Fairness Testing: A Comprehensive Survey and Analysis of Trends

ZHENPENG CHEN, University College London, London, United Kingdom
JIE M. ZHANG, King's College London, London, United Kingdom
MAX HORT, Simula Research Laboratory, Oslo, Norway
MARK HARMAN, University College London, London, United Kingdom
FEDERICA SARRO, University College London, London, United Kingdom

Unfair behaviors of Machine Learning (ML) software have garnered increasing attention and concern among software engineers. To tackle this issue, extensive research has been dedicated to conducting fairness testing of ML software, and this article offers a comprehensive survey of existing studies in this field. We collect 100 papers and organize them based on the testing workflow (i.e., how to test) and testing components (i.e., what to test). Furthermore, we analyze the research focus, trends, and promising directions in the realm of fairness testing. We also identify widely adopted datasets and open-source tools for fairness testing.

CCS Concepts: • **Software and its engineering → Software creation and management**; • **Computing methodologies → Machine learning;**

Additional Key Words and Phrases: Machine learning, fairness testing, survey, analysis, trends

ACM Reference Format:
Zhenpeng Chen, Jie M. Zhang, Max Hort, Mark Harman, and Federica Sarro. 2024. Fairness Testing: A Comprehensive Survey and Analysis of Trends. *ACM Trans. Softw. Eng. Methodol.* 33, 5, Article 137 (June 2024), 59 pages. https://doi.org/10.1145/3652155

1 INTRODUCTION

Machine Learning (ML)-enabled software, commonly referred to as ML software, has gained widespread adoption in critical areas of society, including hiring [131], credit assessment [119], and criminal justice [113]. However, the use of such software has also led to instances of unfair decision-making, particularly when sensitive attributes such as sex, race, age, and occupation are involved [121]. For instance, an ML-enabled recidivism assessment system employed by US courts was found to incorrectly label black defendants as higher-risk individuals compared to white defendants [25].

Zhenpeng Chen, Federica Sarro, and Mark Harman are supported by the ERC Advanced Grant No.741278 (EPIC: Evolutionary Program Improvement Collaborators). Max Hort is supported by the Research Council of Norway through the secureIT project (IKTPLUSS #288787).
Authors' addresses: Z. Chen, M. Harman, and F. Sarro, Department of Computer Science, University College London, Gower Street, London WC1E 6BT, United Kingdom; e-mails: zp.chen@ucl.ac.uk, mark.harman@ucl.ac.uk, f.sarro@ucl.ac.uk; J. M. Zhang (Corresponding author), Department of Informatics, King's College London, 30 Aldwych, London WC2B 4BG, United Kingdom; e-mail: jie.zhang@kcl.ac.uk; M. Hort, Simula Research Laboratory, Kristian Augusts gate 23, 0164 Oslo, Norway; e-mail: maxh@simula.no.

ACM 1049-331X/2024/06-ART137
https://doi.org/10.1145/3652155

The unfair behaviors exhibited by ML software can have profound ethical implications, resulting in unacceptable outcomes, particularly by disadvantaging minority groups and protected categories. Consequently, there is a growing concern and heightened awareness within the research community regarding the issue of unfairness and its impact.

The exploration of fairness issues gained significant momentum in the late 1960s [187], when psychometricians began investigating the fairness of educational tests [142, 143]. In the 1980s, researchers delved into the social impact of technology [206]. Subsequently, the study of fairness expanded to include the domains of ML [240] and **Software Engineering (SE)** [159], beginning around 2008, in response to the rapid adoption of ML in software applications supporting decision-making processes.

From the SE perspective, fairness is a non-functional software property that should be treated as a first-class entity throughout the entire SE process [104, 121]. Ahmad et al. [98] emphasized the increasing importance of fairness as a requirement that should be considered during the *requirements engineering* phase of ML software development. Alidoosti [100] argued for the inclusion of fairness considerations in the *design* process of software architecture. Albarghouthi et al. [99] framed fairness as correctness properties for ML program *verification*. Zhang et al. [304] described fairness as a significant *testing* property for ML software. Additionally, Zhang et al. [300] viewed fairness as the objective of *repairing* ML software.

In this article, we also approach fairness from the SE perspective and define imperfections in software systems that result in a misalignment between desired fairness conditions and actual outcomes as *fairness bugs*. Our focus is on *fairness testing* of ML software, which aims to uncover fairness bugs through code execution. Fairness testing represents an important aspect of software fairness research and is closely intertwined with other activities in the SE process. It verifies whether software systems meet fairness requirements, exposes fairness bugs introduced during software implementation, and guides software repair efforts to address fairness issues, among other related SE activities.

Compared to traditional software testing, fairness testing presents distinct challenges. For example, in traditional software testing, a test oracle typically relies on the output of a single input. In fairness testing, the oracle problem is more challenging, because inputs and outputs from different demographic groups need to be considered simultaneously. Additionally, there are diverse fairness definitions, some of which may even conflict with each other or be mathematically impossible to satisfy concurrently [123, 282]. Since different definitions can require different test oracles, designing fairness testing techniques to accommodate this multitude of definitions poses a significant problem as well.

The significance of fairness testing and its associated challenges has led to a notable increase in research efforts in this field. Figure 1 illustrates the cumulative number of publications on fairness testing until 2023, revealing a growing interest and emphasizing the relevance of this survey. Notably, 89% of fairness testing publications have emerged since 2019, indicating the emergence of this new domain of software testing.

This article offers a comprehensive survey of fairness testing in ML software. The collected papers are sourced from various venues including SE, artificial intelligence, computer security, and human-computer interaction. We categorize these papers based on two key aspects: the fairness testing workflow (i.e., how to test) and fairness testing components (i.e., what to test). Furthermore, we conduct an analysis of research trends and identify potential research opportunities for the fairness testing community. Additionally, we provide an overview of publicly accessible datasets and open-source tools available for fairness testing.

Previous surveys have explored various aspects of fairness in ML and related fields. Mehrabi et al. [224] and Pessach and Shmueli [243] surveyed fairness research on ML algorithms, while

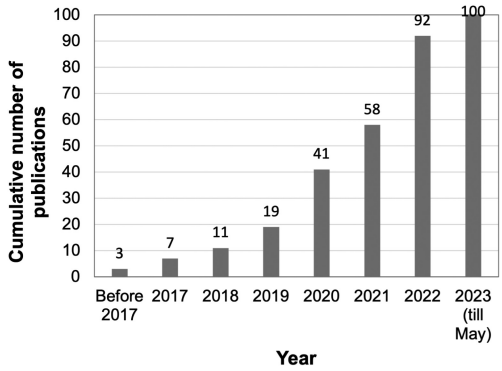

Fig. 1. Cumulative number of publications on fairness testing.

Hort et al. [181] focused on bias[1] mitigation methods for ML classifiers. Sun et al. [267], Berk et al. [113], and Pitoura et al. [245] surveyed techniques for improving fairness in specific ML tasks, such as natural language processing, criminal justice risk assessment, and ranking. Tushev et al. [277] surveyed software design strategies for fairness in digital sharing economy applications. Hutchinson and Mitchell [187] provided a historical perspective on fairness assessment tests across disciplines, including education and hiring. Zhang et al. [304] conducted a broader survey on ML testing, considering fairness as one of several testing properties. A recent systematic literature review [263] focused on software fairness, covering only 20 fairness testing papers. In contrast, our survey specifically concentrates on fairness testing of ML software. To the best of our knowledge, this is the first comprehensive survey specifically dedicated to the literature on fairness testing.

To summarize, this work makes the following contributions:

— It provides a comprehensive survey of 100 fairness testing papers, encompassing diverse research communities.
— It defines fairness bug and fairness testing and provides an overview of the testing workflow and testing components related to fairness testing.
— It compiles a summary of public datasets and open-source tools for fairness testing, providing a navigation for researchers and practitioners interested in the field.
— It analyzes research trends and identifies promising research opportunities in fairness testing, aiming to foster further advancements in this area.

The structure of the article is illustrated in Figure 2, and the detailed survey methodology is presented in Section 3.

2 PRELIMINARIES

In this section, we begin by presenting widely adopted fairness definitions. Subsequently, we offer a definition of fairness bug and fairness testing from the perspective of SE. We further outline the testing workflow and testing components of fairness testing.

2.1 Definition of Fairness

The definition of fairness plays a crucial role in establishing the fairness conditions that software systems are expected to meet. Over the years, researchers and practitioners have proposed and

[1]The terms "bias" and "unfairness" are often used interchangeably in the literature, as they both denote deviations from "fairness" [224].

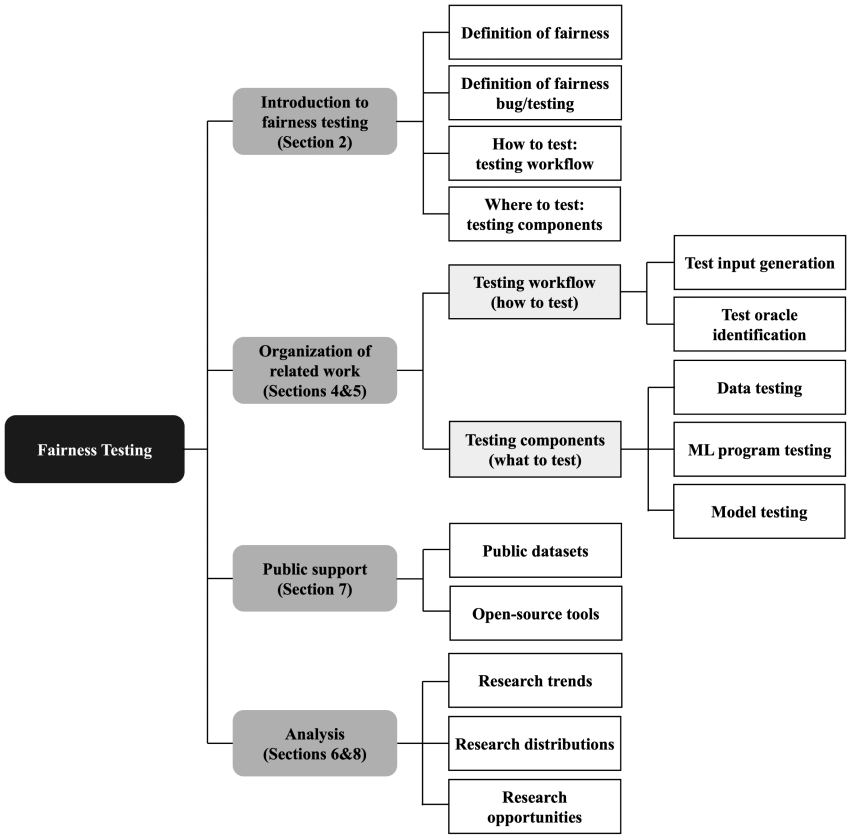

Fig. 2. Structure of this article.

