness in real-world deployments, where robustness is a prerequisite rather than an option.

These adversarial vulnerabilities highlight a deeper, still-unresolved challenge: robustness generalization (Shafahi et al. 2019; Wang et al. 2024). Robustness can be viewed along two axes. (i) Training-specific robustness refers to a model's ability to resist adversarial attacks drawn from the same distribution used during adversarial training. (ii) Generalized robustness, by contrast, captures the model's resilience to unseen or stronger attacks that differ from those encountered during training. A widely adopted defense, adversarial training, augments training data with perturbed examples generated by fixed attack strategies such as Projected Gradient Descent PGD (Madry et al. 2018). While effective at improving training-specific robustness, it suffers from poor performance under alternative attacks. This is because optimization often converges to a narrow local minimum tailored to the specific threat model, leaving the model brittle against shifts in attack distribution (Singh, Croce, and Hein 2023).

By viewing such failures through the lens of software bugs, researchers have adapted two main approaches from software engineering. The first is formal verification, which aims to provide certifiable robustness guarantees for neural networks (Singh et al. 2019; Paulsen, Wang, and Wang 2020; Ugare et al. 2023; Xue et al. 2023). However, these methods suffer from high computational overheads, especially when applied to deep and highly nonlinear models, making them difficult to scale to real-world architectures. The second avenue is deep learning testing and fuzzing. These existing techniques help improve basic robustness by exploring different regions of the input space. Early work proposed structural metrics such as neuron coverage (Pei et al. 2017; Ma et al. 2018; Guo et al. 2018), while subsequent efforts incorporated input diversity (Odena et al. 2019), natural perturbations (Sensei (Gao et al. 2020)), and loss-based sensitivity cues (RobOT (Wang et al. 2021a)). However, despite these

*Co-corresponding Authors: Xi Xiao and Xiaogang Zhu

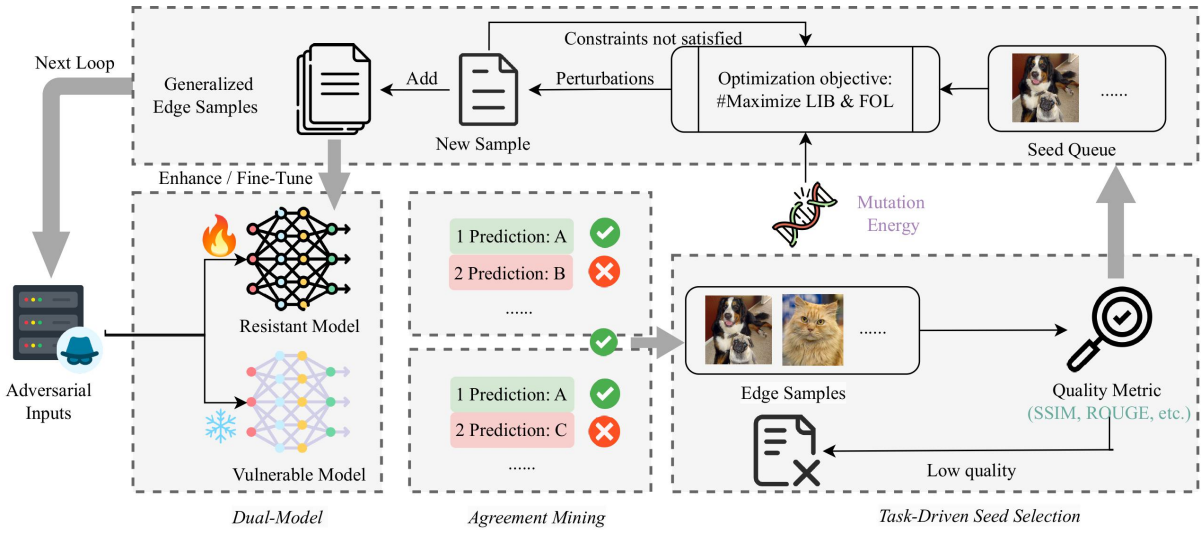

Figure 1: An overview of TwinFuzz, with a dual-model design for agreement mining and task-driven seed selection to guide the fuzzing process under the $\mathcal{L}IB$ metric.

advances, such methods often fall short in improving generalized robustness, which requires resilience to previously unseen or stronger adversarial attacks. No existing coverage or cost metric is explicitly designed to guide the improvement of generalized robustness.

To address the challenge of robustness generalization, we propose a dual-model fuzzing framework structured around three standard components—input generator, executor, and defect monitor—following the classical fuzzing paradigm (Zhu et al. 2022) (see Fig. 1). Since the decision boundary defines where distinctions are made, it reveals the regions with the highest potential for improving generalization. Central to our framework is an entropy-based metric, the Lagrangian Information Bottleneck ($\mathcal{L}$IB), which supports lightweight computation and guides the exploration of decision-boundary regions to enhance generalization. In parallel, we adopt first-order loss (FOL) to promote training-specific robustness. Specifically, (i) the input generator initializes from adversarial examples produced by off-the-shelf attacks, filters for high-quality seeds (i.e., adversarial samples where the two models agree without excessive perturbations), and applies gradient-based mutations guided by both FOL and $\mathcal{L}$IB to generate new inputs; (ii) the executor employs a dual-model setup consisting of a (adversarially trained) resistant model and a more error-prone vulnerable model. During each inference, the resistant model provides predictions and gradients for metric feedback, while the vulnerable model serves as an implicit oracle for identifying risky prediction agreements; (iii) the defect monitor filters uninformative or overly perturbed inputs through two stages: agreement mining, which retains inputs where both models agree—flagging them as edge cases likely near the decision boundary; and task-driven seed selection, which discards low-quality samples using domain-aware criteria (e.g., SSIM for vision tasks) to ensure imperceptibility and semantic validity. Overall, this architecture enables a lightweight yet targeted fuzzing workflow that not only improves robustness against known attacks but also enhances generalization to unseen threats via iterative retraining on high-risk, semantically meaningful failures.

