Testing Techniques in Deep Learning Systems: A Survey

XUEQI DANG, University of Luxembourg, Luxembourg
YINGHUA LI*, Nanjing University of Science and Technology, China
WENDKÛUNI C. OUÉDRAOGO, University of Luxembourg, Luxembourg
MIKE PAPADAKIS, University of Luxembourg, Luxembourg
JACQUES KLEIN, University of Luxembourg, Luxembourg
TEGAWENDÉ F. BISSYANDÉ, University of Luxembourg, Luxembourg
YVES LE TRAON, University of Luxembourg, Luxembourg

Deep learning (DL) systems have achieved remarkable success across domains such as computer vision, natural language processing, and autonomous driving. However, their deployment in safety-critical and security-sensitive scenarios necessitates thorough testing to ensure reliability and trustworthiness. Unlike traditional software, DL models lack well-defined logic and rely heavily on data, introducing unique challenges such as the oracle problem, high labeling cost, and bias detection. Although early survey has provided high-level overviews of DL testing, the field has evolved significantly over the past five years, with a surge of new techniques and a growing emphasis on practical concerns like test efficiency and fairness. This paper presents a comprehensive survey and analysis of 236 DL testing studies, covering six major research communities. We categorize the existing techniques based on six quality properties: efficiency, robustness & security, fairness, interpretability, correctness, and privacy. We provide in-depth analyses of the papers in each category using comparative analyses, distributional trends, and temporal evolution, obtaining 15 core findings. Furthermore, we highlight the emerging frontier of testing for DL, presenting recent trends, unique challenges, and promising future directions.

CCS Concepts: • **Software and its engineering** → **Software testing and debugging**; • **Computer systems organization** → *Neural networks.*

Additional Key Words and Phrases: Deep Learning Testing, Software Testing, Deep Neural Network

ACM Reference Format:
Xueqi Dang, Yinghua Li, Wendkûuni C. Ouédraogo, Mike Papadakis, Jacques Klein, Tegawendé F. Bissyandé, and Yves LE Traon. 2026. Testing Techniques in Deep Learning Systems: A Survey. *ACM Comput. Surv.* 1, 1 (September 2026), 67 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Deep learning (DL) has achieved remarkable breakthroughs across a wide range of domains, including computer vision [68, 170, 240], natural language processing [44, 61], healthcare [69, 70],

*Corresponding author.

Authors' addresses: Xueqi Dang, xueqi.dang@uni.lu, University of Luxembourg, Luxembourg; Yinghua Li, yinghua.li@njust.edu.cn, Nanjing University of Science and Technology, China; Wendkûuni C. Ouédraogo, wendkuuni.ouedraogo@uni.lu, University of Luxembourg, Luxembourg; Mike Papadakis, michail.papadakis@uni.lu, University of Luxembourg, Luxembourg; Jacques Klein, jacques.klein@uni.lu, University of Luxembourg, Luxembourg; Tegawendé F. Bissyandé, tegawende.bissyande@uni.lu, University of Luxembourg, Luxembourg; Yves LE Traon, yves.letraon@uni.lu, University of Luxembourg, Luxembourg.

0360-0300/2026/9-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

and autonomous systems [19, 94]. As DL models become increasingly integrated into safety-critical and security-sensitive applications, their quality and reliability are becoming central concerns for both the software engineering (SE) and machine learning (ML) communities. In particular, failures in DL systems may lead to severe real-world consequences, which highlights the urgent need for comprehensive and structured quality assurance practices.

However, testing DL systems poses fundamentally new challenges that distinguish them from traditional software. Conventional software is governed by explicit logic and deterministic control flows, enabling established testing techniques to operate via clear specifications, code coverage criteria, and test assertions [307]. In contrast, DL systems are data-driven and trained on massive datasets, leading to implicit decision boundaries that are difficult to interpret or verify. These characteristics introduce new challenges for testing. Among them, one of the most fundamental is the oracle problem [281]. In the context of DL testing, oracles are typically not automated and often rely on manual labeling to determine expected outcomes. As a consequence, the cost of labeling large-scale test data can be substantial [75], since manual annotation often demands significant human effort, time cost, and domain-specific expertise. Moreover, ensuring the fairness of DL models has emerged as another critical concern [42]. This is because DL models may inadvertently learn and amplify biases present in the training data, leading to discriminatory outcomes against certain demographic or social groups.

In recent years, the research community has proposed a wide range of testing techniques tailored to the characteristics of DL models (as shown in Figure 1). These techniques address diverse quality goals such as correctness, robustness, security, interpretability, fairness, and privacy. Despite this growing body of work, the field remains fragmented: these techniques are distributed across different research subfields, and most existing surveys [43, 198] tend to focus on specific subtopics, lacking a unified methodological foundation and systematic analysis. Moreover, these works are developed across multiple research communities (e.g., the software engineering and machine learning communities), but there is a lack of comparative analysis across communities, namely, an analysis that examines how different communities approach similar testing goals using different methodologies, underlying assumptions, and research emphases.

In 2020, Zhang *et al.* [281] conducted a comprehensive review of machine learning testing, summarizing key challenges and categorizing existing techniques along several important dimensions. This work laid an important foundation for the field. However, research on DNN testing began to take shape around 2017, and the body of work available in 2020 was still relatively limited compared to the much larger and more diverse body of research available today. As the landscape of deep learning testing has evolved rapidly in recent years, there is a pressing need for an updated and more comprehensive survey that captures recent advances, re-evaluates previous taxonomies, and synthesizes trends across a broader and richer collection of studies.

In particular, recent years have witnessed a surge of new research targeting several popular testing properties. For example, as shown in the study by Chen *et al.* [43], there was a significant increase in the number of papers on fairness testing from 2020 to 2023. In addition, prior to 2020, fault detection techniques aimed at improving efficiency were primarily based on coverage-based methods, which leveraged neuron coverage metrics [172, 191]. However, over the past five years, a variety of more advanced techniques have emerged, including confidence-based methods (which utilize the model's prediction uncertainty), surprise-based methods (which assess the novelty of inputs with respect to training data), learning-based approaches (which train auxiliary models to predict the misclassification probability of test inputs), and even more recent innovations such as CertPri [301] and ATS [81]. These developments motivate a fresh, in-depth review that not only updates the field but also enables researchers to better understand the state of the current landscape, identify gaps, and explore new research opportunities.

In this paper, we present a structured and comprehensive survey of DL testing techniques, with a particular emphasis on their underlying methodological principles. We adopt a taxonomy-driven approach to organize and analyze 236 representative papers drawn from seven major research communities. For each category of techniques, we identify mainstream research directions and trends, and analyze cross-community differences, particularly highlighting the differences between the software engineering (SE) and machine learning (ML) communities in terms of research focus, methodological assumptions, and testing objectives. Our study obtained 15 key findings that collectively reveal the methodological evolution of DL testing, the shifting research priorities across communities, and emerging challenges that remain underexplored.

We adopt a broad and unified definition of "DL testing": any activity designed to reveal defects, vulnerabilities, or undesirable behaviors in DL models. Based on our literature analysis, we classify testing techniques according to six major quality objectives (as shown in Figure 2): 1) efficiency, 2) fairness, 3) interpretability, 4) robustness and security, 5) correctness, and 6) privacy. Each objective is associated with different strategies such as test prioritization, coverage analysis, adversarial input generation, or fairness bug detection [42, 75, 172].

Notably, we performed a thorough manual checking process on the collected papers to ensure both their relevance and quality. As a result, 80.1% of the surveyed papers are published in top-tier venues (Core Rank A*), covering a broad range of research communities. These include SE venues such as TSE, TOSEM, ICSE, FSE, ASE, and ISSTA; machine learning venues such as NeurIPS, ICLR, ICML, and JMLR; computer vision venues including CVPR, ICCV, and ECCV; natural language processing venues such as ACL, EMNLP, NAACL, and TACL; security venues like IEEE S&P, USENIX Security, CCS, and NDSS; and other relevant venues in data mining (e.g., KDD, WWW, TKDE) and systems/networking (e.g., OSDI, SOSP, NSDI).

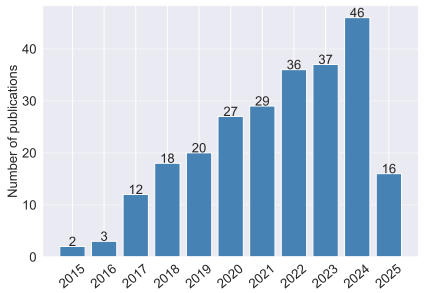


Fig. 1. Deep Learning Testing Techniques Publications

In summary, this paper makes the following contributions:

- **Landscape.** The paper presents a comprehensive and up-to-date overview of deep learning testing research to date, covering 236 papers published from 2015 to 2025 across six major research communities.
- **Survey.** The paper conducts a structured survey of DL testing techniques across various publishing venues, such as software engineering, machine learning, security and data mining.
- **Analysis.** The paper conducts in-depth analyses of the surveyed papers, including comparing approaches in the SE and ML communities (in terms of methodologies, research focus, and testing objectives), identifying trends in emerging research directions, and distinguishing mainstream approaches within each subfield. In total, the study obtains 15 key findings.
- **Horizons.** The paper discusses challenges and outlines research opportunities that can promote future advancements in the field of DL testing.

2 PRELIMINARIES

This section reviews the fundamental terminology in deep learning, aiming to establish a clear foundation for understanding the testing methodologies and analytical techniques discussed in later sections.

2.1 Deep Learning

Deep learning is a subfield of machine learning that employs neural networks with multiple layers to automatically learn hierarchical representations from data. A typical deep learning system is composed of the following components:

- **Dataset:** The performance of deep learning models heavily depends on the availability of large-scale and high-quality datasets. These datasets serve as the foundation for training, validating, and evaluating models, and are typically divided into the following categories:
  - *Training data:* Used to learn the parameters of the model. Through iterative optimization methods such as backpropagation and stochastic gradient descent, the model learns the mapping from input features to output targets.
  - *Validation data:* Used during training to fine-tune hyperparameters (e.g., learning rate, regularization factors) and to monitor model performance, helping to detect and prevent overfitting.
  - *Test data:* Utilized after training to assess the model's generalization capability on unseen data. A well-designed test set provides a realistic estimation of how the model will perform in practical deployment scenarios.

  Benchmark datasets such as ImageNet [58], COCO [164], MNIST [147], and CIFAR [143] have played a crucial role in advancing deep learning research by enabling fair comparisons across models and tasks.
- **Model:** Deep learning models are composed of multilayer neural networks that automatically learn hierarchical data representations through successive transformations. Typical architectures include CNNs [147] for spatial data, RNNs [225] for sequential data, and generative models such as GANs [90] and VAEs [140] . These models are trained using optimization algorithms like stochastic gradient descent (SGD) [23] and adaptive methods such as Adam [139] to minimize loss functions over large datasets.
- **Learning Program:** The procedural logic for defining the model architecture, specifying training objectives (e.g., loss functions), and implementing optimization strategies.
- **Framework:** Software libraries such as TensorFlow [189], PyTorch [131], MXNet [39], and Caffe [124] that abstract and accelerate model development, training, and deployment. Many frameworks also support distributed computing and GPU acceleration.

Deep learning has demonstrated performance across a diverse set of tasks covering multiple domains, including computer vision [63, 133], natural language processing [61, 188], speech recognition [15, 89], and autonomous systems [129, 130]. Representative task categories include:

- **Classification:** Assigning discrete labels to inputs, such as in image recognition or sentiment analysis.
- **Regression:** Predicting continuous values, e.g., age estimation or trajectory prediction.
- **Sequence Modeling:** Capturing temporal or structured dependencies, essential in language modeling, speech recognition, and video analysis.
- **Generative Modeling:** Learning data distributions to synthesize realistic samples, using models like GANs and VAEs.
- **Control and Decision-Making:** Optimizing actions in interactive environments, foundational to reinforcement learning applications such as autonomous driving and game playing.

2.2 Deep Learning Testing

Testing DL systems involves evaluating and validating models with respect to various quality attributes, including but not limited to performance, robustness, correctness, and security. As DNNs are increasingly integrated into safety-critical and high-stakes domains such as autonomous driving, healthcare, and finance, testing practices have become essential to ensure their reliability and trustworthiness [10, 149]. Compared to traditional software systems, DNNs introduce unique challenges for testing. One major distinction is the lack of a clearly defined oracle (i.e., the expected output could be ambiguous or subjective). Moreover, unlike traditional software engineering, where test oracles can often be automatically derived from formal specifications, assertions, or reference implementations, DL models typically lack such precise specifications. As a result, oracle construction for DNNs often requires manual labeling, which introduces additional cost and potential uncertainty. This has led to a growing body of work aimed at reducing labeling effort through techniques such as input prioritization, sampling, and test selection.

From another perspective, unlike traditional software systems, deep learning models are also susceptible to fairness concerns. This is primarily due to their data-driven nature. DNNs learn patterns directly from training data, which could reflect societal biases or underrepresent certain demographic groups. As a result, models can exhibit disparate performance across subpopulations, raising concerns about equity and trust. In recent years, research on fairness testing and mitigation techniques for DNNs has gained increasing attention in both academia and industry [42, 43]. In this paper, we organize the existing literature on DL testing based on the specific quality attributes targeted by different testing techniques, aiming to provide a structured and comprehensive understanding of the field, identify common trends and challenges, and highlight promising directions for future research.

3 TAXONOMY OF DEEP LEARNING TESTING TECHNIQUES

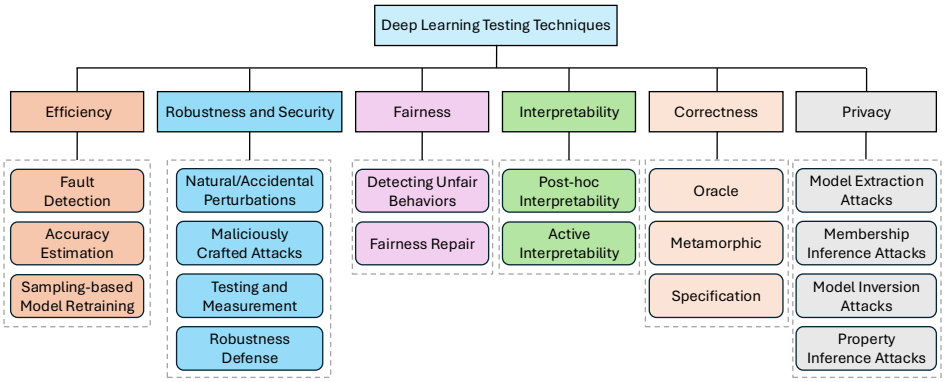


Fig. 2. Article structure

This section presents a taxonomy of deep learning testing techniques. It categorizes existing methods based on key testing goals, including efficiency (how to reduce cost), robustness and security (how to resist perturbations and attacks), fairness (how to ensure equity), interpretability (how to explain), correctness (how to detect label errors), and privacy (how to protect data). Figure 2 illustrates the overall structure of this taxonomy. Our taxonomy is methodology-centric, which organizes testing techniques based on their underlying assumptions, mechanisms, and evaluation paradigms. This taxonomy offers two key contributions: 1) It reveals the underlying methodological structure shared across disparate fields, such as test input prioritization or coverage-based adequacy metrics appearing both in SE and ML contexts; 2) It enables cross-community comparison, helping

researchers identify reusable patterns, adapt techniques across domains, and better understand the testing landscape.

3.1 Efficiency Techniques

These techniques focused on reducing the labeling costs (i.e., the effort required to annotate test inputs with ground truth labels) for testing the deep learning systems. The motivation is that, labeling all test inputs to verify the correctness of DL models can be expensive: 1) Manual annotation remains a mainstream approach. 2) Test sets can be large-scale. 3) Domain-specific expertise can be required to accurately label test inputs. For example, in the medical field, labeling images for diabetic retinopathy (DR) detection requires specialized knowledge in ophthalmology [289]. Existing methods can be categorized into: fault detection techniques (cf. Section 5.1), accuracy estimation techniques (cf. Section 5.2), and sampling-based model retraining techniques (cf. Section 5.3).

3.1.1 Fault Detection.

Fault detection methods aim to identify tests that are more likely to be misclassified by the evaluated deep learning model. By labeling and executing these tests earlier, developers/testers can speed up debugging, improve testing efficiency, or leverage these potentially misclassified tests for efficient retraining to improve model performance. From the perspective of strategy, these methods can be broadly categorized into: test input prioritization (ranking), test input selection (selecting), test adequacy evaluation (measurement), and test input generation (creating).

(1) **Test Input Prioritization:** Test input prioritization aims to identify and prioritize potentially misclassified test inputs, without discarding any of the tests. Labelling such fault-revealing inputs earlier facilitates earlier debugging, thereby enhancing the efficiency of DNN testing. The concept is inspired from traditional software testing, where test prioritization seeks to detect faults earlier during test execution [206], typically guided by coverage metrics.
(2) **Test Input Selection:** Test input selection aims to select a subset of test data that is more likely to be misclassified [174]. Unlike test prioritization, these methods do not retain all test cases. Instead, they select a portion of the tests to accelerate the debugging process or to retrain the deep learning model to improve its performance.
(3) **Test Adequacy Evaluation:** Test adequacy evaluation aims to assess whether a test set is sufficiently effective at revealing faults in the DL model [135]. Some mainstream methods include neuron coverage [172], surprise adequacy [135], and mutation score [173].
(4) **Test Input Generation:** Test input generation aims to automatically construct inputs that expose potential faults or inconsistent behaviors in DL models. Early methods include DeepXplore [191], which first introduced neuron coverage and differential testing for test generation. Subsequent research proposed a variety of generation approaches, including coverage-guided fuzzing [102, 186, 229, 263, 272], search-based testing [231, 237, 309], generative modeling [284], symbolic execution [93, 222], and metamorphic testing [67].

3.1.2 Accuracy Estimation.

Accuracy estimation aims to estimate the accuracy of the entire test set by selecting only a small subset of tests from the original test set (or even not selecting any tests) [37].

(1) **Sampling-based methods:** These methods select a subset of representative tests from the original test set to label, thereby reducing the labelling efforts. Representative methods include CSE [159], PACE [37], and DeepSample [99].
(2) **Labelling-free methods:** These methods estimate model accuracy without requiring any test data labeling, thereby avoiding the cost of manual annotation. They operate by analyzing

internal representations, model confidence, or training data. Representative methods include Aries [116] and AETTA [148].

3.1.3 *Sampling-based Model Retraining.*

Sampling-based model retraining approaches focus on fine-tuning the pre-trained DL model with carefully selected and labeled test inputs to efficiently improve the model's performance. In the ***SE community***, numerous existing studies [75, 81, 104, 135, 174, 254] have also leveraged fault detection methods for retraining, thereby improving the performance of deep learning models. For example, the fault detection metric DeepGini has demonstrated high effectiveness in model retraining [75]. In the ***ML community***, deep active learning-based methods (DAL) were initially proposed for efficiently selecting informative samples to train classifiers [96, 167]. Recent studies [150] have also demonstrated their applicability in selecting informative samples for fine-tuning deep neural networks to reduce labeling effort.

3.2 Fairness Techniques

Fairness emphasizes that, in the process of automated decision-making, deep learning models should avoid treating individuals or groups unfairly due to sensitive attributes such as race, gender, or age. It is important because unfair model decisions can lead to real-world harms such as discrimination, loss of opportunity, and societal bias amplification, especially in high-stakes domains like hiring, lending, and criminal justice [178].

3.2.1 *Techniques for Detecting Unfair Behaviors.*

These techniques aim to detect unfair behaviors in deep learning model predictions. Among them, one of the most common approaches is fairness-oriented test input generation, which synthesizes inputs (such as counterfactual pairs or demographically diverse samples) to trigger and reveal discriminatory behaviors.

(1) **Random Generation:** These methods sample test inputs randomly from the input space to evaluate whether the model exhibits unfair behavior when sensitive attributes are altered [12].
(2) **Search-based Generation:** These methods employ techniques such as heuristic search, evolutionary algorithms, or gradient guidance to actively explore regions of the input space that are most likely to trigger fairness violations. Representative methods include Aequitas [238] (black-box methods), ADF [285] (gradient-guided white-box method), NeuronFair [302] (neuron-level interpretability method), and DICE [183] (information-theoretic method).
(3) **GAN-based Generation:** These approaches leverage Generative Adversarial Networks (GANs) to generate synthetic image samples by modifying sensitive attributes (e.g., gender or skin tone) while preserving other semantic content [264]. They are commonly applied in fairness testing for computer vision tasks, including face recognition and gender classification [8].
(4) **Template-based Generation:** These methods craft manual templates with placeholders for sensitive attributes (e.g., "`<person>` got a promotion last week.") and substitute them with words from different demographic groups to check whether the model produces different predictions [146].
(5) **Verification-based Generation:** These methods formalize fairness properties as logical constraints and use symbolic reasoning (e.g., SMT solvers) to analyze models for provable guarantees or counterexamples [18].
(6) **Other Methods:** In addition to the above mainstream approaches, several emerging methods have also been proposed:
    - **LLM-based Generation:** These methods leverage large language models (LLMs) to generate counterfactual examples for detecting biases [31].

- **Grammar-based Generation:** These methods use context-free grammars to automatically create well-formed and syntactically diverse test inputs for Natural Language Processing (NLP) models [219].

3.2.2 *Fairness Repair Techniques.*

Fairness Repair Techniques aim to mitigate unfair biases in deep learning models, thereby enabling fairer predictions across protected groups.

(1) **Data-level Repair:** These repair methods focus on modifying or augmenting the data distribution before training to reduce model bias. Mainstream approaches include re-sampling [26] and GAN-based augmentation [197].
(2) **Model-level Repair:** These repair methods focus on introducing fairness constraints or architectural modifications directly into the training process of the model. These methods intervene during model optimization to prevent the learning of biased representations. Mainstream strategies include adversarial debiasing [151], fairness-aware loss functions [228], and causal regularization [128].
(3) **Post-hoc Repair:** Post-hoc repair improve fairness after training. They applies methods like distillation [177], fine-tuning [71], and prompting [84] to pre-trained models to improve the fairness.

3.3 Interpretability Techniques

Interpretability refers to the degree to which a human can understand the internal mechanics or decision-making process of a machine learning model. In the context of deep learning, which often involves complex and opaque architectures, interpretability helps uncover why a model makes a particular prediction, what features it relies on, and whether its reasoning aligns with domain knowledge or ethical expectations. Interpretability is especially important in high-stakes applications such as healthcare, finance, and legal decision-making, where understanding model behavior is critical for ensuring safety, fairness, trust, and regulatory compliance. This section introduces two main categories of interpretability methods: *post-hoc techniques*, which explain trained models without altering their structure, and *active techniques*, which enhance interpretability by modifying the training process or architecture.

3.3.1 *Post-hoc Interpretability Techniques.*

These techniques focus on explaining or understanding a trained neural network without modifying its architecture or training process. The mainstream methods can be categorized as follows:

(1) **Rule-based Explanation:** These methods explain model behavior by extracting human-readable decision rules (such as decision trees or if-then rules). The rules can be local (explaining why a particular instance is classified a certain way) or can be global (approximating the overall function of the model). These explanations are particularly useful in safety-sensitive applications like healthcare or finance [100, 292].
(2) **Attribution-based Explanation:** These techniques assign importance scores to input features based on their contribution to the model's output. In computer vision [121, 217, 224], they are commonly visualized as saliency maps highlighting influential pixel regions.
(3) **Hidden Semantics Interpretation:** These techniques aim to understand the internal representations learned by neural networks, especially the semantics of intermediate layers or hidden neurons. For example, in convolutional neural networks trained for image classification, certain neurons may consistently activate in response to specific visual patterns such as animal heads, textures, or object parts.

(4) **Explanation by Example:** These methods explain a model's decision by referring to specific examples, such as similar inputs from the training set that most influenced the prediction. This approach can be especially helpful when abstract rules or numerical attributions are hard to interpret.

3.3.2 Active Interpretability Methods.

These methods aim to actively modify the training process or model architecture to enhance the interpretability of the model. The current mainstream methods fall into four categories:

(1) **Decision tree-based methods:** These methods modify the model architecture or training objectives to induce a decision-tree-like behavior.
(2) **GAM-based methods:** Inspired by classical generalized additive models (GAMs) [106], these models are designed to output the additive effects of individual input features. Their architecture is structured to make each feature's influence separable and interpretable, making them explainable.
(3) **Model optimization:** These approaches incorporate interpretability-related objectives directly into the training loss. They actively guide the model to learn representations that correspond to human expectations.
(4) **Interpretable modules:** These methods insert interpretable components into the network architecture (such as prototype layers or concept bottlenecks) during design time.

While active interpretability techniques have attracted increasing attention in the broader machine learning (ML) community, there is currently very little research in the software engineering (SE) community that falls under the category of "active interpretability methods". Most interpretability methods in the SE field are currently post-hoc (which can be referred to Section 7.1).

3.4 Robustness and Security Techniques

These techniques aim to evaluate the capability of deep learning models to resist perturbations or adversarial manipulations, thereby ensuring secure and reliable behavior. Robustness has become a critical requirement in safety-sensitive domains such as autonomous driving, medical diagnosis, and financial services [34], where model failures may lead to severe consequences.

3.4.1 Testing Against Natural/Accidental Perturbations.

This category focuses on evaluating the robustness of deep learning models against natural or accidental perturbations that frequently occur in real-world environments. Such perturbations include various forms of image degradation or transformation caused by environmental conditions, sensor noise, geometric distortions, occlusions, or rendering artifacts [203]. Although these perturbations are not malicious in nature, they can significantly affect model performance and generalization. The goal of these testing techniques is to simulate realistic input variations and assess model stability under such conditions. A more detailed review of representative perturbation types and corresponding testing methodologies is provided in Section 8.1.

3.4.2 Testing Against Maliciously Crafted Attacks.

These techniques aim to evaluate the robustness of deep learning systems against deliberately engineered input (such as adversarial examples or data poisoning) designed to exploit model vulnerabilities.

(1) **Data-oriented attacks:** These attacks target the training or inference data of deep learning models by manipulating input samples. For example, *poisoning attacks* inject malicious samples into the training set to implant backdoors or degrade performance [98, 211], while *adversarial attacks* add imperceptible perturbations to inference-time inputs to cause mispredictions [91, 176].

(2) **Model-oriented attacks:** These attacks manipulate the model's architecture, parameters, or training process to embed malicious behavior. Common examples include *model poisoning attacks*, which degrade performance or cause targeted errors through malicious parameter updates or tampering with deployed models [288]; *model-reuse attacks*, where malicious code is embedded in pre-trained components that later trigger incorrect behavior in downstream systems [123]; and *neural trojan attacks*, which embed hidden triggers that activate on specific inputs during inference while maintaining normal behavior otherwise [169].

3.4.3 *Testing and Measurement Frameworks.*

Testing and measurement frameworks aim to support the evaluation of robustness in deep learning models under adversarial and non-adversarial settings. Key focus areas include:

(1) **Metric-driven Evaluation Frameworks:** These frameworks emphasize designing diverse and fine-grained evaluation metrics to capture robustness from multiple perspectives. Common metrics include neuron coverage, perturbation imperceptibility, confidence deviation, and decision boundary distance [101].
(2) **Automated Attack-based Evaluation:** This category focuses on integrating multiple complementary adversarial attacks into parameter-free and fully automated testing suites. Such frameworks ensure reproducible robustness evaluation without manual tuning, and are widely used in benchmarking [27, 50, 52].
(3) **Formal Verification:** Formal methods aim to provide provable robustness guarantees within constrained input spaces. Examples include symbolic analysis or abstract interpretation [85].
(4) **Testing-guided Robustness Optimization:** These frameworks bridges testing and model improvement by using test results to guide retraining or repair [221, 246].