Table 1. Widely Adopted Fairness Definitions

Name	Fairness type
Fairness through unawareness [171]	Individual fairness
Fairness through awareness [154]	Individual fairness
Counterfactual fairness [202]	Individual fairness
Causal fairness [160]	Individual fairness
Statistical parity [109]	Group fairness
Equalized odds [176]	Group fairness
Equal opportunity [176]	Group fairness

explored various fairness definitions [123, 179, 225]. This section aims to present the fairness definitions that have received widespread adoption in the literature [304], as listed in Table 1.

These definitions primarily fall into two categories: *individual fairness* and *group fairness* [224]. Individual fairness requires that software should produce similar predictive outcomes for similar individuals, while group fairness requires software to treat different demographic groups in a similar manner. Fairness assessment in the context of ML software often relies on *sensitive attributes*, also known as *protected attributes*, which represent characteristics that require protection against unfairness, such as sex, race, age, and physical ability. By considering sensitive attributes, the population can be categorized into privileged groups and unprivileged groups.

To facilitate the formalization of fairness, we introduce the necessary notations:

— X: Denotes the input feature vector of ML software, excluding sensitive attributes. $X^{(i)}$ denotes the feature vector for individual i.
— A: Denotes sensitive attributes, where $A^{(i)}$ denotes the sensitive attributes for individual i.
— Y: Denotes the actual outcome of ML software.
— $\hat{Y}$: Denotes the predicted outcome, with $\hat{Y}(X^{(i)}, A^{(i)})$ indicating the predicted outcome for individual i.
— $P(\cdot)$: Denotes the probability function.

For Y and $\hat{Y}$, we use F to denote a favorable outcome and UF to denote an unfavorable outcome. Given that ML inputs of diverse data types can be encoded as vectors, these definitions of fairness are applicable to a wide range of data types.

2.1.1 Individual Fairness. We first introduce four widely adopted definitions of individual fairness.

Fairness through unawareness [171] assumes that a software system can achieve fair outcomes by refraining from explicitly using sensitive attributes in the decision-making process. By excluding these attributes, the system cannot rely on them and is thus expected to produce the same outcome for individuals with identical non-sensitive features. Formally, fairness through awareness assumes that, for individuals i and j, if $X^{(i)} = X^{(j)}$, then $\hat{Y}(X^{(i)}) = \hat{Y}(X^{(j)})$.

Fairness through awareness [154] requires a software system to produce similar outcomes for similar individuals. To achieve this, two distance metrics are employed: $d(\cdot, \cdot)$ measures the similarity of individuals for a specific task, and $D(\cdot, \cdot)$ measures the similarity of probability distributions over outcomes. Formally, fairness through awareness dictates that $D(\hat{Y}(X^{(i)}, A^{(i)}), \hat{Y}(X^{(j)}, A^{(j)})) \leq d((X^{(i)}, A^{(i)}), (X^{(j)}, A^{(j)}))$. In other words, the distributions over predicted outcomes for two individuals should be indistinguishable within their measured similarity/distance. Thus, if two individuals are similar, then they should receive similar predicted outcomes.

Counterfactual fairness [202] states that an individual's prediction should remain the same in the real world as well as in a counterfactual world where the individual belongs to a different demographic group (i.e., the sensitive attribute is different). In practical applications, the input X may contain features that have a causal relationship with the sensitive attribute A. Therefore, when the attribute $A = a$ is changed to the counterfactual value $A = a'$, the input $X = x$ will also be transformed to $X = x'$, where features causally linked to the sensitive attribute will be altered accordingly. Formally, under any context $X = x$ and $A = a$: $P(\hat{Y}(x, a) = y | X = x, A = a) = P(\hat{Y}(x', a') = y | X = x, A = a)$, for all y and for any value a' attainable by A.

Causal fairness [160] captures the causal relationship between sensitive attributes and outcomes. It involves running a software system on an input, modifying the sensitive attribute(s), and checking whether this modification leads to a change in the output. Formally, causal fairness requires that for any individual i, there should be no other instance j that simultaneously satisfies the following conditions: (1) $X^{(i)} = X^{(j)}$; (2) $A^{(i)} \neq A^{(j)}$; (3) $\hat{Y}(X^{(i)}, A^{(i)}) \neq \hat{Y}(X^{(j)}, A^{(j)})$.

A notable distinction between fairness through unawareness and other fairness definitions lies in its deliberate exclusion of sensitive attributes from the decision-making process. In contrast, fairness through awareness incorporates sensitive attribute information depending on the definition of individual similarity; counterfactual fairness and causal fairness make use of sensitive attributes to comprehend their relationship with outcomes.

Next, we discuss the difference among fairness through awareness, counterfactual fairness, and causal fairness. Fairness through awareness is a more general concept compared to counterfactual fairness and causal fairness. The alignment between fairness through awareness and counterfactual fairness emerges when the similarity metric is crafted by contemplating the likeness of

Table 2. Advantages and Disadvantages of Different Individual Fairness Definitions

Name	Advantages	Disadvantages
Fairness through unawareness	Straightforward solution by avoiding explicit use of sensitive attributes.	Ignores correlations between non-sensitive features and sensitive attributes.
Fairness through awareness	(1) Considers both the similarity of individuals and the similarity of outcome distributions. (2) Flexible similarity definition for different scenarios.	(1) Choice of distance metrics can impact results and may require fine-tuning. (2) Sensitivity to the definition of similarity, which can vary across scenarios.
Counterfactual fairness	(1) Incorporates causal reasoning to identify discriminatory effects. (2) Considers an impact of changes in sensitive attributes on both non-sensitive features and predictions.	(1) Requires knowledge of causal relationships between non-sensitive features and sensitive attributes. (2) Practical implementation may be challenging, especially when causal relationships are complex.
Causal fairness	(1) Incorporates causal reasoning to identify discriminatory effects. (2) Easy to implement compared to counterfactual fairness.	Ignores potential relationships between non-sensitive features and sensitive attributes.

samples in counterfactual worlds [202]. However, when the similarity metric is grounded in the similarities of non-sensitive attributes, fairness through awareness can be likened to causal fairness [282]. While counterfactual fairness and causal fairness share similarities in their approach to individual fairness—both adopting a causal standpoint that evaluates fairness by manipulating sensitive attribute information—they diverge in certain aspects. Specifically, causal fairness simplifies this perspective by modifying only sensitive attribute information without considering the causal relationships among sensitive attributes and other features. In contrast, counterfactual fairness changes both sensitive attribute information and non-sensitive features that are causally linked to sensitive attributes.

Table 2 summarizes advantages and disadvantages of different individual fairness definitions.

2.1.2 Group Fairness. We introduce three widely adopted definitions of group fairness. To better illustrate the distinctions between these definitions, we use a credit score software system as an example, with sex as the protected attribute of applicants.

Statistical parity [109], also known as **demographic parity**, requires the probability of a favorable outcome to be the same among different demographic groups. In other words, a software system satisfies statistical parity if, for any two possible values (denoted as a and a') of the sensitive attribute set, the probability of obtaining favorable outcomes is the same: $P[\hat{Y} = F|A = a] = P[\hat{Y} = F|A = a']$. For instance, the credit score system should assign a good score to male and female applicants in equal proportions.

Equalized odds [176] requires that the privileged and the unprivileged groups have equal true positive rates and equal false positive rates. In other words, the prediction is independent of sensitive attributes when the target label Y is fixed. For any two possible values (denoted as a and a') of the sensitive attribute set, $P[\hat{Y} = F|A = a, Y = y] = P[\hat{Y} = F|A = a', Y = y]$ for $y \in \{F, UF\}$. In our example, the probability of correctly assigning a good predicted credit score to an applicant with an actual good credit score, and the probability of incorrectly assigning a good predicted credit score to an applicant with an actual bad credit score, should be equal for male and female applicants.

Equal opportunity [176] states that the privileged and the unprivileged groups have equal true positive rates. In other words, the prediction is independent of sensitive attributes when the target label Y is fixed as F: For any two possible values (denoted as a and a') of the sensitive attribute set, $P[\hat{Y} = F|A = a, Y = F] = P[\hat{Y} = F|A = a', Y = F]$. In our example, the probability of accurately assigning a good predicted credit score to an applicant with an actual good credit score should be the same for male and female applications.

Table 3 summarizes advantages and disadvantages of different group fairness definitions.

More about fairness definitions can be found in recent work on surveying, analyzing, and comparing them. Mitchell et al. [225] compiled a comprehensive catalog of fairness definitions found

Table 3. Advantages and Disadvantages of Different Group Fairness Definitions

Name	Advantages	Disadvantages
Statistical parity	(1) Easy to understand and implement. (2) Promotes fairness at a high level, focusing on overall outcome balance.	(1) May not address nuanced biases in specific performance metrics. (2) Ignores variations in predictive performance between groups.
Equalized odds	(1) Considers both positive and negative predictive performance. (2) Addresses potential disparities in error rates between groups.	(1) Stricter conditions may be challenging to satisfy in practice. (2) May be sensitive to class imbalances and prevalence differences.
Equal opportunity	(1) Focuses specifically on fairness in positive predictions. (2) Emphasizes equal opportunities for positive outcomes.	(1) Does not consider false positive rates, potentially overlooking negative consequences. (2) Similar to equalized odds, may face challenges in practical implementation.

in the literature, offering a valuable resource for understanding different perspectives. Verma and Rubin [282] provided insights into the rationale behind existing fairness definitions and examined their relationships through a detailed case study. Castelnovo et al. [123] delved into the differences, implications, and orthogonal aspects of various fairness definitions, shedding light on their distinct characteristics. Additionally, Mehrabi et al. [224] developed a taxonomy of fairness definitions proposed specifically for ML algorithms.

2.2 Definition of Fairness Bug and Fairness Testing

A software bug is an imperfection in a computer program that causes a discordance between existing and required conditions [9]. Based on this definition, prior research [304] defines "ML bug" as any imperfection in an ML item that causes a discordance between the existing and the required conditions and "ML testing" as any activity designed to reveal ML bugs. Aligning with the terminology used in SE, we define fairness bug and fairness testing as follows:

— **Definition (Fairness Bug).** *A fairness bug refers to any imperfection in a software system that causes a discordance between the existing and required fairness conditions.*

The required fairness conditions depend on the fairness definition adopted by the developers of the software under test. The imperfections can manifest as unfair predictions that violate individual fairness or discrepancies in outcomes among different demographic groups that surpass a predetermined threshold for group fairness. Previous studies have also referred to such imperfections as fairness defects [121] or fairness issues [159]. In this article, we adopt the term "fairness bug" as a representative of these related concepts, as "bug" carries a broader meaning [9].

The presence of fairness bugs has spurred research efforts towards developing automated testing techniques to detect these bugs, i.e., fairness testing.