In summary, our key contributions are as follows:

- We propose a dual-model fuzzing framework to enhance generalized robustness in DL models, guided by a lightweight metric, Lagrangian information bottleneck.
- We devise *agreement mining*, a label-free oracle mechanism that leverages prediction consistency between a resistant and a vulnerable model to isolate edge samples.
- We introduce a task-driven seed selection strategy that preserves quality and fidelity, improving fuzzing effectiveness under constrained computational budgets.
- We implement a prototype TwinFuzz, and evaluate it across diverse datasets and model architectures, demonstrating its effectiveness in boosting both training-specific and generalized robustness.

Related Work

Adversarial Training

Adversarial attacks can severely degrade both the accuracy and generalization of neural networks. To mitigate such vulnerabilities, a fundamental defense strategy is adversarial training (Madry et al. 2018). This technique involves augmenting the training data with adversarially perturbed samples to improve model robustness. Formally, adversarial training can be formulated as the following min-max optimization problem:

$$\min_{\theta} \mathbb{E}(x, y) \sim \mathcal{D} \left[\max |\delta| \leq \epsilon \mathcal{L}(f_\theta(x + \delta), y) \right], \quad (1)$$

where δ is an adversarial perturbation bounded by ϵ, and $\mathcal{L}$ denotes the loss function. The inner maximization seeks a perturbation δ that maximizes the model's prediction error,

simulating a worst-case attack within an ϵ -bounded neighborhood of the input. The outer minimization then updates the model parameters θ to minimize this worst-case loss, training the model to resist such adversarial perturbations and improve its robustness.

Deep Learning Testing

To enhance the robustness of deep neural network (DNN), researchers have proposed various test generation and prioritization methods. In test generation, works like DeepXplore (Pei et al. 2017) and DLFuzz (Guo et al. 2018) aim to increase neuron coverage to uncover incorrect behaviors of models. Subsequent works, such as ADAPT (Lee et al. 2020) and RobOT (Wang et al. 2021a), introduce refined metrics to improve testing effectiveness, though some studies question the value of neuron coverage (Harel-Canada et al. 2020). Recent methods like DistXplore (Wang et al. 2023) and BET (Wang et al. 2022) focus on out-of-distribution robustness and blackbox testing.

In test prioritization, techniques like LSA (Kim, Feldt, and Yoo 2019), DeepGini (Feng et al. 2020), ATS (Gao et al. 2022), and RTS (Sun et al. 2023) help reduce labeling effort by selecting high-value inputs. However, methods like DeepGini fail under adversarial or domain-shifted conditions (Wang et al. 2021b). Inspired by this, our approach prioritizes tests based on intrinsic quality metrics (e.g., SSIM) without relying on DNN-specific behaviors. DAT (Hu et al. 2022) shows that with 70% out-of-distribution data, even well-designed selection strategies perform worse than random. Inspired by this, we select test cases solely based on their intrinsic quality, ignoring DNN-specific properties.

Methodology

Overview of TwinFuzz Framework

To address the problem of robustness generalization, as illustrated in Fig. 1, our proposed framework TwinFuzz, follows a fuzzing paradigm, consisting of three major components - generator, executor, and defect monitor (Zhu et al. 2022). The aim of this framework is to generate edge samples that are perceptually similar to the original examples, enabling exploration of the high-risk regions for enhancing robustness generalization. Edge samples are inputs near the decision boundary where both resistant and vulnerable models agree. 1) *Input Generator.* Implemented as the entropy-guided mutator in TwinFuzz, it takes high-quality seeds and mutates them to produce candidate inputs through gradient-based optimization guided by our devised $\mathcal{L}$IB and FOL for generalized and training-specific robustness. This process requires only minimal computation per mutation. 2) *Executor.* It is realized as a dual-model scheme composed of a resistant model (with adversarial training) and a vulnerable baseline model. During each execution, the resistant model provides forward predictions and gradients for optimization, while the vulnerable model acts as a reference to detect risky prediction agreements. This scheme serves both metric feedback and oracle guidance. 3) *Defect Monitor.* This component operates in two stages: first, Agreement Mining identifies inputs on which both the resistant and the vulnerable models concur, treating these as high-risk edge samples; second, Task-Driven Seed Selection applies a task-specific threshold (for example, SSIM in computer vision) to ensure that perturbations remain imperceptible and to filter out low-quality adversarial examples.

$\mathcal{L}$-Information Bottleneck

Despite the importance of robustness generalization, there remains a lack of effective and lightweight metrics to guide the fuzzing process and model enhancement. To address this gap, we introduce an Information Bottleneck (IB) based objective that serves as the core of our mutator. Under the IB framework, a neural network is prone to poor generalization when it fails to compress task-relevant information - specifically, when the mutual information $I(X;Z)$ remains high (Tishby and Zaslavsky 2015). In our context, this insight motivates a maximized direction of the IB Lagrangian to mutate new samples and expose vulnerabilities regarding the model's generalization capability.