3.4.4 *Robustness Defense Techniques.*

These techniques aim to improve model robustness by mitigating the impact of adversarial or accidental perturbations [24]. They serve as follow-up actions to robustness testing, addressing vulnerabilities identified during the testing process. These techniques can be broadly categorized into the following types:

(1) **Mitigation Techniques:** These techniques proactively enhance model robustness through training-time or structural modifications. Common approaches include adversarial training [92, 176], input pre-processing (e.g., denoising or compression) [218, 266], model architecture changes (e.g., distillation, regularization) [295], and ensemble methods that aggregate predictions from multiple models to reduce vulnerability [235].
(2) **Detection Techniques:** These methods aim to identify adversarial inputs at inference time to prevent them from influencing model decisions. Techniques range from auxiliary classifiers [180], and distribution-based outlier detection [95], to transformation consistency checks [185, 266], and behavior-based anomaly detectors that monitor internal activations [41, 184].
(3) **Adaptive Test-Time Defenses:** These defenses operate during inference by dynamically adjusting input representations or model responses to mitigate adversarial effects [144, 252], often without requiring changes to the training process.

3.5 Correctness Techniques

These techniques for deep learning focus on verifying whether a model's predictions and behaviors align with expected functionality or predefined specifications. While certain techniques (such as fault detection and accuracy estimation) are related to correctness, they are primarily designed to improve testing efficiency and are therefore discussed separately in Section 5. To avoid redundancy,

this section concentrates on correctness-oriented methods whose primary goal is to directly assess behavioral correctness, including approaches based on test oracles, metamorphic relations, and formal or informal specifications.

(1) **Oracle-based/Oracle-Free Testing** Oracle-based approaches use predefined mechanisms to judge correctness, while oracle-free techniques leverage strategies such as cross-model comparison, differential testing, or statistical indicators to detect inconsistencies without requiring ground truth labels.
(2) **Metamorphic Testing** Metamorphic testing validates model correctness by checking whether outputs remain consistent or change predictably when inputs are modified through specific transformations. It is particularly useful when no reliable oracle is available.
(3) **Specification-based Testing** Specification-based testing verifies model behavior against defined properties or constraints, such as safety conditions or input-output invariants. These specifications can be manually written, formally defined, or learned from observed failures, enabling validation of model conformance to expected behaviors.

3.6 Privacy Testing Techniques

Privacy in the context of deep learning refers to the protection of sensitive information from unintended leakage during model training or inference. This includes both the privacy of individuals whose data is used to train the model and the confidentiality of proprietary model parameters or internal representations [182]. Given the data-driven nature of deep learning models, even black-box access can potentially leak critical information, raising significant privacy concerns. Privacy is especially important in domains such as healthcare, finance, and personalized services [127, 168], where data often contains personally identifiable or confidential information.

Privacy-oriented testing techniques aim to evaluate the extent to which deep learning models unintentionally expose sensitive information. The purpose is to test the confidentiality boundaries of a model by attempting to steal proprietary parameters, reveal sensitive training data, or reconstruct internal representations. Some popular categories of attacks are as follows:

(1) **Membership Inference Attacks:** Attackers infer whether a specific data sample was part of the model's training set by analyzing the model's response to that input, thereby revealing information about data participation [215]. This poses a threat to data privacy guarantees.
(2) **Model Extraction Attacks:** Attackers query the model via its API to collect input-output pairs and reconstruct a functionally equivalent substitute model. This threatens the intellectual property and commercial value of the original model.
(3) **Model Inversion Attacks:** Attackers exploit the model's output (e.g., confidence scores) to recover sensitive features or the full data sample.
(4) **Property Inference Attacks:** Attackers aim to infer certain latent properties present in the training dataset that are neither part of the labels nor used as input features. For example, in a disease prediction task, although the "gender" attribute is not included as an input feature, the training dataset may consist of a specific proportion of male and female patients. By analyzing the model's outputs, the attacker is able to infer this gender ratio. Such inference can lead to the leakage of sensitive information.

4 PAPER COLLECTION

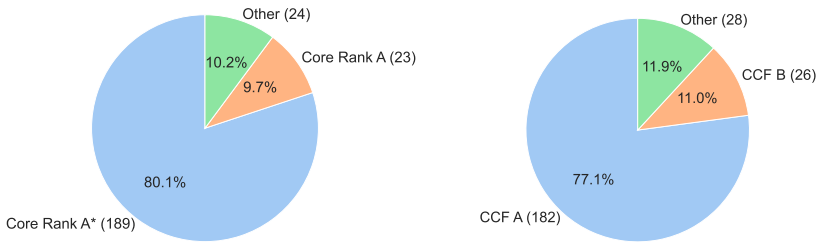
In this section, we introduce the scope of our survey, the methodology for collecting and filtering relevant papers, and a statistical analysis of the collected literature.

4.1 Survey Scope

In this study, we adopt a broad and unified definition of "DL testing": any activity designed to reveal defects, vulnerabilities, or undesirable behaviors in DL models. To understand the current landscape of deep learning testing techniques, we collect and analyze 236 relevant research papers published between 2015 and 2025. Our focus covers multiple testing properties of deep learning systems, including efficiency, robustness, fairness, interpretability, correctness, and privacy. Specifically, we use the following selection criteria for collecting papers:

- The paper proposes a novel technique, framework, or method that targets at least one testing property (e.g., robustness, fairness, privacy) or component (e.g., data, learning program) of deep learning systems.
- The paper provides an empirical evaluation or in-depth analysis of existing testing methodologies, tools, or benchmarks in the context of DL system testing.
- The paper introduces datasets, benchmarks, or evaluation protocols that are specifically designed for testing purposes.

In addition, we prioritize papers published in top-tier venues, including SE conferences and journals such as ICSE, FSE, ASE, ISSTA, TOSEM, and TSE; ML venues like NeurIPS, ICLR, ICML, and JMLR; computer vision conferences such as CVPR, ICCV, and ECCV; NLP venues including ACL, EMNLP, NAACL, and TACL; security conferences like IEEE S&P, USENIX Security, CCS, and NDSS; as well as other relevant venues in data mining (e.g., KDD, WWW, TKDE) and systems/networking (e.g., OSDI, SOSP, NSDI). As shown in Figure 3, 80.1% of the surveyed papers are published in Core Rank A* venues, and 9.7% in Core Rank A venues. Similarly, 77.1% appear in CCF A venues, and 11.0% in CCF B venues. Here, CCF refers to the Chinese Computer Federation's conference ranking system, which is widely used in China to evaluate publication quality. CCF A and B denote top-tier and high-quality conferences, respectively.



(a) Core Rank Venue Distribution (b) CCF Venue Distribution

Fig. 3. Core Rank and CCF Distribution

4.2 Paper Collection Methodology

We follow a keyword-based retrieval methodology, inspired by the approach adopted in previous surveys [115]. As summarized in Table 1, for each major category in our taxonomy, we construct search keywords using combinations of terms such as "deep learning", "neural network", and "test", together with property-specific phrases. For example, to retrieve papers related to efficiency-oriented testing, we use queries such as "deep learning neural network test prioritization" and "deep learning accuracy estimation". The complete list of search keywords used in our study is presented in Table 1.

Our sources include digital libraries such as IEEE Xplore, ACM Digital Library, SpringerLink, arXiv, and Google Scholar. Our process consists of two main steps. First, we perform a keyword-based search using the taxonomy shown in Table 1. For each category, we construct search queries

Table 1. Keyword list

Category	Keywords
Efficiency	deep learning / neural network test prioritization deep learning / neural network accuracy estimation deep learning / neural network fault detect deep learning / neural network test select deep learning / neural network efficient deep learning / neural network misclassification detect deep learning / neural network retrain
Robustness	deep learning / neural network robustness deep learning / neural network security deep learning / neural network natural/accidental deep learning / neural network robustness measure deep learning / neural network robustness defense
Fairness	deep learning / neural network fair deep learning / neural network bias
Interpretability	deep learning / neural network interpretability/interpretable deep learning / neural network explain deep learning / neural network attribution
Correctness	deep learning / neural network oracle deep learning / neural network metamorphic test deep learning / neural network specification test
Privacy	deep learning / neural network membership deep learning / neural network privacy deep learning / neural network inference deep learning / neural network extract/steal deep learning / neural network inversion

that combine core terms. Then, following the previous study [115], we use citation-based search to identify potentially missing papers. In this step, we search for papers on Google Scholar and consider: 1) the references cited in the collected papers, and 2) the papers that cite them.

Furthermore, we performed a **manual review** of all initially retrieved papers after the keyword-based search. Our goal was to ensure that the included papers not only matched the keyword criteria but also aligned with the intended scope of this survey. We prioritized studies published in CCF A or B venues, or in Core Rank A* and Core Rank A conferences or journals.

4.3 Collection Results

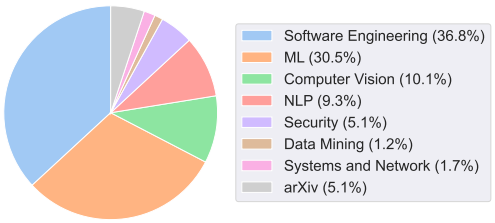


Fig. 4. Publication Venue Distribution

In total, we collected 236 papers related to the testing of deep learning systems, published between 2013 and 2025.

Temporal Distribution: Figure 1 illustrates the temporal distribution of the collected publications. The X-axis represents the publication year, while the Y-axis indicates the number of papers published annually. We observe a significant growth in research interest starting from 2017, with the number of publications peaking around 2023. This trend highlights the increasing attention to testing methodologies for deep learning systems in recent years.

Venue Distribution Figure 4 shows the distribution of the surveyed papers across different research communities. The major proportion of the works were published in Software Engineering venues

(36.8%) and Machine Learning venues (30.5%). The remaining papers are distributed across other domains, including Computer Vision (10.1%), Natural Language Processing (9.3%), and Security (5.1%).

5 EFFICIENCY TECHNIQUES

As deep learning systems grow in complexity and scale, testing them becomes increasingly expensive in terms of both computational cost and manual labeling effort. Moreover, unlike traditional software systems, where test oracles can often be explicitly defined, deep learning models lack automatic oracle mechanisms and therefore require manual labeling for testing, which can result in substantial human and time costs. As a result, improving the efficiency of deep learning testing has become a critical goal in both research and practice. In this section, we review efficiency-oriented testing techniques from three key perspectives: 1) fault detection, 2) accuracy estimation, and 3) sampling-based model retraining. These techniques aim to minimize labeling cost and computational overhead while preserving or enhancing testing effectiveness. Figure 5 illustrates the workflow of these techniques, organized according to the three aspects described above.

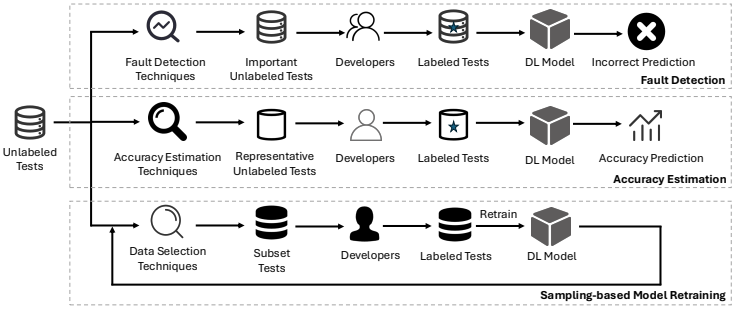


Fig. 5. Workflow of efficiency-oriented techniques

5.1 Fault Detection

Fault detection plays a central role in testing DL systems, aiming to identify test inputs that are likely to trigger incorrect model behaviors. Compared to traditional software systems where test oracles can be defined, DL models typically lack precise, manually defined specifications, making it difficult to determine whether a given output is correct. Consequently, fault detection techniques for DL systems rely on various heuristics and analysis strategies to prioritize or identify potentially misclassified inputs.

5.1.1 Coverage-based Methods.

These methods leverage coverage criteria (e.g., neuron coverage, layer coverage, and MC/DC-inspired structural coverage) to evaluate test inputs and detect tests that are more likely to be misclassified. As one of the earliest fault detection approaches for deep learning testing, coverage-based methods are adapted from traditional software engineering [191], where coverage metrics (e.g., statement, branch, or path coverage) are widely used to assess test adequacy.

(1) Neuron Coverage This metric measures the proportion of neurons activated in a deep neural network when test inputs are executed, thus guiding test selection, prioritization, or test input generation in DNN testing. Pei *et al.* [191] (SOSP 2017) first proposed the concept of neuron coverage, which is inspired from code coverage in traditional software testing [88]. Specifically, a neuron is considered activated if its output exceeds a predefined threshold for a given input. Neuron coverage is then defined as the ratio of unique activated neurons to the total number of neurons in the network across a test suite. Building on this foundational idea, Ma *et al.* [172]

(ASE 2018) further extended neuron coverage by proposing a set of fine-grained testing criteria. To validate coverage-guided testing in real-world safety-critical domains, Tian *et al* [231] (ICSE 2018) proposed DeepTest, which explores the internal logic of deep neural networks on DNN-driven self-driving cars. DeepTest automatically generates realistic synthetic inputs and leverages metamorphic relations to detect incorrect behaviors without requiring manual specifications.

Later efforts expanded neuron coverage into new testing paradigms. Guo *et al.* [102] (ESEC/FSE 2018) proposed DLFuzz, a fuzzing-inspired testing framework that mutates inputs via gradient-based optimization to maximize neuron coverage while inducing mispredictions. Similarly, Yu *et al.* [272] (TNNLS 2022) introduced Test4Deep, a white-box differential testing method that activates under-exercised neurons to uncover behavioral inconsistencies without requiring manual test oracle labeling. More recently, Qi *et al.* [195] (IST 2024) significantly extended the neuron coverage landscape by proposing 80 novel metrics derived from both neuron-level and layer-level activation patterns, monitored not only during inference but throughout training.

(2) MC/DC coverage In addition to neuron-level coverage metrics, researchers have also explored structural testing criteria inspired by traditional software engineering. Sun *et al.* [223] (TECS 2019) proposed a set of structural coverage criteria for DNNs, inspired by the Modified Condition/Decision Coverage (MC/DC) criterion in traditional software testing, which requires that each condition in a compound decision is shown to independently influence the overall decision outcome. They abstract neurons or groups of neurons as "conditions" and "decisions" across adjacent layers, defining four criteria that capture how changes in one part of the network affect downstream behavior, thus enabling fine-grained testing of DNNs.

(3) Coverage-guided Fuzzing Another line of work integrates coverage criteria with automated input mutation, forming the basis of coverage-guided fuzzing in deep learning testing. Odena *et al.* [186] (ICML 2019) proposed TensorFuzz, a coverage-guided fuzzing (CGF) framework for neural networks. TensorFuzz leverages approximate nearest neighbor algorithms to define coverage in the activation space of neural networks. To expand the diversity and depth of coverage exploration, Xie *et al.* [263] (ISSTA 2019) developed DeepHunter, a CGF framework that integrates multiple coverage criteria with adaptive seed selection and metamorphic mutation strategies. More recently, Bai *et al.* [16] (IST 2024) proposed CriticalFuzz, which advances CGF by incorporating the concept of critical neurons (neurons that are frequently activated during training and strongly influence model output). By prioritizing the coverage of these neurons, CriticalFuzz improves fault detection capability.

(4) Other coverage criteria Beyond activation frequency or structural criteria, a set of recent works has explored novel forms of coverage that aim to model the decision logic or semantic behavior of DNNs more precisely. Gerasimou *et al.* [86] (ICSE 2020) proposed DeepImportance, which uses relevance propagation to identify neurons with high semantic impact on output decisions. Xie *et al.* [262] (TOSEM 2022) introduced Neuron Path Coverage (NPC), which abstracts decision logic as neuron activation paths and measures how thoroughly a test suite exercises these paths. Extending this line further, Shi *et al.* [214] (IST 2025) proposed a Markov model-based coverage criterion that formalizes the decision logic of deep neural networks as probabilistic transitions between critical neurons, extracted using Layer-wise Relevance Propagation. Overall, these emerging coverage criteria aim to capture the internal decision logic and semantic behaviors of deep neural networks more precisely than traditional structural or activation-based approaches.

5.1.2 Confidence-based Methods.

Confidence-based methods assume that if a deep learning model is less confident in classifying a test input, the input is more likely to be misclassified. These methods are known for their simplicity and **computational efficiency**, with low runtime and operational costs, making them widely

applicable in practical testing scenarios. As an early representative in this category, Feng *et al.* [75] (ISSTA 2020) proposed DeepGini, a classical confidence-based method. DeepGini leverages the Gini score to quantify the model's confidence for each test input and ranks all the tests in the test set based on their Gini scores.

To systematically examine the effectiveness of such metrics, Weiss *et al.* [256] (ISSTA 2022) conducted a replicability study evaluating a diverse set of confidence-based techniques, including Vanilla Softmax, Prediction-Confidence Score (PCS), Entropy, and Monte Carlo (MC) Dropout. Their findings reveal that these standard uncertainty quantifiers perform comparably to DeepGini in terms of fault detection effectiveness. Building upon these insights, Recently, Chen *et al.* [36] (ASE 2024) proposed FAST, which improves high-confidence error detection by dynamically pruning low-contributing features to recalibrate prediction confidence. Li *et al.* [160] (ISSTA 2024) introduced DATIS, which is currently the state-of-the-art approach. DATIS employs a two-step process: it first identifies test inputs with high uncertainty by leveraging distances to nearest neighbor training samples, and subsequently filters redundant inputs based on inter-input distances to maximize fault diversity.

5.1.3 Surprise-based Methods.

Surprise adequacy (SA) measures the "surprise" of an input by comparing the neural activation behavior of a deep learning system on that input against its training data. It was first proposed by Kim *et al.* [135] (ICSE 2019). In their paper, they introduced two specific metrics: LSA, which estimates the likelihood of an input's activation trace, and DSA, which compares its distance to same-class versus different-class training samples. To demonstrate the practical value of SA in real-world settings, Kim *et al.* [137] (ESEC/FSE 2020) present an industrial case study conducted in collaboration with Hyundai Motor Company, demonstrating how the SA metric can be effectively applied to reduce manual labelling efforts in the development of DNN-based semantic segmentation systems for autonomous driving. Subsequently, Weiss *et al.* [255] (DeepTest 2021) recognized the computational inefficiency of SA as a key limitation. To address this issue, they developed a performance-optimized implementation that reduces runtime by up to 97%. Recently, Kim *et al.* [136] (TOSEM 2023) analyzed three variants of SA across a diverse set of DL models and benchmarks, including safety-critical autonomous driving systems, demonstrating that SA metrics effectively quantify the behavioral novelty of inputs relative to training data and are strongly correlated with the model's prediction correctness.

5.1.4 Learning-based Methods.

Learning-based methods train auxiliary models to automatically learn features from test inputs and use these features to predict the probability that the test inputs will be misclassified by the DL model. This approach has recently gained increasing attention due to its high effectiveness. As an early effort in this category, Wang *et al.* [254] (ICSE 2021) proposed PRIMA, which leverages mutation analysis combined with learning-to-rank techniques to prioritize tests that are more likely to be mispredicted by the model. Subsequently, Chen *et al.* [35] (ASE 2023) proposed ActGraph, which builds a dynamic activation graph to capture high-order neuron interaction features for training a learning-to-rank model that prioritizes test inputs likely to expose DNN misbehaviors. Demir *et al.* [57] (ISSTA 2024) further generalized this learning-based strategy, proposing a meta-model-based test prioritization framework that combines multiple uncertainty metrics to predict misclassification likelihood using a logistic regression model.

In parallel, several domain-specific learning-based methods have emerged to tackle challenges in specialized DNN architectures and data modalities. For instance, Dang *et al.* [54] (TOSEM 2023) proposed GraphPrior for Graph Neural Networks (GNNs). They designed GNN-oriented mutation rules and employed ranking models to estimate the probability of each test being misclassified.

Targeting 3D vision tasks, Li *et al.* [154] (TOSEM 2024) proposed PCPrior. By combining spatial, mutation, prediction, and uncertainty features, PCPrior constructs rich representations of test inputs and leverages a learning-to-rank model to prioritize tests more likely to be misclassified. In the context of speech recognition, Yang *et al.* [269] (TOSEM 2025) introduced Prophet, which leverages word-level error prediction using pre-trained language models to estimate the likelihood that a test case will reveal recognition error.

5.1.5 Other Methods.

In addition to the above mainstream categories, prior studies have also introduced several other popular fault detection approaches that demonstrated effectiveness in some specific scenarios. For instance, Ma *et al.* [175] (ESEC/FSE 2021) proposed a differential testing approach that uses subspecialized models as references to detect errors by identifying prediction inconsistencies on specific data slices. Gao *et al.* [81] (ICSE 2022) proposed Adaptive Test Selection (ATS), which prioritizes inputs that enhance geometric coverage in the output space to efficiently detect faults with fewer tests. Zheng *et al.* [301] (ASE 2023) proposed CertPri, a certifiable test prioritization method that ranks inputs by their estimated movement cost in feature space to detect bugs early across tasks.

5.2 Accuracy Estimation

While the previous section introduced fault detection as a means to improve testing efficiency by identifying potentially misclassified inputs, another important line of work focuses on improving efficiency through accuracy estimation, which aims to select representative test cases to estimate the accuracy of the entire test set. In some cases, accuracy estimation can even be performed without labeling any test data.

5.2.1 Sampling-based methods.

These methods sampled a subset of representative test inputs from the entire test set to label, thereby reducing the total labelling cost. A representative early effort in this direction is Cross Entropy-based Sampling (CES) and Confidence-based Stratified Sampling (CSS) proposed by Li *et al.* [159] (ESEC/FSE 2019). CES improves estimation efficiency by conditioning on the internal representations learned by the DNN and selecting representative test inputs that minimize the cross-entropy between the empirical distribution and a target distribution over this latent space. In contrast, CSS stratifies the operational data based on prediction confidence and samples proportionally from each stratum.

Subsequent work by Chen *et al.* [37] (TOSEM 2020) was the first to introduce clustering for test selection in DNNs. They proposed PACE, which uses input clustering and adaptive sampling to select diverse, representative test inputs for practical accuracy estimation in DNNs. Meanwhile, some studies approached the problem from the perspective of neuron coverage. For example, Zhou *et al.* [305] (ISSRE 2020) proposed DeepReduce, a two-phase input reduction method that uses multi-objective optimization to preserve neuron coverage and match output distribution for cost-effective accuracy estimation. Most recently, Guerriero *et al.* [99] (ICSE 2024) introduced DeepSample, a family of sampling-based testing techniques designed for DNNs. DeepSample incorporates advanced concepts from statistical sampling theory, such as stratification, unequal probability sampling, and the use of auxiliary variables (e.g., confidence, DSA, LSA) to guide test selection.

5.2.2 Labelling-Free Methods.

Compared to sampling-based methods, which still require a small amount of labeled data, labelling-free methods aim to eliminate the need for any manual annotation. These techniques

estimate model accuracy purely based on model outputs, internal behaviors, or structural properties, thus requiring no labeling of any data. The first relevant work in the software engineering (SE) community is Aries, proposed by Hu *et al.* [116] (ICSE 2023). Aries estimates DNN accuracy on unlabeled data by analyzing the distance of data points to the decision boundary. Aries leverages dropout-induced uncertainty to partition data into confidence-based buckets and infers accuracy using bucket-wise statistics from the original labeled test set. This method avoids manual labeling and effectively handles distribution shifts in real-world settings.

In the machine learning (ML) community, Zheng *et al.* [303] (NeurIPS 2023) extended the research to the graph domain by introducing GNNEvaluator, which estimates GNN performance on unlabeled graphs by learning from synthetic, discrepancy-aware data and using only model outputs without test labels or model access. More recently, Lee *et al.* [148] (CVPR 2024) proposed AETTA, a novel accuracy estimation framework for test-time adaptation (TTA) scenarios. AETTA estimates test accuracy by measuring the disagreement between the adapted model's predictions and multiple dropout-based stochastic inferences.

5.3 Sampling-based Model Retraining

An important objective of fault detection is to enable targeted improvements through retraining the original model. In this regard, retraining a model using carefully selected test inputs (particularly those that are more likely to expose model faults) has emerged as an effective strategy to enhance model performance. Sampling-based model retraining techniques focus on achieving this goal by fine-tuning pre-trained models on a strategically chosen subset of test data, thereby minimizing annotation costs while maximizing retraining effectiveness. In the SE community, several fault detection methods proposed in the original papers are directly used for retraining [75, 81]. In contrast, the ML community has introduced various active learning methods that have been shown to be effective for fine-tuning [78, 244, 261]. Below, we review representative approaches from both lines of work.

5.3.1 Fault Detection Techniques for Retraining.

First, we review several methods originally proposed for fault detection that have also been directly applied to model retraining in their respective papers. Feng *et al.* [75] (ISSTA 2020) proposed DeepGini for fault detection and demonstrated that DeepGini performs well in selecting inputs to guide retraining: on some datasets, retraining with the top 10% of test inputs selected by DeepGini outperforms retraining with test inputs selected by other methods. Similar observations have also been made in several subsequent studies, such as ATS proposed by Gao *et al.* [81] (ICSE 2022), CertPri by Zheng *et al.* [301] (ASE 2023), and DATIS by Li *et al.* [160] (ISSTA 2024), all demonstrating that their fault detection techniques can be used to guide sample-based retraining.

In contrast to the above methods, where retraining is a secondary application, another body of work has directly focused on test selection for the primary purpose of enhancing retraining. For example, Shen *et al.* [213] (ASE 2020) proposed Multiple-boundary Clustering and Prioritization (MCP) to enhance neural network retraining by identifying and prioritizing test inputs near multiple decision boundaries. Hu *et al.* [114] (TOSEM 2022) investigated the effectiveness of various test selection strategies for DNN retraining under data distribution shifts.

From a repair perspective, Li *et al.* [152] (ISSTA 2022) proposed HybridRepair, which enhances model performance through retraining with both labeled and unlabeled data. HybridRepair identifies failure-prone regions using a local entropy metric, selectively labels representative samples, and propagates labels within dense clusters to maximize supervision under budget constraints. Additionally, it incorporates semi-supervised consistency regularization on carefully selected unlabeled data to increase training density and enhance repair effectiveness.

5.3.2 *Deep Active Learning Techniques for Retraining.*

Existing research has also highlighted the potential of deep active learning (DAL) techniques for retraining deep learning models [150]. Since DAL is a major research direction in the ML community and has almost no existing work in the SE domain [150], the following discussion focuses on briefly outlining its core objectives and methods rather than providing a comprehensive review. The goal is to help readers understand how DAL can be leveraged for sample selection during model retraining. For a more comprehensive survey, we refer readers to the work by Ren *et al.* [199].

A primary class of DAL methods is *uncertainty-based sampling*, which selects unlabeled instances where the model exhibits low confidence, using acquisition functions such as entropy, margin, least confidence, or Bayesian disagreement [113, 205, 244, 261]. Another common approach is *representative-based sampling*, which selects samples that best capture the underlying distribution of the unlabeled data, typically through clustering or density-aware strategies [13, 138, 268]. A third category is *influence-based sampling*, which focuses on selecting inputs expected to have the greatest impact on model performance or parameter updates, often relying on gradient analysis, loss prediction, or influence functions [119, 251, 271, 296]. Lastly, *hybrid methods* integrate multiple strategies (such as combining uncertainty with diversity) to balance informativeness and sample coverage [48].

5.4 Analysis of Efficiency Techniques

To better understand the contributions of existing research in improving the efficiency of testing deep learning systems, we conducted a multi-dimensional analysis of the literature.