— **Definition (Fairness Testing)**. *Fairness testing refers to any activity designed to reveal fairness bugs through code execution.* Formally, consider a software system S under test, a set of inputs I, the required fairness condition C, and the existing fairness condition C'. Fairness testing refers to any activity involving the execution of I on S to identify a discordance between C and C'.

This definition categorizes fairness testing as a subset of software testing, specifically excluding manual inspection and formal verification methods. Since we focus on fairness testing of ML software, we follow the recent ML testing survey [304] to consider two key aspects of this emerging testing domain: *testing workflow* and *testing components*.

2.3 Fairness Testing Workflow

The fairness testing workflow refers to *how* to conduct fairness testing with different testing activities. This section outlines its key activities.

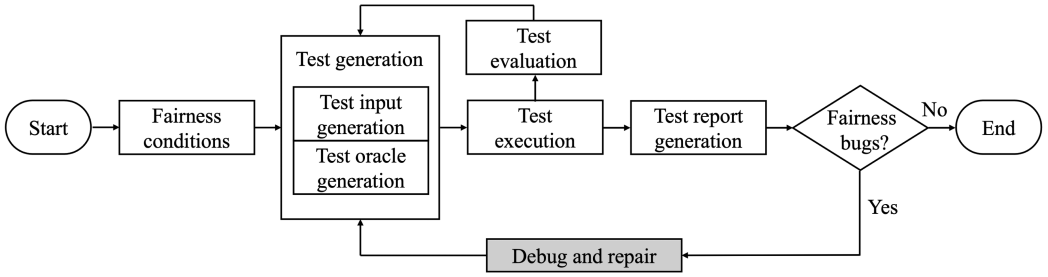

Fig. 3. Workflow of fairness testing.

Figure 3 presents an overview of the fairness testing workflow for ML software, which builds upon the established ML testing workflow in previous work [304]. In this process, software engineers establish and specify the desired fairness conditions (also often referred to as fairness requirements) for the software under test. These fairness conditions primarily stem from individual fairness or group fairness definitions. Based on the defined fairness conditions, test oracles are identified and created. Test inputs are obtained either through sampling from collected data or generation. The test inputs are then executed on the software under test to determine if any violations of the fairness conditions occur. Engineers assess the adequacy of the tests, evaluating their effectiveness in uncovering fairness bugs. Simultaneously, the results of the test execution provide a bug report that assists engineers in reproducing, locating, and resolving any identified fairness bugs. The fairness testing process can be iterated to ensure that the repaired software aligns with the desired fairness conditions. Upon satisfying the fairness conditions, the software can be deployed for use. Following deployment, engineers can employ runtime monitoring to continually test the software using new real-world inputs, ensuring ongoing adherence to the fairness conditions.

Note that, as in ML testing [304], test inputs in fairness testing can take diverse forms based on the testing components of ML software. Common testing components include data, ML programs, and ML models, as described in Section 2.4. For testing data, test inputs can be programs capable of detecting data bias [304]; for testing ML programs and models, test inputs typically involve data instances used to validate ML behaviors [304]. Unlike traditional software testing, which can rely on a single data instance as the test input, fairness testing of ML programs and models typically uses pairs of data instances (for individual fairness testing) or a set of data instances with diverse demographic information (for group fairness testing).

Section 4 organizes fairness testing papers based on the testing workflow perspective. The existing fairness testing literature primarily addresses test input generation (Section 4.1) and test oracle identification (Section 4.2), while other testing activities remain open challenges and present research opportunities for the community (discussed in Section 8).

2.4 Fairness Testing Components

Software testing can be conducted for different testable parts within a software system [203]. In the context of fairness testing, this section focuses on introducing the testable components within ML software.

We denote the entire ML software used by end-users as S, where S encompasses both ML components and non-ML components [216, 230], as illustrated in Figure 4. In the ML components, an ML model M is trained using large-scale data D, comprising sensitive attributes A, remaining feature vectors X, and data labels Y. An ML program G is implemented based on ML frameworks F (e.g., scikit-learn [239], TensorFlow [96], and Keras [172]) [134, 304], and it specifies the structure

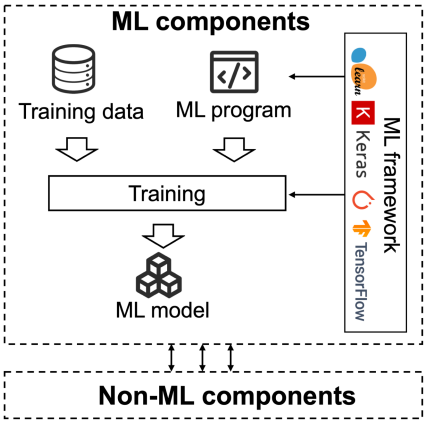

Fig. 4. Components to test in ML software.

of the desired ML model and the process of obtaining it through the training data [134]. To translate ML components into a functional software system, various non-ML components denoted as C are necessary, such as data storage, user interfaces, and monitoring infrastructures. All these components are interconnected and work together to achieve the software's objective.

In Section 2.2, we define fairness testing as any activity aimed at uncovering fairness bugs in the ML software S. According to this definition, fairness testing for S should encompass testing activities on each component, including the training data D, ML program G, ML frameworks F, ML model M, and non-ML components C. This layered approach to fairness testing addresses diverse aspects of the ML software S, facilitating a comprehensive evaluation that identifies and rectifies biases at various stages. The collaboration of these testing components collectively fulfills the mission of fairness testing in the ML software.

Data Testing. ML follows the data-driven programming paradigm. Training data D determines the decision logic of ML models M to a large extent [304]. Therefore, training data is considered as an important component to test in the ML testing literature [120, 304]. Since bias in training data is demonstrated to be a main root cause of ML software unfairness [125], training data is also an important component for fairness testing. In the fairness literature, data testing aims to identify bias within the training data D. As elucidated in Section 2.3, data testing involves the use of programs as test inputs, designed to specifically detect three types of data bias: feature bias, label bias, and selection bias. Feature bias detection focuses on identifying whether the features X of training data contain bias [210]. Label bias detection aims to identify whether factors unrelated to label determination influence the label generation process, resulting in biased data labels Y [127]. Selection bias detection targets the identification of unexpected correlations between sensitive attributes A and data labels Y within the distribution of training data [293].

ML Program Testing. An ML program G encodes the process by which an ML model M is obtained based on the training data. It has been an important fairness testing component of ML software [304]. A fairness bug may arise as improper data processing [116], training algorithm selection [183], and hyper-parameter settings (i.e., configuration options that control the learning process) [274] in ML programs. In the fairness literature, ML program testing aims to uncover issues within the implementation of the ML program G that contribute to the unfairness of the ML software S.

Framework Testing. ML frameworks F (also called ML libraries) implement ML algorithms internally and provide high-level **Application Programming Interfaces (APIs)** for developers

to build ML models without knowledge of the inner working of these algorithms. The ML testing literature [228, 233, 244, 289, 292] has considered ML frameworks as an important testing component and detected bugs inside ML frameworks that lead to accuracy problems in the final ML model M. Nevertheless, existing framework testing studies are primarily related to ML performance (e.g., accuracy). In contrast, framework testing for fairness aims to detect issues inside ML frameworks F that result in unfairness of the ML software S. To the best of our knowledge, to date, there has been no framework testing work for detecting fairness bugs.

Model Testing: Most fairness testing techniques are model-centric [97, 156, 160, 278, 295, 310, 313, 318]. They consider the ML model M as the testing component and aim to reveal fairness bugs based on the input-output behaviors of M. Fairness testing of ML models can be performed in a white, black, or gray-box manner [274]. Black-box testing is a technique of testing without having any knowledge of the internal working of ML software (e.g., code and data); white-box testing tests an ML software system taking into consideration its internal working; gray-box testing is to test with limited knowledge of the internal working of the software under test [194].

Non-ML component testing: From the SE perspective, ML software is beyond the ML models and also includes non-ML components C [215]. These non-ML components may also affect the fairness of ML software. For instance, data storage practices that result in the exclusion or underrepresentation of specific demographic groups can lead to biased training data, subsequently resulting in biased ML models. Similarly, biased user interfaces may inadvertently exhibit favoritism or bias against particular user groups. Testing non-ML components for fairness seeks to identify any unfairness within the ML software S attributable to the non-ML components C. However, to the best of our knowledge, there has been no non-ML component testing work in the fairness testing literature.

Section 5 organizes the related papers from the testing component perspective. The existing literature primarily focuses on data testing (Section 5.1), ML program testing (Section 5.2), and model testing (Section 5.3).

3 SURVEY METHODOLOGY

This section introduces the scope and the paper collection process of our survey.

3.1 Survey Scope

We aim to define, collect, and curate the disparate literature, arguing and demonstrating that there does, indeed, exist a coherent area of research in the field that can be termed "fairness testing."

We apply the following inclusion criteria when collecting papers. The papers that satisfy any of these criteria are included in this survey.

- The paper introduces the general idea or one of the related aspects of fairness testing of ML software.
- The paper presents an approach, study, framework, or tool that targets at fairness testing of ML software.

We do not include papers about the issues of fairness in network systems and hardware systems. Moreover, we filter out papers that are about fairness definitions but do not consider them in the context of testing. We also do not include papers about gender diversity/inclusion and cognitive bias in software development, because our survey focuses on SE product fairness, not SE process fairness.

3.2 Paper Collection

We first collected papers by keyword searching on the DBLP publication database [73], which covers arXiv (a widely used open-access archive), more than 1,800 journals, and 5,800 academic

conferences and workshops in computer science [87]. DBLP has been extensively used in previous surveys in SE [132, 162, 221, 304, 305], and a recent ML testing survey [304] demonstrates that papers collected from other popular publication databases are a subset of those collected from DBLP.

We developed the search keywords through an iterative trial-and-error procedure [212] conducted by the first two authors, with input and discussion from the other authors. Initially, we started with a general search string "fairness testing" to gather initial relevant papers. Subsequently, we carefully examined the titles, abstracts, and keywords of these papers to identify additional keywords and phrases. Through brainstorming sessions, we expanded and refined the list of search strings by incorporating related terms, synonyms, and variations. The iterative process allowed us to continuously improve the search keyword list based on the outcomes of the searches, ensuring that it accurately captured the relevant literature on fairness testing.

The final keywords used for searching included *("**fair**" OR "**bias**" OR "**discrimination**") AND ("**software**" OR "**learning**" OR "**bug**" OR "**defect**" OR "**fault**" OR "**algorithm**" OR "**test**" OR "**detect**" OR "**evaluat**" OR "**discover**" OR "**identify**" OR "**find**" OR "**uncover**" OR "**reveal**" OR "**recogniz**" OR "**unveil**")*. As a result, we conducted a total of $3 \times 16 = 48$ searches on DBLP on March 12, 2023, and obtained 7,674 hits. Then, the first two authors manually inspected each hit paper to check whether it was in the scope of our survey and selected 67 relevant papers.