Definition 1 *Information Bottleneck Lagrangian. Given a neural network with parameters θ, an input variable X, an output variable Y and a latent representation Z. The IB Lagrangian is defined as follows:*

$$\mathcal{L}\text{IB}(\theta; X, Y) = \beta I(X;Z) - I(Y;Z), \tag{2}$$

where $I(X;Z)$ reflects the compression degree of Z to X, and $I(Y;Z)$ measures how well Z preserves the information necessary to predict Y. The parameter $\beta > 0$ controls the trade-off between compression and prediction fidelity. To ensure the fuzzing effectiveness and enable the computation of $\mathcal{L}$IB with unknown distributions, we treat a deep neural network as a Markov chain ($X \to Z \to \hat{Y}$) and apply the data processing inequality to estimate the lower-bound $I(X;Z) \geq I(X;\hat{Y})$.

Proposition 1 *Given a deterministic neural network with fixed behaviors, let X denote the input variable, Z the latent representation Z, and $\hat{Y}$ the prediction. The mutual information $I(X;Z)$ can be practically lower-bounded by:*

$$I(X;Z) \geq I(X;\hat{Y}) = H(\hat{Y}) - H(\hat{Y}|X) = H(\hat{Y}). \tag{3}$$

Note that $H(\hat{Y}|X) = 0$ due to the fixed network behaviors. Based on prior work (Achille and Soatto 2018b,a), we approximate $I(Y;Z)$ as $\mathcal{L}_{CE}(\theta; X, Y)$. This yields a fully calculable lower bound of $\mathcal{L}$IB:

$$\mathcal{L}\text{IB}(\theta; X, Y) \geq \beta H(\hat{Y}) + \mathcal{L}_{CE}(\theta; X, Y). \tag{4}$$

In addition, we adopt a neuron coverage metric for training-specific robustness - *first-order loss (FOL)* (Wang et al. 2021a):

$$\text{FOL}(X) = \epsilon \left\| \nabla_X f_\theta(X^t) \right\|_2. \tag{5}$$

To this end, we have an overall optimization objective as:

$$\max \lambda \cdot FOL(X) + \mathcal{L}_{\text{IB}}(\theta; X, Y), \tag{6}$$

where λ is a parameter on how much we favor FOL.

Agreement Mining

The core idea behind *agreement mining* is to identify informative adversarial samples by examining prediction consistency between two models. Specifically, for each adversarial input, we compare the outputs of a *resistant* model (trained to withstand adversarial attacks) and a more error-prone *vulnerable* baseline model. Inputs where the two models *agree* (highlighted in green in Fig. 1) are retained as high-risk edge samples and seed candidates; those with *discrepant* predictions (red) are discarded.

This design rests on a key statistical intuition: adversarial examples typically cause misclassification in the vulnerable model. When the robust model makes the same prediction, this suggests that the input likely resides near the decision boundary - where the model's output is unstable and confidence is low. In this way, the vulnerable model's output serves as a *negative oracle*, and prediction agreement becomes a proxy for locating hard-to-classify, high-risk inputs. Crucially, this selection process requires no ground-truth labels, making it highly suitable for continuous or large-scale testing scenarios.

Within the fuzzing framework, agreement mining acts as an early-stage filter that significantly narrows the search space, improving testing effectiveness. By focusing only on boundary-adjacent inputs, it prioritizes the most informative failure cases and provides strong candidates for mutation in the next stage. Furthermore, because the mechanism relies only on model agreement, it is agnostic to network architecture, output space, and task type, thereby supporting broad generalization across models and domains.

Task-Driven Seed Selection

Under constrained testing budgets - such as limited time or computational resources - prioritizing high-quality seeds becomes essential for efficient fuzzing. In practice, many adversarial samples, from which seeds are derived, introduce excessive perturbations that result in noticeable perceptual artifacts, reducing their practical utility and undermining attack imperceptibility.

To ensure the quality of generated inputs, we apply a filtering step that discards overly perturbed samples. In the context of computer vision tasks, we use the Structural Similarity Index Measure (SSIM) as a criterion to assess the perceptual similarity between an adversarial image X' and its original counterpart X. SSIM evaluates structural, luminance, and contrast similarity, offering a more human-aligned measure of visual fidelity than simple pixel-wise differences. By enforcing a threshold on SSIM, we retain only adversarial examples that are visually similar to the originals, thus balancing effectiveness and imperceptibility. In detail, we have:

$$SSIM(X, X') = l(X, X')^{\gamma_l} c(X, X')^{\gamma_c} s(X, X')^{\gamma_s}, \quad (7)$$

where $l(X, X')$, $c(X, X')$, $s(X, X')$ denote luminance, contrast, and structural similarity components, respectively, γ_l, γ_c, and γ_s control their relative weights. Samples that pass a specified SSIM threshold are retained as high-quality seeds for subsequent mutation.

While our evaluation is conducted on visual data, the seed selection module is inherently modality-agnostic. By replacing SSIM with appropriate quality metrics tailored to other domains (e.g., BLEU for text or PESQ for audio), the same mechanism can be extended beyond images, enabling broad applicability across diverse tasks.

Seed Mutation and Model Refinement

Building on agreement mining and task-driven seed selection, the retained edge samples are enqueued as *seeds* for fuzzing. These seeds are not only adversarially informative - lying close to the decision boundary where model predictions are most unstable - but also perceptually similar to the original examples, ensuring semantic validity. By focusing mutation on such samples, TWINFUZZ explores high-risk regions of the input space, generating diverse and meaningful test cases that can expose model vulnerabilities and improve robustness through iterative retraining.