5.4.1 *Fault Detection Techniques.*

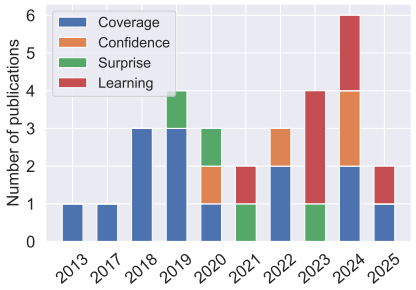


Fig. 6. Trend of fault detection techniques with year

Figure 6 shows the publication trends of fault detection methods from 2013 to 2025. Coverage-based methods have maintained consistent attention over the years, forming a foundational line of research. Representative works include neuron coverage and its extensions (e.g., DeepXplore, DeepGauge, DLFuzz), coverage-guided fuzzing (e.g., DeepHunter, CriticalFuzz), and recent alternatives like importance-based and probabilistic coverage. Confidence-based techniques, such as DeepGini [75], have been shown to outperform coverage methods in both effectiveness and efficiency (with runtime even less than 1 second), making them suitable for practical deployment. Learning-based methods emerged in 2021 and have rapidly gained traction, peaking in 2023 and remaining prominent. These methods employ supervised learning to rank test inputs based on mutation features (e.g., PRIMA, PCPrior), uncertainty or linguistic patterns (e.g., Meta-model, Prophet), or geometric and structural signals (e.g., CertPri, ActGraph). Their experimental results demonstrate that learning-based methods surpass all existing test prioritization techniques [153, 254]. Their rapid evolution reflects increasing interest in intelligent, data-driven testing solutions.

Finding 1: *Coverage-based testing remains a foundational strategy and has received considerable research attention in recent years. Confidence-based methods, in particular, have been shown to be simpler, more efficient, and less time-consuming, making them especially attractive for practical and scalable deployment. Meanwhile, recent years have witnessed growing interest in learning-based methods, primarily due to their high fault detection performance. Although they are generally less efficient than confidence-based approaches, the runtime cost of many learning-based methods remains within an acceptable range.*

5.4.2 *Accuracy Estimation Techniques.*

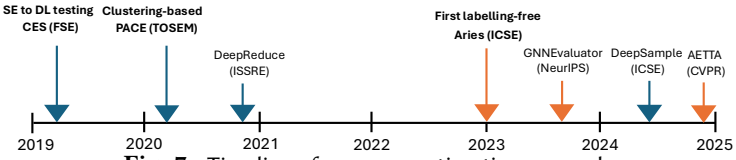


Fig. 7. Timeline of accuracy estimation research

The timeline in Figure 7 reveals the evolution of accuracy estimation methods for DNNs, which shows a clear shift from sampling-based to labeling-free approaches. We can see that the evolution of accuracy estimation techniques for deep learning models reflects a clear shift from cost-reduction strategies to labelling-free innovations. Early research in the SE community focused on sampling-based methods that aim to reduce labeling cost by selecting a small but representative subset of test data. For example, CES and CSS use internal representations and confidence strata for stratified sampling [159]; PACE introduces clustering and MMD-based subset selection to ensure distributional coverage [37]; DeepReduce adds multi-objective optimization to maintain both neuron coverage and output distribution fidelity [305]; and DeepSample incorporates statistical sampling theory and auxiliary signals like uncertainty or surprise [99]. These advances highlight the community's ongoing effort to balance cost efficiency and accuracy estimation quality through smarter test selection.

Since 2023, a new paradigm has emerged: labeling-free accuracy estimation, which bypasses manual annotation altogether by modeling the relationship between internal behaviors and prediction accuracy. Aries [116] uses dropout-based uncertainty and decision boundary proximity to estimate accuracy under distribution shifts. GNNEvaluator [303] regresses GNN accuracy from output embeddings on unlabeled graphs by learning structural discrepancies, while AETTA [148] estimates test-time accuracy by analyzing prediction disagreement and entropy under test-time adaptation. These methods do not require access to ground truth, making them highly attractive for post-deployment quality monitoring or large-scale testing.

Finding 2: *Accuracy estimation for deep learning has evolved from sampling-based approaches that minimize labeling cost through representative test selection to labeling-free techniques that infer model performance directly from internal behaviors or prediction uncertainty. This transition reflects a paradigm shift in test selection: from approximate empirical estimation to behavioral modeling, enabling scalable and annotation-free quality assessment.*

5.4.3 *Sampling-based Retraining Techniques.*

In the SE community, several fault detection techniques originally proposed for identifying incorrect behavior (such as DeepGini [75], ATS [81], CertPri [301], and DATIS [160]) have been repurposed to guide retraining. These methods rank test inputs based on metrics such as confidence, structural deviation, or certified robustness, and select samples that are more likely to be misclassified to fine-tune the model. Empirical studies [174] show that using a small subset of

potentially misclassified inputs can enhance DNN model performance. In addition, methods like MCP [213], DAT [114], and HybridRepair [152] are proposed with retraining as a primary goal. These methods focus on boundary-aware sampling, OOD-aware selection, and hybrid labeling strategies (e.g., combining labeled and unlabeled data with consistency regularization), offering more targeted retraining pipelines under real-world constraints.

In contrast, the ML community contributes numerous deep active learning techniques that are primarily aimed at efficient model training. These methods are organized around four main strategies: (1) Uncertainty-based approaches (e.g., Entropy [261], BALD [78]) select samples for which the model is least confident; (2) Representative-based methods (e.g., clustering [268], density estimation [138]) focus on covering the data distribution effectively by selecting central samples; (3) Influence-based techniques (e.g., gradient-based selection [251], predicted loss [271]) prioritize inputs that are expected to cause significant model changes upon retraining; and (4) Hybrid strategies (e.g., Cluster-Margin [48]) aim to combine uncertainty and diversity to improve selection robustness. Recent surveys [150] pointed out that such DAL techniques can be adapted for retraining or fine-tuning deep learning models, offering label-efficient ways to improve model performance under limited annotation budgets.

Finding 3: *In the SE community, testing techniques that effectively identify faulty behaviors in deep learning models can often be repurposed for targeted retraining, revealing their cross-task adaptability. In the ML community, deep active learning methods originally developed for efficient training data labeling can be leveraged for retraining. Important categories include uncertainty-based sampling (e.g., entropy or margin-based selection), representative-based sampling (e.g., clustering or density estimation), influence-based sampling (e.g., gradient or loss impact analysis), and hybrid approaches that combine multiple strategies to balance informativeness and diversity.*

5.4.4 *Comparative Analysis and Insights.*

We conduct a comparative analysis to reveal important differences in the applicability, efficiency, and detection effectiveness of different types of test prioritization methods (e.g., coverage-based, confidence-based, and learning-based). Table 2 summarizes the key characteristics across these categories.

Table 2. Comparison of Fault Detection Methods

Method Type	Model Access	Interpretable	Generalizability	Cost	Strengths	Weaknesses
Coverage-based	White-box	High	Medium	Medium	Structured exploration and test guidance	Not applicable in black-box settings
Confidence-based	Black-box	Medium	High	Low	Efficient and widely applicable	Solely relies on model confidence calibration
Surprise-based	White-box	Medium	High	High	Good at finding novel / boundary cases	High computation and data dependency
Learning-based	Mixed	Variable	High	High	Learns complex detection patterns; effective	Requires model training and feature engineering; sometimes less efficient
Others	Varies	Varies	Varies	Varies	Effective in domain-specific scenarios	Limited generality

The comparative analysis in Table 2 highlights that different fault detection techniques offer varying trade-offs across interpretability, generalizability, efficiency, and applicability. Coverage-based and surprise-based methods typically require white-box access and provide relatively better interpretability, but they are less generalizable and not suitable in black-box deployment scenarios. Confidence-based methods, on the other hand, operate in black-box settings and are highly efficient. This is because they rely solely on the model's output probabilities (e.g., softmax scores) without requiring access to internal structures or additional computation-intensive procedures.

Learning-based techniques show strong generalizability and are capable of capturing complex detection patterns across domains, but they demand more computational cost and typically rely on auxiliary training or feature engineering. This is because the ranking models used for fault detection are typically trained on large feature representations extracted from DNNs, which requires additional computation time. For example, the average time of the learning-based PRIMA [254] approach is around 10 minutes. Other domain-specific methods may offer practical benefits in narrow settings but lack broader applicability.

Finding 4: *Coverage-based techniques enable interpretable analysis through internal activation patterns. Confidence-based methods offer high efficiency and black-box compatibility. Surprise-based approaches are effective at identifying boundary cases. Learning-based techniques demonstrate strong effectiveness by capturing complex fault patterns across diverse domains.*

6 FAIRNESS TECHNIQUES

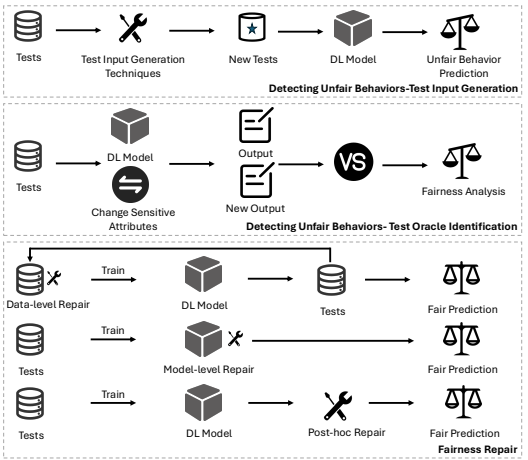


Fig. 8. Workflow of fairness testing techniques

Fairness refers to the "absence of any prejudice or favoritism toward an individual or a group based on their inherent or acquired characteristics" [178]. Early works focused on defining fairness notions and formalizing bias metrics, such as demographic parity, equality of opportunity, and counterfactual fairness [105]. Since 2014, several workshops and conferences specialized in fairness for machine learning, such as FAT/ML, AIES, and FAccT. With the large-scale deployment of Computer Vision (CV) and Natural Language Processing (NLP) models [20, 25], researchers began to investigate fairness through empirical bias detection techniques. For example, Buolamwini *et al.* [25] examined commercial facial analysis systems in computer vision and revealed that these models exhibited significant demographic disparities, especially for individuals with darker skin tones and those labeled as female. In the context of NLP, Bolukbasi *et al.* [20] focused on word embeddings commonly used in language-based recommendation and search systems, and found that such models encoded and amplified gender stereotypes.

After 2020, the rise of multimodal models introduced new fairness challenges [21]. The widespread use of large language models such as the GPT series has led to more complex and less interpretable forms of bias propagation [155]. Current research trends focus on evaluating model fairness in real-world applications, integrating fairness into the training pipeline, and mitigating bias without compromising performance [190]. This section surveys existing fairness-oriented research from two

perspectives: unfairness detection and fairness repair. Figure 8 presents the workflow of fairness testing techniques.

6.1 Techniques for Detecting Unfair Behaviors

These techniques aim to detect unfair behaviors of deep learning systems [43]. The most widely used and classic approach to date is test input generation, which aims to expose unfair behaviors in deep learning models by automatically generating inputs that reveal disparities across sensitive attributes. The mainstream categories are surveyed as follows.

1) Random Generation One of the earliest lines of work in this space focuses on randomized input generation. Galhotra *et al.* [79] (FSE 2017) proposed Themis, which employs randomized input generation combined with systematic pruning and statistical testing to identify causal discrimination. Cases where a system's decision changes as a result of modifying a sensitive attribute, indicating that the attribute has a causal influence on the outcome. Building on this idea, Angell *et al.* [12] (ESEC/FSE 2018) extended the original Themis prototype, presenting Themis v2.0, which is an open-source framework for automatically generating test suites to detect both group and causal discrimination.

2) Search-based Generation Beyond random testing, another popular direction in this area is search-based generation. Compared to random methods, it is more effective and efficient in detecting unfairness. One representative work is Aequitas proposed by Udeshi *et al.* [238] (ASE 2018), which detects individual fairness violations in black-box models through a two-phase search strategy combining global and local exploration. Subsequently, Zhang *et al.* [285] (ICSE 2020) proposed ADF (Adversarial Discrimination Finder), which leverages gradient information to generate individual discriminatory inputs through a two-phase adversarial sampling strategy, thereby efficiently identifying fairness violations in deep neural networks by targeting regions near decision boundaries where such violations are more likely to occur.

Zheng *et al.* [302] (ICSE 2022) extends the idea by utilizing internal model information, proposing NeuronFair for interpreting and mitigating discrimination in deep neural networks by identifying biased neurons. From an information-theoretic perspective, Monjezi *et al.* [183] (ICSE 2023) proposed DICE. DICE introduces a quantitative notion of individual discrimination (QID), which measures the extent to which protected attributes influence model decisions, using entropy-based metrics. It leverages causal analysis to localize neurons responsible for fairness violations.

3) GAN-based Generation Apart from the aforementioned efforts in the software engineering community, the machine learning field has also proposed a surge of research on fairness, among which GAN-based generation represents one of the most popular approaches. For instance, Dash *et al.* [56] (WACV 2022) ImageCFGen, which combines structural causal models (SCMs) with a GAN-based inference framework. The method generates counterfactual images where specific sensitive attributes are altered while trying to keep all other aspects unchanged. By comparing a model's predictions on the original and counterfactual images, the framework enables the detection of potential biases with respect to those sensitive attributes.

Following this direction, Algaba *et al.* [9] proposed LUCID-GAN, which generates canonical inputs using a conditional generative model. LUCID-GAN samples realistic inputs conditioned on model predictions, enabling fairness evaluation even for non-differentiable black-box models. By comparing the distributions of protected features and proxy features, LUCID-GAN reveals patterns of potential direct, proxy, and intersectional discrimination.

4) Template-based Generation While generative and search-based methods offer flexibility in exploring the input space, another line of research adopts a more controlled and interpretable approach through template-based generation. One early example in this direction is the work of Vig *et al.* [239] (NeurIPS 2020), who proposed an interpretation framework based on causal mediation

analysis to investigate gender bias in language models. Their approach involves constructing templated sentences and systematically varying gender-related terms to conduct controlled interventions. By evaluating the model's responses to these interventions, the method quantifies the extent to which specific internal components (such as individual neurons or attention heads) mediate and transmit gender bias.

Extending template-based testing to the conversational domain, Wan *et al.* [242] (ESEC/FSE 2023) proposed BiasAsker for identifying social biases in conversational AI systems. BiasAsker adopts a template-based question generation method, where predefined linguistic templates (e.g., yes-no questions, choice questions, and wh-questions) are instantiated using combinations of 841 social groups and 8,110 biased properties. Zhang *et al.* [283] (TOSEM 2023) introduced TestSGD, which leverages a template-based mechanism to identify group discrimination in neural networks. TestSGD constructs interpretable templates in the form of rule sets (i.e., logical conjunctions over sensitive attributes) that define candidate subgroups. These rule sets are used to partition the input space and evaluate disparities in model predictions.

5) Verification-based Generation While most existing fairness testing approaches rely on input generation and empirical analysis, a less-explored but promising direction is to leverage formal verification techniques for fairness assessment. Biswas and Rajan [18] (ICSE 2023) proposed Fairify, an SMT-based framework for verifying individual fairness in neural networks. Fairify partitions the input space and statically prunes inactive neurons using interval analysis and heuristic profiling, significantly reducing the complexity of the verification problem. It then encodes fairness properties as logical constraints over duplicated network instances and leverages an SMT solver to either certify fairness or generate counterexamples.

6) Other Methods Beyond the major categories outlined above, several innovative approaches have emerged that explore alternative perspectives or modalities for fairness testing. Fryer *et al.* [77] used large language models to generate counterfactual text pairs that differ only in sensitive attributes, enabling effective probing of counterfactual fairness in text classifiers. Soremekun *et al.* [219] (TSE 2022) proposed ASTRAEA, a grammar-based framework that uses metamorphic relations to generate discriminatory NLP inputs and localize fairness violations for targeted testing.

Among the various approaches for detecting unfair behaviors in deep learning systems, search-based test input generation has emerged as the most effective and widely adopted strategy. This is supported by its consistent presence in top SE venues (e.g., Aequitas [238], ADF [285], NeuronFair [302], and DICE [183]) and its relative frequency (4 out of 12 surveyed papers). These techniques leverage optimization-based strategies such as gradient search and information gain to generate inputs near decision boundaries—locations where fairness violations are most likely to occur. Compared to earlier random generation methods (e.g., Themis), search-based methods offer higher detection rates and better scalability across domains. This trend suggests that optimization-guided strategies are becoming the de facto standard for fairness testing in DL systems.

6.2 Fairness Repair Techniques

Fairness repair techniques aim to mitigate biases in neural networks by addressing imbalances and spurious correlations related to sensitive attributes in data. Below, we survey existing research from three perspectives: data-level repair (before training), model-level repair (during training) and post-hoc repair (after training).

6.2.1 Data-level Repair.

Training data plays a crucial role in shaping model behavior, as biases or imbalances in the data distribution can be directly learned and amplified by neural networks. Consequently, improving the fairness of deep learning systems typically begins with addressing issues in the data itself. Data-level

repair techniques modify or augment the training data distribution (typically through resampling, data generation, or perturbation) to mitigate bias before model training. As a model-agnostic and easily deployable strategy, this class of techniques has seen widespread adoption in both research and practice [190].

1) Re-sampling As one of the most intuitive and widely adopted fairness repair strategies, Re-sampling reduces model bias toward majority groups by redistributing the number of samples across different sensitive attribute groups in the training dataset. In the ***ML community***, significant efforts have been devoted to mitigating bias at the data level, as data imbalance and representation bias are widely recognized as primary sources of unfairness in model predictions. Li *et al.* [157] (CVPR 2019) proposed REPAIR, which focuses on reducing representation bias in datasets. By assigning lower weights to examples that are easily classified under a specific representation, REPAIR performs bias-aware resampling to construct more balanced datasets. Building on the idea of balancing sensitive attributes, Cao *et al.* [26] (WACV 2022) proposed a fairness-aware age prediction framework to curate training data. They re-sample from a combined data pool to ensure uniform representation across age, gender, and ethnicity by selecting equal numbers of samples for each sensitive feature combination.

Meanwhile, Tanjim *et al.* [228] (WACV 2022) extended resampling techniques to generative tasks by proposing a novel task of high-resolution image generation conditioned on multiple sensitive attributes (e.g., occupation, race, and gender). They introduced a resampling strategy that enforces a uniform distribution over attribute combinations during training. Specifically, rare classes are over-sampled such that each attribute combination is equally likely to be selected in mini-batches.

In the ***SE community***, unlike the ML community's focus on rebalancing data distributions, researchers emphasize data debugging (i.e., identifying and removing bias). A representative work is MAAT proposed by Chen *et al.* [42] (ESEC/FSE 2022), which constructs a fairness-oriented model by debugging the training data from the "We're All Equal" perspective to identify and remove bias-inducing instances. MAAT then combines this model with a performance-oriented one through weighted fusion of their probabilistic outputs, forming an ensemble decision model that balances fairness and performance.

2) GAN-based augmentation Another popular approach is to leverage Generative Adversarial Networks (GANs) to generate new samples for protected groups, thereby expanding the training data and reducing model bias toward majority groups. In the ***ML community***, an early effort in this direction is by Yucer *et al.* [275] (CVPR 2020), who leverages CycleGAN-based adversarial image-to-image translation to mitigate racial bias in face recognition. Their method generates racially transformed versions of individual face images while preserving identity-related characteristics. Ramaswamy *et al.* [197] (CVPR 2021) subsequently tackled a different challenge: reducing the unintended correlation between protected attributes (e.g., gender, age) and target labels in visual recognition. Rather than altering images in pixel space, their approach operates in the GAN's latent space by perturbing latent vectors to synthesize image pairs that share the same label but differ only in the protected attribute. This enables a more targeted de-biasing mechanism that aims to disentangle protected attributes from prediction outcomes.

While prior methods enforce fairness by controlling the content of generated samples (e.g., perturbing latent vectors to decorrelate protected and target attributes), the recent work IGGAN [294] (WACV 2024) instead targets the adversarial optimization process. It introduces an information gap by applying heterogeneous data augmentation across multiple discriminators, ensuring that all players (generator and discriminators) operate under incomplete and asymmetric information.

6.2.2 Model-level Repair.

While data-level repair methods focus on modifying the training data to mitigate bias before model learning begins, model-level repair techniques intervene during the learning process itself.
1) Adversarial Debiasing Among various model-level fairness approaches, adversarial debiasing has emerged as one of the most influential and widely adopted techniques. In the ***ML community***, a foundational approach is proposed by Zhang *et al.* [278] (AIES 2018), who introduced a general adversarial debiasing framework that jointly trains a predictor and a discriminator to mitigate undesirable demographic biases in deep learning models. The predictor learns to perform the primary task (e.g., classification), while the discriminator attempts to infer protected attributes such as gender or zip code from the predictor's output.

Extending adversarial debiasing to the unsupervised setting, Li *et al.* [151] (CVPR 2020) propose Deep Fair Clustering (DFC) for clustering high-dimensional visual data. By formulating a minimax game between an encoder and a discriminator, DFC learns feature representations that are both statistically independent of sensitive attributes and favorable for downstream clustering tasks. Recently, Zheng *et al.* [300] proposed an adversarial debiasing framework based on variational autoencoder to remove demographic bias from self-supervised 3D CT embeddings while preserving performance on clinical tasks.

In the ***SE community***, Tao *et al.* (ESEC/FSE 2022) proposed RULER [230], which identifies unfair behaviors by generating individual discriminatory instances (IDIs) through adversarial perturbations of sensitive and bounded non-sensitive attributes. These instances are then used in an iterative training process that effectively improves fairness while maintaining model accuracy.
2) Causal Approaches While adversarial debiasing methods aim to remove sensitive information from learned representations through adversarial training, another important line of model-level repair leverages causal inference to explicitly reason about and control the influence of sensitive attributes. In the ***ML community***, Kim *et al.* [134] pioneered this direction with DCEVAE (AAAI 2021), which disentangles causal and correlated factors in latent space to estimate counterfactual effects without requiring a full causal graph. Building on this idea of counterfactual reasoning, Dash *et al.* [56] (WACV 2022) extended causal modeling to image domains, proposing ImageCFGen, a GAN-based framework using structural causal models to generate counterfactual images by altering only target attributes, with a regularizer to enforce fair predictions. More recently, Schrouff *et al.* [209] (NeurIPS 2024) showed that fairness preprocessing is ineffective without causal grounding. They reframed data balancing as causal intervention and highlighted the need for causal graphs in mitigation. Extending further to dynamic environments,

While causal fairness in the ML community often centers on modeling latent structures or generating counterfactual examples via deep generative models, work in the ***SE community*** tends to emphasize practical integration with software systems, model debugging, and interpretability grounded in causal analysis. Chakraborty *et al.* [30] (ESEC/FSE 2020) proposed Fairway to detect and mitigate algorithmic bias. Fairway combines pre-processing and in-processing techniques, identifying biased training samples using a novel situation-testing-inspired method and removes these "ambiguous" data points to reduce bias. Building on this line, Zhang *et al.* [282] (ESEC/FSE 2022) used causal analysis to adaptively recommend fairness repair methods by estimating the influence of features and neurons on discrimination via Average Causal Effect.
3) Loss Functions These methods are designed to modify or augment objective functions, or to propose new loss functions that incorporate fairness constraints into the model training process.

An early and influential example is the work of Hendricks *et al.* [109] (ECCV 2018), who investigated gender bias amplification in image captioning and proposed the Equalizer model. Equalizer model includes two complementary loss functions: Appearance Confusion Loss (ACL), which encourages confusion when gender evidence is absent, and Confident Loss (Conf), which promotes confident gender predictions when evidence is present. While Hendricks *et al.*'s work targets

fairness in language generation, Tanjim *et al.* [228] (WACV 2022) focused on fairness in visual generation, proposing a bias mitigation framework tailored to high-resolution image synthesis. Their method addresses severe class imbalance in multi-attribute conditional generation by integrating weighted loss functions and gradient penalty regularization into the discriminator. In addition, they combine oversampling with data augmentation to enhance both the fidelity and diversity of generated samples, thereby promoting fairer outcomes in generative settings.

Recently, Khalili *et al.* [132] (ICML 2023) extended the paradigm of fairness-aware loss design by formalizing and operationalizing the fairness notion of Equalized Loss, which requires the expected loss to be approximately equal across sensitive groups. While prior works incorporated fairness into model training through task-specific or empirically motivated loss formulations, Khalili *et al.*'s approach provides a general framework applicable across classification and regression tasks.

6.2.3 Post-hoc Repair.

Unlike data-level or model-level interventions that operate during the training process, post-hoc repair techniques improve fairness after model training. These methods avoid retraining the entire model from scratch and instead introduce targeted modifications through techniques such as fine-tuning, knowledge distillation, or inference-time interventions. This research area has been primarily explored within the machine learning (ML) community, with relatively limited contributions from the software engineering (SE) literature. Therefore, for each of the methods discussed below, we provide only a selection of representative classic approaches and recent advancements, rather than attempting to offer a comprehensive survey.

1) Fine-tuning One prominent post-hoc strategy is fine-tuning. These approaches involve adjusting the parameters of a pre-trained model using additional fairness-aware data to reduce bias in its predictions. One representative example is FairTune [66] (ICLR 2024), a framework that enhances fairness in medical image analysis by optimizing parameter-efficient fine-tuning (PEFT) with fairness as the guiding objective. Recently, Zhao *et al.* [299] (CVPR 2025) proposed a data-driven fine-tuning method for fairness without protected labels, using synthetic images and selectively updating fairness-sensitive, domain-robust parameters.

2) Distillation In contrast to fine-tuning, another promising direction is distillation-based fairness, which focuses on transferring fairness from a teacher model to a student model by training the student to replicate the teacher's predictions on fairness-aware data. Jung *et al.* [125] (CVPR 2021) proposed MFD, a distillation-based framework that uses MMD regularization to align group-conditioned features with group-agnostic teacher representations, reducing bias without harming performance. Recently, Zhao *et al.* [297] (NeurIPS 2024) proposed ABSLD, which improves adversarially robust fairness by adaptively scaling soft label smoothness to reduce class-wise robustness disparities.

3) Prompting Compared to the two aforementioned methods, prompting-based methods offer a more lightweight alternative. These techniques operate solely at inference time, using carefully designed prompts to steer model outputs toward fairer behavior, without any parameter updates. One early example of this approach is proposed by Schick *et al.* [208] (TACL 2021), who introduce a prompting-based framework to mitigate bias in pretrained language models through two key mechanisms: self-diagnosis and self-debiasing. Self-diagnosis enables models to autonomously assess whether their outputs exhibit undesirable attributes, using only internal knowledge and natural language descriptions of these attributes. Based on this, self-debiasing leverages contrastive prompting during decoding to reduce the probability of generating biased content, without requiring any additional training data or parameter updates. Building on this line of work, Zayed *et al.* [277] (ACL 2024) critically examine the limitations of existing prompt-based fairness metrics and introduce CAIRO, which leverages prompt augmentation and selection to significantly enhance metric

Table 3. Summary of fairness testing techniques in the SE and ML communities

Task/Description	SE	ML
Detection	Search-based [183, 285, 302] Template-based [242, 283] Grammar-based [219] verification-based [18]	Random generation [12, 79] Search-based [238] GAN-based [9, 56] Template-based [239, 242] LLM-based [77]
Repair	Data debugging & balancing [42] Adversarial debiasing [230] Causal-based [30, 282]	Resampling [26, 157, 228] GAN-based augmentation [197, 275, 294] Adversarial debiasing [151, 278, 300] Causal-based debiasing [56, 134, 165, 209] Fairness-aware loss [109, 132, 228] Fine-tuning [66, 299] Knowledge distillation [125, 297] Prompt-based [208, 277]

agreement. This method demonstrates that careful prompt engineering can improve the consistency of fairness assessments across diverse metrics.

6.3 Analysis of Fairness Techniques

We summarize our analysis of fairness-related research by comparing the methodological focuses of the software engineering (SE) and machine learning (ML) communities across two key dimensions: *detection* and *repair* techniques. Table 3 presents a comparison of representative techniques from both communities, organized by core fairness tasks.