The fairness testing community is diverse, with publications appearing in various venues and adopting different terminologies. To capture papers that might have been overlooked by our keywords and ensure a comprehensive coverage of the field, we further employed a snowballing approach. This process, conducted in April and May 2023, aimed to identify transitively dependent papers and expand our paper collection. Both backward and forward snowballing approaches [188] were employed. In *backward snowballing*, we examined the references in each collected paper and identified those lying in our scope; in *forward snowballing*, we used Google Scholar to identify papers of our interest from those that cited the collected papers. We iteratively repeated the snowballing process until we reached a fixed point, where no new relevant papers were identified. Through this process, we were able to retrieve an additional 25 papers, contributing to a more comprehensive coverage of the fairness testing literature.

To ensure that our survey is comprehensive and accurate, we also contacted the authors of the papers that we collected via keyword searching and snowballing. We provided them with our paper and asked them to check whether our description of their work was correct. This interaction allowed us to refine our understanding of their contributions and make necessary revisions to our descriptions. Furthermore, during these communications, authors directed our attention to 15 additional papers that we had not initially included in our collection. Among the suggested papers, 8 met our inclusion criteria and were deemed relevant to our survey. These 8 papers were subsequently added to our repository, further enhancing the coverage of the fairness testing literature in our survey.

Table 4 shows the statistics of the paper collection process. In summary, we consider 67 + 25 + 8 = 100 papers for this survey.

3.3 Paper Analysis

To ensure a rigorous analysis of the collected papers and to enhance the quality and accuracy of the survey, we employed thematic synthesis, a well-established method in SE literature reviews [186]. This method allowed us to systematically organize and analyze the papers in a structured manner.

The analysis process was led by the first two authors, who extensively read and examined the full text of each paper. Their objective was to develop a comprehensive understanding of the testing workflow and test components described in the papers. Through meticulous manual analysis,

Table 4. Statistics of the Collected Papers

Keyword	Hits
fair | bias | discrimination + software	236
fair | bias | discrimination + learning	2,399
fair | bias | discrimination + bug	20
fair | bias | discrimination + defect	41
fair | bias | discrimination + fault	94
fair | bias | discrimination + algorithm	2,121
fair | bias | discrimination + test	409
fair | bias | discrimination + detect	1,013
fair | bias | discrimination + evaluat	838
fair | bias | discrimination + discover	172
fair | bias | discrimination + identify	119
fair | bias | discrimination + find	96
fair | bias | discrimination + uncover	33
fair | bias | discrimination + reveal	60
fair | bias | discrimination + recogniz	12
fair | bias | discrimination + unveil	11
After manual inspection	67
After snowballing	92
After collecting author feedback	100

they extracted relevant information, identified common patterns and discerned major themes that emerged across the collection of papers.

Using these identified themes as a framework, the authors categorized and organized the related papers under different thematic headings. This approach facilitated a coherent and structured representation of the survey's content, enabling readers to easily navigate and comprehend the key insights and findings from the analyzed papers.

During the analysis, in instances where disagreements arose, the first two authors held discussion meetings involving other co-authors. These meetings served as a platform to address conflicts and reach a consensus on the extracted data and the placement of papers within the identified themes. The co-authors, all of whom have published fairness-related papers in top-tier venues, contributed their expertise and acted as arbitrators, ensuring the resolution of any disagreements and maintaining the integrity of the analysis.

Finally, to ensure the integrity and reliability of the survey's findings, all authors independently double-checked the content. This review process aimed to identify any potential errors, inconsistencies, or omissions.

Additionally, as mentioned before, to further enhance the quality of the survey, we shared the draft with the authors of the collected papers. This collaborative approach allowed us to gather valuable input, feedback, and validation from experts in the field. By incorporating their insights, we ensured that the survey accurately represented the original papers' findings and perspectives, strengthening the overall credibility and robustness of our analysis.

4 FAIRNESS TESTING WORKFLOW

We first introduce existing techniques that support the key activities involved in fairness testing, i.e., test input generation and test oracle identification.

Table 5. Test Generation Techniques and the Fairness Definitions Adopted

Category	Authors [Ref], Year	Venue	Fairness definitions
Random generation	Galhotra et al. [101, 160], 2017	ESEC/FSE	causal fairness, statistical parity
	Angell et al. [101], 2018	ESEC/FSE	causal fairness, statistical parity
Search-based generation	Udeshi et al. [278], 2018	ASE	causal fairness
	Sano et al. [252], 2022,	SEKE	causal fairness
	Aggarwal et al. [97], 2019	ESEC/FSE	causal fairness
	Fan et al. [156], 2022	ICSE	causal fairness
	Ma et al. [218], 2022	arXiv	causal fairness
	Zhang et al. [313], 2020	ICSE	causal fairness
	Zhang et al. [315] 2021	IEEE TSE	causal fairness
	Zhang et al. [310], 2021	ISSTA	causal fairness
	Zheng et al. [318], 2022	ICSE	causal fairness
	Monjezi et al. [227], 2023	ICSE	causal fairness
	Xie and Wu [295], 2020	TrustCom	causal fairness
	Perera et al. [242], 2022	EMSE	causal fairness
	Tao et al. [270], 2022	ESEC/FSE	causal fairness
	Xiao et al. [294], 2023	ISSTA	causal fairness
	Patel et al. [237], 2022	ICST	causal fairness
	Haffar et al. [175], 2022	MDAI	counterfactual fairness
	Zhang et al. [312], 2023	ACM TOSEM	statistical parity
	Cabrera et al. [122], 2019	VAST	group fairness
Verification-based generation	Sharma and Wehrheim [255], 2020	ICTSS	causal fairness
	Sharma et al. [254], 2021	ICMLA	causal fairness
	Kitamura et al. [197], 2022	SSBSE	causal fairness
	Zhao et al. [317], 2022	SSBSE	causal fairness
Template-based generation	Díaz et al. [151], 2018	CHI	causal fairness
	Zhang et al. [312], 2023	ACM TOSEM	statistical parity
	Liu et al. [214], 2020	COLING	causal fairness, statistical parity
	Kiritchenko and Mohammad [196], 2018	NAACL workshop	causal fairness
	Mehrabi et al. [222], 2020	HT	parity in performance across groups
	Wang et al. [290], 2022	ACL	parity in performance across groups
	Sharma et al. [256], 2020	NeurIPS workshop	causal fairness
	Sheng et al. [257], 2019	IJCNLP	causal fairness
	Huang et al. [185], 2020	EMNLP	causal fairness
	Vig et al. [283], 2020	NeurIPS	counterfactual fairness
	Dhamala et al. [150], 2021	FAccT	causal fairness
	Smith et al. [261], 2022	EMNLP	statistical parity
	Ribeiro et al. [251], 2020	ACL	causal fairness
	Wan et al. [286], 2023	ESEC/FSE	absolute fairness, relative fairness
	Ma et al. [219], 2020	IJCAI	counterfactual fairness
	Asyrofi et al. [102], 2021	IEEE TSE	counterfactual fairness
	Yang et al. [299], 2021	ESEC/FSE	distributional fairness
	Sun et al. [268], 2020	ICSE	counterfactual fairness
	Sun et al. [269], 2022	ICSE	counterfactual fairness
Grammar-based generation	Ezekiel et al. [262], 2022	IEEE TSE	counterfactual fairness
GAN-based generation	Denton et al. [149], 2019	CVPR workshop	causal fairness
	Joo and Kärkkäinen [191], 2020	arXiv	causal fairness
	Zhang et al. [314], 2021	arXiv	causal fairness
	Muthukumar [229], 2019	CVPR workshop	causal fairness
	Dash et al. [146]	WACV	counterfactual fairness
	Balakrishnan et al. [106], 2020	ECCV	parity in robustness across groups
	Pu et al. [247], 2022	ICSE workshop	parity in robustness across groups
Signal transformation -based generation	Rajan et al. [250], 2022	FASE	parity in robustness across groups

4.1 Test Input Generation

In the area of fairness testing, test input generation aims to automatically produce instances that can induce discrimination and reveal fairness bugs in software systems. We organize relevant research based on the techniques adopted, including random test input generation, search-based test input generation, verification-based test input generation, and domain-specific test input generation. Table 5 summarizes these techniques and indicates the fairness definitions that they adopt.

4.1.1 Random Test Input Generation. Random testing evaluates a software system with inputs randomly chosen from its input space, and it can be used to infer the quantitative estimate of a system's operational reliability [153].

Galhotra et al. [160] and Angell et al. [101] introduced Themis, a random test input generation approach for fairness. Themis randomly assigns values to non-sensitive attributes and varies the values of sensitive attributes. By observing the behavior of the system under test with these inputs, Themis quantifies the occurrence of discriminatory instances in the input space. Discrimination is measured using two frequency values. The first value represents the proportion of generated inputs where altering the sensitive attributes leads to a change in the output (*causal fairness*). The second value captures the disparity in receiving favorable outcomes among different demographic groups within the generated inputs (*statistical parity*).

4.1.2 Search-based Test Input Generation. Despite the effectiveness of Themis, random generation can lead to a low success rate of discriminatory input generation [156], so the following fairness testing work generates test inputs using search-based techniques. Search-based test generation uses meta-heuristic search techniques to guide the generation process and make this process more efficient and effective [177, 178, 204]. It has been employed in an increasing number of fairness testing techniques to explore the input space of the software under test.

Two-phase search-based techniques. In the realm of fairness testing, most search-based test input generation techniques employ a two-phase approach for generating individual discriminatory instances. These techniques operate based on the *causal fairness* definition, which implies that altering sensitive attributes should not impact the outcomes.

The two phases involved are the global search and the local search:

— During the *global search* phase, the technique conducts an exploration of the input space to identify an initial set of individual discriminatory instances. These instances consist of pairs where altering the sensitive attributes results in divergent outcomes.
— During the *local search* phase, the technique focuses on searching for additional individual discriminatory instances in the neighborhood of those found during the global search. This phase is based on the hypothesis that if a discriminatory input exists in the input space, then there exist more discriminatory inputs located closer to it [278]. The hypothesis draws inspiration from the robustness property of ML models, where similar inputs should yield similar outputs. Hence, the discriminatory inputs and their neighborhood are likely to be similarly discriminatory, especially for robust models [278].

In the following, we introduce how existing search-based fairness testing techniques materialize the two phases:

Udeshi et al. [278] introduced Aequitas, the first fairness testing approach based on a two-phase search framework. In the global search phase, Aequitas randomly explores the input space to uncover discriminatory instances. In the local search phase, Aequitas perturbs the non-sensitive attribute of the discriminatory instances found in the global phase to explore their neighboring inputs and identify more discriminatory instances. Furthermore, Aequitas employs the generated cases to estimate the proportion of inputs that violate causal fairness, offering statistical evidence for fairness bugs. Sano et al. [252] proposed KOSEI, which modifies the local search of Aequitas [278]. KOSEI individually applies perturbations to all non-protected attributes, in contrast to Aequitas, which probabilistically selects one attribute at a time. Additionally, KOSEI enables users to set an iteration limit for the perturbation process.