Fuzzing begins with adversarial examples produced by standard attack methods. For each fuzzing iteration, a seed is dequeued and passed through the *executor* - the resistant model under test - which returns two key metrics: FOL (First-Order Local sensitivity) and $\mathcal{L}$IB. Unlike prior neuron-coverage–based approaches (Pei et al. 2017; Guo et al. 2018), our mutation strategy avoids costly profiling steps. Each optimization step requires only a single backward pass to compute the gradient of FOL, and two log-based operations to compute $\mathcal{L}$IB, leading to significantly lower computational overhead.

Each seed is assigned a mutation budget, namely energy, indicating how many mutations it can undergo. In each iteration, we compute a composite objective that jointly maximizes both FOL and $\mathcal{L}$IB in Eq. (6), resulting in a directional gradient $\nabla_{\mathcal{L}}$ that points toward regions of high sensitivity and semantic ambiguity. New test samples are generated by perturbing the seed along this direction, scaled with a randomized step size to promote diversity. If the seed's energy is depleted, a new seed is drawn from the queue. When the queue becomes empty, TWINFUZZ will cycle through all the seeds again. When the fuzzing phase ends, all accumulated generalized edge samples are used to fine-tune the resistant model with the adversarial learning.

Experiments

Experiment Setup

Datasets and Models. We evaluate TWINFUZZ on six representative datasets, including MNIST (LeCun et al. 1998), Fashion-MNIST (Xiao, Rasul, and Vollgraf 2017), CIFAR-10 (Krizhevsky, Hinton et al. 2009), SVHN (Netzer et al. 2011), PACS (Li et al. 2017) and ImageNet (Deng et al. 2009); and nine widely used DNN models in real-world scenarios: LeNet-4 (LeCun et al. 1998), LeNet-5 (LeCun et al. 1998), ResNet-20, ResNet-50 (He et al. 2016a), ResNet-50-v2 (He et al. 2016b), VGG16 (Simonyan and Zisserman 2014), Inception-v3 (Szegedy et al. 2016), ViT-Tiny, and ViT-Base (Dosovitskiy et al. 2020). Table 1 summarizes the relevant details, including the dataset, model, clean test accuracy before applying adversarial training or DL testing

Dataset[1]	Model	Test Acc.	Retrain[2]
MNIST	LeNet-4	99.04%	98.97%
	LeNet-5	99.08%	99.02%
Fashion-MNIST	LeNet-4	90.48%	90.33%
	LeNet-5	90.67%	90.51%
CIFAR-10	VGG16	76.98%	75.77%
	Inception-v3	77.86%	77.14%
	ResNet-20	75.99%	75.25%
SVHN	VGG16	93.48%	92.85%
	ResNet-20	92.06%	91.76%
PACS	ResNet-50	74.46%	74.41%
	ResNet-50-v2	71.00%	70.49%
ImageNet-1K	ViT-Tiny	72.93%	71.32%
	ViT-Base	77.64%	76.84%

[1] For later reference, M: MNIST; F: Fashion-MNIST; C: CIFAR-10; S: SVHN; I: ImageNet-1K; L4: LeNet-4; L5: LeNet-5; VG: VGG16; In: Inception-v3; R20: ResNet-20; ViTT: ViT-Tiny; ViTB: ViT-Base. [2] Retrain: Clean accuracy of retrained models using TWINFUZZ.

Table 1: Datasets and corresponding models.

(*Test Acc.*), and clean accuracy of retrained models using TWINFUZZ (*Retrain*). Specifically, PACS is chosen for its domain diversity - spanning Photos, Cartoons, Sketches, and Art - making it ideal for evaluating domain generalization. We use Photos, Cartoons, and Sketches as training domains and reserve the Art domain for testing.

Pre-adversarial Training. We conduct pre-adversarial training for all models involved in our experiments. For the MNIST, Fashion-MNIST, CIFAR-10, SVHN, and ImageNet datasets, we apply adversarial perturbations using both PGD and the Fast Gradient Sign Method (FGSM) to 20% of the training data. These adversarial examples are then combined with the clean samples to train the resistant models. For the PACS dataset, which contains high-resolution images and comparatively fewer samples, we apply adversarial perturbations to only 10% of the training data. This conservative ratio helps mitigate the risk of the resistant model overfitting to adversarial patterns, ensuring better generalization.

Evaluation Metric. Following the common practice in prior work (Gao et al. 2020; Wang et al. 2021a), we adopt robust accuracy as the primary evaluation metric. Robust accuracy is defined as the proportion of adversarial test samples in a perturbed validation set that are *not misclassified* by the model, i.e., samples for which the model retains correct predictions despite adversarial perturbations. We refer to these as robust test cases. Let $N_{\text{validation}}$ denotes the total number of samples in the validation set and N_{robust} the number of robust test cases among them. The robust accuracy:

$$\text{robust accuracy} = \frac{N_{\text{robust}}}{N_{\text{validation}}}. \tag{8}$$

We carefully curate the validation sets to ensure fair and meaningful comparisons across all methods. Details on the dataset construction and selection criteria are discussed in the following sections.