6.3.1 *Fairness Detection Techniques: ML vs SE.*

For detection techniques, the SE community consistently emphasizes structured and verifiable testing methods, grounded in classical software testing paradigms such as search-based input generation, grammar-based analysis, and formal verification. Representative SE works such as Aequitas [238], ADF [285], and NeuronFair [302] leverage input space optimization and model introspection to detect discriminatory behaviors. Moreover, recent advances such as TestSGD [283] and ASTRAEA [219] adapt rule-based partitioning and grammar-driven generation to identify group-level disparities with clear semantic interpretability and test oracle design, which aligns closely with SE goals of test traceability and fault localization.

In contrast, the ML community adopts a more generative and data-centric perspective. Techniques such as GAN-based test generation [9, 56] and LLM-based prompt perturbation [77] are commonly used to synthesize counterfactual inputs, typically operating in image/text. While these approaches excel at scale and visual diversity, they typically lack formal guarantees and fine-grained test control. This difference in methodology reflects the underlying priorities of the two communities: SE emphasizes test accountability, interpretability, and diagnosability, whereas ML prioritizes expressiveness and statistical generality. Notably, SE research has also introduced verification-based approaches such as Fairify [18], using SMT-based solvers to formally verify fairness properties, which is an approach virtually absent in ML fairness testing literature.

> **Finding 5:** *Fairness detection in the SE community is dominated by structured and explainable strategies (e.g., search-based input generation, template instantiation, and formal verification) that reflect core software engineering values of interpretability, diagnosability, and trustworthiness. In contrast, the ML community adopts more generative, data-oriented strategies (e.g., GAN-based or LLM-based test synthesis) that prioritize test diversity and coverage, often at the cost of traceability.*

6.3.2 *Fairness Repair Techniques: ML vs SE.*

For repair techniques, the SE community consistently adopts an engineering-driven perspective that emphasizes interpretability and modular repair, in contrast to the ML community's stronger focus on algorithmic optimization and statistical fairness criteria. Specifically, at the *data level*, SE researchers (e.g., MAAT [42], Fairway [30]) tend to focus on debugging the training set to remove bias-inducing instances. In contrast, the ML literature emphasizes rebalancing data distributions through resampling [157], synthetic augmentation [228], or latent-space manipulation via GANs [197].

At the *model level*, the SE community prioritizes actionable feedback and diagnosis, exemplified by methods like RULER [230], which identifies and retrains on individual discriminatory instances (IDIs), or adaptive frameworks [282] that dynamically select fairness strategies based on causal influence analysis. In contrast, ML methods frequently pursue general-purpose frameworks based on adversarial debiasing [278] and fairness-aware loss functions [132], often without deep integration into system-level workflows.

Finally, while post-hoc fairness repair in ML covers fine-tuning, distillation, and prompting [66, 125, 208], such techniques are still underexplored in SE, where interpretability and cost-efficiency in software integration remain key concerns. This methodological divergence reflects the underlying goals of the two communities: whereas ML aims to maximize fairness metrics across standardized benchmarks, SE focuses on practical deployment, system robustness, and human understanding.

Finding 6: *The SE community consistently prioritizes fairness repair techniques that offer high interpretability and actionable insights, often grounded in causal reasoning or debugging-based workflows. These methods are closely aligned with software engineering goals such as traceability and modularity with testing and verification. In contrast, the ML community adopts a broader but less interpretable repair strategies focused on statistical fairness and algorithmic generality.*

7 INTERPRETABILITY TECHNIQUES

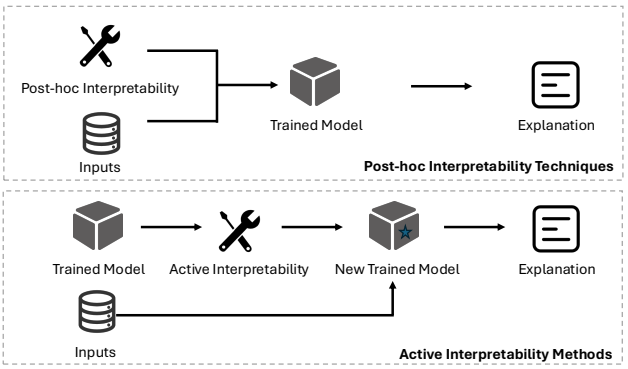


Fig. 9. Workflow of interpretability-oriented testing techniques

Interpretability techniques aim to reveal the internal logic, decision pathways, or influential features of a model, thereby improving trust and transparency. As machine learning (ML) systems become increasingly integrated into high-stakes domains (e.g., healthcare [220], finance [227], and autonomous driving [276]), ensuring their interpretability has emerged as a critical concern for both developers and stakeholders. For example, in the healthcare domain, interpretable models are essential for enabling clinicians to validate machine-generated diagnoses, and ensure that AI-assisted decisions align with medical guidelines and ethical standards [220].

Interpretability techniques can be broadly categorized into post-hoc interpretability and active interpretability. Post-hoc interpretability refers to methods that explain model behavior after the model has been trained, typically without modifying its internal structure. Active interpretability, on the other hand, involves designing or training models with interpretability in mind from the outset, often by imposing architectural constraints or incorporating interpretability objectives during training. Figure 9 exhibits the workflow of interpretability-oriented testing techniques.

7.1 Post-hoc Interpretability Techniques

Post-hoc interpretation techniques are currently the more mainstream and widely adopted approach. They aim to explain trained neural networks without modifying their architecture or training procedures. These methods are particularly valuable when working with complex or black-box models, such as deep neural networks, where direct inspection of internal logic is challenging or even infeasible. By providing insights into what the model has learned and why it makes certain predictions, post-hoc interpretability serves as a crucial tool for validating, debugging, and trusting ML systems after deployment. The mainstream approaches include rule-based explanations, attribution-based methods, hidden semantics interpretation, and example-based explanations.

7.1.1 Attribution-based Explanation.

Attribution-based methods are widely adopted in both academia and industry, as they provide intuitive feature-level importance scores applicable across domains and model architectures. Specifically, Attribution-based explanation assigns importance scores to input features based on their contribution to the model's output. In the ***ML community***, a representative example of such methods is LIME, introduced by Ribeiro *et al.* [200] (KDD 2016). LIME is designed to explain the predictions of black-box classifiers. It approximates the complex model locally around a specific instance using a simple, interpretable model, thus providing human-understandable explanations. LIME is widely regarded as one of the landmark works in the field. Building on this foundation, recent work has sought to better understand and evaluate the behavior of attribution methods across different settings. Duan *et al.* [65] (ECCV 2024) present Meta-Rank, an open-source benchmark for attribution-based explanation methods in deep neural networks. Their study highlights the inconsistency of evaluation results across datasets, models, and protocols, and introduces a unified metric, Meta-Rank, which aggregates performance rankings across diverse settings into a comprehensive leaderboard.

While the ML community largely focuses on general tasks like image or text classification, the SE community mainly targets deep learning-based code models, aiming to address domain-specific challenges such as structural semantics and robustness. Li *et al.* [162] (ASE 2023) proposed Robin for deep learning-based code classifiers. Robin introduces a hybrid interpreter-approximator architecture and leverages adversarial training and mixup-based data augmentation. This design improves the fidelity and robustness of explanations across various code classification tasks. Recently, Chen *et al.* [38] (FSE 2024) proposed DeciX, a model-agnostic method for code generation that builds a dependency-aware causal graph to explain predictions by capturing both input-output and output-output token relationships.

In the area of attribution-based explanation, the ML and SE communities demonstrate different methodological focuses. The ***ML community*** has primarily focused on building general-purpose explanation frameworks (e.g., LIME [200]), targeting tasks such as image and text classification. These methods are often evaluated based on their usability, approximation fidelity, and general applicability. Recent work such as Meta-Rank [65] further emphasizes benchmarking and cross-dataset consistency, reflecting the community's interest in explanation methods that are broadly applicable and statistically comparable across models and domains. In contrast, the SE community

concentrates on applying and adapting attribution techniques to the unique challenges of deep learning models for code, which aligns well with the core focus of software engineering. These models typically involve structurally complex and syntactically rich inputs and require explanations that are consistent with programming language semantics.

> **Finding 7:** *Attribution-based explanation research in the SE community focuses on domain-specific challenges in code-related tasks. In contrast, ML research emphasizes general-purpose methods with broad applicability across domains and focuses on standardized evaluation and benchmarking.*

7.1.2 Rule-based Explanation.

In contrast to attribution-based methods that focus on feature-level importance, rule-based interpretability techniques aim to approximate a model's behavior through explicit, human-readable decision logic. Specifically, these methods explain the behavior of a trained black-box model by extracting human-readable decision rules (such as decision trees or if-then rules) to approximate the model's overall logic. A representative example is Anchors, introduced by Ribeiro *et al.* [201] (AAAI 2018). Anchors are high-precision if-then rules that capture locally sufficient conditions for a model's prediction. Compared to attribution-based methods like LIME [200], Anchors provide better coverage guarantees and more precise interpretability, particularly for complex models such as deep neural networks. Lopardo *et al.* [171] (AISTATS 2023) conducted the first theoretical analysis of Anchors on text data. They theoretically analyzed its behavior on interpretable models such as if-then rules and linear classifiers, and empirically investigated neural networks. Their findings reveal a strong connection between the selected anchors and model parameters such as partial derivatives. Expanding on local explanations, Wu *et al.* [257] (AAAI 2020) introduced regional tree regularization, which enhances interpretability by enabling a deep neural network's behavior to be approximated by simple decision trees within predefined input regions. This promotes human-simulable logic in the model's decision process, enabling domain experts to better understand model predictions.

In the SE domain, rule-based explanations have been adopted to enhance both trust and debuggability of deep learning systems. Cito *et al.* [46] (ESEC/FSE 2021) proposed a model-agnostic technique that employs rule induction to globally explain model mispredictions. Their approach identifies interpretable rule-based patterns over input features that are strongly correlated with prediction errors, thereby enabling systematic debugging of DL models used in software engineering tasks. More recently, Ahmed *et al.* [5] (ICSE 2024) proposed DeepInfer, which assesses DNN prediction trustworthiness by inferring input preconditions using a semantic abstraction based on weakest precondition reasoning.

For rule-based explanation, the ML community primarily develops general-purpose rule extraction techniques that approximate a model's behavior using human-readable logic such as if–then rules or decision trees. Classical methods such as Anchors [201] offer high-precision local rules capturing sufficient conditions for model predictions, while follow-up studies (e.g., Lopardo et al. [171]) provide theoretical analyses connecting anchor selection to model characteristics such as partial derivatives. Other ML efforts, such as regional tree regularization [257], focus on improving interpretability by approximating neural network behavior with simple decision trees within predefined subspaces. Overall, ML research emphasizes general applicability and theoretical grounding across a variety of prediction tasks. In contrast, the SE community focuses on leveraging rule-based explanations to enhance trustworthiness and debuggability in deep learning systems used for software engineering tasks. SE models often operate in domains that require strong guarantees, making rule-based reasoning particularly valuable. For example, Cito et al. [46] employ rule induction to highlight input patterns associated with mispredictions, enabling systematic debugging of

DL models. DeepInfer [5] introduces rule-based abstractions grounded in weakest-precondition semantics to infer data constraints under which model predictions remain trustworthy. These SE-oriented approaches emphasize semantic alignment and developer interpretability with formal reasoning, reflecting the community's focus on actionable and verifiable explanations.

> **Finding 8:** *Rule-based explanation research in the SE community prioritizes actionable and verifiable reasoning that supports trustworthiness and debugging, whereas the ML community focuses on general-purpose and theoretically grounded rule extraction for broad interpretability.*

7.1.3 Hidden Semantics Interpretation.

In contrast to attribution- and rule-based methods that focus on input-level signals or explicit logic approximations, hidden semantics interpretation seeks to reveal the internal concepts learned by deep models by analyzing the behavior of their hidden neurons. Specifically, it identifies neurons that consistently respond to interpretable visual features, such as object parts (e.g., animal heads) or geometric shapes (e.g., wheels), thereby providing insights into the internal representations learned by the model. In the ***ML community***, Dalvi *et al.* [53] (AAAI 2019) introduced supervised (Linguistic Correlation Analysis) and unsupervised (Cross-model Correlation Analysis) techniques to identify neurons that capture specific linguistic properties or play a crucial role in model behavior. Their work advances post-hoc interpretability by enabling fine-grained analysis of how language information is distributed and localized across neurons in deep learning models.

In the ***SE community***, Gill *et al.* [87] (ICSE 2025) introduced TraceFL for federated learning (FL). TraceFL operates at the central aggregator and leverages the concept of neuron provenance. This fine-grained mechanism captures the flow of information from client models to the fused global model by identifying influential neurons activated during inference and mapping them back to the originating client neurons. By computing neuron-level gradients and contributions, TraceFL constructs an end-to-end provenance graph that quantifies each client's responsibility for a given prediction.

While both the ML and SE communities leverage hidden semantics interpretation to analyze internal neuron behavior, their emphases reflect domain-specific needs and challenges. In the ML community, the focus lies in understanding how individual neurons encode general linguistic or semantic properties within deep NLP models. Techniques such as those proposed by Dalvi *et al.* [53] aim to localize interpretable linguistic patterns, thereby enhancing transparency in model predictions. In contrast, the SE community adopts a more system-level perspective, as exemplified by TraceFL [87], which emphasizes the traceability and accountability of neuron-level contributions in distributed learning settings. Rather than explaining what a neuron represents, the goal is to quantify where it comes from (i.e., which client model contributed to a specific activation), thus supporting debugging and trust assessment in federated software systems.

> **Finding 9:** *While the ML community uses hidden semantics interpretation to understand what internal neurons represent, the SE community focuses on tracing where neuron activations come from to support debugging and build trust in distributed or federated learning systems.*

7.1.4 Explanation by Example.

Explanation by Example is one category of interpretability methods but is not considered mainstream [292]. It explains a model's prediction by referring to similar training examples that most influenced the decision. Currently, only the ML community has a small number of related papers, so we did not conduct a comparison between the SE and ML communities. A representative method in this category is Influence Functions, proposed by Koh and Liang [141] (ICML 2017). Influence

Functions estimates the impact of each training instance on a given prediction by approximating how the model's parameters would change if that instance were upweighted or removed. This allows users to trace predictions back to the most influential training examples, thereby providing clear, example-based explanations of model behavior.

7.2 Active Interpretability Methods

In contrast to post-hoc approaches that generate explanations after model training, active interpretability methods aim to enhance interpretability by integrating explanation mechanisms directly into the training process or model architecture. These methods are not yet mainstream and have only been explored in the ML community. Therefore, we did not conduct a comparative analysis between the SE and ML communities.

1) Decision tree A representative active interpretability approach is Neural-Backed Decision Trees (NBDT), proposed by Wan *et al.* [241] (ICLR 2021). NBDT replaces the final linear classification layer of a neural network with a differentiable sequence of decision rules, enhancing both accuracy and interpretability.

2) Generalized additive model (GAM) Another line of work draws inspiration from classical statistical models to design neural architectures that retain functional transparency. Agarwal *et al.* [4] (NeurIPS 2021) introduced Neural Additive Models (NAMs), a class of interpretable neural networks that combine the functional transparency of generalized additive models (GAMs) with the expressive power and scalability of deep learning. By structuring the model as a linear combination of feature-specific subnetworks, NAMs offer precise per-feature interpretability via learned shape functions, while retaining competitive predictive performance.

3) Model optimization Beyond architectural design, another approach integrates interpretability objectives directly into the training process. Ross *et al.* [204] (IJCAI 2017) proposed to incorporate model explanations into the training objective by regularizing input gradients. This approach encourages models not only to make accurate predictions but also to make their decision logic consistent with human domain knowledge. By constraining explanations during training, the method improves generalization under distributional shifts and enables the discovery of multiple models with comparable accuracy but qualitatively different explanations in an unsupervised setting.

4) Interpretable modules Another important direction focuses on introducing interpretable components into deep neural networks. Chen *et al.* [32] (NeurIPS 2019) proposed ProtoPNet, which integrates case-based reasoning into the model architecture to improve interpretability. By incorporating a prototype layer into the network, ProtoPNet enables predictions to be made through part-level comparisons between input image regions and learned prototypical parts from the training set. Each prototype is directly visualizable and corresponds to a latent patch of a real training image, making the decision process interpretable.

5) Other methods In addition to the representative methods in each category discussed above, recently, Arya *et al.* [14] (NeurIPS 2024) proposed B-cosification, which utilize lightweight architectural transformations for modifying existing deep neural networks to achieve interpretability. By replacing standard linear layers with B-cos transformation and removing bias terms, B-cosification enables faithful and human-interpretable explanations in the form of dynamic linear summaries. Moreover, Zhu *et al.* [308] (CVPR 2025) introduced non-parametric part prototype learning for interpretable image classification. This approach substantially enhances the interpretability of prototypical part networks (ProtoPNets) by clustering deep features extracted from self-supervised Vision Transformers to derive semantically part prototypes for each class. These prototypes could capture diverse object parts, enabling more intuitively understandable visual explanations.

7.3 Analysis of Interpretability Techniques

Building on the prior review of existing interpretability techniques, this section analyzes how these methods are adopted and adapted in the context of deep learning system testing. Specifically, we compare the interpretability-oriented efforts in the machine learning (ML) and software engineering (SE) communities, focusing on the methodological landscape and domain-specific research priorities.
Current Landscape: Interpretability remains an emerging focus in the SE community, with comparatively fewer related works. Existing SE studies primarily adapt rule-based, attribution-based, and hidden semantics methods for model understanding, typically tailored to software-specific domains such as code classification (e.g., Robin [162], DeciX [38]) or federated systems (e.g., TraceFL [87]). These approaches aim to support debugging and local trust assessment, aligning with traditional SE goals of fault localization and specification assurance.

On the other hand, the ML community advances a wider landscape of interpretability techniques across both post-hoc and active categories. Post-hoc efforts include landmark methods such as LIME [200], Anchors [201], and influence functions [141], complemented by recent benchmarking efforts (e.g., Meta-Rank [65]) that systematize evaluation. Active methods (e.g., Neural-backed Decision Trees [241], Neural Additive Models [4], and ProtoPNet [32]) directly optimize model structure or training objectives to embed interpretability in the learning process, typically achieving model-intrinsic transparency.

> **Finding 10:** *Interpretability-oriented research in the SE community remains in a developmental stage, with a smaller body of work focused on adapting classical rule-based, attribution-based, and semantic analysis methods to improve interpretability in DL systems. In contrast, the ML community exhibits rapidly evolving progress across a broader methodological landscape, advancing both post-hoc and active interpretability.*

Domain Priorities: SE research emphasizes interpretability techniques tailored to software tasks, prioritizing local explainability to support debugging, specification checking, and trust in distributed settings. Representative approaches include attribution-based techniques for robustness (Robin [162]), counterfactual explanations [47], and neuron provenance tracking (TraceFL [87]). By contrast, ML interpretability research tends to be domain-agnostic, aiming for general-purpose explanations applicable across vision, language, and multimodal tasks. Moreover, the ML community focuses on building scalable, model-agnostic explanation mechanisms and standardized evaluation frameworks to ensure interpretability across diverse architectures and deployment settings.

> **Finding 11:** *SE research prioritizes local interpretability for debugging, traceability, and specification assurance in domain-specific tasks. In contrast, ML research focuses on domain-agnostic interpretability, emphasizing generality and standardized evaluation across diverse tasks and architectures.*

8 ROBUSTNESS AND SECURITY TECHNIQUES

Testing techniques for robustness and security are designed to evaluate how deep learning systems respond to unexpected, accidental, or maliciously crafted inputs. We organize the existing work into four categories: 1) testing against natural or accidental perturbations (cf. Section 8.1), 2) testing against maliciously crafted attacks (cf. Section 8.2), 3) testing and measurement frameworks (cf. Section 8.3), and 4) robustness defense techniques (cf. Section 8.4). Figure 10 presents the workflow of robustness and security testing techniques, summarized from the perspectives of the four categories above. As there already exist numerous comprehensive and mature surveys on DNN robustness [6, 60, 64, 265, 286], we only highlight some representative works.

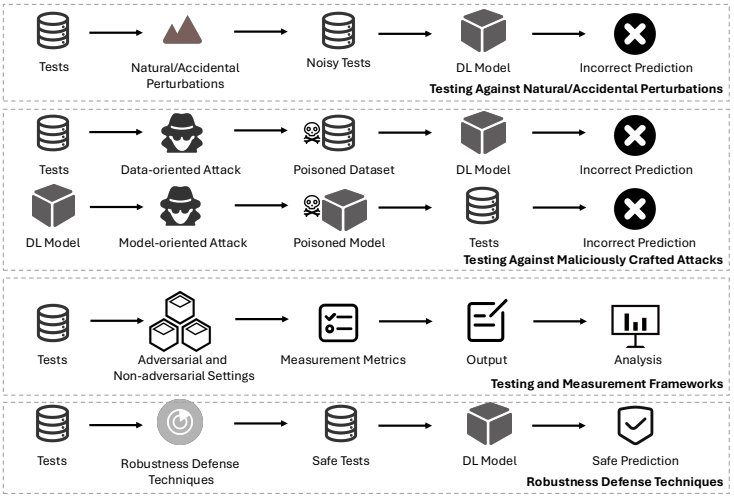


Fig. 10. Workflow of robustness and security testing techniques

8.1 Testing Against Natural/Accidental Perturbations

This category includes testing approaches that simulate real-world noise, corruptions, or distributional shifts. Examples include noise, blur, brightness shifts, and background variations. Such perturbations are not crafted with malicious intent but can still significantly affect model performance.

In the ***ML community***, robustness testing against natural and accidental perturbations has evolved from standardized benchmark creation to input-space and model-level robustness enhancement. Early efforts primarily focused on establishing reproducible and scalable evaluation frameworks to quantify the impact of real-world corruptions on deep neural networks. A significant milestone in this direction was the creation of ImageNet-C and ImageNet-P by Hendrycks *et al.* [111] (ICLR 2019), which provided a comprehensive suite of algorithmically generated corruptions (e.g., noise, blur, weather distortions) and perturbation sequences for evaluating model stability. These benchmarks quickly became the standard for measuring robustness against natural corruptions and inspired a wave of research on robustness-aware learning.

Building on these benchmarks, subsequent research has explored more advanced data augmentation techniques to improve model robustness. For instance, Hendrycks *et al.* [110] (ICCV 2021) proposed DeepAugment, a novel data augmentation technique that leverages image-to-image neural networks to generate semantically consistent but structurally diverse distortions by perturbing internal representations of deep models during the forward pass. These distortions simulate natural corruptions such as blur, texture variation, and style shifts.

In parallel, other works explored systematic robustness diagnostics to evaluate and understand the generalization limits of augmentation-based training. For instance, Mintun *et al.* [181] (NeurIPS 2021) focused on analyzing the effectiveness of various data augmentations in improving robustness against natural corruptions. They proposed the Minimal Sample Distance (MSD), a perceptual similarity metric that quantifies the alignment between augmentation schemes and test-time corruptions. Their work critically evaluates existing augmentations, revealing that many fail to generalize to unseen or perceptually dissimilar corruptions. To address this gap, they also introduced new corruption benchmarks tailored for such evaluations.

More recently, the research focus has extended into the model-internal space, exploring how noise injection at the parameter level can promote robustness. Trinh *et al.* [236] (NeurIPS 2024) proposed Data Augmentation via Multiplicative Perturbations (DAMP), which introduces multiplicative

weight noise during training to enhance the robustness of deep neural networks across diverse natural corruption types, such as noise (e.g., Gaussian, shot, impulse), blur (e.g., defocus, motion, zoom) and weather effects (e.g., snow, frost, fog).

In the ***SE community***, robustness testing against natural and accidental perturbations has gradually evolved from exploratory test case generation to robustness analysis techniques. Early efforts primarily focused on enhancing environmental diversity through synthetic data generation, aiming to simulate realistic scenarios that deep learning models may encounter post-deployment [194, 284]. For example, Zhang *et al.* [284] (ASE 2018) proposed DeepRoad, which employs Generative Adversarial Networks (GANs) to produce driving scenes under diverse weather conditions, including extreme scenarios such as heavy snow and hard rain. Zhong *et al.* [304] (FASE 2021) investigated the local robustness of deep neural networks under natural and accidental perturbations, such as rotations, translations, rain, and fog. They introduced the concept of neighbor accuracy to quantify per-input robustness and proposed both white-box (DeepRobust-W) and black-box (DeepRobust-B) methods to effectively localize non-robust inputs that could lead to incorrect model behavior.

As deep learning systems were increasingly applied to safety-critical domains such as autonomous driving, researchers began to repurpose traditional software testing methodologies (e.g., fuzzing and mutation testing) to expose hidden model weaknesses under real-world conditions [231]. Gao *et al.* [82] (ICSE 2020) proposed Sensei, which repurposes software testing methods, specifically mutation-based fuzz testing, for data augmentation of deep neural networks. Sensei employs genetic search to generate challenging input variants, thereby improving DNN robustness against naturally occurring environmental perturbations. More recent work has shifted toward fine-grained robustness diagnostics, focusing on quantifying and localizing model vulnerabilities at the level of individual inputs [246, 249, 304]. This evolution reflects the SE community's emphasis on testing efficiency, input-level fault detection, and practical integration into verification workflows. The following studies exemplify this progression in testing strategies.

8.2 Testing Against Maliciously Crafted Attacks

In contrast to natural or accidental perturbations, which arise from environmental or operational noise, this category targets the evaluation of model robustness against maliciously crafted inputs intentionally designed to induce model failures.

8.2.1 Data-oriented Attacks.

Data plays a central role in deep learning systems, serving as both the foundation for model training and the basis for reliable inference. The quality of training and testing data directly influences a model's behavior and generalization capabilities. However, this strong data dependence also makes deep learning models inherently vulnerable to data manipulation. Data-oriented attacks exploit this vulnerability by deliberately tampering with input data (either at training time or inference time) to induce misbehavior without modifying the model's architecture or learning algorithm.

(1) Adversarial Attacks Among data-oriented attacks, adversarial attacks have emerged as the most extensively studied class of techniques for evaluating the robustness of deep learning models at inference time. These attacks generate carefully crafted perturbations on input data to mislead the model into producing incorrect predictions, while keeping the perturbations imperceptible to humans. Common approaches include gradient-based methods such as the Fast Gradient Sign Method (FGSM) [91] and Projected Gradient Descent (PGD) [176], optimization-based attacks such as the Carlini and Wagner (C&W) attack [28], and black-box or ensemble-based methods such as AutoAttack [52] and Square Attack [11]. These representative approaches have become standard

baselines for adversarial robustness evaluation across a wide range of deep learning architectures and tasks.

Recent advances have further expanded this line of research toward improving the efficiency, transferability, and realism of adversarial examples. For instance, Abdollahpoorrostam *et al.* [2] (NeurIPS 2024) proposed SuperDeepFool, a parameter-free attack that generates minimal ℓ_2-norm perturbations through a projection-based alignment with the decision boundary, achieving competitive success rates at substantially lower computational cost. Guo *et al.* [103] (CVPR 2025) further introduced the Multi-Objective Set-based Attack (MOS Attack), which optimizes multiple surrogate loss functions simultaneously to discover synergistic perturbation patterns, yielding improved attack efficiency and generalization across models. For a broader taxonomy of adversarial attacks, we refer the reader to authoritative surveys [6, 17, 29, 198, 306], which offer comprehensive overviews of the field.