Aggarwal et al. [97] proposed SG, which combines symbolic generation and local explainability for search-based discriminatory instance generation. In the global search phase, SG utilizes a local model explainer to approximate the decision-making process of ML software by constructing a decision tree. Symbolic execution is then employed to cover various paths in the decision tree, aiming to search for discriminatory inputs. In the local search phase, SG perturbs the non-sensitive

attribute of these discovered inputs to explore their neighborhood within the input space, thereby generating additional discriminatory inputs.

Fan et al. [156] introduced ExpGA, an explanation-guided approach for generating discriminatory instances. Initially, ExpGA employs interpretable methods to identify seed instances that are more likely to produce discriminatory instances when their feature values are slightly modified compared to other instances. Subsequently, using these seed instances as inputs, ExpGA leverages a genetic algorithm for local search, enabling efficient generation of a substantial number of discriminatory instances.

Ma et al. [218] proposed I&D to enhance the initial seed selection for the global search of the two-phase search-based generation approach. It achieves this by generating a chiral model, which alters the protected attributes of the training data. This chiral model helps identify initial discriminatory instances by detecting differences in predictions compared to the original model. I&D employs the SHAP value [217], a game-theory-based approach, to explain the variation in prediction behavior between the chiral and original models for each initial discriminatory instance. Moreover, it clusters the initial discriminatory instances based on their SHAP values and selects diverse instances from each cluster in a round-robin fashion for future use in the global search.

In addition to these general techniques, there have been several *two-phase search-based generation techniques specifically proposed for* ***Deep Neural Networks (DNNs)***, including ADF [313, 315], EIDIG [310], NeuronFair [318], and DICE [227].

Zhang et al. [313, 315] proposed ADF, a gradient-guided generation approach for DNNs. In the global search phase, ADF locates the discriminatory instances near the decision boundary by iteratively perturbing a seed input towards the decision boundary with the guidance of gradient. In the local search phase, ADF further uses gradients as the guidance to search the neighborhood of the found individual discriminatory instances to discover more discriminatory instances.

Zhang et al. introduced EIDIG [310], which inherits and improves ADF by integrating a momentum term into the iterative search for identifying discriminatory instances. The momentum term enables the memorization of the previous trend and helps to escape from local optima, which ensures a higher success rate of finding discriminatory instances. In addition, EIDIG reduces the frequency of gradient calculation by exploiting the prior knowledge of gradients to accelerate the local search phase.

Zheng et al. [318] proposed NeuronFair, which uses the identified biased neurons to guide the generation of discriminatory instances for DNNs. In the global search phase, NeuronFair identifies the biased neurons that cause discrimination via neuron analysis and searches for discriminatory instances with the optimization object of increasing the **ActDiff (activation difference)** values of the biased neurons. In the local search phase, NeuronFair uses the generated discriminatory instances as seeds and perturbs non-sensitive attributes of them to generate more discriminatory instances.

Monjezi et al. [227] introduced DICE, an information-theoretic approach for fairness testing of DNNs. In the global search phase, DICE uses gradient-guided clustering to explore the input space and identify instances with the maximum quantitative individual discrimination. This discrimination is quantified as the dependence on sensitive attributes using information-theoretic principles. In the local phase, promising instances from the global phase are used to generate additional discrimination instances by exploring their neighborhood.

Other search-based techniques. Besides the two-phase approach, there are also other search-based generation techniques that have been proposed in the fairness testing literature.

Xie and Wu [295] used reinforcement learning to develop a black-box search strategy for generating instances that violate *causal fairness*. Their approach frames the task of generating discriminatory instances as a reinforcement learning problem, with the ML model under test treated as part

of the environment. The reinforcement learning agent interacts with the environment by taking actions to search for discriminatory inputs while receiving feedback in the form of rewards and observing the resulting state. Through iterative interactions, the agent learns an optimal policy for efficiently generating discriminatory inputs.

Perera et al. [242] presented SBFT, a search-based fairness testing approach for *regression-based ML models*. SBFT measures unfairness degree as the maximum difference in regression results between pairs of instances that only differ in terms of their sensitive attributes (i.e., *causal fairness*). SBFT begins with an initial set of randomly selected test inputs from the input space. Genetic algorithms are then applied to these inputs, aiming to generate the next generation of test inputs that exhibit the desired fairness degree.

Tao et al. [270] introduced RULER, a fairness testing technique that departs from the strict definition of *causal fairness*. Unlike existing approaches that require coupling samples differing only in sensitive attributes, RULER relaxes this constraint. It allows simultaneous perturbations on both sensitive and non-sensitive attributes to search for more discriminatory instances, because there may not exist discriminatory instances that strictly satisfy the causal fairness. RULER imposes perturbation constraints: sensitive attributes must remain within their valid value ranges, while non-sensitive attributes are bounded by a small range.

Xiao et al. [294] proposed LIMI, an approach for generating natural discriminatory instances based on *causal fairness*. LIMI uses a **generative adversarial network (GAN)** to imitate the decision boundary of the model under test in the latent space. By approximating the decision boundary with a surrogate linear boundary, LIMI can search for instances that closely align with the original data distribution. LIMI performs vector manipulations on latent vectors to move them towards the surrogate boundary. Vector calculations then identify two potential discriminatory candidates in close proximity to the real decision boundary.

Patel et al. [237] applied combinatorial t-way testing [201] to fairness testing. Combinatorial t-way testing is a coverage-based data sampling method that can generate diverse datasets by applying logical constraints to specify the sampling space. The approach creates an input parameter model from the training dataset and uses the model to generate a t-way test set. For each test, it mutates protected attributes to search for discriminatory instances.

Haffar et al. [175] introduced two distinct generation approaches for tabular data and images. First, they employed guided adversarial generation on tabular data to search for counterfactuals. Second, they employed GANs to generate counterfactuals for image data. The generation objective is to modify the predicted label with minimal adjustments to the input. If one of the modified attributes happens to be a protected attribute, then it suggests that the model is exhibiting unfairness against that particular attribute.

Zhang et al. [312] introduced TestSGD, a group fairness testing approach for DNNs that specifically focuses on the *statistical parity* of subgroups defined by conjunctions of protected attributes. TestSGD first generates rule sets to capture frequent subgroups. Each rule splits the input space into two groups. To accurately measure group discrimination, TestSGD introduces small, uniform perturbations to the original training samples to search for more samples. Then, it calculates the demographic parity score based on the generated samples.

Cabrera et al. [122] also focused on fairness testing of subgroups and proposed a testing tool named FAIRVIS. In real-world applications, the number of subgroups to analyze can be overwhelming. To address this challenge, FAIRVIS offers methods to search this large space and find potential issues more efficiently. It introduces a subgroup generation technique that recommends intersectional groups where a model may be underperforming. This technique involves clustering the training dataset to identify statistically similar subgroups and using an entropy-based approach to

determine important features and calculate group fairness metrics for the clusters. Finally, FAIRVIS presents generated subgroups sorted by their group fairness metrics.

4.1.3 Verification-based Test Input Generation. Since our survey focuses on fairness testing, we do not discuss work focusing solely on formal verification of fairness properties (see, e.g., References [99, 111, 164, 165, 190, 265, 279]). Nevertheless, there is a research stream focusing on generating discriminatory test inputs through fairness verification using **Satisfiability Modulo Theories (SMT)** solving. These techniques are called verification-based test input generation [254].

Sharma et al. [254, 255] introduced fairCheck and MLCheck, two verification-based techniques for generating test inputs to assess fairness. These approaches approximate the black-box model under test with a white-box model, leveraging its predictions. The fairness property and the white-box model are translated into logical formulas using SMT solvers. Test cases are subsequently automatically generated in an attempt to violate the specified fairness property, employing the Z3 SMT solver [147] to check for satisfiability.

Kitamura et al. [197] developed VBT-CT, a technique that integrates combinatorial t-way testing into verification-based fairness testing. As described before, combinatorial t-way testing can generate diverse datasets by applying logical constraints to specify the sampling space. By incorporating combinatorial t-way testing into the generation process of verification-based testing, VBT-CT enhances the detection capability of discriminatory data.

Zhao et al. [317] introduced VBT-X, an approach that integrates hash-based sampling [130] into the test generation phase of verification-based fairness testing. Hash-based sampling techniques are capable of producing diverse solutions for a given logical formula, offering the advantage of generating varied solutions with reasonable computational overhead. By leveraging the diverse sampling capability of hash-based sampling, VBT-X enhances the effectiveness of test generation in verification-based fairness testing.

4.1.4 Domain-specific Test Input Generation. Recently, an increasing number of approaches have been proposed for test input generation in specific application domains. These approaches aim to generate natural inputs that belong to the data distribution of a practical application scenario. This section introduces such domain-specific test input generation in typical domains: natural language processing, computer vision, and speech recognition.

Natural language processing. Test input generation for **Natural Language Processing (NLP)** systems is primarily based on that a fair NLP system should produce similar results for pairs of similar texts.

Researchers can collect text data from the wild and then mutate words related to sensitive attributes to generate test inputs. For example, Díaz et al. [151] conducted a study where they collected sentences containing the word “old” from a blogger and replaced it with “young” to detect age-related fairness issues in sentiment analysis systems. They also applied mutations to common adjectives instead of age-related words. By using word embeddings, they obtained analogs for “older” and “younger” versions of these adjectives. These variants were used as test inputs to identify fairness issues related to age. Zhang et al. [312] generated new samples by substituting sensitive terms in the collected texts with alternative terms within the same sensitive attribute category. For instance, for the gender category, the sensitive terms can be bisexual, female, gay, and so on. Then they calculate the statistical parity based on the original samples and the generated ones.

Similarly, Liu et al. [214] conducted fairness testing on generative and retrieval dialogue models. They created gender and race word lists, including male-female word pairs and African-American English-Standard English word pairs. From a large dialogue corpus, they selected contexts containing words from these lists and created parallel contexts by replacing the words with their counterparts. The original and generated texts were then compared based on the response of

dialogue models using four measurements: diversity, politeness, sentiment, and attribute words. The attribute word analysis involved comparing the probability of attribute words (e.g., career and family-related) appearing in the responses across different groups' contexts. This is a mixed adoption of causal fairness and statistical parity.

Generation based on handcrafted templates: A large proportion of fairness testing studies for NLP systems involve the use of handcrafted templates to generate test inputs. These templates consist of short sentences with placeholders (e.g., "*<person> goes to the school in our neighborhood*") that can be filled with different words to test for violations of causal fairness. Due to the simplicity of the handcrafted templates, most of such test generation studies adopt *causal fairness* that detects whether NLP systems produced similar outcomes for texts that differ only in sensitive attributes.