Model[1]	DeepXplore			DLFuzz		
	Deepfool	BIM	MIM	Deepfool	BIM	MIM
M-L4	0.094	0.390	0.262	0.099	0.839	0.726
M-L5	0.142	0.603	0.513	0.123	0.5204	0.407
F-L4	0.285	0.510	0.513	0.334	0.678	0.638
F-L5	0.323	0.536	0.500	0.300	0.245	0.247
C-VG	0.263	0.047	0.063	0.240	0.079	0.104
C-In	0.383	0.120	0.055	0.317	0.057	0.076
C-R20	0.299	0.043	0.087	0.3083	0.127	0.140
S-VG	0.225	0.015	0.045	0.303	0.014	0.034
S-R20	0.379	0.209	0.172	0.415	0.244	0.216
I-ViTT	0.143	0.095	0.189	0.184	0.160	0.200
I-ViTB	0.154	0.103	0.216	0.212	0.075	0.245
AVG	0.245	0.243	0.238	0.258	0.276	0.276
Model	**ADAPT**			**RobOT**		
	Deepfool	**BIM**	**MIM**	**Deepfool**	**BIM**	**MIM**
M-L4	0.284	0.899	0.775	0.564	0.911	0.867
M-L5	0.260	0.801	0.639	0.556	0.872	0.891
F-L4	0.435	0.716	0.728	0.587	0.791	0.751
F-L5	0.414	0.605	0.591	0.573	0.731	0.699
C-VG	0.307	0.208	0.189	0.332	0.257	0.269
C-In	0.415	0.197	0.176	0.420	0.327	0.369
C-R20	0.365	0.190	0.221	0.385	0.310	0.299
S-VG	0.319	0.260	0.195	0.420	0.472	0.419
S-R20	0.498	0.325	0.297	0.593	0.488	0.448
I-ViTT	0.275	0.214	0.334	0.365	0.478	0.453
I-ViTB	0.258	0.147	0.255	0.359	0.395	0.416
AVG	0.348	0.415	0.399	0.469	0.548	0.535
Model	**TWINFUZZ**			**Pre-adv Train**		
	Deepfool	**BIM**	**MIM**	**Deepfool**	**BIM**	**MIM**
M-L4	0.816	0.918	0.876	0.125	0.901	0.841
M-L5	0.777	0.899	0.904	0.117	0.854	0.899
F-L4	0.747	0.827	0.861	0.398	0.775	0.714
F-L5	0.712	0.811	0.831	0.383	0.684	0.619
C-VG	0.511	0.457	0.491	0.268	0.146	0.166
C-In	0.536	0.473	0.508	0.353	0.217	0.234
C-R20	0.524	0.428	0.465	0.260	0.229	0.203
S-VG	0.795	0.710	0.655	0.271	0.327	0.266
S-R20	0.874	0.740	0.767	0.473	0.381	0.342
I-ViTT	0.483	0.524	0.512	0.264	0.403	0.388
I-ViTB	0.466	0.502	0.498	0.239	0.376	0.332
AVG	**0.658**	**0.663**	**0.670**	0.286	0.481	0.455

[1] Model: dataset-model pairs refer to Table 1.

Table 2: Training-specific robustness performance under three attack sources.

Overall Performance for Robustness Improvement

We evaluate the effectiveness of robustness improvements from two perspectives: (i) training-specific robustness, where the validation attack matches the attack source used during DL testing, and (ii) generalized robustness, where the validation attack differs from the testing source to assess cross-attack resilience.

We conduct robustness improvement experiments across eleven selected model–dataset pairs. Test cases are gener-

Model[1]	RobOT			TwinFuzz		
	BIM-D[2]	D-BIM	D-MIM	BIM-D[3]	D-BIM	D-MIM
M-L4	0.338	0.867	0.851	0.671	0.909	0.865
M-L5	0.305	0.795	0.865	0.625	0.870	0.898
F-L4	0.468	0.775	0.721	0.655	0.805	0.799
F-L5	0.409	0.520	0.430	0.626	0.742	0.703
C-VG	0.297	0.198	0.216	0.356	0.358	0.376
C-In	0.387	0.282	0.314	0.464	0.388	0.455
C-R20	0.316	0.279	0.268	0.476	0.401	0.396
S-VG	0.355	0.390	0.341	0.639	0.659	0.594
S-R20	0.532	0.423	0.401	0.718	0.672	0.627
I-ViTT	0.293	0.405	0.397	0.345	0.458	0.443
I-ViTB	0.277	0.315	0.364	0.315	0.408	0.424
AVG	0.362	0.477	0.470	**0.536**	**0.607**	**0.598**

[1] Model: dataset-model pairs refer to Table 1 [2] Format X-Y: Use X for testing and Y for validation. [3] D: Deepfool.

Table 3: Generalized robustness under across attack sources.

ated using our proposed TwinFuzz, as well as four baseline DL testing methods: RobOT (a robustness-oriented approach), DeepXplore, DLFuzz, and ADAPT (all based on neuron coverage). For each method, we generate test cases equivalent to 10% of the training set size, which are then used to fine-tune the pre-adversarially trained models. To ensure fairness, all methods are initialized with the same adversarial input set.

Same-Attack Validation. To evaluate training-specific robustness, we consider three representative adversarial attacks: DeepFool (Moosavi-Dezfooli, Fawzi, and Frossard 2016), BIM (Kurakin, Goodfellow, and Bengio 2018), and MIM (Dong et al. 2018). These attacks were selected to ensure diversity across perturbation strategies while avoiding redundancy. Specifically, DeepFool targets minimal perturbations for decision boundary crossing, BIM applies iterative gradient-based updates for stronger perturbations, and MIM incorporates momentum to stabilize attack directions. Although alternative attacks such as CW (Carlini and Wagner 2017), UAP (Moosavi-Dezfooli et al. 2017), and L-BFGS (Szegedy et al. 2013) are also popular, their design philosophies largely overlap with the selected attacks, particularly in their reliance on gradient optimization or minimality objectives. Our choice therefore ensures broad coverage of attack types without unnecessary duplication.