While adversarial attack research in the ***ML community*** largely focuses on perturbing inputs such as images, audio, or text to evaluate or degrade model performance across tasks like classification and detection, the ***SE community*** emphasizes attacks that target software-specific artifacts (e.g., source code, comments, programs) and software engineering tasks (e.g., code generation, bug detection). Zhou *et al.* [306] (TOSEM 2022) proposed ACCENT for deep learning-based code comment generation models. ACCENT crafts adversarial code snippets by substituting programmer-defined identifiers while preserving the code's syntactic correctness and semantic equivalence, thereby misleading models to produce incorrect or irrelevant comments. Zhang *et al.* [279] (ESEC/FSE 2023) proposed DeepRover, a fuzzing-based attack framework designed for assessing the robustness of DNNs in image classification tasks. DeepRover generates adversarial examples by introducing minor input perturbations that can mislead DNN decisions and has been shown to be more query-efficient and effective than prior blackbox attack methods. Huang *et al.* [117] (ICSE 2025) propose ITGen for deep code models. Unlike prior techniques that focus on replacing identifiers based on static importance scores or context, ITGen integrates feedback from failed attacks to refine its generation strategy. By employing a bit vector–based representation of code variants and coupling it with an enhanced Bayesian optimization framework, ITGen effectively mitigates local optima bias and improves search efficiency.

(2) Poisoning Attacks Compared to adversarial attacks, poisoning attacks have received relatively less attention but remain an important and emerging line of research. It differs fundamentally from adversarial attacks in that they corrupt the training process rather than the inference process. While adversarial attacks manipulate inputs at test time to cause mispredictions, poisoning attacks inject carefully crafted malicious data points into the training set to implant backdoors, bias decision boundaries, or degrade model performance over time.

In the ***ML community***, early research primarily focused on *backdoor poisoning* attacks, in which models are trained to perform normally on clean inputs, but deliberately misclassify inputs that contain attacker-specified triggers. For instance, Gu *et al.* [98] proposed BadNets, which demonstrated that inserting poisoned training data embedded with subtle triggers could reliably implant backdoors without degrading validation accuracy or changing the model architecture. Building on this, Chen *et al.* [40] introduced more practical targeted backdoor attacks under a realistic and constrained threat model, where the attacker lacks knowledge of the model and training set and can only inject a small number of poisoned examples. They proposed both instance-key and pattern-key attack strategies, showing high attack success rates even in low-resource settings.

Subsequent research expanded into *clean-label poisoning*, a subtler paradigm where the labels of poisoned examples remain correct, thus evading obvious manual inspection. Shafahi *et al.* [211] (NeurIPS 2018) showed that even without label tampering, poisoned samples could shift the model's

decision boundary toward targeted misclassifications. As poisoning attacks moved toward distributed and privacy-preserving settings, Fang *et al.* [74] (USENIX 2020) revealed that federated learning systems remain vulnerable to local poisoning attacks by malicious participants, even when using Byzantine-resilient aggregation. They highlighted the systemic risks of poisoning in collaborative training.

In parallel, defenses also began to emerge. For example, Xia *et al.* [259] (IJCAI 2022) proposed *ARGD*, a distillation-based defense that aligns attention-based feature correlations between teacher and student models, successfully removing backdoors while preserving performance on clean data. More recently, the scope of poisoning attacks has further expanded to dynamic learning settings. Abbasi *et al.* [1] (CVPR 2024) introduced *BrainWash*, a novel attack targeting continual learning systems. Unlike prior attacks aiming for misclassification, BrainWash induces *targeted forgetting* by reconstructing proxy data for past tasks through model inversion, and subtly poisoning new task data to erase prior knowledge. This highlights a shift in poisoning research from static misbehavior injection to long-term model manipulation across time.

The works above represent several influential poisoning attack techniques across different settings and threat models. For a more comprehensive overview of poisoning attacks and defenses (including taxonomy, methodologies, and recent trends), we recommend that readers refer to the corresponding surveys [45, 73, 233, 245].

While poisoning attacks in the ***ML community*** typically aim to inject misclassification behavior into models for image, text, or audio tasks (such as label flipping, backdoor triggering, or decision boundary manipulation), the ***SE community*** focuses on task-specific consequences that arise from poisoning software engineering artifacts. These artifacts include source code, code-comment pairs, or code repositories used to train models for generation, retrieval, or vulnerability detection. Rather than inducing generic mispredictions, SE-oriented poisoning attacks often aim to manipulate downstream behaviors such as code suggestion ranking, vulnerable pattern generation, or model-guided patching. For example, Wan *et al.* [243] (ESEC/FSE 2022) demonstrates that by poisoning a small fraction of the training corpus with carefully crafted malicious code snippets (containing triggers and baits), attackers can significantly manipulate search rankings, pushing vulnerable code into top results without degrading overall model performance. Cotroneo *et al.* [49] (ICPC 2024) proposed a targeted data poisoning strategy for NL-to-code models by injecting vulnerable code into training data without changing the natural language descriptions.

8.2.2 Model-oriented Attacks.

While data-oriented attacks manipulate the input data used for training or inference, another line of attacks directly targets the model itself. Attackers modify the model's architecture, parameters, or training process to inject malicious behavior [108].

In the ***ML community***, model-oriented attacks have primarily focused on embedding hidden backdoors or weakening robustness by tampering with model parameters. Early works in this area typically assumed strong attacker capabilities, such as full access to model architectures or training procedures, enabling direct injection of malicious behaviors into the model. Liu *et al.* [169] (NDSS 2018) proposed Neural Trojan Attacks, which embed hidden backdoors into neural networks, enabling malicious behavior to be triggered by specific inputs while maintaining normal performance. Ji *et al.* [123] (CCS 2018) proposed model-reuse attacks, where attackers subtly modify pre-trained models so that, after integration, host systems produce attacker-chosen outputs for certain inputs without affecting normal behavior on others.

Building upon these early explorations, subsequent research has increasingly shifted towards more stealthy and realistic threat models that require weaker assumptions, making attacks more feasible in practical deployment scenarios. Kurita *et al.* [145] (ACL 2020) reveal that pre-trained

models are vulnerable to weight poisoning attacks, where backdoors are injected into model weights and persist through fine-tuning. These attacks allow adversaries to control model outputs by inserting specific trigger keywords, without degrading clean task performance. Yu *et al.* [273] (ICML 2023) proposed adversarial parameter attacks that subtly perturb model parameters to preserve clean accuracy while severely reducing adversarial robustness.

Compared to the **ML community**, which adopts a model-centric and algorithm-level perspective focusing on the internal mechanisms of learning algorithms, the **SE community** approaches model-oriented attacks from a system-level and deployment-aware perspective. Researchers in SE focus on how ML models behave once they are integrated into complex software systems, such as mobile applications, cloud services, or edge devices. For example, Li *et al.* [156] (ICSE 2021) proposed DeepPayload, which injects malicious logic into compiled models by modifying their data-flow graphs without access to training data or retraining. Huang *et al.* [120] (ICSE 2021) investigated the feasibility of adversarial attacks on deep learning models embedded in real-world Android apps. They proposed a comprehensive attack pipeline that identifies fine-tuned on-device models by matching them with highly similar pre-trained models (e.g., from TensorFlow Hub), enabling effective adversarial attacks. Zhang *et al.* [288] (IEEE Trans. Comput., 2023) propose a reference-data-independent model poisoning attack, which enables adversaries to manipulate neural networks without access to the training dataset or data belonging to the target label.

When comparing the SE and ML communities, we also see that model-oriented attacks in the ML domain often assume *strong attacker capabilities*, such as full or partial access to training data, model parameters, or the training pipeline. These studies typically explore vulnerabilities such as backdoor injection, weight poisoning, and adversarial fine-tuning during model development. The attacks are primarily designed to compromise model integrity or robustness in controlled experimental settings. In contrast, in the SE community, the attacker's assumptions are often *weaker but more practical*, reflecting real-world constraints. For example, attacks may be carried out through reverse engineering of compiled models, tampering with model files in deployment environments, or manipulating software interfaces that trigger model predictions. The SE community places greater emphasis on the feasibility and stealthiness of attacks in production environments, even when access to training data or the ability to retrain models is not available.

8.3 Testing and Measurement Frameworks

This category covers testing frameworks and tools designed to support robustness evaluation. As both natural perturbations (e.g., noise, blur, occlusion) and adversarial attacks become increasingly sophisticated, the need for standardized, reproducible, and interpretable evaluation methodologies has become critical.

In the ML community, there has been a growing consensus around the necessity for formalized evaluation standards. Carlini *et al.* [27] critically reviewed common pitfalls in evaluating adversarial defenses and proposed foundational principles for robustness evaluation, including adaptive attacks, clearly defined threat models, and reproducible reporting practices. Their insights shaped much of the subsequent tooling and benchmarking efforts. Croce *et al.* [52] (ICML 2020) further operationalized these ideas by introducing AutoAttack, a fully automatic and parameter-free evaluation suite combining multiple strong adversarial attacks. This significantly reduced inconsistencies across studies and enabled reproducible comparison of robustness claims. Building on AutoAttack, the authors established RobustBench [50], an open-source leaderboard that tracks state-of-the-art adversarial robustness results across models and datasets, reinforcing community-wide transparency and benchmarking.

Expanding on robustness metrics, Guo *et al.* [101] proposed a comprehensive evaluation framework that integrates 23 diverse metrics (ranging from neuron coverage to perturbation imperceptibility) to analyze robustness from both model- and data-oriented dimensions. This multidimensional view reflects a trend in the ML community to move beyond simple adversarial accuracy as a robustness proxy. More recently, Simonetto *et al.* [216] (NeurIPS 2024) addressed the gap in robustness evaluation for tabular data, introducing TabularBench, a benchmark covering critical domains such as healthcare and finance. Meanwhile, Li *et al.* [158] (NeurIPS 2024) proposed GREAT Score, which estimates global robustness using generative models to approximate the data manifold, offering a non-attack-based perspective to measuring a model's resilience.

In contrast to the ***ML community***, which primarily evaluates robustness using standardized benchmarks and attack pipelines, the ***SE community*** places greater emphasis on the practical testability, diagnosability, and deploy-time behaviors of DNNs within software systems. SE approaches typically incorporate metrics and strategies inspired by software testing, runtime analysis, and verification. For example, Zhang *et al.* [287] (ICSE 2020) approached adversarial robustness from an uncertainty-guided testing perspective. They proposed generating test cases based on uncertainty metrics such as prediction confidence and variation ratio to uncover both rare benign and adversarial behaviors. This approach aligned with principles of boundary testing in traditional SE. Tian *et al.* [232] (ICSE 2020) introduced DeepInspect, which shifts the robustness analysis from individual inputs to class-level semantics. They designed a neuron activation probability matrix and a weight-based baseline to detect systematic confusion and bias patterns missed by per-sample evaluation.

From a practical engineering perspective, Wang *et al.* [246] (ICSE 2021) proposed RobOT, which not only detects vulnerabilities but also closes the loop with retraining. They introduced lightweight metrics strongly correlated with robustness to prioritize impactful test inputs, illustrating a tight integration between testing and model improvement. This is similar to test-driven development in SE.

Complementing these empirical testing approaches, Xue *et al.* [267] (ISSTA 2023) proposed DualApp, a formal verification framework that enhances robustness measurement through theoretical guarantees. By leveraging a dual-approximation method that constructs tighter bounds on activation functions, DualApp offers a more precise estimation of robustness margins. This line of work addresses a key limitation of empirical methods (lack of formal assurance) by bridging the gap between software verification and DNN safety analysis. Further expanding the scope of test input selection, Sun *et al.* [221] (TSE 2023) introduced a robust test selection (RTS) framework that prioritizes failure-revealing inputs. By integrating noise filtering, majority voting, and a probability tier matrix, RTS aims to uncover diverse and non-redundant failure patterns.

Overall, the ML and SE communities adopt fundamentally different perspectives in robustness evaluation. ML research emphasizes standardized benchmarks, reproducibility, and attack-driven comparisons, aiming to measure model robustness in controlled settings. In contrast, SE approaches embed robustness testing within the software engineering lifecycle, focusing on diagnosability, test-retrain integration, formal verification, and deployment-aware behavior. While ML-centric methods support generalizable benchmarking, SE-centric frameworks offer greater practical relevance and actionable guidance for improving model reliability in real-world systems. Bridging these two views (combining empirical effort with system-level validation) remains a critical direction for future robustness research.

8.4 Robustness Defense Techniques

Robustness defense techniques are designed to enhance the robustness of deep learning models.

8.4.1 Mitigation techniques.

Mitigation techniques focused on proactively improving the robustness of deep learning models against attacks.

(1) Adversarial Training Adversarial training is one of the most widely studied defense techniques in the ***ML community***, which improves model robustness by incorporating adversarial examples into the training process. The initial idea, introduced by Goodfellow *et al.* [92] through FGSM, demonstrated that injecting adversarial examples during training can improve model robustness. This was later formalized by Madry *et al.* [176] as a minimax optimization problem, establishing a principled foundation for robust training. Subsequent work, such as TRADES [280] (ICML 2019), introduced regularized loss functions to balance robustness and accuracy, while more recent studies [290] (ICML 2024) explored connections between input-perturbation-based and weight-perturbation-based training objectives (e.g., SAM). Despite its effectiveness, adversarial training remains computationally intensive and often overfits to specific attack types [198], motivating ongoing research on generalization, efficiency, and robustness verification. The above discussion highlights representative literature in adversarial training. For a broader and more systematic overview, we refer readers to the survey by Ren *et al.* [198].

In the ***SE community***, adversarial training has received relatively limited attention compared to the ML community. For example, Zhang *et al.* [293] (ASE 2022) proposed Dare, a model training framework which goes beyond traditional adversarial training by transforming classification models into isomorphic regression models. This transformation enables the model to capture small perturbations more sensitively through a fine-grained loss function.

(2) Pre-processing Pre-processing is another major line of defense in adversarial robustness, which aims to enhance the robustness of models against adversarial examples through input transformations. Most existing work on input pre-processing defenses has been conducted within the *ML community*, with rare contributions from the *SE community*. For example, Xu *et al.* [266] (NDSS 2018) proposed Feature Squeezing, which reduces the degrees of freedom of input samples through color bit-depth reduction and spatial smoothing, thereby making adversarial perturbations less effective. Song *et al.* [218] (ICLR 2018) proposed PixelDefend, which improves the robustness of classifiers against adversarial examples by slightly modifying the input image to move it back toward high-probability regions of the training data distribution. Gao *et al.* [83] (NeurIPS 2022) investigated the limitations of stochastic pre-processing defenses against adversarial examples. Their study demonstrated that many existing stochastic defenses incorporate insufficient randomness, rendering them vulnerable even to standard attacks without requiring advanced techniques.

(3) Ensemble Methods These methods are based on Ensemble Learning [62], which combines multiple models to solve a problem by aggregating their predictions. Early work by Tramèr *et al.* [235] (ICLR 2018) laid the foundation by introducing ensemble adversarial training, which mixes adversarial inputs from multiple sources during training. Later advancements such as iGAT [59] (NeurIPS 2023) and EED [126] (CVPR 2025) further refined this idea by strategically allocating adversarial examples and promoting structural diversity within ensembles, improving both white-box robustness and computational efficiency.

Ensemble methods are not currently considered a mainstream approach in adversarial defense, but they remain a representative line of research. Their limited adoption is primarily due to the following reasons: 1) High computational overhead. Combining multiple models increases storage and inference costs; 2) Limited effectiveness under white-box attacks. As noted by Tramèr *et al.* [235], ensemble defenses provide only marginal improvements against white-box adversaries; and 3) Scalability challenges. Achieving significant robustness gains often requires sufficient diversity among base models, which is difficult to realize in practice.

(4) Structural-based Methods In the context of software engineering research, there has been increasing interest in structure-based defenses, which aim to protect ML models by adjusting model architecture, controlling access or deployment mechanisms, or shaping input/output distributions to reduce adversarial vulnerability. Zhang *et al.* [295] (ICSE 2023) introduced FedSlice for crowdsourcing federated learning (CFL). FedSlice slices the server-side model into heterogeneous segments, distributing different slices to participants. This strategy prevents any single participant from accessing the full model, thereby mitigating threats such as adversarial, gradient leakage, and inference attacks.

8.4.2 Detection Techniques.

Unlike training-based or architecture-level defenses that aim to improve a model's robustness, detection techniques focus on identifying adversarial or malicious inputs at inference time, in order to reject them or to alert the system. Early efforts in the ***ML community*** primarily explored statistical differences between benign and adversarial inputs. Grosse *et al.* [95] introduced a statistical testing approach using Maximum Mean Discrepancy (MMD) and Energy Distance to differentiate input distributions. Xu *et al.* [266] (NDSS 2018) proposed Feature Squeezing, which applies input transformations (e.g., color depth reduction and smoothing) to amplify the detectability of adversarial perturbations. Around the same time, Meng and Chen [180] (CCS 2017) proposed MagNet, combining detectors and autoencoder-based reformers to both identify and recover adversarial examples through manifold projection.

More recently, in the context of text models, Nguyen *et al.* [185] introduced VoteTRANS, a training-free detection method that applies multiple input transformations and detects adversarial texts based on label inconsistency across these variants. Mumcu and Yilmaz [184] proposed a detection method based on feature regression across layers. By learning to predict deeper-layer features from earlier activations, their approach identifies adversarial examples through elevated prediction errors.

While the ***ML community*** emphasizes statistical patterns or internal feature behaviors for adversarial detection, the ***SE community*** focuses more on practical, testability-oriented strategies that leverage model behavior under mutations, pruning, or runtime analysis. Wang *et al.* [247] (ICSE 2019) integrated model mutation testing with statistical hypothesis testing for detection. The approach is based on the observation that adversarial examples are significantly more sensitive to model mutations compared to benign inputs. Chen *et al.* [41] (ASE 2021) proposed Graph-Guided Testing (GGT), which detects adversarial samples by generating diverse pruned models guided by graph-theoretic properties of the neural network architecture. More recently, Zhao *et al.* [298] (TOSEM 2024) introduced the A2D (Attack as Detection) framework, which repurposes adversarial attack cost as a robustness proxy to identify abnormal inputs, including adversarial, backdoored, and mislabeled examples. Peng *et al.* [192] (TOMM 2024) proposed AED-PADA, which improves generalization of adversarial example detectors by introducing Principal Adversarial Domain Adaptation. Their framework identifies principal adversarial domains (PADs) covering broad adversarial feature spaces and applies multi-source domain adaptation to enhance detection performance on unseen attacks.

Overall, adversarial detection techniques in the **ML community** tend to emphasize theoretical soundness, generalization across tasks, and alignment with statistical or representational properties of neural networks. For example, methods such as MagNet [180], VoteTRANS [185], and layer-wise feature regression [184] explore intrinsic data or feature consistency to distinguish adversarial inputs across modalities. In contrast, the **SE community** often adopts a more pragmatic perspective, emphasizing testability and deployment feasibility in operational environments. Techniques such as mutation-based detection [247] and graph-guided testing [41] leverage model behavior under

controlled perturbations or structural variations, aligning detection with well-established software testing paradigms. While ML methods push the boundary of detection accuracy and theoretical generality, SE methods contribute complementary insights by integrating robustness evaluation into practical, system-level testing workflows.

8.4.3 *Adaptive Test-time Defenses.*

Adaptive test-time defenses aim to dynamically adjust inputs or model components at inference time to mitigate adversarial effects, often without retraining. Compared to training-time defenses, these methods are typically more flexible but also incur additional runtime overhead due to their reliance on auxiliary modules.

In the **ML community**, recent efforts have explored both input purification and model adaptation strategies. Croce *et al.* [51] (ICML 2022) provided a systematic categorization of such methods and revealed that, despite their complexity, many adaptive defenses offer marginal gains over static ones under strong adversaries. To reduce overhead, Kulkarni and Weng [144] (ECCV 2024) proposed IG-Defense, a training-free and efficient method that leverages interpretability to identify and mask non-critical neurons during inference. Wang *et al.* [252] (CVPR 2025) focused on vision-language models and introduced TAPT, which adaptively tunes textual and visual prompts during inference to enhance zero-shot adversarial robustness.

By contrast, the ***SE community*** has conducted limited work in this area, and adaptive test-time defense remains an underexplored research direction. Huang *et al.* [118] (ISSTA 2022) proposed a robustness quantification framework based on ϵ-weakened robustness, which relaxes traditional binary definitions. Instead of requiring absolute correctness under all perturbations, their framework allows a bounded error margin, thereby making robustness evaluation more practical in real-world deployment scenarios.

8.5 Analysis of Robustness and Security Techniques

Below, we provide an overall analysis of this entire section. Table 4 summarizes representative robustness and security testing techniques from both the SE and ML communities, categorized into four areas: natural perturbations, malicious attacks, testing frameworks, and defense strategies. In the ***SE community***, many works focus on adapting traditional software testing methodologies (such as fuzzing, mutation testing, and formal verification) for evaluating robustness of deep learning systems under both natural and adversarial perturbations. For natural or accidental perturbations, approaches like fuzzing-based test augmentation [82] and robustness localization techniques [304] have been employed to expose hidden model vulnerabilities in realistic scenarios.

Against maliciously crafted inputs, the literature has proposed fuzzing-based adversarial example generation [279], adversarial code transformation [117, 306], and logic or model poisoning strategies [156, 288]. For measurement and evaluation, several SE frameworks introduce failure pattern discovery [221, 232], robustness indicators based on uncertainty and confidence [246, 287], and formal verification methods for provable robustness [267].

In the ***ML community***, robustness and security testing efforts concentrate more on data-centric and model-centric strategies that are broadly applicable across domains. For natural or accidental perturbations, benchmark-driven data augmentation methods (e.g., ImageNet-C [111]) and metrics-based robustness evaluations [181] are widely adopted. To assess vulnerability under adversarial conditions, ML research explores a variety of malicious attack types, including gradient-based [92, 176], optimization-based [28], and parameter perturbation attacks [273].

In terms of measurement and evaluation, the community emphasizes standardized platforms and scalable metrics, including AutoAttack [52], RobustBench [50], and the GREAT score [158]. To enhance robustness, ML researchers have developed a set of defense techniques, ranging from

Table 4. Summary of robustness and security testing techniques in the SE and ML communities

Category of Technology	SE	ML
Natural/Accidental Perturbations	GANs-based [284] Fuzzing-Based [82] Robustness localization [304]	Benchmarking [111] Data Augmentation [110, 236] Metrics [181]
Maliciously Crafted Attacks	Fuzzing-based attack [279] Adversarial Code [117, 306] Code poisoning [49, 243] Logic injection [156] Model poisoning [288]	Gradient-based [92, 176] Optimization-based [28, 103] Data poisoning [1, 98] Weight/parameter perturbation [123, 169, 273]
Testing and Measurement Frameworks	Failure Pattern Discovery [221, 232] Robustness Indicator [246, 287] Formal robustness verification [267]	Methodological foundations [27] Benchmarking [50, 52, 216] Metrics [101, 158]
Robustness Defense Techniques	Adversarial Training [293] Structural-based [295] Adversarial Detection [41, 192, 247, 298] Test-time Defenses [118]	Adversarial Training [92, 176, 280, 290] Pre-processing [83, 218, 266] Ensemble Defenses [59, 126, 235] Adversarial Detection [95, 180, 266] Test-time Defenses [51, 144, 252]

adversarial training [176], preprocessing [266], and ensemble methods [235], to adaptive test-time defenses [252].

Finding 12: *Robustness and security research in the SE community emphasizes system-level testing and verifiable analysis by adapting classical software engineering techniques (e.g., fuzzing, mutation testing, formal verification). In contrast, the ML community prioritizes data- and model-centric robustness strategies, along with standardized benchmarking and evaluation pipelines for cross-domain generalizability.*

9 CORRECTNESS TECHNIQUES

Correctness-Oriented Testing Techniques for deep learning testing refer to a set of techniques aimed at verifying and ensuring the correctness of DL models in terms of their predictions and behaviors, or their conformance to given specifications. It is important to note that some techniques (such as those for fault detection and accuracy estimation) are also related to ensuring correctness. However, since their primary objective is to improve testing efficiency by reducing labeling efforts or computational costs, they are discussed in Section 5 under efficiency-oriented testing. To avoid overlap between sections, this section focuses on correctness-oriented techniques that cannot be clearly categorized into other groups, such as those explicitly validating model behaviors using test oracles, metamorphic relations, or specification-based assertions. Figure 11 exhibits the workflow of correctness-oriented testing techniques.

9.1 Oracle-based/Oracle-Free Testing

In traditional software testing, test oracles serve as ground truth for determining whether program behavior is correct. However, in deep learning (DL) systems, where models operate on high-dimensional, probabilistic functions, oracle definition becomes inherently challenging [163]. To address this, the SE community has developed a range of *oracle-based* and *oracle-free* strategies, including differential testing, statistical proxies, and learned approximations.

1) Differential testing A widely adopted strategy is differential testing, which bypasses the need for ground-truth labels by cross-referencing outputs from multiple DL frameworks or models. For example, Pham *et al.* [193] (ICSE 2019) proposed CRADLE, which detects inconsistencies across major backends (e.g., TensorFlow vs. CNTK) under identical specifications. Wang *et al.* [253] (FSE 2020)

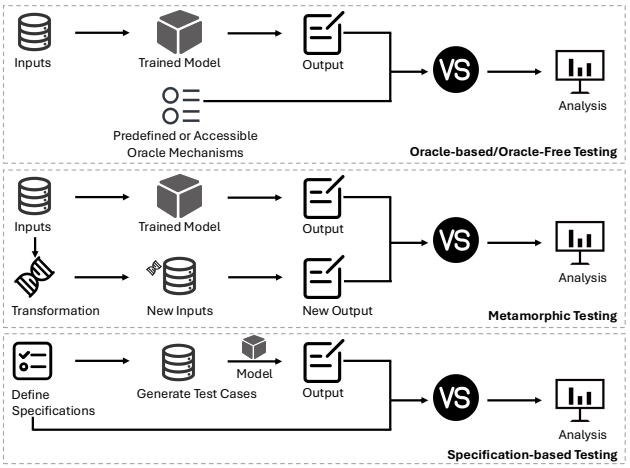


Fig. 11. Workflow of correctness-oriented testing techniques

extended this idea in LEMON by introducing model mutations to amplify behavioral divergence across libraries. Gu *et al.* [97] (ICSE 2022) further advanced this approach with Muffin, a fuzzing-based framework that constructs randomized neural architectures to uncover inconsistencies in loss, gradients, and predictions across training pipelines.

More recently, Wang *et al.* [248] (TSE 2024) introduced D^3, the first differential testing framework targeting distributed DL systems. D^3 defines a distributed equivalence rule asserting that outputs from identical inputs across distributed configurations (e.g., GPU count, sharding, quantization) should remain consistent. This enables automated generation of distributed scenarios to detect implementation bugs and numerical instabilities.

2) Statistical oracles Another branch of work leverages *statistical oracles*, such as DeepGini [75] (ISSTA 2020), which uses prediction uncertainty (via Gini impurity) as a proxy for error likelihood. These approaches prioritize tests that expose incorrect behaviors without requiring ground truth labels. In addition, studies such as DeepXplore [191] (SOSP 2017) adopted a cross-referencing oracle across multiple models trained for the same task. If these models disagree on the output for a given input, it is flagged as a potentially incorrect case. This mechanism effectively surfaces corner cases and erroneous behaviors in autonomous systems. A broader analysis of oracle challenges in ML systems is provided by Liem and Panichella [163] (ICSE 2020), who categorize oracle types and analyze practical barriers in deploying them for real-world DL models. By analyzing ImageNet's ILSVRC2012 benchmark through heuristics based on entropy and semantic similarity, they detect structural flaws in label taxonomy, inconsistent class relationships (e.g., synonyms, homonyms, meronyms), and limitations in data representation. Their findings suggest that high model accuracy does not necessarily reflect meaningful visual understanding, and that oracle quality can significantly impact both model validity and safety in real-world deployments.

9.2 Metamorphic Testing

Metamorphic Testing (MT) is a widely adopted approach for verifying the correctness of deep learning models in the absence of reliable test oracles. Instead of checking individual outputs against expected labels, MT validates whether the model behaves consistently under input transformations that preserve or predictably alter its semantics.