Kiritchenko and Mohammad [196] created 11 templates focused on gender and race, using predefined values for the placeholder *<person>* representing different names and noun phrases referring to females/males or African/European American. Then they compared emotion and sentiment scores that the systems predict on pairs of sentences differ only in one word corresponding to race or gender.[2]

Mehrabi et al. [222] created nine templates to compare the performance of name entity recognition systems with respect to recognizing male and female names. Wang et al. [290] focused on machine translation systems and developed 30 templates to test their ability to determine the correct gender of a name.

Sharma et al. [256] designed three templates specifically for gender-related fairness issues in natural language inference systems, with gender-specific hypotheses using the placeholder *<gender>*.

Similarly, other researchers have designed templates for detecting fairness bugs in natural language generation and machine translation systems. Sheng et al. [257], Huang et al. [185], Vig et al. [283], and Dhamala et al. [150] used templates with placeholders for sensitive attributes such gender and race.

Smith et al. [261] developed a set of manually created templates covering 13 different demographic attributes. These templates were used to identify bias in language models by examining *statistical disparity*, which measures the differences at the group level in the output or assigned probabilities of the model, which arise due to the presence of different identity or demographic information within the input text.

Another approach, CheckList [251], utilizes predefined templates to evaluate NLP systems on various capabilities, including fairness.

Wan et al. [286] introduced BiasAsker, a test input generation approach aimed at measuring absolute bias and relative bias of conversational systems towards various demographic groups. Absolute bias refers to direct expressions of bias, such as statements like "Group A is smarter than Group B," while relative bias involves generating different responses to questions about different groups. To obtain social groups and biased properties, they constructed a comprehensive social bias dataset, which includes a total of 841 groups and 8,110 biased properties. To detect both types of bias, they designed templates and rules for generating Yes-No Questions, Choice-Questions, and Wh-Questions. These generated inputs serve as a means to evaluate and identify bias in conversational systems.

Automated generation: While handcrafted templates have been effective in detecting fairness issues in NLP systems, researchers argue that the generated test inputs relying on them may be simplistic and limited [246]. Furthermore, the generated simplistic test inputs may overlook complex scenarios where multiple words related to sensitive attributes are present, which is often the

[2]Sex and gender are different concepts that are often used interchangeably. We keep the original usage of the two words in each paper to reserve fidelity.

case in practical applications. To address this limitation, automated test generation techniques should carefully consider each word that can rely on sensitive attributes and generate natural counterfactual inputs that align with the *counterfactual fairness* definition.

To address the problem, Ma et al. [219] introduced the automated framework MT-NLP. It employs advanced NLP techniques to identify human-related tokens in the input text and utilizes word analogy techniques to mutate these tokens, generating discriminatory test inputs. Language fluency metrics are then used to filter out unrealistic inputs.

Asyrofi et al. [102] proposed BiasFinder, an automated approach for creating diverse and complex test inputs to uncover fairness bugs in NLP systems. By leveraging NLP techniques such as coreference resolution and named entity recognition, BiasFinder identifies words associated with demographic characteristics and replaces them with placeholders to form templates. Concrete values are then filled into these templates, generating a large number of text mutants for testing metamorphic relationships.

Yang et al. [299] proposed BiasRV, an approach for testing bias in deployed sentiment analysis systems. By utilizing BiasFinder, they generate a template for a given text and create mutants based on that template to assess if the system exhibits biased predictions. They define *distributional fairness*, which examines whether mutants from different demographic groups are treated similarly. Specifically, they expect the distribution of predicted sentiments for these groups of mutants to be closely aligned if the system is fair. Unlike traditional statistical parity, which measures overall fairness, distributional fairness evaluates whether the system makes biased predictions for specific inputs.

Ezekiel et al. [262] developed ASTRAEA, a grammar-based fairness testing approach for generating discriminatory inputs in NLP systems. ASTRAEA incorporates input grammars covering various NLP tasks and biases. By randomly exploring the input grammars and mutating sensitive attribute-related words using alternative tokens, ASTRAEA generates initial test inputs and checks their satisfaction of metamorphic relations.

For fairness testing of machine translation systems, Sun et al. [268, 269] proposed TransRepair and CAT. TransRepair conducts sentence mutations by replacing words with context-similar alternatives, while CAT identifies and replaces words using isotopic replacement. The resulting mutants, along with the original input sentence, serve as test inputs for evaluating the fairness of machine translation.

Computer vision. To detect fairness issues in **Computer Vision (CV)** systems, researchers often examine how the system's output changes when the sensitive attribute of a person in the input image is altered while keeping other factors constant (i.e., generating *counterfactual* images). This idea forms the basis of test input generation for CV systems.

GANs [170] are commonly used for image transformations in ML testing [304]. However, conventional GANs face challenges in generating precise changes required for fairness testing. For example, changing hair color without affecting other facial features or hair style can be difficult. To address this, recent efforts have adapted and improved conventional GANs for generating test inputs in CV systems. Denton et al. [149] developed a face-generative model that maps latent codes to images and inferred directions in latent code space to manipulate specific sensitive attributes. Test inputs were generated by traversing these inferred directions. Joo and Kärkkäinen [191] employed the FaderNetwork architecture [205], where specific known attributes of an image are input separately to the generator. Zhang et al. [314] utilized CycleGAN [319], which limits changes to non-sensitive attributes, to generate discriminatory inputs.

Muthukumar [229] criticized GAN-based approaches for their inability to effectively modify a single attribute while keeping other non-related attributes unchanged. This limitation makes it challenging to identify the exact cause of unfair outcomes. For instance, in gender classification

where discrimination occurs between different skin types, it is possible that factors such as hairstyle, facial structure, cosmetics, or clothing contribute to the disparity rather than skin type alone. To address this issue, Muthukumar proposed a solution where face images were represented in the YCrCb color space [184], and techniques such as luminance mode-shift and optimal transport [158] were utilized to alter the skin type of a face.

However, these approaches may fail to account for causal relationships between attributes when generating discriminatory images. Although they claim to produce counterfactual images, they often rely on a causal fairness definition rather than strict counterfactual fairness. To generate test inputs that reflect real-world scenarios, it is crucial to consider the downstream effects resulting from changes in sensitive attributes. For instance, in a chest MRI classification system, a patient's age can influence the relative size of their organs [146]. Therefore, altering the age without considering the causal relationship between age and organ size would lack realism. In response to this limitation, Dash et al. [146] introduced ImageCFGen, a fairness testing method that incorporates knowledge from a causal graph and utilizes an inference mechanism within a GAN-like framework. This approach enables the generation of discriminatory images that adhere to the definition of counterfactual fairness.

Unlike the aforementioned studies that employ individual fairness definitions, Balakrishnan et al. [106] introduced a new concept of group fairness, investigating the parity in robustness against transformations across various demographic groups. Specifically, they proposed an image generation method that produces synthesized samples considering multiple sensitive attributes and compares error rates of gender classifiers across various subgroups before and after the synthesis. For generation, the approach modifies multiple attributes simultaneously to create grid-like sets of matched images called transects. This is achieved by navigating the latent space of the generator in directions specific to each attribute.

Similarly, Pu et al. [247] investigated the parity in robustness of deepfake detectors by introducing makeup as a feature perturbation across various demographic groups. Specifically, they employed common makeup alterations such as eyeshadows, eyeliners, lipstick, and blushes to perturb the face images before feeding these images to the deepfake detection models. Subsequently, they compared the accuracy differences of the detectors in detecting deepfakes between original and synthetic images for both male and female images, thus quantifying the extent of gender disparity in the two accuracy differences. In addition, they discovered that salient regions near the lips have the greatest impact on the fairness of the tested models.

Speech recognition. The fairness testing literature has dedicated less attention to speech recognition compared to NLP and CV. Rajan et al. [250] introduced AequeVox, a fairness testing approach tailored for speech recognition systems. Similar to Balakrishnan et al. [106] and Pu et al. [247], AequeVox measures group fairness as the parity in robustness against perturbations across demographic groups. It generates test inputs by applying eight common real-life metamorphic perturbations to speech signals, such as noise, drop, and low/high pass filters. Then it calculates the error rate increase in speech recognition for various demographic groups after applying these perturbations. If the difference in error rate increase surpasses a predetermined threshold for different groups, then the recognition system is considered to have fairness bugs.

4.1.5 Test Input for Data Testing. ML program testing and model testing typically rely on data instances as test inputs, whose generation process has been detailed in the previous sections. For data testing, test inputs are typically programs capable of detecting data bias, as described in Section 2.3. Typically, researchers manually implement these programs based on testing objectives, which include identifying feature bias, label bias, and selection bias.

To detect feature bias, Peng et al. [241] implemented logistic regression and decision tree algorithms as models to infer relationships between sensitive attributes and non-sensitive

features. This approach facilitated the identification of non-sensitive features that are correlated with sensitive attributes and contribute to fairness issues. Li et al. [210] implemented linear regression models to analyze the association between non-sensitive features and sensitive attributes to identify feature bias. Zhang et al. [306, 308, 309] implemented a causal graph to identify features causally leading to unfair outcomes. Li and Xu [211] optimized a hyperplane within the latent space of a generative model, offering a solution to detect unknown biased attributes and enhance the understanding of feature bias. Black et al. [117] developed optimal transport projection programs [284] to map data instances from the privileged/unprivileged group to their counterparts in the unprivileged/privileged group, conducting analyses on these flipsets to detect features contributing to unfairness.

To detect label bias, Chakraborty et al. [125–127] implemented situation testing programs to uncover biased data instances. Additionally, Chen and Joo [133] tailored testing programs based on facial action units to detect label bias in widely used datasets for facial expression recognition.

To detect selection bias, researchers [125, 126, 137] implemented programs to analyze the statistical association between sensitive attributes and outcome labels in the training data. Simultaneously, some researchers [220] scrutinized dataset representation to ascertain potential underrepresentation issues of demographic groups.

4.2 Test Oracle Identification

Given an input for a software system, the challenge of distinguishing the corresponding desired behavior from the potentially incorrect behavior is called the "*test oracle problem*" [110]. Test oracle identification is one of the key problems in ML testing [304]. The test oracle of fairness testing determines whether the software is behaving as fairness requirements and enables the judgment of whether a fairness bug exists. Existing work employs two types of test oracles for fairness testing: metamorphic relations and statistical measurements.

4.2.1 Metamorphic Relations as Test Oracles. A metamorphic relation is a relationship between the software input change and the output change that we expect to hold across multiple executions [110]. Suppose a system that implements the function $\sin(x)$, then $\sin(x) = sin(\pi + x)$ is a metamorphic relation. This relation can be used as a test oracle to help detect bugs. If $\sin(x)$ differs from $sin(\pi + x)$, then we can conclude that the system under test has a bug without the need for examining the specific values output by the system.