Robust accuracy, defined as the proportion of adversarial inputs correctly classified, is used to assess model resilience. As shown in Table 2, our framework TwinFuzz consistently outperforms the baselines. Specifically, TwinFuzz achieves average robust accuracies of 65.8%, 66.3%, and 67.0% across DeepFool, BIM, and MIM, respectively. This marks improvements of 37.2%, 18.2%, and 21.5% over the pre-adversarially trained models, and gains of 18.9%, 11.5%, and 13.5% over RobOT. In contrast, we observe a decline in robustness when using neuron coverage–based testing. Even the best-performing ADAPT reduces robust accuracy by 6.6% and 5.6% under BIM and MIM, compared to pre-adversarially trained baselines. These results indicate that neuron coverage is not a reliable proxy for robustness enhancement and may even introduce harmful gradients.

Furthermore, we find that the effectiveness of adversarial training varies by dataset. For MNIST and Fashion-MNIST, the pre-trained models show reasonable robustness to BIM and MIM. However, on more complex datasets like CIFAR-10 and SVHN, robustness against the same attacks is significantly weaker. The larger input dimensionality and more entangled decision boundaries give each attack many more viable perturbation directions. As a result, the training process fits the specific perturbations seen during training but fails to cover the much broader adversarial surface that exists in these complex domains. In other words, the *overfitting* here is not merely attack-specific but also *data-manifold specific*: the harder the manifold, the more brittle the defence. This clarifies why training-specific robustness does not automatically translate into reliable protection - even when the evaluation uses the same attack family.

Cross-Attack Validation. We conduct cross-source evaluation experiments, in which the attacks used for generating training seeds differ from those used for validation. This setup tests whether the model's robustness extends beyond the perturbation patterns observed during retraining. For instance, in the BIM–Deepfool configuration, we begin by generating adversarial examples using BIM, which serve as the initial seeds. These are then passed to DL testing methods, including TwinFuzz and RobOT, to produce additional test cases. The newly generated samples are subsequently used to fine-tune a pre-adversarially trained model. Finally, we evaluate the robust accuracy of the enhanced model against adversarial inputs generated by a different attack method - in this case, Deepfool.

As shown in Table 3, TwinFuzz achieves average robust accuracies of 53.6%, 60.7%, and 59.8% across three attack pairings, consistently outperforming RobOT by margins of 17.4%, 13.0%, and 12.8%, respectively. These results demonstrate the effectiveness of our TwinFuzz and the proposed $\mathcal{L}$IB metric in producing semantically meaningful and transferable test cases that strengthen model robustness under distributional shifts.

Ablation Study

Contribution of Agreement Mining. As shown in Table 4, removing agreement mining (TwinFuzz_NoAM) leads to a significant drop in robust accuracy compared to the full version of TwinFuzz. Specifically, average performance decreases by 18.9%, 11.3%, and 14.0% under Deepfool, BIM, and MIM attacks, respectively. This decline highlights the importance of agreement mining in isolating informative edge samples. These samples drive the model away from local optima and expose generalizable weaknesses. Without this filtering stage, adversarial inputs may be less informative or redundant, resulting in diminished robustness gains. Interestingly, the performance gap between TwinFuzz_NoAM and RobOT is smaller under BIM and MIM. This is because resistant models already exhibit stronger robustness against these attacks and RobOT's objective focuses solely on training-specific robustness. This shows that

Model[1]	TwinFuzz_NoSS			TwinFuzz_NoAM			TwinFuzz_No$\mathcal{L}$IB		
	Deepfool	BIM	MIM	Deepfool	BIM	MIM	Deepfool	BIM	MIM
M-L4	0.404	0.895	0.869	0.458	0.908	0.852	0.514	0.815	0.795
M-L5	0.421	0.872	0.895	0.447	0.870	0.902	0.495	0.821	0.884
F-L4	0.538	0.783	0.753	0.564	0.795	0.740	0.479	0.724	0.724
F-L5	0.554	0.725	0.679	0.589	0.715	0.664	0.514	0.699	0.634
C-VG	0.405	0.299	0.311	0.394	0.301	0.287	0.425	0.327	0.300
C-In	0.512	0.300	0.303	0.491	0.329	0.316	0.547	0.304	0.324
C-R20	0.447	0.289	0.270	0.431	0.296	0.290	0.415	0.312	0.269
S-VG	0.515	0.495	0.368	0.528	0.508	0.471	0.620	0.429	0.459
S-R20	0.666	0.538	0.490	0.653	0.489	0.515	0.675	0.425	0.468
I-ViTT	0.356	0.437	0.406	0.324	0.399	0.396	0.370	0.372	0.385
I-ViTB	0.308	0.459	0.415	0.285	0.440	0.398	0.357	0.425	0.440
Avg Drop	0.192↓	0.109↓	0.146↓	0.189↓	0.113↓	0.140↓	0.166↓	0.149↓	0.153↓

[1] Model: dataset-model pairs refer to Table 1.

Table 4: Ablation study for TwinFuzz.

Model	RobOT			TwinFuzz			Pre-adv Train		
	Deepfool	BIM	MIM	Deepfool	BIM	MIM	Deepfool	BIM	MIM
PACS-ResNet-50	0.323	0.207	0.163	**0.339**	**0.224**	**0.174**	0.314	0.220	0.137
PACS-ResNet-50-v2	0.438	0.093	0.148	**0.453**	**0.121**	**0.174**	0.433	0.109	0.149

Table 5: Generalized robustness performance on cross-domain datasets.

agreement mining not only improves overall performance but also plays a key role in targeting transferable, high-risk failure cases that enhance robustness generalization.