Early work by Dwarakanath *et al.* [67] (ISSTA 2018) first applied metamorphic testing to deep learning models. Specifically, they proposed and validated MRs for a ResNet-based image classifier (such as permutations of RGB channels and convolution operation order) to detect implementation

bugs. They demonstrated that violations of the defined MRs can effectively uncover implementation faults in deep neural networks, thereby addressing the oracle problem in deep learning testing. Building on this foundation, Wang and Su [250] (ASE 2020) extended MT beyond classification to object detection, proposing MetaOD, which enforces relational consistency between original and modified inputs through operations such as inserting or duplicating objects.

MT has also been extended to the deep learning software toolchain. Xiao *et al.* [260] proposed MT-DLComp, a framework targeting DL compilers, which are critical components responsible for transforming high-level DNN models into optimized low-level executables across heterogeneous hardware platforms. By defining semantics-preserving metamorphic relations and applying iterative model mutations, MT-DLComp effectively detects logic bugs in the compilation pipeline without relying on ground-truth outputs.

9.3 Specification-based Testing

Specification-based testing focuses on validating model predictions against explicitly defined properties or constraints. Seshia *et al.* [210] (ATVA 2018) surveyed the landscape of formal specifications for neural networks, categorizing properties like input–output invariances and safety constraints, and highlighting their significance in enabling property-driven verification. Sharma *et al.* [212] (ICMLA 2021) proposed MLCheck, a property-driven testing framework that allows users to formally specify high-level behaviors and generate test cases via constraint solving, without requiring user-defined test generators or manual annotations. While traditional specification-based testing relies on predefined properties, recent work such as SEDE proposed by Fahmy *et al.* [72] (TOSEM 2023) infers hazard-triggering conditions from observed failures and uses them as learned specifications for retraining and safety analysis. SEDE leverages simulators commonly used in cyber-physical domains to systematically generate inputs that resemble failure-inducing test cases. It then applies rule learning to derive human-readable expressions that identify commonalities among failing inputs. By combining heatmap-based clustering with tailored evolutionary search, SEDE identifies both failure-inducing and non-failure-inducing inputs to improve root cause inference.

9.4 Analysis of Correctness Techniques

We conducted an analysis of the aforementioned 13 representative papers on deep learning testing techniques. Our findings reveal that: the majority of these works originated from the **SE community**, with limited representation from the **ML community**. Specifically, 10 out of 13 papers (77%) were published in established SE venues, including ICSE [97, 193], FSE [253], ASE [250], ISSTA [67, 75, 260], and TOSEM [72]. Only one paper, MLCheck [212], appeared in a mixed AI venue (ICMLA), and none were published in core ML conferences. This distribution reflects a clear division of focus between the two communities. The SE community has taken a leading role in developing testability- and verification-oriented methodologies, particularly in challenging areas such as test oracle construction, metamorphic testing without labeled data, and specification-driven validation. These areas align closely with core SE concerns regarding software reliability, formal reasoning, and system-level correctness. By contrast, the ML community has largely prioritized goals such as improving accuracy, adversarial robustness, and generalization under distribution shifts. Testing and verification are often treated as downstream evaluation steps rather than primary research problems. ML research favors benchmark-driven empirical evaluation over formal verification, making correctness analysis less central in the publication pipeline.

Finding 13: *The majority of correctness-oriented testing techniques originate from the SE community. This reflects a greater emphasis on reliability, oracle construction, and behavioral validation, which remain underexplored in mainstream ML research.*

In addition to evaluating the community distribution of correctness-oriented techniques, we also analyzed their capacity for automation and their ease of integration into real-world deep learning development pipelines, which are two qualities that are essential for practical deployment and continuous validation in software engineering contexts. Our findings indicate that while most techniques are designed with minimal human supervision in mind, they differ significantly in their levels of automation and system integration readiness. For example, differential testing methods such as CRADLE [193], LEMON [253], and Muffin [97] can be seamlessly integrated into CI/CD pipelines due to their fully automated workflows and independence from labeled data. These tools can automatically generate tests, detect inconsistencies, and report errors across DL libraries with minimal human intervention. Similarly, D^3 [248] provides automated test scenario generation for distributed training setups, making it highly suitable for modern production-scale environments.

In contrast, some specification- and metamorphic-based methods involve manual steps or require domain-specific simulators, which could limit their adoption. For instance, MLCheck [212] requires developers to encode formal properties, and SEDE [72] relies on domain simulators and evolutionary search, which could increase deployment overhead. Although these methods provide richer fault localization and explainability, their integration into continuous development workflows may require additional tooling and infrastructure.

Finding 14: *Correctness-oriented testing techniques vary in their degree of automation and integration readiness. While many SE-developed tools support full automation and scalable deployment (e.g., via differential testing), others emphasize semantic expressiveness at the cost of higher integration overhead.*

10 PRIVACY TECHNIQUES

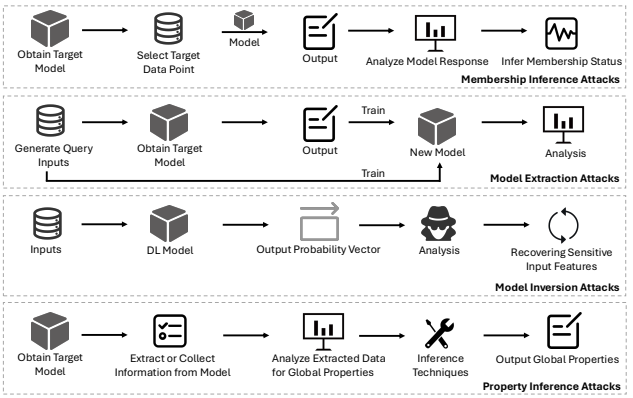


Fig. 12. Workflow of privacy-oriented testing techniques

As DL systems become increasingly integrated into sensitive domains such as healthcare, finance, and personalized services, concerns over the unintended leakage of private information have grown sharply. Recent studies have shown that deep models can memorize and reveal sensitive details from their training data, raising serious privacy risks [166]. For example, Rigaki and Garcia [202] highlight that privacy attacks like membership inference, model inversion, and property inference are not only technically feasible but are also becoming increasingly sophisticated, posing risks to

both data owners and model providers. Despite this growing body of work, privacy testing has received relatively limited attention in the software engineering testing community, with almost no dedicated research. This reveals a significant gap between privacy-focused attack/defense research in ML and SE-grounded testing methodologies. In the following, we review some representative approaches in the ML community and conduct corresponding analyses. For a broader understanding of privacy attacks and defenses in machine learning, we refer the reader to the comprehensive surveys [166, 202]. Figure 12 presents a workflow of these testing strategies.

10.1 Membership Inference Attacks

Membership inference (MIAs) attacks are among the earliest and most extensively studied privacy attacks, primarily focusing on determining whether a specific data record was part of a model's training set. Specifically, these attacks exploit differences in how a model responds to training versus non-training samples (such as confidence scores, output distributions, loss values, or gradients) and learn these patterns to infer whether a given input was used during training. It was first introduced by Shokri *et al.* [215] (IEEE S&P 2017). They demonstrated that models trained using popular machine learning-as-a-service (MLaaS) platforms (e.g., Google [22] and Amazon [55]) are vulnerable to such attacks even in a black-box setting. Subsequent work broadened the scope of MIAs along multiple dimensions. Salem *et al.* [207] showed that MIAs can succeed with fewer resources, using threshold-based or metric-based confidence scores. Yeom *et al.* [270] introduced the concept of advantage-based attacks, showing that MIAs can be directly linked to generalization error, and formalized the theoretical conditions under which membership leakage is likely.

Beyond discriminative models, generative models such as GANs and VAEs were also shown to be vulnerable. Hayes *et al.* [107] first demonstrated this for GANs, and later Chen *et al.* [33] (CCS 2020) presented a comprehensive taxonomy of MIA scenarios based on access granularity—black-box, white-box, and the nuanced partial black-box setting—highlighting that even limited access can leak membership. Recently, Wu *et al.* [258] (ICLR 2024) proposed YOQO (You Only Query Once), a highly efficient label-only membership inference attack that introduces the concept of "improvement area" (i.e., region in the input space that is sensitive to the inclusion of a target sample in the training set). By optimizing a single query sample within this region, YOQO accurately infers membership using only one model query per target.

10.2 Model Extraction Attacks

In addition to membership inference attacks, model extraction attacks have also become a major focus in the study of ML privacy. These attacks focus on reconstructing a substitute model by querying the original model, thereby posing a risk to its intellectual property. Tramèr *et al.* [234] (USENIX Security 2016) first demonstrate the feasibility of model extraction attacks against real-world machine learning systems. Their work showed that an adversary with black-box access to a prediction API without any knowledge of the model's parameters or training data can accurately reconstruct the model's decision logic. In many cases, they could even recover its exact parameters.

Subsequent efforts pushed the attack frontier toward more complex models. For example, Orekondy *et al.* [187] (CVPR 2019) proposed Knockoff Nets, showing that vision-based deep neural networks can be extracted using task-relevant, unlabeled data through transfer-set construction. This demonstrated that attackers can mount effective black-box extraction even without access to the same data distribution. Further work by Jagielski *et al.* [122] (USENIX Security 2020) explored high-fidelity extraction, aiming not just to reproduce output behavior but to match internal representations, showing that even gradient information leakage can be mimicked under certain

conditions. More recently, Rigaki and Garcia [202] (ACM Computing Surveys) reviewed the categorization of model extraction attacks based on attacker goals (e.g., functionality stealing vs. model replication) and capabilities (e.g., full prediction vector vs. label-only access). They emphasized the growing relevance of extraction as a precursor to other attacks such as membership inference or adversarial input generation.

Defenses have co-evolved alongside these attacks. Liu *et al.* [166] noted early heuristic-based defenses, such as output truncation and access rate limiting, though these were often insufficient against adaptive strategies. More advanced defenses like Tang *et al.* [226] (USENIX Security 2024) introduced information-theoretic formulations, aiming to minimize the mutual information between true and perturbed predictions, offering robustness against both weak and strong adversaries.

10.3 Model Inversion Attacks

In addition to membership inference and model extraction, model inversion attacks have also emerged as a crucial category of privacy threats. They aim to reconstruct sensitive input features or approximate raw data by analyzing the outputs of trained models. Fredrikson *et al.* [76] (CCS 2015) first introduced the concept of model inversion, showing that adversaries with access to prediction confidences can infer sensitive attributes of training inputs, such as reconstructing facial images or genomic markers.

Later research extended inversion attacks to more complex models. Hitaj *et al.* (2017) [112] (CCS 2017) demonstrated that collaborative learning settings (e.g., federated learning) are also vulnerable. By introducing a malicious participant, they used GANs to generate realistic reconstructions of training data from other participants, despite the privacy-preserving design of the learning protocol. This indicated that inversion threats exist even in decentralized and privacy-aware ML systems.

The integration of generative models into inversion attacks further increased their reconstruction quality. Zhang *et al.* [291] (CVPR 2020) showed that by training a GAN with a publicly available prior distribution, adversaries can effectively invert deep models to reconstruct training samples. Their work confirmed that models with high accuracy often memorize input-specific information, making them more vulnerable. Recently, the focus has shifted toward query-efficient black-box inversion. Li *et al.* [161] (CVPR 2025) proposed SMILE, a novel inversion framework that significantly reduces the number of queries required while maintaining high-fidelity reconstruction. SMILE achieves this through a surrogate-based initialization and a gradient-free search process in the generative model's latent space, enabling practical inversion in more realistic threat settings.

10.4 Property Inference Attacks

Property inference attacks aim to extract statistical or structural attributes of the training dataset (e.g., class distribution, data collection environment, or demographic composition) that are not directly exposed in the model's inputs or outputs. Property inference attacks are not yet as widely studied or deployed as membership inference or model extraction attacks, but they represent an important and growing area of research. This class of attacks was first systematically explored by Ganju *et al.* [80] (CCS 2018), who showed that fully connected neural networks can leak global properties of their training data through their learned parameters. Their work addressed the challenge of neuron permutation invariance by proposing two technical solutions: a neuron sorting procedure to create a canonical representation, and a DeepSets-based meta-classifier that naturally accommodates permutation invariance. This resulted in improved inference accuracy across multiple benchmark datasets.

Building on this foundation, property inference was extended to more complex model architectures and data modalities. For example, Melis *et al.* [179] (SP 2019) demonstrated that collaborative learning systems (such as federated learning) can leak unintended properties through gradient updates, even when individual data samples are never shared. These works highlighted that property inference is not limited to centralized, parameter-access scenarios, but also applies to real-world decentralized DL pipelines. More recently, Yuan *et al.* [274] (NeurIPS 2024) advanced the field by focusing on graph neural networks (GNNs), where structural and relational properties are of particular interest. Their proposed attack replaces the costly shadow training approach with a model approximation strategy, achieving high performance at lower computational overhead. Moreover, they introduce diversity-enhancing techniques based on edit distances and approximation error bounds, which improve attack generalizability across different graph distributions.

10.5 Analysis of Privacy Techniques

Based on the review of the above papers, we observe that privacy testing has emerged as a well-established research direction in the machine learning (ML) community, but remains underexplored in software engineering (SE). The ML community has developed a rich taxonomy of privacy attacks (including membership inference, model extraction, inversion, and property inference) with growing empirical effort. Techniques range from classical statistical exploits to modern generative-model-based strategies (e.g., GAN-driven inversion, label-only attacks, query-efficient extraction), typically evaluated across benchmarks and threat models. This line of work reflects a security-centric paradigm: understanding how sensitive information leaks and under what conditions models fail to preserve data confidentiality. In contrast, the SE community has paid relatively little attention to privacy testing. While SE research has adapted classical techniques (e.g., for robustness or fairness testing), there is a lack of systematic methodologies for assessing privacy leakage in learning-enabled systems. This gap reveals an underexplored direction: integrating privacy evaluation into the broader software testing lifecycle, where privacy risks could be proactively identified, monitored, and mitigated as part of standard pipelines.

> **Finding 15:** *Privacy testing in the ML community has evolved into a well-defined research area with standardized attack taxonomies, formal threat models, and empirically validated methods. In contrast, the SE community has yet to establish dedicated privacy testing methodologies, revealing a significant gap between privacy-centric security research and software testing practices.*

11 CHALLENGES AND OPPORTUNITIES

In this section, we outline the key challenges that hinder the development and deployment of effective deep learning testing techniques, and highlight emerging opportunities that may drive future research directions.

11.1 Challenges in DL Testing

11.1.1 High Cost of Labeling and Expert Dependency.

Many state-of-the-art testing techniques rely on labeled test sets to evaluate performance or detect faults. However, the labeling proces (particularly in safety-critical domains like medical imaging or legal document classification) could require substantial manual effort and domain expertise, making the testing process costly. Furthermore, manual labeling at scale could introduce additional problems. Human annotators may produce inconsistent labels due to subjectivity, fatigue, or ambiguity in the data. As a result, label noise could undermine the validity of testing metrics and may lead to false conclusions about model performance. Additionally, as deep learning systems

evolve and new data continuously arrive, the need for repeated labeling could intensify, resulting the cost and scalability bottleneck.

11.1.2 Oracle Uncertainty in DL Testing.
Unlike traditional software systems with deterministic behaviors, deep learning models operate on probabilistic inference, making the definition of ground-truth oracles typically ambiguous. This uncertainty impairs the validation of model predictions, especially for complex tasks such as image generation, sentiment classification, or multimodal reasoning. The absence of explicit test oracles complicates the design of adequacy metrics and the interpretation of testing outcomes.

11.1.3 Lack of Unified Testing Standards. The proliferation of diverse testing criteria (e.g., neuron coverage, surprise adequacy, mutation score) has resulted in a fragmented landscape where comparison across techniques remains challenging. The absence of standard evaluation protocols, benchmarks, or test adequacy definitions limits reproducibility and inhibits the integration of research outcomes.

11.1.4 Insufficient Testing of Non-Standard Learning Paradigms.
Most existing testing methods are optimized for supervised classification tasks, while other paradigms such as generative modeling, reinforcement learning, and online learning receive comparatively limited attention. These learning settings pose unique testing challenges, such as delayed rewards, dynamic environments, or non-deterministic outputs.

11.1.5 Generalization Under Distribution Shifts.
Numerous testing techniques typically assume that test data shares the same distribution as training data (in-distribution). However, real-world deployments frequently face distribution shifts, either natural (e.g., environmental changes, user behavior) or synthetic (e.g., corruptions, adversarial perturbations). Such shifts can significantly degrade model performance and expose vulnerabilities not captured during in-distribution testing.

Despite its practical importance, most existing methods lack mechanisms to simulate or evaluate robustness under distribution shift. Traditional accuracy metrics typically ignore performance drops in specific subgroups or rare conditions. Although benchmarks like ImageNet-C [111] and WILDS [142] have been proposed, well-organized shift-aware testing remains limited in practice.

11.2 Research Opportunities in DL testing

11.2.1 Zero-Label Testing.
Emerging paradigms in self-supervised learning and representation analysis offer promising directions for label-efficient testing. Techniques such as AETTA [148] and Aries [116] demonstrate how model outputs and internals can serve as proxies for error likelihood, enabling test selection and accuracy estimation without any ground truth labels. This direction is particularly promising for online testing, continuous integration, or deployment-time monitoring, where labels are either unavailable or delayed. Further research can explore combining zero-label testing to build more adaptive and label-efficient testing pipelines.

11.2.2 Benchmarking Under Realistic Data Shifts.
Deep learning models typically suffer performance degradation when faced with data that deviates from the training distribution. Such distribution shifts. While existing benchmarks such as ImageNet-C/A/R/O and WILDS have made progress in assessing robustness under controlled and natural shifts, their coverage remains limited in modality, dynamics, and evaluation scope. Future research could focus on constructing multi-modal and task-diverse benchmark suites that more accurately reflect real-world deployment conditions. In addition, the development of standardized

evaluation protocols will be crucial to enabling fair comparisons and advancing shift-resilient model design.

11.2.3 Continual Testing under Concept Drift. In real-world applications, input distributions shift over time (a phenomenon known as concept drift) which means that the ground truth could evolve as time progresses. Existing test suites are typically static and thus fail to capture such dynamic changes. Future research could explore continual testing pipelines, leveraging techniques from online learning, adaptive sampling, and streaming data to maintain test effectiveness in the presence of temporal drift.

11.2.4 Testing Multimodal Deep Learning Systems.

With the rise of multimodal architectures such as CLIP [196], Flamingo [7], and GPT-4V [3], deep learning systems now process inputs across multiple modalities (e.g., text, images, audio). However, testing methods remain largely unimodal, failing to capture cross-modal inconsistencies or modality-specific vulnerabilities. Opportunities lie in: 1) designing cross-modal test adequacy metrics; 2) generating coherent multimodal adversarial examples and 3) detecting semantic inconsistency across modalities (e.g., image describes a cat but text refers to a dog).

11.2.5 Fairness Testing for Multimodal and Generative Models. Fairness testing and social bias detection remain insufficiently studied in the context of multimodal and generative models. For instance, models that generate text from images may inadvertently propagate gender or racial biases present in either modality. There is a growing need to develop testing techniques tailored to this setting, including multimodal fairness test generation, cross-modal counterfactual fairness evaluation, and demographic group sensitivity analysis, to ensure that deep learning systems make fair decisions.

12 CONCLUSION

This paper presents a comprehensive survey and taxonomy of testing techniques for deep learning (DL) systems, covering 236 studies across six key quality properties: efficiency, robustness and security, fairness, interpretability, correctness, and privacy. Through a fine-grained taxonomy and in-depth analysis of the surveyed papers, we provide a detailed landscape of DL testing, highlighting current research hotspots and obtaining several crucial findings. The study also identifies key challenges and highlights future research opportunities for testing in DL domains. We hope this work provides a useful reference for understanding the current landscape of DL testing. By organizing existing studies, highlighting challenges, and summarizing emerging trends, we aim to support future efforts toward improving the reliability and trustworthiness of DL-based systems.

AVAILABILITY

Our package is available at

https://github.com/yinghuali/Survey.

REFERENCES

[1] Ali Abbasi, Parsa Nooralinejad, Hamed Pirsiavash, and Soheil Kolouri. 2024. BrainWash: A Poisoning Attack to Forget in Continual Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 24057–24067.

[2] Mahed Abroshan, Seyed-Mohsen Moosavi-Dezfooli, et al. 2024. SuperDeepFool: a new fast and accurate minimal adversarial attack. *Advances in Neural Information Processing Systems* 37 (2024), 98537–98562.

[3] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[4] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton. 2021. Neural additive models: Interpretable machine learning with neural nets. *Advances in neural information processing systems* 34 (2021), 4699–4711.

[5] Shibbir Ahmed, Hongyang Gao, and Hridesh Rajan. 2024. Inferring data preconditions from deep learning models for trustworthy prediction in deployment. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering*. 1–13. https://doi.org/10.1145/3597503.3623333

[6] Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access* 6 (2018), 14410–14430.

[7] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems* 35 (2022), 23716–23736.

[8] Vítor Albiero, Kevin Bowyer, Kushal Vangara, and Michael King. 2020. Does face recognition accuracy get better with age? deep face matchers say no. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 261–269. https://doi.org/10.1109/WACV45572.2020.9093357

[9] Andres Algaba, Carmen Mazijn, Carina Prunkl, Jan Danckaert, and Vincent Ginis. 2023. LUCID–GAN: Conditional Generative Models to Locate Unfairness. In *World Conference on Explainable Artificial Intelligence*. Springer, 346–367. https://doi.org/10.1007/978-3-031-44070-0_18

[10] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565* (2016).

[11] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. 2020. Square attack: a query-efficient black-box adversarial attack via random search. In *European conference on computer vision*. Springer, 484–501. https://doi.org/10.1007/978-3-030-58592-1_29

[12] Rico Angell, Brittany Johnson, Yuriy Brun, and Alexandra Meliou. 2018. Themis: Automatically testing software for discrimination. In *Proceedings of the 2018 26th ACM Joint meeting on european software engineering conference and symposium on the foundations of software engineering*. 871–875. https://doi.org/10.1145/3236024.3264590

[13] David Arthur and Sergei Vassilvitskii. 2006. *k-means++: The advantages of careful seeding*. Technical Report. Stanford.

[14] Shreyash Arya, Sukrut Rao, Moritz Böhle, and Bernt Schiele. 2024. B-cosification: Transforming Deep Neural Networks to be Inherently Interpretable. In *Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS)*. https://github.com/shrebox/B-cosification arXiv:2411.00715.

[15] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems* 33 (2020), 12449–12460.

[16] Tongtong Bai, Song Huang, Yifan Huang, Xingya Wang, Chunyan Xia, Yubin Qu, and Zhen Yang. 2024. CriticalFuzz: A critical neuron coverage-guided fuzz testing framework for deep neural networks. *Information and Software Technology* 172 (2024), 107476. https://doi.org/10.1016/j.infsof.2024.107476

[17] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2154–2156.

[18] Sumon Biswas and Hridesh Rajan. 2023. Fairify: Fairness verification of neural networks. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1546–1558. https://doi.org/10.1109/ICSE48619.2023.00134

[19] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. 2016. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316* (2016).

[20] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems* 29 (2016).

[21] Brandon M Booth, Louis Hickman, Shree Krishna Subburaj, Louis Tay, Sang Eun Woo, and Sidney K D'Mello. 2021. Bias and fairness in multimodal machine learning: A case study of automated video interviews. In *Proceedings of the 2021 international conference on multimodal interaction*. 268–277. https://doi.org/10.1145/3462244.3479897

[22] Praveen Borra. 2024. The evolution and impact of google cloud platform in machine learning and AI. *Available at SSRN 4914163* (2024).

[23] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 177–186. https://doi.org/10.1007/978-3-7908-2604-3_16

[24] Panagiotis Bountakas, Apostolis Zarras, Alexios Lekidis, and Christos Xenakis. 2023. Defense strategies for adversarial machine learning: A survey. *Computer Science Review* 49 (2023), 100573. https://doi.org/10.1016/j.cosrev.2023.100573

[25] Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*. PMLR, 77–91.

[26] Yushi Cao, David Berend, Palina Tolmach, Guy Amit, Moshe Levy, Yang Liu, Asaf Shabtai, and Yuval Elovici. 2022. Fair and accurate age prediction using distribution aware data curation and augmentation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 3551–3561. https://doi.org/10.1109/WACV51458.2022.00292

[27] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. 2019. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705* (2019).

[28] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*. Ieee, 39–57. https://doi.org/10.1109/SP.2017.49

[29] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. 2018. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069* (2018).

[30] Joymallya Chakraborty, Suvodeep Majumder, Zhe Yu, and Tim Menzies. 2020. Fairway: a way to build fair ML software. In *Proceedings of the 28th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*. 654–665. https://doi.org/10.1145/3368089.3409697

[31] Ivi Chatzi, Nina Corvelo Benz, Eleni Straitouri, Stratis Tsirtsis, and Manuel Gomez-Rodriguez. 2024. Counterfactual token generation in large language models. *arXiv preprint arXiv:2409.17027* (2024).

[32] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. 2019. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems* 32 (2019).

[33] Dingfan Chen, Ning Yu, Yang Zhang, and Mario Fritz. 2020. Gan-leaks: A taxonomy of membership inference attacks against generative models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*. 343–362. https://doi.org/10.1145/3372297.3417238

[34] Huaming Chen and M Ali Babar. 2024. Security for machine learning-based software systems: A survey of threats, practices, and challenges. *Comput. Surveys* 56, 6 (2024), 1–38. https://doi.org/10.1145/3638531

[35] Jinyin Chen, Jie Ge, and Haibin Zheng. 2023. ActGraph: prioritization of test cases based on deep neural network activation graph. *Automated Software Engineering* 30, 2 (2023), 28. https://doi.org/10.1007/s10515-023-00396-8

[36] Jialuo Chen, Jingyi Wang, Xiyue Zhang, Youcheng Sun, Marta Kwiatkowska, Jiming Chen, and Peng Cheng. 2024. FAST: Boosting Uncertainty-based Test Prioritization Methods for Neural Networks via Feature Selection. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*. 895–906. https://doi.org/10.1145/3691620.3695472

[37] Junjie Chen, Zhuo Wu, Zan Wang, Hanmo You, Lingming Zhang, and Ming Yan. 2020. Practical accuracy estimation for efficient deep neural network testing. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 29, 4 (2020), 1–35. https://doi.org/10.1145/3394112

[38] Simin Chen, Zexin Li, Wei Yang, and Cong Liu. 2024. DeciX: Explain Deep Learning Based Code Generation Applications. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 2424–2446. https://doi.org/10.1145/3660814

[39] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274* (2015).

[40] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).

[41] Zuohui Chen, Renxuan Wang, Jingyang Xiang, Yue Yu, Xin Xia, Shouling Ji, Qi Xuan, and Xiaoniu Yang. 2021. Detecting adversarial samples with graph-guided testing. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 1196–1198. https://doi.org/10.1109/ASE51524.2021.9678732

[42] Zhenpeng Chen, Jie M Zhang, Federica Sarro, and Mark Harman. 2022. MAAT: a novel ensemble approach to addressing fairness and performance bugs for machine learning software. In *Proceedings of the 30th ACM joint european software engineering conference and symposium on the foundations of software engineering*. 1122–1134. https://doi.org/10.1145/3540250.3549093

[43] Zhenpeng Chen, Jie M Zhang, Federica Sarro, and Mark Harman. 2024. Fairness improvement with multiple protected attributes: How far are we?. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13. https://doi.org/10.1145/3597503.3639083

[44] KR1442 Chowdhary. 2020. Natural language processing. *Fundamentals of artificial intelligence* (2020), 603–649. https://doi.org/10.1007/978-81-322-3972-7_19

[45] Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Sebastiano Vascon, Werner Zellinger, Bernhard A Moser, Alina Oprea, Battista Biggio, Marcello Pelillo, and Fabio Roli. 2023. Wild patterns reloaded: A survey of machine learning security against training data poisoning. *Comput. Surveys* 55, 13s (2023), 1–39.