Metamorphic relation is a type of pseudo oracle commonly adopted to automatically mitigate the oracle problem in ML testing. It has also been widely studied for fairness testing of ML software. Specifically, existing work mainly performs fairness-related metamorphic transformation on the input data or training data of ML software and expects these transformations to not change or yield expected changes in the prediction. Next, we classify and discuss this work according to whether the metamorphic transformations operate on sensitive attributes.

Metamorphic transformations through mutating sensitive attributes. The mutation of sensitive attributes is a widely used technique for generating metamorphic relations in the context of *individual fairness*. Assessing violations of individual fairness, such as counterfactual fairness and causal fairness, typically requires the comparison of paired data instances that vary in their sensitive attributes.

For ML software designed for *classification* tasks, a widely employed metamorphic relation for fairness testing involves comparing pairs of instances with different sensitive attributes but similar non-sensitive attributes, expecting them to yield the same classification outcome. For instance, a fair loan application system should make identical decisions for two applicants who differ only in their gender.

This metamorphic relation has found extensive application in testing the fairness of software systems and guiding the generation of fairness tests across various domains, including tabular data classification [97, 101, 160, 278, 295, 310, 313, 315], text classification [102, 185, 196, 219, 251, 256, 262, 290], and image classification [146, 148, 149, 191, 314, 318], among others.

In the case of tabular data, researchers typically select the sensitive attributes of interest in the dataset (e.g., gender and race) and modify the values of those attributes (e.g., from "male" to "female") to assess if the software violates the fairness metamorphic relation for those instances (i.e., adopting the *causal fairness* definition). Tao et al. [270] proposed that altering only the sensitive attributes can lead to unnatural test inputs and therefore relaxed this constraint by allowing small permutations to be applied to non-sensitive attributes simultaneously. Similarly, counterfactual fairness [202] necessitates modifying the non-sensitive attributes that are causally influenced by the sensitive attributes whenever the sensitive attributes are modified.

For text data, researchers generate inputs by filling sensitive attribute-related placeholders in predefined templates and mutating the sensitive attributes within these templates, or identify the entities related to the sensitive attributes and simultaneously transform all the identified entities to create valid test inputs that reflect real-world scenarios. For example, when dealing with gender, researchers identify and modify person names, gender pronouns, gender nouns, and so on.

For image data, researchers predominantly rely on advanced deep learning techniques such as GANs [170] to transform images across different sensitive attributes. The methodologies for altering sensitive attributes in different types of data are extensively discussed in Section 4.1.

For ML software designed for *regression* tasks, the prediction outcomes are continuous values rather than discrete labels, which poses a challenge in determining metamorphic relations for assessing fairness. Specifically, it is difficult to ascertain whether two predicted continuous outcomes are different enough to indicate fairness issues in the software under test. To address this challenge, Udeshi et al. [278] introduced the use of a threshold to define metamorphic relations. In this approach, the difference between the outcomes of two similar instances that vary only in their sensitive attribute must be smaller than a manually specified threshold. Similarly, Perera et al. [242] proposed the concept of fairness degree, which quantifies the maximum difference in predicted values across all pairs of instances that are similar except for their sensitive attribute. The fairness degree can be utilized and defined to construct metamorphic relations and guide the generation of test inputs.

For ML software designed for *generation* tasks, it is also challenging to determine the metamorphic relations. In the case of natural language generation systems, it is particularly difficult to evaluate the identity or similarity of generated text. To address this, researchers [150, 185, 214, 257] have employed various existing natural language processing techniques to measure text similarity. These techniques include sentiment classification, perplexity and semantic similarity measurement, politeness measurement, diversity measurement, toxicity classification, and regard classification applied to machine-generated text. As a result, metamorphic relations for text generation systems require that pairs of inputs, which are identical except for the sensitive attribute, should yield generated text with consistent sentiment polarity, perplexity, semantics, and regard. This serves as a measure of similarity in the generated text. In the context of machine translation systems, where translations are generated based on input sentences, Sun et al. [268, 269] have proposed generating test oracles based on the metamorphic relationship between translation inputs and outputs. Specifically, they expect translation outputs for the original input sentence and its mutants, considering sensitive characteristics, to exhibit a certain level of consistency modulo the mutated words. To validate this consistency, similarity metrics are employed as test oracles to measure the degree of agreement between translated outputs.

Metamorphic transformations through mutating non-sensitive attributes. Some researchers [247, 250] adopt a different approach to generate metamorphic relations for fairness testing by focusing on the mutation of non-sensitive attributes. They employ metamorphic transformations as perturbations applied to samples and subsequently assess the ML software's robustness to these perturbations across various demographic groups. This methodology can be viewed as an implementation of group fairness, as it involves comparing the software's performance across different groups in response to the perturbations.

Balakrishnan et al. [106] produced synthesized samples taking multiple sensitive attributes into account and compared error rates of gender classifiers across various subgroups before and after the synthesis.

Pu et al. [247] employed makeup as a form of perturbation applied to face images. They then compared the accuracy differences as bias factors between male and female individuals on both the original face images and synthetic images generated by introducing these perturbations.

Rajan et al. [250] applied eight metamorphic transformations to speech signals and measured the increase in error rates of speech recognition for different demographic groups after these transformations.

4.2.2 Statistical Measurements as Test Oracles. Researchers have proposed various statistical fairness measurements that align with different fairness definitions. While these measurements do not serve as direct oracles for fairness testing, they provide a quantitative way to assess the fairness of the software under test. For instance, in the case of statistical parity, researchers calculate the favorable rate among demographic groups and identify fairness violations by comparing these rates. This comparison involves measuring the difference between the rates, known as **Statistical Parity Difference (SPD)**, or computing the ratio of the rates, known as **Disparate Impact (DI)** [303]. If the calculated SPD or DI exceeds a predefined threshold, then it indicates the presence of fairness bugs in the software under test.

There is a wide array of statistical fairness measurements available, with the IBM AIF360 toolkit [37] alone offering over 70 such measurements. Determining the appropriate measure of fairness is a requirements engineering problem involving negotiation among various stakeholders and different interpretations [108]. For certain social-critical application scenarios, domain knowledge is required for fairness testing and the *prediction-modeler* (e.g., data scientists and software engineers) needs to work together with the *decision-maker* (e.g., product managers and business strategists) [253]. How to achieve this is still an open challenge. However, once a fairness measure is determined, the actual statistical measurement used for testing is typically straightforward. Providing a comprehensive description and comparison of each measurement is beyond the scope of this survey. Verma and Rubin [282] conducted a survey and categorized several widely adopted statistical fairness measurements. In this survey, we expand upon their categorization based on our collected papers and present representative measurements from each category.

Measurements based on predicted outcomes. Some measurements are calculated based on the predicted outcomes of the software for privileged and unprivileged groups. For example, the aforementioned **Statistical Parity Difference (SPD)** [109] measures the difference in favorable rates among different demographic groups; DI [157] measures the ratio of the favorable rate of the unprivileged group against that of the privileged group.

Measurements based on predicted and actual outcomes. Some measurements not only consider the predicted outcomes for different demographic groups, but also compare them with the actual outcomes recorded in the collected data. For example, the **Equal Opportunity Difference (EOD)** [176] measures the difference in the true-positive rates of privileged and unprivileged groups, where the true-positive rates are calculated by comparing the predicted and actual

outcomes. Another widely adopted measurement that lies in this category is **Average Odds Difference (AOD)** [176], which refers to the average of the false-positive rate difference and the true-positive rate difference between unprivileged and privileged groups.

Measurements based on predicted probabilities and actual outcomes. Some measurements take the predicted probability scores and actual outcomes into account. For example, for any given predicted score, the calibration measurement calculates the difference in the probability of having a favorable outcome for privileged and unprivileged groups [141]; the measurement of balance for positive class calculates the difference of average predicted probability scores in the favorable class between privileged and unprivileged groups [198].

Measurements based on neuron activation. As DNNs are widely used in software systems to support the decision-making process, researchers have started to leverage the internal behaviors of DNNs to design statistical fairness measurements. Tian et al. [273] proposed a new statistical measurement based on neuron activation for DNNs. First, they computed a neuron activation vector for each label class based on the test inputs. Specifically, for a class c, each element of its neuron activation vector represents how frequently a corresponding neuron is activated by all members in the test inputs belonging to class c. Then they computed the distance between neuron activation vectors of different classes as the fairness measurement. If two classes do not show a similar distance with regards to a third class, then they consider that the DNN under test contains fairness bugs.

Measurements based on situation testing. Some researchers designed statistical measurements to approximate situation testing, which is a legal experimental procedure of seeking pairs of instances that have similar characteristics apart from the sensitive attribute value but obtain different prediction outputs [272]. Thanh et al. [272] leveraged the k-nearest neighbor classification to approximate situation testing. They first divided the dataset into the privileged group and the unprivileged group based on the sensitive attribute. Then, for each instance r in the dataset, they found the k-nearest neighbors in the two groups and denoted them as sets K_p and K_u, respectively. Finally, they calculated the proportions of instances, for which the outcome is the same as r in K_p and K_u, and measured the difference between the two proportions. If the difference is larger than a given threshold, then the instance r is considered unfairly treated. Zhang et al. [307] improved the measurement proposed by Thanh et al. [272]. They designed a new distance function that measures the distance between data instances to improve the k-nearest neighbor classification. Their function considers only the set of attributes that are identified as the direct causes of the outcome by Causal Bayesian Networks [238].

Measurements based on optimal transport projections. Several measurements [117, 258, 271] are proposed based on optimal transport projections [284], which seek for a transformation map between two probability measures. Black et al. [117] mapped the set of women in the data to their male correspondents, with the optimal transport projection to minimize the sum of the distances between a woman and the man to which she is mapped (called her counterpart). Then they extracted the positive flipset, which contained the women with favorable outcomes whose counterparts did not. They also extracted the negative flipset, which was the set of women with unfavorable outcomes whose counterparts are favorable. Finally, they calculated the size difference of the positive and the negative flipsets to measure the unfairness of the system under test.

Measurements for ranking systems. Applying the aforementioned statistical fairness measurements directly to ranking systems, which are extensively utilized in various domains such as hiring and university admissions [163, 199], poses significant challenges. To address this challenge, some researchers have tackled the ranking problem by transforming it into a classification problem and subsequently applying existing statistical fairness measurements. For example, researchers [259, 297] have used statistical parity difference as a fairness measurement,

assessing whether individuals from different groups have equal representation among desirable outcomes, such as securing top positions in the ranking. Pairwise fairness is another common statistical metric employed in the context of ranking systems [114, 200, 232]. It necessitates that a ranking system ensures an equal likelihood of a clicked item being ranked above another relevant unclicked item across different demographic groups. Recently, a fairness testing approach for deep recommender systems, namely, FairRec [173], supports the measurement of differences in evaluation metrics between demographic groups through three types of statistical measurements: recommendation performance, alignment of recommended item popularity with user preferences, and diversity in recommendations.