Contribution of Task-Driven Seed Selection. As shown in Table 4, removing task-driven seed selection from TwinFuzz (TwinFuzz_NoSS) leads to a drop in robust accuracy. The average performance declines by 19.2%, 10.9%, and 14.6% under different attack settings, respectively, compared to the full version of TwinFuzz. This substantial degradation highlights the importance of quality filtering in robustness improvement. Without seed selection, distorted or overly perturbed adversarial samples are retained, which can mislead the retraining process and introduce spurious noise. In contrast, task-driven seed selection filters out low-quality inputs using task-relevant perceptual similarity constraints (e.g., SSIM), preserving semantically valid samples that are more likely to generalize.

Effectiveness of $\mathcal{L}$IB for Robustness

We assess the impact of $\mathcal{L}$IB on robustness from three key aspects: (1) resistance to cross-source attacks, (2) ablation study, and (3) cross-domain dataset generalization. The first aspect has already been discussed in the previous section.

To evaluate cross-domain dataset generalization, we conduct experiments on the PACS dataset, using Art Painting as the held-out target domain and the remaining three domains (Photo, Cartoon, and Sketch) as the training set. Following the standard protocol, we perform adversarial training using only 10% of the available training data to avoid overfitting. Cross-domain adversarial robustness presents a particularly difficult challenge: as the model is trained on source domains intensively, its ability to generalize to unseen target domains tends to degrade. Table 5 presents the results on this setting. TwinFuzz consistently improves robust accuracy over the pre-adversarial trained baseline. Specifically, it achieves gains of 2.5%, 0.4%, and 3.7% on ResNet-50, and 2.0%, 1.2%, and 2.5% on ResNet-50-v2, across the three attack types. These results demonstrate that TwinFuzz remains effective even under significant domain shift, though the overall gains are understandably smaller compared to in-domain evaluations.

Furthermore, our ablation study highlights the critical contribution of the proposed $\mathcal{L}$IB metric. As shown in Table 4, removing $\mathcal{L}$IB from TwinFuzz results in a substantial drop in robust accuracy: 16.6%, 14.9%, and 15.3%, respectively. These results affirm that $\mathcal{L}$IB is not only effective but also indispensable for guiding robustness generalization.

Conclusion

This work proposes TwinFuzz, a dual-model fuzzing framework to improve robustness generalization in deep learning. Guided by a lightweight Information-Bottleneck metric $\mathcal{L}$IB, it mutates inputs toward uncertain decision boundaries. It uses prediction agreement between a resistant and a vulnerable model to identify risky edge samples and applies task-driven seed selection to maintain perceptual quality. These samples are then reused to fine-tune the model. Extensive experiments demonstrate that TwinFuzz achieves state-of-the-art performance. TwinFuzz is effective, label-free, and easily extendable to other settings.

References

Achille, A.; and Soatto, S. 2018a. Emergence of invariance and disentanglement in deep representations. *Journal of Machine Learning Research*, 19(50): 1–34.

Achille, A.; and Soatto, S. 2018b. Information dropout: Learning optimal representations through noisy computation. *IEEE transactions on pattern analysis and machine intelligence*, 40(12): 2897–2905.

Cao, Y.; Xiao, X.; Sun, R.; Wang, D.; Xue, M.; and Wen, S. 2023. Stylefool: Fooling video classification systems via style transfer. In *IEEE Symposium on Security and Privacy*, 1631–1648. IEEE.

Carlini, N.; and Wagner, D. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 39–57. Ieee.

Chen, L.; Wu, P.; Chitta, K.; Jaeger, B.; Geiger, A.; and Li, H. 2024. End-to-end autonomous driving: Challenges and frontiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. In *IEEE Conference on Computer Vision and Pattern Recognition*, 9185–9193.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Du, H.; Shi, H.; Zeng, D.; Zhang, X.-P.; and Mei, T. 2022. The elements of end-to-end deep face recognition: A survey of recent advances. *ACM Computing Surveys*, 54(10s): 1–42.

Duan, R.; Chen, Y.; Niu, D.; Yang, Y.; Qin, A. K.; and He, Y. 2021. Advdrop: Adversarial attack to dnns by dropping information. In *IEEE/CVF International Conference on Computer Vision*, 7506–7515.

Feng, Y.; Shi, Q.; Gao, X.; Wan, J.; Fang, C.; and Chen, Z. 2020. DeepGini: prioritizing massive tests to enhance the robustness of deep neural networks. In *ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2020, 177–188. ISBN 9781450380089.

Gao, X.; Feng, Y.; Yin, Y.; Liu, Z.; Chen, Z.; and Xu, B. 2022. Adaptive test selection for deep neural networks. In *International Conference on Software Engineering*, 73–85.

Gao, X.; Saha, R. K.; Prasad, M. R.; and Roychoudhury, A. 2020. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *ACM/IEEE International Conference on Software Engineering*, 1147–1158.

Guo, J.; Jiang, Y.; Zhao, Y.; Chen, Q.; and Sun, J. 2018. Dlfuzz: Differential fuzzing testing of deep learning systems. In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 739–743.

Harel-Canada, F.; Wang, L.; Gulzar, M. A.; Gu, Q.; and Kim, M. 2020. Is neuron coverage a meaningful measure for testing deep neural networks? In *ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 851–862.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 630–645. Springer.