[46] Jürgen Cito, Isil Dillig, Seohyun Kim, Vijayaraghavan Murali, and Satish Chandra. 2021. Explaining mispredictions of machine learning models using rule induction. In *Proceedings of the 29th ACM joint meeting on European software*

engineering conference and symposium on the foundations of software engineering. 716–727. https://doi.org/10.1145/3468264.3468614

[47] Jürgen Cito, Isil Dillig, Vijayaraghavan Murali, and Satish Chandra. 2022. Counterfactual explanations for models of code. In *Proceedings of the 44th international conference on software engineering: software engineering in practice*. 125–134. https://doi.org/10.1145/3510457.3513081

[48] Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. 2021. Batch active learning at scale. *Advances in Neural Information Processing Systems* 34 (2021), 11933–11944.

[49] Domenico Cotroneo, Cristina Improta, Pietro Liguori, and Roberto Natella. 2024. Vulnerabilities in ai code generators: Exploring targeted data poisoning attacks. In *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*. 280–292. https://doi.org/10.1145/3643916.3644416

[50] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. 2020. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670* (2020).

[51] Francesco Croce, Sven Gowal, Thomas Brunner, Evan Shelhamer, Matthias Hein, and Taylan Cemgil. 2022. Evaluating the adversarial robustness of adaptive test-time defenses. In *International Conference on Machine Learning*. PMLR, 4421–4435.

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

[53] Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, Anthony Bau, and James Glass. 2019. What is one grain of sand in the desert? analyzing individual neurons in deep nlp models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6309–6317. https://doi.org/10.1609/aaai.v33i01.33016309

[54] Xueqi Dang, Yinghua Li, Mike Papadakis, Jacques Klein, Tegawendé F Bissyandé, and Yves LE Traon. 2023. GraphPrior: Mutation-based Test Input Prioritization for Graph Neural Networks. *ACM Transactions on Software Engineering and Methodology* (2023). https://doi.org/10.1145/3607191

[55] Piali Das, Nikita Ivkin, Tanya Bansal, Laurence Rouesnel, Philip Gautier, Zohar Karnin, Leo Dirac, Lakshmi Ramakrishnan, Andre Perunicic, Iaroslav Shcherbatyi, et al. 2020. Amazon SageMaker Autopilot: a white box AutoML solution at scale. In *Proceedings of the fourth international workshop on data management for end-to-end machine learning*. 1–7. https://doi.org/10.1145/3399579.3399870

[56] Saloni Dash, Vineeth N Balasubramanian, and Amit Sharma. 2022. Evaluating and mitigating bias in image classifiers: A causal perspective using counterfactuals. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. 915–924.

[57] Demet Demir, Aysu Betin Can, and Elif Surer. 2024. Test Selection for Deep Neural Networks using Meta-Models with Uncertainty Metrics. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 678–690. https://doi.org/10.1145/3650212.3680312

[58] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. IEEE, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[59] Yian Deng and Tingting Mu. 2023. Understanding and improving ensemble adversarial defense. *Advances in Neural Information Processing Systems* 36 (2023), 58075–58087.

[60] Yao Deng, Tiehua Zhang, Guannan Lou, Xi Zheng, Jiong Jin, and Qing-Long Han. 2021. Deep learning-based autonomous driving systems: A survey of attacks and defenses. *IEEE Transactions on Industrial Informatics* 17, 12 (2021), 7897–7912.

[61] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 4171–4186. https://doi.org/10.18653/v1/N19-1423

[62] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. 2020. A survey on ensemble learning. *Frontiers of Computer Science* 14 (2020), 241–258. https://doi.org/10.1007/s11704-019-8208-z

[63] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*. https://openreview.net/forum?id=YicbFdNTTy

[64] Nathan Drenkow, Numair Sani, Ilya Shpitser, and Mathias Unberath. 2021. A systematic review of robustness in deep learning for computer vision: Mind the gap? *arXiv preprint arXiv:2112.00639* (2021).

[65] Jiarui Duan, Haoling Li, Haofei Zhang, Hao Jiang, Mengqi Xue, Li Sun, Mingli Song, and Jie Song. 2024. On the Evaluation Consistency of Attribution-Based Explanations. In *European Conference on Computer Vision*. Springer,

206–224. https://doi.org/10.1007/978-3-031-72897-6_12

[66] Raman Dutt, Ondrej Bohdal, Sotirios A. Tsaftaris, and Timothy Hospedales. 2024. FairTune: Optimizing Parameter Efficient Fine Tuning for Fairness in Medical Image Analysis. In *Proceedings of the International Conference on Learning Representations (ICLR)*. https://github.com/Raman1121/FairTune

[67] Anurag Dwarakanath, Manish Ahuja, Samarth Sikand, Raghotham M Rao, RP Jagadeesh Chandra Bose, Neville Dubash, and Sanjay Podder. 2018. Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. In *Proceedings of the 27th ACM SIGSOFT international symposium on software testing and analysis*. 118–128. https://doi.org/10.1145/3213846.3213858

[68] Andre Esteva, Katherine Chou, Serena Yeung, Nikhil Naik, Ali Madani, Ali Mottaghi, Yun Liu, Eric Topol, Jeff Dean, and Richard Socher. 2021. Deep learning-enabled medical computer vision. *NPJ digital medicine* 4, 1 (2021), 5. https://doi.org/10.1038/s41746-020-00376-2

[69] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. 2017. Dermatologist-level classification of skin cancer with deep neural networks. *nature* 542, 7639 (2017), 115–118. https://doi.org/10.1038/nature21056

[70] Andre Esteva, Alexandre Robicquet, Bharath Ramsundar, Volodymyr Kuleshov, Mark DePristo, Katherine Chou, Claire Cui, Greg Corrado, Sebastian Thrun, and Jeff Dean. 2019. A guide to deep learning in healthcare. *Nature medicine* 25, 1 (2019), 24–29. https://doi.org/10.1038/s41591-018-0316-z

[71] Farshid Faal, Ketra Schmitt, and Jia Yuan Yu. 2023. Reward modeling for mitigating toxicity in transformer-based language models. *Applied Intelligence* 53, 7 (2023), 8421–8435. https://doi.org/10.1007/s10489-022-03944-z

[72] Hazem Fahmy, Fabrizio Pastore, Lionel Briand, and Thomas Stifter. 2023. Simulator-based explanation and debugging of hazard-triggering events in DNN-based safety-critical systems. *ACM Transactions on Software Engineering and Methodology* 32, 4 (2023), 1–47.

[73] Jiaxin Fan, Qi Yan, Mohan Li, Guanqun Qu, and Yang Xiao. 2022. A survey on data poisoning attacks and defenses. In *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)*. IEEE, 48–55.

[74] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Gong. 2020. Local model poisoning attacks to {Byzantine-Robust} federated learning. In *29th USENIX security symposium (USENIX Security 20)*. 1605–1622.

[75] Yang Feng, Qingkai Shi, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. 2020. Deepgini: prioritizing massive tests to enhance the robustness of deep neural networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 177–188. https://doi.org/10.1145/3395363.3397357

[76] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 1322–1333. https://doi.org/10.1145/2810103.2813677

[77] Zee Fryer, Vera Axelrod, Ben Packer, Alex Beutel, Jilin Chen, and Kellie Webster. 2022. Flexible text generation for counterfactual fairness probing. *arXiv preprint arXiv:2206.13757* (2022).

[78] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *International conference on machine learning*. PMLR, 1183–1192.

[79] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: testing software for discrimination. In *Proceedings of the 2017 11th Joint meeting on foundations of software engineering*. 498–510. https://doi.org/10.1145/3106237.3106277

[80] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. 2018. Property inference attacks on fully connected neural networks using permutation invariant representations. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 619–633. https://doi.org/10.1145/3243734.3243834

[81] Xinyu Gao, Yang Feng, Yining Yin, Zixi Liu, Zhenyu Chen, and Baowen Xu. 2022. Adaptive test selection for deep neural networks. In *Proceedings of the 44th international conference on software engineering*. 73–85. https://doi.org/10.1145/3510003.3510232

[82] Xiang Gao, Ripon K Saha, Mukul R Prasad, and Abhik Roychoudhury. 2020. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *Proceedings of the acm/ieee 42nd international conference on software engineering*. 1147–1158.

[83] Yue Gao, Ilia Shumailov, Kassem Fawaz, and Nicolas Papernot. 2022. On the limitations of stochastic pre-processing defenses. *Advances in neural information processing systems* 35 (2022), 24280–24294.

[84] Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. *arXiv preprint arXiv:2009.11462* (2020).

[85] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. 2018. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE symposium on security and privacy (SP)*. IEEE, 3–18. https://doi.org/10.1109/SP.2018.00058

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

//doi.org/10.1145/3377811.3380391

[87] Waris Gill, Ali Anwar, and Muhammad Ali Gulzar. 2025. TraceFL: Interpretability-Driven Debugging in Federated Learning via Neuron Provenance. In *Proceedings of the 47th IEEE/ACM International Conference on Software Engineering (ICSE)*. IEEE/ACM. https://doi.org/10.1109/ICSE55347.2025.00128

[88] Milos Gligoric, Alex Groce, Chaoqiang Zhang, Rohan Sharma, Mohammad Amin Alipour, and Darko Marinov. 2013. Comparing non-adequate test suites using coverage criteria. In *Proceedings of the 2013 International Symposium on Software Testing and Analysis*. 302–313. https://doi.org/10.1145/2483760.2483769

[89] Yuan Gong, Yu-An Chung, and James Glass. 2021. AST: Audio Spectrogram Transformer. In *Proc. Interspeech 2021*. 571–575. https://doi.org/10.21437/Interspeech.2021-698

[90] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

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

[92] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1412.6572

[93] Divya Gopinath, Kaiyuan Wang, Mengshi Zhang, Corina S Pasareanu, and Sarfraz Khurshid. 2018. Symbolic execution for deep neural networks. *arXiv preprint arXiv:1807.10439* (2018).

[94] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. 2020. A survey of deep learning techniques for autonomous driving. *Journal of field robotics* 37, 3 (2020), 362–386. https://doi.org/10.1002/rob.21918

[95] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. 2017. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280* (2017).

[96] Bin Gu, Zhou Zhai, Cheng Deng, and Heng Huang. 2020. Efficient active learning by querying discriminative and representative samples and fully exploiting unlabeled data. *IEEE Transactions on Neural Networks and Learning Systems* 32, 9 (2020), 4111–4122. https://doi.org/10.1109/TNNLS.2020.3016928

[97] Jiazhen Gu, Xuchuan Luo, Yangfan Zhou, and Xin Wang. 2022. Muffin: Testing deep learning libraries via neural architecture fuzzing. In *Proceedings of the 44th International Conference on Software Engineering*. 1418–1430. https://doi.org/10.1145/3510003.3510092

[98] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[99] Antonio Guerriero, Roberto Pietrantuono, and Stefano Russo. 2024. DeepSample: DNN sampling-based testing for operational accuracy assessment. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–12. https://doi.org/10.1145/3597503.3639584

[100] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. 2018. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)* 51, 5 (2018), 1–42. https://doi.org/10.1145/3236009

[101] Jun Guo, Wei Bao, Jiakai Wang, Yuqing Ma, Xinghai Gao, Gang Xiao, Aishan Liu, Jian Dong, Xianglong Liu, and Wenjun Wu. 2023. A comprehensive evaluation framework for deep model robustness. *Pattern Recognition* 137 (2023), 109308. https://doi.org/10.1016/j.patcog.2023.109308

[102] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. 2018. Dlfuzz: Differential fuzzing testing of deep learning systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 739–743. https://doi.org/10.1145/3236024.3264835

[103] Ping Guo, Cheng Gong, Xi Lin, Fei Liu, Zhichao Lu, Qingfu Zhang, and Zhenkun Wang. 2025. MOS-Attack: A Scalable Multi-Objective Adversarial Attack Framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Poster No. 466, ExHall, 13 June 2025.

[104] Yao Hao, Zhiqiu Huang, Hongjing Guo, and Guohua Shen. 2023. Test input selection for deep neural network enhancement based on multiple-objective optimization. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 534–545. https://doi.org/10.1109/SANER56733.2023.00056

[105] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. *Advances in neural information processing systems* 29 (2016).

[106] Trevor J Hastie. 2017. Generalized additive models. *Statistical models in S* (2017), 249–307.

[107] Jamie Hayes, Luca Melis, George Danezis, and Emiliano De Cristofaro. 2017. Logan: Membership inference attacks against generative models. *arXiv preprint arXiv:1705.07663* (2017).

[108] Yingzhe He, Guozhu Meng, Kai Chen, Xingbo Hu, and Jinwen He. 2020. Towards security threats of deep learning systems: A survey. *IEEE Transactions on Software Engineering* 48, 5 (2020), 1743–1770. https://doi.org/10.1109/TSE.2020.3034721

[109] Lisa Anne Hendricks, Kaylee Burns, Kate Saenko, Trevor Darrell, and Anna Rohrbach. 2018. Women also snowboard: Overcoming bias in captioning models. In *Proceedings of the European conference on computer vision (ECCV)*. 771–787.

[110] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. 2021. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF international conference on computer vision*. 8340–8349. https://doi.org/10.1109/ICCV48922.2021.00823

[111] Dan Hendrycks and Thomas Dietterich. 2019. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=HJz6tiCqYm

[112] Briland Hitaj, Giuseppe Ateniese, and Fernando Perez-Cruz. 2017. Deep models under the GAN: Information leakage from collaborative deep learning. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 603–618.

[113] Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745* (2011).

[114] Qiang Hu, Yuejun Guo, Maxime Cordy, Xiaofei Xie, Lei Ma, Mike Papadakis, and Yves Le Traon. 2022. An empirical study on data distribution-aware test selection for deep learning enhancement. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 4 (2022), 1–30. https://doi.org/10.1145/3511598

[115] Qiang Hu, Yuejun Guo, Xiaofei Xie, Maxime Cordy, Lei Ma, Mike Papadakis, and Yves Le Traon. 2024. Test optimization in dnn testing: a survey. *ACM Transactions on Software Engineering and Methodology* 33, 4 (2024), 1–42.

[116] Qiang Hu, Yuejun Guo, Xiaofei Xie, Maxime Cordy, Mike Papadakis, Lei Ma, and Yves Le Traon. 2023. Aries: Efficient testing of deep neural networks via labeling-free accuracy estimation. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1776–1787. https://doi.org/10.1109/ICSE48619.2023.00152

[117] Li Huang, Weifeng Sun, and Meng Yan. 2025. Iterative Generation of Adversarial Example for Deep Code Models. In *2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE Computer Society, 623–623. https://doi.org/10.1109/ICSE55347.2025.00086

[118] Pei Huang, Yuting Yang, Minghao Liu, Fuqi Jia, Feifei Ma, and Jian Zhang. 2022. ϵ-weakened robustness of deep neural networks. In *Proceedings of the 31st ACM SIGSOFT international symposium on software testing and analysis*. 126–138. https://doi.org/10.1145/3533767.3534373

[119] Siyu Huang, Tianyang Wang, Haoyi Xiong, Jun Huan, and Dejing Dou. 2021. Semi-supervised active learning with temporal output discrepancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3447–3456. https://doi.org/10.1109/ICCV48922.2021.00343

[120] Yujin Huang, Han Hu, and Chunyang Chen. 2021. Robustness of on-device models: Adversarial attack to deep learning models on android apps. In *2021 IEEE/ACM 43rd International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 101–110. https://doi.org/10.1109/ICSE-SEIP52600.2021.00019

[121] Aya Abdelsalam Ismail, Hector Corrada Bravo, and Soheil Feizi. 2021. Improving deep learning interpretability by saliency guided training. *Advances in Neural Information Processing Systems* 34 (2021), 26726–26739.

[122] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. 2020. High accuracy and high fidelity extraction of neural networks. In *29th USENIX security symposium (USENIX Security 20)*. 1345–1362.

[123] Yujie Ji, Xinyang Zhang, Shouling Ji, Xiapu Luo, and Ting Wang. 2018. Model-reuse attacks on deep learning systems. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 349–363. https://doi.org/10.1145/3243734.3243757

[124] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. 2014. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*. 675–678. https://doi.org/10.1145/2647868.2654889

[125] Sangwon Jung, Donggyu Lee, Taeeon Park, and Taesup Moon. 2021. Fair feature distillation for visual recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12115–12124. https://doi.org/10.1109/CVPR46437.2021.01194

[126] Yoojin Jung and Byung Cheol Song. 2025. Two is Better than One: Efficient Ensemble Defense for Robust and Compact Models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 9696–9706.

[127] Sachiko Kanamori, Taeko Abe, Takuma Ito, Keita Emura, Lihua Wang, Shuntaro Yamamoto, Le Trieu Phong, Kaien Abe, Sangwook Kim, Ryo Nojima, et al. 2022. Privacy-preserving federated learning for detecting fraudulent financial transactions in japanese banks. *Journal of Information Processing* 30 (2022), 789–795.

[128] Sunghun Kang, Gwangsu Kim, and Chang D Yoo. 2022. Fair facial attribute classification via causal graph-based attribute translation. *Sensors* 22, 14 (2022), 5271. https://doi.org/10.3390/s22145271

[129] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. 2023. Champion-level drone racing using deep reinforcement learning. *Nature* 620, 7976 (2023), 982–987. https://doi.org/10.1038/s41586-023-06419-4

[130] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. 2021. Deep Drone Acrobatics (Extended Abstract). In *Proceedings of the Thirtieth International Joint Conference on Artificial*

Intelligence, IJCAI-21, Zhi-Hua Zhou (Ed.). International Joint Conferences on Artificial Intelligence Organization, 4780–4783. https://doi.org/10.24963/ijcai.2021/650

[131] Nikhil Ketkar, Jojo Moolayil, Nikhil Ketkar, and Jojo Moolayil. 2021. Introduction to pytorch. *Deep learning with python: learn best practices of deep learning models with PyTorch* (2021), 27–91. https://doi.org/10.1007/978-1-4842-5364-9_2

[132] Mohammad Mahdi Khalili, Xueru Zhang, and Mahed Abroshan. 2023. Loss balancing for fair supervised learning. In *International Conference on Machine Learning*. PMLR, 16271–16290.

[133] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. 2022. Transformers in vision: A survey. *ACM computing surveys (CSUR)* 54, 10s (2022), 1–41. https://doi.org/10.1145/3505244

[134] Hyemi Kim, Seungjae Shin, JoonHo Jang, Kyungwoo Song, Weonyoung Joo, Wanmo Kang, and Il-Chul Moon. 2021. Counterfactual fairness with disentangled causal effect variational autoencoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 8128–8136. https://doi.org/10.1609/aaai.v35i9.16990

[135] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 1039–1049. https://doi.org/10.1109/ICSE.2019.00108

[136] Jinhan Kim, Robert Feldt, and Shin Yoo. 2023. Evaluating surprise adequacy for deep learning system testing. *ACM Transactions on Software Engineering and Methodology* 32, 2 (2023), 1–29. https://doi.org/10.1145/3546947

[137] Jinhan Kim, Jeongil Ju, Robert Feldt, and Shin Yoo. 2020. Reducing dnn labelling cost using surprise adequacy: An industrial case study for autonomous driving. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1466–1476. https://doi.org/10.1145/3368089.3417065

[138] Yeachan Kim and Bonggun Shin. 2022. In defense of core-set: A density-aware core-set selection for active learning. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 804–812. https://doi.org/10.1145/3534678.3539476

[139] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[140] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[141] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *International conference on machine learning*. PMLR, 1885–1894.

[142] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*. PMLR, 5637–5664.

[143] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[144] Akshay Kulkarni and Tsui-Wei Weng. 2024. Interpretability-Guided Test-Time Adversarial Defense. In *European Conference on Computer Vision*. Springer, 466–483. https://doi.org/10.1007/978-3-031-72913-3_26

[145] Keita Kurita, Paul Michel, and Graham Neubig. 2020. Weight Poisoning Attacks on Pretrained Models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 2793–2806. https://doi.org/10.18653/v1/2020.acl-main.249

[146] Keita Kurita, Nidhi Vyas, Ayush Pareek, Alan W Black, and Yulia Tsvetkov. 2019. Measuring bias in contextualized word representations. *arXiv preprint arXiv:1906.07337* (2019).

[147] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 2002. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (2002), 2278–2324. https://doi.org/10.1109/5.726791

[148] Taeckyung Lee, Sorn Chottananurak, Taesik Gong, and Sung-Ju Lee. 2024. AETTA: Label-Free Accuracy Estimation for Test-Time Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 28643–28652. https://doi.org/10.1109/CVPR52733.2024.02706

[149] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. 2017. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific reports* 7, 1 (2017), 1–14. https://doi.org/10.1038/s41598-017-17876-z

[150] Dongyuan Li, Zhen Wang, Yankai Chen, Renhe Jiang, Weiping Ding, and Manabu Okumura. 2024. A survey on deep active learning: Recent advances and new frontiers. *IEEE Transactions on Neural Networks and Learning Systems* (2024). https://doi.org/10.1109/TNNLS.2024.3396463

[151] Peizhao Li, Han Zhao, and Hongfu Liu. 2020. Deep fair clustering for visual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9070–9079. https://doi.org/10.1109/CVPR42600.2020.00909

[152] Yu Li, Muxi Chen, and Qiang Xu. 2022. HybridRepair: towards annotation-efficient repair for deep learning models. In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 227–238. https://doi.org/10.1145/3533767.3534408

[153] Yinghua Li, Xueqi Dang, Lei Ma, Jacques Klein, and Tegawendé F Bissyandé. 2024. Prioritizing test cases for deep learning-based video classifiers. *Empirical Software Engineering* 29, 5 (2024), 111.

[154] Yinghua Li, Xueqi Dang, Lei Ma, Jacques Klein, Yves Le Traon, and Tegawendé F Bissyandé. 2024. Test input prioritization for 3d point clouds. *ACM Transactions on Software Engineering and Methodology* 33, 5 (2024), 1–44. https://doi.org/10.1145/3643676

[155] Yingji Li, Mengnan Du, Rui Song, Xin Wang, and Ying Wang. 2023. A survey on fairness in large language models. *arXiv preprint arXiv:2308.10149* (2023).

[156] Yuanchun Li, Jiayi Hua, Haoyu Wang, Chunyang Chen, and Yunxin Liu. 2021. Deeppayload: Black-box backdoor attack on deep learning models through neural payload injection. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 263–274. https://doi.org/10.1109/ICSE43902.2021.00035

[157] Yi Li and Nuno Vasconcelos. 2019. Repair: Removing representation bias by dataset resampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9572–9581. https://doi.org/10.1109/CVPR.2019.00980

[158] Zaitang Li, Pin-Yu Chen, and Tsung-Yi Ho. 2024. GREAT score: Global robustness evaluation of adversarial perturbation using generative models. *Advances in Neural Information Processing Systems* 37 (2024), 39158–39189.

[159] Zenan Li, Xiaoxing Ma, Chang Xu, Chun Cao, Jingwei Xu, and Jian Lü. 2019. Boosting operational dnn testing efficiency through conditioning. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 499–509. https://doi.org/10.1145/3338906.3338930

[160] Zhong Li, Zhengfeng Xu, Ruihua Ji, Minxue Pan, Tian Zhang, Linzhang Wang, and Xuandong Li. 2024. Distance-Aware Test Input Selection for Deep Neural Networks. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 248–260. https://doi.org/10.1145/3650212.3652125

[161] Ziang Li, Hongguang Zhang, Juan Wang, Meihui Chen, Hongxin Hu, Wenzhe Yi, Xiaoyang Xu, Mengda Yang, and Chenjun Ma. 2025. From Head to Tail: Efficient Black-box Model Inversion Attack via Long-tailed Learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 29288–29298.

[162] Zhen Li, Ruqian Zhang, Deqing Zou, Ning Wang, Yating Li, Shouhuai Xu, Chen Chen, and Hai Jin. 2023. Robin: a novel method to produce robust interpreters for deep learning-based code classifiers. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 27–39. https://doi.org/10.1109/ASE56229.2023.00164

[163] Cynthia CS Liem and Annibale Panichella. 2020. Oracle issues in machine learning and where to find them. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*. 483–488. https://doi.org/10.1145/3387940.3391490

[164] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part v 13*. Springer, 740–755. https://doi.org/10.1007/978-3-319-10602-1_48

[165] Yujie Lin, Chen Zhao, Minglai Shao, Baoluo Meng, Xujiang Zhao, and Haifeng Chen. 2024. Towards Counterfactual Fairness-aware Domain Generalization in Changing Environments. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*. Jeju, South Korea, 4560–4568.

[166] Bo Liu, Ming Ding, Sina Shaham, Wenny Rahayu, Farhad Farokhi, and Zihuai Lin. 2021. When machine learning meets privacy: A survey and outlook. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–36.

[167] Sanmin Liu, Shan Xue, Jia Wu, Chuan Zhou, Jian Yang, Zhao Li, and Jie Cao. 2021. Online active learning for drifting data streams. *IEEE Transactions on Neural Networks and Learning Systems* 34, 1 (2021), 186–200. https://doi.org/10.1109/TNNLS.2021.3091681

[168] Yintong Liu, U Rajendra Acharya, and Jen Hong Tan. 2025. Preserving privacy in healthcare: A systematic review of deep learning approaches for synthetic data generation. *Computer Methods and Programs in Biomedicine* 260 (2025), 108571.

[169] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018)*. Internet Soc. https://doi.org/10.14722/ndss.2018.23291

[170] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*. 10012–10022.

[171] Gianluigi Lopardo, Frederic Precioso, and Damien Garreau. 2023. A Sea of Words: An In-Depth Analysis of Anchors for Text Data. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 206)*. PMLR, 4848–4879. https://proceedings.mlr.press/v206/lopardo23a.html

[172] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. 2018. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 120–131. https://doi.org/10.1145/3238147.3238202

[173] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. 2018. Deepmutation: Mutation testing of deep learning systems. In *2018 IEEE 29th international symposium on software reliability engineering (ISSRE)*. IEEE, 100–111. https://doi.org/10.1109/ISSRE.2018.00021

[174] Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. 2021. Test selection for deep learning systems. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 30, 2 (2021), 1–22. https://doi.org/10.1145/3417330

[175] Yu-Seung Ma, Shin Yoo, and Taeho Kim. 2021. Selecting test inputs for DNNs using differential testing with subspecialized model instances. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1467–1470. https://doi.org/10.1145/3468264.3473131

[176] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rJzIBfZAb

[177] Pratik Mazumder, Pravendra Singh, and Vinay P Namboodiri. 2022. Fair Visual Recognition in Limited Data Regime using Self-Supervision and Self-Distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 3095–3103. https://doi.org/10.1109/WACV51458.2022.00394

[178] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)* 54, 6 (2021), 1–35.

[179] Luca Melis, Congzheng Song, Emiliano De Cristofaro, and Vitaly Shmatikov. 2019. Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 691–706.

[180] Dongyu Meng and Hao Chen. 2017. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 135–147. https://doi.org/10.1145/3133956.3134057

[181] Eric Mintun, Alexander Kirillov, and Saining Xie. 2021. On interaction between augmentations and corruptions in natural corruption robustness. *Advances in Neural Information Processing Systems* 34 (2021), 3571–3583.

[182] Fatemehsadat Mireshghallah, Mohammadkazem Taram, Praneeth Vepakomma, Abhishek Singh, Ramesh Raskar, and Hadi Esmaeilzadeh. 2020. Privacy in deep learning: A survey. *arXiv preprint arXiv:2004.12254* (2020).