Hu, Q.; Guo, Y.; Cordy, M.; Xie, X.; Ma, L.; Papadakis, M.; and Le Traon, Y. 2022. An empirical study on data distribution-aware test selection for deep learning enhancement. *ACM Transactions on Software Engineering and Methodology*, 31(4): 1–30.

Kim, J.; Feldt, R.; and Yoo, S. 2019. Guiding deep learning system testing using surprise adequacy. In *International Conference on Software Engineering*, 1039–1049. IEEE.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2018. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, 99–112. Chapman and Hall/CRC.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Lee, S.; Cha, S.; Lee, D.; and Oh, H. 2020. Effective white-box testing of deep neural networks with adaptive neuron-selection strategy. In *ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2020, 165–176. ISBN 978-1-4503-8008-9.

Li, D.; Yang, Y.; Song, Y.-Z.; and Hospedales, T. M. 2017. Deeper, broader and artier domain generalization. In *IEEE International Conference on Computer Vision*, 5542–5550.

Ma, L.; Juefei-Xu, F.; Zhang, F.; Sun, J.; Xue, M.; Li, B.; Chen, C.; Su, T.; Li, L.; Liu, Y.; et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *ACM/IEEE International Conference on Automated Software engineering*, 120–131.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; Fawzi, O.; and Frossard, P. 2017. Universal adversarial perturbations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1765–1773.

Moosavi-Dezfooli, S.-M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2574–2582.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; Ng, A. Y.; et al. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, volume 2011, 4. Granada.

Odena, A.; Olsson, C.; Andersen, D.; and Goodfellow, I. 2019. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In *International Conference on Machine Learning*, 4901–4911. PMLR.

Paulsen, B.; Wang, J.; and Wang, C. 2020. Reludiff: Differential verification of deep neural networks. In *ACM/IEEE International Conference on Software Engineering*, 714–726.

Pei, K.; Cao, Y.; Yang, J.; and Jana, S. 2017. Deepxplore: Automated whitebox testing of deep learning systems. In *Symposium on Operating Systems Principles*, 1–18.

Shafahi, A.; Najibi, M.; Ghiasi, M. A.; Xu, Z.; Dickerson, J.; Studer, C.; Davis, L. S.; Taylor, G.; and Goldstein, T. 2019. Adversarial training for free! *Advances in neural information processing systems*, 32.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, G.; Gehr, T.; Püschel, M.; and Vechev, M. 2019. An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*, 3(POPL): 1–30.

Singh, N. D.; Croce, F.; and Hein, M. 2023. Revisiting adversarial training for imagenet: Architectures, training and generalization across threat models. *Advances in Neural Information Processing Systems*, 36: 13931–13955.

Sun, W.; Yan, M.; Liu, Z.; and Lo, D. 2023. Robust Test Selection for Deep Neural Networks. *IEEE Transactions on Software Engineering*, 49(12): 5250–5278.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2818–2826.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tishby, N.; and Zaslavsky, N. 2015. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop*, 1–5. IEEE.

Ugare, S.; Banerjee, D.; Misailovic, S.; and Singh, G. 2023. Incremental verification of neural networks. *ACM on Programming Languages*, 7(PLDI): 1920–1945.

Wang, J.; Chen, J.; Sun, Y.; Ma, X.; Wang, D.; Sun, J.; and Cheng, P. 2021a. Robot: Robustness-oriented testing for deep learning systems. In *International Conference on Software Engineering*, 300–311. IEEE.

Wang, J.; Qiu, H.; Rong, Y.; Ye, H.; Li, Q.; Li, Z.; and Zhang, C. 2022. BET: black-box efficient testing for convolutional neural networks. In *ACM SIGSOFT International Symposium on Software Testing and Analysis*, 164–175.

Wang, L.; Xie, X.; Du, X.; Tian, M.; Guo, Q.; Yang, Z.; and Shen, C. 2023. DistXplore: Distribution-guided testing for evaluating and enhancing deep learning systems. In *ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 68–80.

Wang, Z.; Li, X.; Zhu, H.; and Xie, C. 2024. Revisiting adversarial training at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 24675–24685.

Wang, Z.; You, H.; Chen, J.; Zhang, Y.; Dong, X.; and Zhang, W. 2021b. Prioritizing test inputs for deep neural networks via mutation analysis. In *IEEE/ACM International Conference on Software Engineering*, 397–409. IEEE.

Weng, J.; Hu, K.; Yao, T.; Wang, J.; and Wang, Z. 2023. Federated Unsupervised Cluster-Contrastive learning for person Re-identification: A coarse-to-fine approach. *Computer Vision and Image Understanding*, 237: 103831.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xue, Z.; Liu, S.; Zhang, Z.; Wu, Y.; and Zhang, M. 2023. A tale of two approximations: Tightening over-approximation for DNN robustness verification via under-approximation. In *ACM SIGSOFT International Symposium on Software Testing and Analysis*, 1182–1194.

Yue, W.; Zhang, J.; Hu, K.; Xia, Y.; Luo, J.; and Wang, Z. 2024. SurgicalSAM: Efficient class promptable surgical instrument segmentation. In *AAAI Conference on Artificial Intelligence*, volume 38, 6890–6898.

Zheng, L.; Chiang, W.-L.; Sheng, Y.; Zhuang, S.; Wu, Z.; Zhuang, Y.; Lin, Z.; Li, Z.; Li, D.; Xing, E.; et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36: 46595–46623.

Zhu, X.; Wen, S.; Camtepe, S.; and Xiang, Y. 2022. Fuzzing: a survey for roadmap. *ACM Computing Surveys*, 54(11s): 1–36.