[183] Verya Monjezi, Ashutosh Trivedi, Gang Tan, and Saeid Tizpaz-Niari. 2023. Information-theoretic testing and debugging of fairness defects in deep neural networks. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 1571–1582. https://doi.org/10.1109/ICSE48619.2023.00136

[184] Furkan Mumcu and Yasin Yilmaz. 2025. Detecting Adversarial Examples. arXiv:2410.17442 [cs.LG] https://arxiv.org/abs/2410.17442

[185] Hoang-Quoc Nguyen-Son, Seira Hidano, Kazuhide Fukushima, Shinsaku Kiyomoto, and Isao Echizen. 2023. Vote-TRANS: Detecting adversarial text without training by voting on hard labels of transformations. *arXiv preprint arXiv:2306.01273* (2023).

[186] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. 2019. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In *International Conference on Machine Learning*. PMLR, 4901–4911. https://proceedings.mlr.press/v97/odena19a.html

[187] Tribhuvanesh Orekondy, Bernt Schiele, and Mario Fritz. 2019. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4954–4963.

[188] Daniel W Otter, Julian R Medina, and Jugal K Kalita. 2020. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems* 32, 2 (2020), 604–624. https://doi.org/10.1109/TNNLS.2020.2979670

[189] Bo Pang, Erik Nijkamp, and Ying Nian Wu. 2020. Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics* 45, 2 (2020), 227–248. https://doi.org/10.3102/1076998619872761

[190] Otavio Parraga, Martin D More, Christian M Oliveira, Nathan S Gavenski, Lucas S Kupssinskü, Adilson Medronha, Luis V Moura, Gabriel S Simões, and Rodrigo C Barros. 2025. Fairness in Deep Learning: A survey on vision and language research. *Comput. Surveys* 57, 6 (2025), 1–40.

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

[192] Heqi Peng, Yunhong Wang, Ruijie Yang, Beichen Li, Rui Wang, and Yuanfang Guo. 2025. AED-PADA: Improving Generalizability of Adversarial Example Detection via Principal Adversarial Domain Adaptation. *ACM Transactions on Multimedia Computing, Communications and Applications* 21, 2 (2025), 1–24. https://doi.org/10.1145/3706061

[193] Hung Viet Pham, Thibaud Lutellier, Weizhen Qi, and Lin Tan. 2019. CRADLE: cross-backend validation to detect and localize bugs in deep learning libraries. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 1027–1038. https://doi.org/10.1109/ICSE.2019.00107

[194] Van-Hau Pham, Nguyen Phuc Chuong, Pham Thanh Thai, Phan The Duy, et al. 2024. A coverage-guided fuzzing method for automatic software vulnerability detection using reinforcement learning-enabled multi-level input mutation. *IEEE Access* 12 (2024), 129064–129080.

[195] Xiaofang Qi, Tiangang Zhu, and Yanhui Li. 2024. Coverage-enhanced fault diagnosis for Deep Learning programs: A learning-based approach with hybrid metrics. *Information and Software Technology* 173 (2024), 107488. https://doi.org/10.1016/j.infsof.2024.107488

[196] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PmLR, 8748–8763.

[197] Vikram V Ramaswamy, Sunnie SY Kim, and Olga Russakovsky. 2021. Fair attribute classification through latent space de-biasing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9301–9310. https://doi.org/10.1109/CVPR46437.2021.00918

[198] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. 2020. Adversarial attacks and defenses in deep learning. *Engineering* 6, 3 (2020), 346–360.

[199] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. 2021. A survey of deep active learning. *ACM computing surveys (CSUR)* 54, 9 (2021), 1–40.

[200] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144. https://doi.org/10.1145/2939672.2939778

[201] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32. https://doi.org/10.1609/aaai.v32i1.11491

[202] Maria Rigaki and Sebastian Garcia. 2023. A survey of privacy attacks in machine learning. *Comput. Surveys* 56, 4 (2023), 1–34.

[203] Alexander Robey, Hamed Hassani, and George J Pappas. 2020. Model-based robust deep learning: Generalizing to natural, out-of-distribution data. *arXiv preprint arXiv:2005.10247* (2020).

[204] Andrew Ross, Michael C Hughes, and Finale Doshi-Velez. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. 2662–2670. https://doi.org/10.24963/ijcai.2017/371

[205] Dan Roth and Kevin Small. 2006. Margin-based active learning for structured output spaces. In *Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings 17*. Springer, 413–424. https://doi.org/10.1007/11871842_40

[206] Ripon K Saha, Lingming Zhang, Sarfraz Khurshid, and Dewayne E Perry. 2015. An information retrieval approach for regression test prioritization based on program changes. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, Vol. 1. IEEE, 268–279. https://doi.org/10.1109/ICSE.2015.47

[207] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. 2018. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246* (2018).

[208] Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp. *Tr ansactions of the Association for Computational Linguistics* 9 (2021), 1408–1424. https://doi.org/10.1162/tacl_a_00434

[209] Jessica Schrouff, Alexis Bellot, Amal Rannen-Triki, Alan Malek, Isabela Albuquerque, Arthur Gretton, Alexander D'Amour, and Silvia Chiappa. 2024. Mind the Graph When Balancing Data for Fairness or Robustness. In *Advances in Neural Information Processing Systems (NeurIPS)*. https://arxiv.org/abs/2406.17433 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

[210] Sanjit A Seshia, Ankush Desai, Tommaso Dreossi, Daniel J Fremont, Shromona Ghosh, Edward Kim, Sumukh Shivakumar, Marcell Vazquez-Chanlatte, and Xiangyu Yue. 2018. Formal specification for deep neural networks. In *Automated Technology for Verification and Analysis: 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings 16*. Springer, 20–34. https://doi.org/10.1007/978-3-030-01090-4_2

[211] Ali Shafahi, W Ronny Huang, Mahyar Najibi, Octavian Suciu, Christoph Studer, Tudor Dumitras, and Tom Goldstein. 2018. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems* 31 (2018).

[212] Arnab Sharma, Caglar Demir, Axel-Cyrille Ngonga Ngomo, and Heike Wehrheim. 2021. MLCheck– Property-Driven Testing of Machine Learning Classifiers. In *Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. https://doi.org/10.1109/ICMLA52953.2021.00123

[213] Weijun Shen, Yanhui Li, Lin Chen, Yuanlei Han, Yuming Zhou, and Baowen Xu. 2020. Multiple-boundary clustering and prioritization to promote neural network retraining. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*. 410–422. https://doi.org/10.1145/3324884.3416621

[214] Ying Shi, Beibei Yin, and Jing-Ao Shi. 2025. Markov model based coverage testing of deep learning software systems. *Information and Software Technology* 179 (2025), 107628. https://doi.org/10.1016/j.infsof.2024.107628

[215] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. 2017. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*. IEEE, 3–18. https://doi.org/10.1109/SP.2017.41

[216] Thibault Simonetto, Salah Ghamizi, and Maxime Cordy. 2025. TabularBench: benchmarking adversarial robustness for tabular deep learning in real-world use-cases. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '24)*. Curran Associates Inc., Red Hook, NY, USA, Article 2492, 37 pages.

[217] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034* (2013).

[218] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2018. PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rJUYGxbCW

[219] Ezekiel Soremekun, Sakshi Udeshi, and Sudipta Chattopadhyay. 2022. Astraea: Grammar-based fairness testing. *IEEE Transactions on Software Engineering* 48, 12 (2022), 5188–5211. https://doi.org/10.1109/TSE.2022.3141758

[220] Gregor Stiglic, Primoz Kocbek, Nino Fijacko, Marinka Zitnik, Katrien Verbert, and Leona Cilar. 2020. Interpretability of machine learning-based prediction models in healthcare. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10, 5 (2020), e1379.

[221] Weifeng Sun, Meng Yan, Zhongxin Liu, and David Lo. 2023. Robust test selection for deep neural networks. *IEEE Transactions on Software Engineering* 49, 12 (2023), 5250–5278. https://doi.org/10.1109/TSE.2023.3330982

[222] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. 2018. Testing deep neural networks. *arXiv preprint arXiv:1803.04792* (2018).

[223] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. 2019. Structural test coverage criteria for deep neural networks. *ACM Transactions on Embedded Computing Systems (TECS)* 18, 5s (2019), 1–23. https://doi.org/10.1145/3358233

[224] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *International conference on machine learning*. PMLR, 3319–3328.

[225] Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 1017–1024.

[226] Minxue Tang, Anna Dai, Louis DiValentin, Aolin Ding, Amin Hass, Neil Zhenqiang Gong, Yiran Chen, et al. 2024. {ModelGuard}:{Information-Theoretic} Defense Against Model Extraction Attacks. In *33rd USENIX Security Symposium (USENIX Security 24)*. 5305–5322.

[227] Pan Tang, Tiantian Tang, and Chennuo Lu. 2024. Predicting systemic financial risk with interpretable machine learning. *The North American Journal of Economics and Finance* 71 (2024), 102088.

[228] Md Mehrab Tanjim, Ritwik Sinha, Krishna Kumar Singh, Sridhar Mahadevan, David Arbour, Moumita Sinha, and Garrison W Cottrell. 2022. Generating and controlling diversity in image search. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 411–419. https://doi.org/10.1109/WACV51458.2022.00396

[229] Chuanqi Tao, Yali Tao, Hongjing Guo, Zhiqiu Huang, and Xiaobing Sun. 2023. DLRegion: Coverage-guided fuzz testing of deep neural networks with region-based neuron selection strategies. *Information and Software Technology* 162 (2023), 107266. https://doi.org/10.1016/j.infsof.2023.107266

[230] Guanhong Tao, Weisong Sun, Tingxu Han, Chunrong Fang, and Xiangyu Zhang. 2022. RULER: discriminative and iterative adversarial training for deep neural network fairness. In *Proceedings of the 30th acm joint european software engineering conference and symposium on the foundations of software engineering*. 1173–1184. https://doi.org/10.1145/3540250.3549169

[231] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*. 303–314. https://doi.org/10.1145/3180155.3180220

[232] Yuchi Tian, Ziyuan Zhong, Vicente Ordonez, Gail Kaiser, and Baishakhi Ray. 2020. Testing DNN image classifiers for confusion & bias errors. In *Proceedings of the acm/ieee 42nd international conference on software engineering*. 1122–1134. https://doi.org/10.1145/3377811.3380400

[233] Vale Tolpegin, Stacey Truex, Mehmet Emre Gursoy, and Ling Liu. 2020. Data poisoning attacks against federated learning systems. In *European symposium on research in computer security*. Springer, 480–501.

[234] Florian Tramèr, Fan Zhang, Ari Juels, Michael K Reiter, and Thomas Ristenpart. 2016. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*. 601–618.

[235] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2018. Ensemble Adversarial Training: Attacks and Defenses. In *International Conference on Learning Representations*. https://openreview.net/forum?id=rkZvSe-RZ

[236] Quoc Trung Trinh, Markus Heinonen, Luigi Acerbi, and Samuel Kaski. 2024. Improving robustness to corruptions with multiplicative weight perturbations. *Advances in Neural Information Processing Systems* 37 (2024), 35492–35516.

[237] Cumhur Erkan Tuncali, Georgios Fainekos, Hisahiro Ito, and James Kapinski. 2018. Simulation-based adversarial test generation for autonomous vehicles with machine learning components. In *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 1555–1562. https://doi.org/10.1109/IVS.2018.8500421

[238] Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. 2018. Automated directed fairness testing. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*. 98–108. https://doi.org/10.1145/3238147.3238165

[239] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. *Advances in neural information processing systems* 33 (2020), 12388–12401.

[240] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. 2018. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience* 2018, 1 (2018), 7068349. https://doi.org/10.1155/2018/7068349

[241] Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Suzanne Petryk, Sarah Adel Bargal, and Joseph E. Gonzalez. 2021. {NBDT}: Neural-Backed Decision Tree. In *International Conference on Learning Representations*. https://openreview.net/forum?id=mCLVeEpplNE

[242] Yuxuan Wan, Wenxuan Wang, Pinjia He, Jiazhen Gu, Haonan Bai, and Michael R Lyu. 2023. Biasasker: Measuring the bias in conversational ai system. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 515–527. https://doi.org/10.1145/3611643.3616310

[243] Yao Wan, Shijie Zhang, Hongyu Zhang, Yulei Sui, Guandong Xu, Dezhong Yao, Hai Jin, and Lichao Sun. 2022. You see what i want you to see: poisoning vulnerabilities in neural code search. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1233–1245. https://doi.org/10.1145/3540250.3549153

[244] Dan Wang and Yi Shang. 2014. A new active labeling method for deep learning. In *2014 International joint conference on neural networks (IJCNN)*. IEEE, 112–119. https://doi.org/10.1109/IJCNN.2014.6889457

[245] Feilong Wang, Xin Wang, and Xuegang Jeff Ban. 2024. Data poisoning attacks in intelligent transportation systems: A survey. *Transportation Research Part C: Emerging Technologies* 165 (2024), 104750.

[246] Jingyi Wang, Jialuo Chen, Youcheng Sun, Xingjun Ma, Dongxia Wang, Jun Sun, and Peng Cheng. 2021. Robot: Robustness-oriented testing for deep learning systems. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 300–311. https://doi.org/10.1109/ICSE43902.2021.00038

[247] Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. 2019. Adversarial sample detection for deep neural network through model mutation testing. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 1245–1256. https://doi.org/10.1109/ICSE.2019.00126

[248] Jiannan Wang, Hung Viet Pham, Qi Li, Lin Tan, Yu Guo, Adnan Aziz, and Erik Meijer. 2024. D 3: Differential Testing of Distributed Deep Learning with Model Generation. *IEEE Transactions on Software Engineering* (2024). https://doi.org/10.1109/TSE.2024.3461657

[249] Jie Wang, Zili Wu, Minyan Lu, and Jun Ai. 2024. An Empirical Study on the Effect of Training Data Perturbations on Neural Network Robustness. *Sensors (Basel, Switzerland)* 24, 15 (2024), 4874.

[250] Shuai Wang and Zhendong Su. 2020. Metamorphic Object Insertion for Testing Object Detection Systems. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 1053–1065. https://doi.org/10.1145/3324884.3416584

[251] Tianyang Wang, Xingjian Li, Pengkun Yang, Guosheng Hu, Xiangrui Zeng, Siyu Huang, Cheng-Zhong Xu, and Min Xu. 2022. Boosting active learning via improving test performance. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 8566–8574. https://doi.org/10.1609/aaai.v36i8.20834

[252] Xin Wang, Kai Chen, Jiaming Zhang, Jingjing Chen, and Xingjun Ma. 2025. Tapt: Test-time adversarial prompt tuning for robust inference in vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 19910–19920.

[253] Zan Wang, Ming Yan, Junjie Chen, Shuang Liu, and Dongdi Zhang. 2020. Deep learning library testing via effective model generation. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 788–799. https://doi.org/10.1145/3368089.3409761

[254] Zan Wang, Hanmo You, Junjie Chen, Yingyi Zhang, Xuyuan Dong, and Wenbin Zhang. 2021. Prioritizing test inputs for deep neural networks via mutation analysis. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 397–409. https://doi.org/10.1109/ICSE43902.2021.00046

[255] Michael Weiss, Rwiddhi Chakraborty, and Paolo Tonella. 2021. A review and refinement of surprise adequacy. In *2021 IEEE/ACM Third International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest)*. IEEE, 17–24. https://doi.org/10.1109/DeepTest52559.2021.00009

[256] Michael Weiss and Paolo Tonella. 2022. Simple techniques work surprisingly well for neural network test prioritization and active learning (replicability study). In *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*. 139–150. https://doi.org/10.1145/3533767.3534375

[257] Mike Wu, Sonali Parbhoo, Michael Hughes, Ryan Kindle, Leo Celi, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. 2020. Regional tree regularization for interpretability in deep neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 6413–6421. https://doi.org/10.1609/aaai.v34i04.6112

[258] Yutong Wu, Han Qiu, Shangwei Guo, Jiwei Li, and Tianwei Zhang. 2024. You only query once: an efficient label-only membership inference attack. In *The Twelfth International Conference on Learning Representations*.

[259] Jun Xia, Ting Wang, Jiepin Ding, Xian Wei, and Mingsong Chen. 2022. Eliminating Backdoor Triggers for Deep Neural Networks Using Attention Relation Graph Distillation. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, Lud De Raedt (Ed.). International Joint Conferences on Artificial Intelligence Organization, 1481–1487. https://doi.org/10.24963/ijcai.2022/206

[260] Dongwei Xiao, Zhibo Liu, Yuanyuan Yuan, Qi Pang, and Shuai Wang. 2022. Metamorphic testing of deep learning compilers. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 6, 1 (2022), 1–28. https://doi.org/10.1145/3508035

[261] Binhui Xie, Longhui Yuan, Shuang Li, Chi Harold Liu, Xinjing Cheng, and Guoren Wang. 2022. Active learning for domain adaptation: An energy-based approach. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 36. 8708–8716. https://doi.org/10.1609/aaai.v36i8.20850

[262] Xiaofei Xie, Tianlin Li, Jian Wang, Lei Ma, Qing Guo, Felix Juefei-Xu, and Yang Liu. 2022. Npc: N euron p ath c overage via characterizing decision logic of deep neural networks. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 31, 3 (2022), 1–27. https://doi.org/10.1145/3490489

[263] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. 2019. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 146–157. https://doi.org/10.1145/3293882.3330579

[264] Depeng Xu, Shuhan Yuan, Lu Zhang, and Xintao Wu. 2018. Fairgan: Fairness-aware generative adversarial networks. In *2018 IEEE international conference on big data (big data)*. IEEE, 570–575. https://doi.org/10.1109/BigData.2018.8622525

[265] Han Xu, Yao Ma, Hao-Chen Liu, Debayan Deb, Hui Liu, Ji-Liang Tang, and Anil K Jain. 2020. Adversarial attacks and defenses in images, graphs and text: A review. *International journal of automation and computing* 17, 2 (2020), 151–178.

[266] Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proceedings 2018 Network and Distributed System Security Symposium (NDSS)*. Internet Society.

[267] Zhiyi Xue, Si Liu, Zhaodi Zhang, Yiting Wu, and Min Zhang. 2023. A tale of two approximations: tightening over-approximation for dnn robustness verification via under-approximation. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*. 1182–1194. https://doi.org/10.1145/3597926.3598127

[268] Yazhou Yang and Marco Loog. 2022. To actively initialize active learning. *Pattern Recognition* 131 (2022), 108836. https://doi.org/10.1016/j.patcog.2022.108836

[269] Zhou Yang, Jieke Shi, Muhammad Hilmi Asyrofi, Bowen Xu, Xin Zhou, DongGyun Han, and David Lo. 2025. Prioritizing speech test cases. *ACM Transactions on Software Engineering and Methodology* 34, 4 (2025), 1–27. https://doi.org/10.1145/3707450

[270] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. 2018. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*. IEEE, 268–282.

[271] Donggeun Yoo and In So Kweon. 2019. Learning loss for active learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 93–102. https://doi.org/10.1109/CVPR.2019.00018

[272] Jing Yu, Shukai Duan, and Xiaojun Ye. 2022. A white-box testing for deep neural networks based on neuron coverage. *IEEE transactions on neural networks and learning systems* 34, 11 (2022), 9185–9197. https://doi.org/10.1109/TNNLS.2022.3156620

[273] Lijia Yu, Yihan Wang, and Xiao-Shan Gao. 2023. Adversarial parameter attack on deep neural networks. In *International Conference on Machine Learning*. PMLR, 40354–40372.

[274] Hanyang Yuan, Jiarong Xu, Renhong Huang, Mingli Song, Chunping Wang, and Yang Yang. 2024. Can Graph Neural Networks Expose Training Data Properties? An Efficient Risk Assessment Approach. In *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (Eds.), Vol. 37. Curran Associates, Inc., 69361–69385. https://proceedings.neurips.cc/paper_files/paper/2024/file/806288e682d8a38c0bf21e37ab38af0a-Paper-Conference.pdf

[275] Seyma Yucer, Samet Akçay, Noura Al-Moubayed, and Toby P Breckon. 2020. Exploring racial bias within face recognition via per-subject adversarially-enabled data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 18–19. https://doi.org/10.1109/CVPRW50498.2020.00017

[276] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. 2020. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access* 8 (2020), 58443–58469.

[277] Abdelrahman Zayed, Gonçalo Mordido, Ioana Baldini, and Sarath Chandar. 2024. Why Don't Prompt-Based Fairness Metrics Correlate?. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Bangkok, Thailand, 9002–9019. https://doi.org/10.18653/v1/2024.acl-long.487

[278] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. 2018. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. 335–340. https://doi.org/10.1145/3278721.3278779

[279] Fuyuan Zhang, Xinwen Hu, Lei Ma, and Jianjun Zhao. 2023. DeepRover: A Query-Efficient Blackbox Attack for Deep Neural Networks. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1384–1394. https://doi.org/10.1145/3611643.3616370

[280] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*. PMLR, 7472–7482.

[281] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* 48, 1 (2020), 1–36. https://doi.org/10.1109/TSE.2019.2962027

[282] Mengdi Zhang and Jun Sun. 2022. Adaptive fairness improvement based on causality analysis. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 6–17. https://doi.org/10.1145/3540250.3549103

[283] Mengdi Zhang, Jun Sun, Jingyi Wang, and Bing Sun. 2023. TESTSGD: Interpretable testing of neural networks against subtle group discrimination. *ACM Transactions on Software Engineering and Methodology* 32, 6 (2023), 1–24. https://doi.org/10.1145/3591869

[284] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems. In *Proceedings of the 33rd ACM/IEEE international conference on automated software engineering*. 132–142.

[285] Peixin Zhang, Jingyi Wang, Jun Sun, Guoliang Dong, Xinyu Wang, Xingen Wang, Jin Song Dong, and Ting Dai. 2020. White-box fairness testing through adversarial sampling. In *Proceedings of the ACM/IEEE 42nd international conference on software engineering*. 949–960. https://doi.org/10.1145/3377811.3380331

[286] Wei Emma Zhang, Quan Z Sheng, Ahoud Alhazmi, and Chenliang Li. 2020. Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11, 3 (2020), 1–41.

[287] Xiyue Zhang, Xiaofei Xie, Lei Ma, Xiaoning Du, Qiang Hu, Yang Liu, Jianjun Zhao, and Meng Sun. 2020. Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. 739–751. https://doi.org/10.1145/3377811.3380368

[288] Xianglong Zhang, Huanle Zhang, Guoming Zhang, Hong Li, Dongxiao Yu, Xiuzhen Cheng, and Pengfei Hu. 2023. Model poisoning attack on neural network without reference data. *IEEE Trans. Comput.* 72, 10 (2023), 2978–2989. https://doi.org/10.1109/TC.2023.3280133

[289] Yinan Zhang and Mingqiang An. 2016. An active learning classifier for further reducing diabetic retinopathy screening system cost. *Computational and mathematical methods in medicine* 2016, 1 (2016), 4345936. https://doi.org/10.1155/2016/4345936

[290] Yihao Zhang, Hangzhou He, Jingyu Zhu, Huanran Chen, Yifei Wang, and Zeming Wei. 2024. On the Duality Between Sharpness-Aware Minimization and Adversarial Training. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*. PMLR.

[291] Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. 2020. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 253–261.

[292] Yu Zhang, Peter Tiňo, Aleš Leonardis, and Ke Tang. 2021. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence* 5, 5 (2021), 726–742. https://doi.org/10.1109/TETCI.2021.3100641

[293] Yingyi Zhang, Zan Wang, Jiajun Jiang, Hanmo You, and Junjie Chen. 2022. Toward improving the robustness of deep learning models via model transformation. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. 1–13. https://doi.org/10.1145/3551349.3556920

[294] Zhaoyu Zhang, Yang Hua, Hui Wang, and Seán McLoone. 2024. Improving the fairness of the min-max game in GANs training. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 2910–2919. https://doi.org/10.1109/WACV57701.2024.00289

[295] Ziqi Zhang, Yuanchun Li, Bingyan Liu, Yifeng Cai, Ding Li, Yao Guo, and Xiangqun Chen. 2023. Fedslice: Protecting federated learning models from malicious participants with model slicing. In *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. IEEE, 460–472. https://doi.org/10.1109/ICSE48619.2023.00049

[296] Guang Zhao, Edward Dougherty, Byung-Jun Yoon, Francis Alexander, and Xiaoning Qian. 2021. Efficient active learning for Gaussian process classification by error reduction. *Advances in Neural Information Processing Systems* 34 (2021), 9734–9746.

[297] Shiji Zhao, Ranjie Duan, Xizhe Wang, and Xingxing Wei. 2024. Improving adversarial robust fairness via anti-bias soft label distillation. *Advances in Neural Information Processing Systems* 37 (2024), 89125–89149.

[298] Zhe Zhao, Guangke Chen, Tong Liu, Taishan Li, Fu Song, Jingyi Wang, and Jun Sun. 2024. Attack as detection: Using adversarial attack methods to detect abnormal examples. *ACM Transactions on Software Engineering and Methodology* 33, 3 (2024), 1–45. https://doi.org/10.1145/3631977

[299] Zengqun Zhao, Ziquan Liu, Yu Cao, Shaogang Gong, and Ioannis Patras. 2025. AIM-Fair: Advancing Algorithmic Fairness via Selectively Fine-Tuning Biased Models with Contextual Synthetic Data. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 28748–28758.

[300] Guangyao Zheng, Michael A Jacobs, Vladimir Braverman, and Vishwa S Parekh. 2025. Towards Fair Medical AI: Adversarial Debiasing of 3D CT Foundation Embeddings. *arXiv preprint arXiv:2502.04386* (2025).

[301] Haibin Zheng, Jinyin Chen, and Haibo Jin. 2023. CertPri: certifiable prioritization for deep neural networks via movement cost in feature space. In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 1–13. https://doi.org/10.1109/ASE56229.2023.00126

[302] Haibin Zheng, Zhiqing Chen, Tianyu Du, Xuhong Zhang, Yao Cheng, Shouling Ji, Jingyi Wang, Yue Yu, and Jinyin Chen. 2022. Neuronfair: Interpretable white-box fairness testing through biased neuron identification. In *Proceedings of the 44th International Conference on Software Engineering*. 1519–1531. https://doi.org/10.1145/3510003.3510123

[303] Xin Zheng, Miao Zhang, Chunyang Chen, Soheila Molaei, Chuan Zhou, and Shirui Pan. 2023. Gnnevaluator: Evaluating gnn performance on unseen graphs without labels. *Advances in Neural Information Processing Systems* 36 (2023), 33606–33623.

[304] Ziyuan Zhong, Yuchi Tian, and Baishakhi Ray. 2021. Understanding local robustness of deep neural networks under natural variations. In *International Conference on Fundamental Approaches to Software Engineering*. Springer International Publishing Cham, 313–337. https://doi.org/10.1007/978-3-030-71500-7_16

[305] Jianyi Zhou, Feng Li, Jinhao Dong, Hongyu Zhang, and Dan Hao. 2020. Cost-effective testing of a deep learning model through input reduction. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 289–300. https://doi.org/10.1109/ISSRE5003.2020.00035

[306] Shuai Zhou, Chi Liu, Dayong Ye, Tianqing Zhu, Wanlei Zhou, and Philip S Yu. 2022. Adversarial attacks and defenses in deep learning: From a perspective of cybersecurity. *Comput. Surveys* 55, 8 (2022), 1–39.

[307] Hong Zhu, Patrick AV Hall, and John HR May. 1997. Software unit test coverage and adequacy. *Acm computing surveys (csur)* 29, 4 (1997), 366–427.

[308] Zhijie Zhu, Lei Fan, Maurice Pagnucco, and Yang Song. 2025. Interpretable Image Classification via Non-parametric Part Prototype Learning. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 9762–9771.

[309] Amirhossein Zolfagharian, Manel Abdellatif, Lionel C Briand, Mojtaba Bagherzadeh, et al. 2023. A search-based testing approach for deep reinforcement learning agents. *IEEE Transactions on Software Engineering* 49, 7 (2023), 3715–3735. https://doi.org/10.1109/TSE.2023.3269804