Measurements considering multiple subgroups. Some researchers extend their analysis to encompass a spectrum of subgroups, representing combinations of various sensitive attributes. This approach allows for a more nuanced examination of fairness in ML software. The consideration of multiple subgroups enhances the scope of fairness measurements, recognizing the diverse perspectives and experiences within different segments of the population. For instance, Chen et al. [139] used intersectional fairness as a measurement, calculating the maximum disparity between any two subgroups. Similarly, Zhang et al. [312] proposed a group fairness testing approach that specifically focuses on the SPD within subgroups.

It is difficult to determine the threshold for statistical measurements to detect the fairness bugs. For example, for the aforementioned SPD, it would be too strict to consider the software under test to be fair only when SPD equals to 0. In practice, practitioners can set a threshold for the measurement under consideration [152]. If the measurement result for the software under test is above or below the specified threshold, then the software is considered to have a fairness bug. Although the threshold could be empirically specified by engineers, it is challenging to determine the appropriate threshold for each fairness measurement.

To alleviate this problem, researchers attempt to use statistical testing based on the measurements to detect fairness bugs. Tramèr et al. [276] proposed FairTest to analyze the associations between software outcomes and sensitive attributes. The software under test is deemed to have a fairness bug if the associations are statistically significant. Taskesen et al. [271] and Si et al. [258] employed a statistical hypothesis test for the fairness measurements based on optimal transport projection. DiCiccio et al. [152] presented a non-parametric permutation testing approach for assessing whether a software system is fair in terms of a fairness measurement. The permutation test is used to test the null hypothesis that a system has equitable performance for two demographic groups (e.g., male or female) with respect to the given measurement. Gursoy et al. [174] used the permutation test to detect whether prediction errors of a regression model are distributed in a statistically significant manner across demographic groups to determine whether the model under test is unfair.

In addition, researchers construct the baseline for the fairness measurement and detect fairness bugs by comparing the measurement value with the baseline. Zhao et al. [316] used the fairness measurements calculated based on training data as their baseline against which to evaluate. Specifically, they used the obtained ML model to annotate unlabeled data instances and revealed situations when the ML process amplified existing bias by comparing the fairness measurements on training data and those on the annotated dataset. Wang and Russakovsky [288] showed that the bias amplification measurement proposed by Zhao et al. [316] conflated different types of bias amplification and failed to account for varying base rates of sensitive attributes. Then they proposed a new, decoupled metric for measuring bias amplification, which takes into account the base rate of each sensitive attribute and disentangles the directions of amplification. Wang et al. [291] presented a fairness testing approach for visual recognition systems that predicted action labels for images containing people. They trained two classifiers to predict gender from a set of ground

truth labels and model predictions. The difference in the predictability of the two models indicated whether the ML process introduced fairness bugs.

4.2.3 Test Oracle for Data Testing. ML model testing relies on metamorphic relations and statistical measurements as test oracles. Moreover, all ML program testing papers that we collect use statistical measurements (in Section 4.2.2) as test oracles. This section describes test oracles for data testing of fairness. Specifically, we summarize the test oracles for detecting feature bias, label bias, and selection bias, respectively.

Feature bias: Peng et al. [241] and Li et al. [210] assumed that feature bias is present when non-sensitive attributes exhibit correlations with sensitive attributes. In contrast, some researchers [211, 306, 308, 309] argued that feature bias manifests if specific features establish causal relationships with unfair outcomes. Black et al. [117] mapped data instances from the privileged/unprivileged group to their counterparts in the unprivileged/privileged group and determined biased features based on divergent outcomes among these counterparts.

Label bias: Chakraborty et al. [125–127] partitioned the training data into privileged and unprivileged groups and then trained two models separately for each group. Subsequently, they employed the two models to predict outcomes for each training data instance, identifying label bias if the predicted outcomes differ.

Selection bias: Chen et al. [137] and Mambreyan et al. [220] assumed that selection bias is present when a statistical association exists between a sensitive attribute and outcome labels in the training data. In contrast, some researchers [192, 275, 287, 298] assumed that different demographic groups should be equally represented in the dataset, implying an equal number of data instances for each group. Some scholars combined both oracles, expecting fair training data to exhibit an equivalent favorable rate between privileged and unprivileged groups (i.e., no statistical association between a sensitive attribute and outcomes) and an equal number of data instances for both groups [125, 126].

5 FAIRNESS TESTING COMPONENTS

This section introduces the fairness testing literature from the perspective of "*what to test*." Just as traditional software testing can be conducted on different testable parts within a software system [203], fairness testing can also be performed on different parts, including training data, ML programs, ML models, ML frameworks, and non-ML components. Figure 5 shows the categorization of this section. Existing studies primarily focus on testing training data, ML programs, and ML models.

5.1 Data Testing

ML models are developed following the data-driven paradigm. This paradigm makes ML models vulnerable to fairness bugs present in data. Specifically, fairness bugs in training data can be learned and propagated throughout the ML model development pipeline, leading to the creation of biased and unfair ML software systems. To tackle this problem, data testing approaches, which detect bugs in ML training data [304], have been proposed for fairness testing. They detect the bias in data features, data labels, and data distribution.

5.1.1 Detecting Feature Bias. Feature bias arises when certain features in the training data exhibit a strong correlation with sensitive attributes, causing them to become the underlying source of software unfairness [210]. Zhang and Harman [303] investigated the impact of the feature set on the fairness of ML models. Their findings demonstrated that the selection of features has a substantial influence on fairness, thereby highlighting the importance of considering testing data features in fairness-related endeavors.

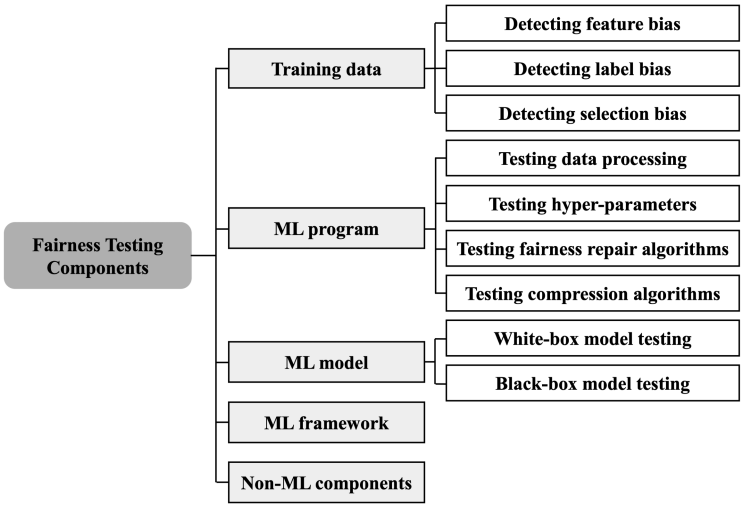

Fig. 5. Testing components of fairness testing.

To identify the features responsible for fairness issues, it is natural to suspect that the software exhibits discrimination against a particular demographic group due to its consideration of sensitive attributes during both the training and prediction phases [127]. To investigate whether the sensitive attribute serves as the underlying cause of fairness problems, Chakraborty et al. [127] conducted an experiment where they removed the sensitive attribute information from the data (i.e., fairness through unawareness). Surprisingly, they found that the resulting machine learning software demonstrated a similar degree of unfairness as observed previously.

A similar discovery was made in a real-world case: Back in 2016, it was revealed that Amazon's same-day delivery service exhibited discriminatory behavior towards neighborhoods with a disproportionately high population of Black residents [22]. Although the ML model behind the service did not explicitly incorporate race information, the presence of correlated attributes in the training data allowed for the possibility of bias. Specifically, it was found that the "Zipcode" information utilized during model training exhibited a strong correlation with race, causing the ML model to indirectly infer race information from it.

To identify non-sensitive features that could potentially contribute to fairness issues, Peng et al. [241] employed logistic regression and decision tree algorithms as models to infer the relationships between sensitive attributes and non-sensitive features. Similarly, Li et al. [210] utilized linear regression to analyze the association between each feature and sensitive attributes, thereby identifying features that may introduce bias.

In contrast, Zhang et al. [306, 308, 309] employed discrimination detection based on causal modeling to detect both direct and indirect discrimination within datasets. They constructed a causal graph to capture the causal relationships between attributes and outcomes. Direct discrimination was modeled as the causal effect occurring along the direct path from sensitive attributes to the outcome. However, indirect discrimination was represented by the causal effects along other paths involving non-sensitive features.

Li and Xu [211] proposed a method to detect unknown biased attributes in a classifier that predicts a target attribute, such as gender, based on input images. The biased attribute in question is distinct from the target attribute. For instance, if a gender classifier exhibits varying predictions for female images based on their skin tones, then the skin tone attribute would be considered

biased. To identify this unknown biased attribute, the authors optimized a hyperplane within the latent space of a generative model. By analyzing the transformation of synthesized counterfactual images generated by the model, human observers can interpret the semantic meaning of the biased attribute hyperplane. For example, images transitioning from "light skin" to "dark skin" indicate the presence of bias associated with skin color.

Black et al. [117] utilized optimal transport projections [284] to map data instances from the unprivileged group to their counterparts in the privileged group. This allowed them to extract the positive flipset, which consists of unprivileged group members with favorable outcomes whose counterparts experienced unfavorable outcomes. Additionally, they computed the negative flipset, consisting of unprivileged group members with unfavorable outcomes whose counterparts experienced favorable outcomes. By analyzing the members of these flipsets, they determined which features contributed to inconsistent classifications.

5.1.2 Detecting Label Bias. Label bias occurs when factors unrelated to the determination of labels influence the process that generates outcome labels [293]. ML models are often developed using data collected over an extended period. During the data collection process, labels are typically assigned by human annotators or algorithms, introducing the potential for human and algorithmic biases to be encoded into the labels.

To detect label bias, Chakraborty et al. [125–127] employed situation testing to identify biased data points and remove them from the training data. They divided the dataset into privileged and unprivileged groups based on the sensitive attribute. Two separate models were then trained on the data from each group. For each training instance, the predictions from both models were compared. If the two models produced divergent results, then there was a probability that the label of that data point was biased.

Chen and Joo [133] utilized facial action units, objective indicators of fundamental muscle actions associated with different facial expressions, to detect label bias in widely used datasets for facial expression recognition. They demonstrated that many expression datasets exhibited significant label bias between different gender groups, particularly concerning expressions of happiness and anger. Furthermore, they found that conventional fairness repair methods were unable to completely mitigate such biases in trained models.

5.1.3 Detecting Selection Bias. Selection bias occurs when the process of sampling training data introduces an unexpected correlation between sensitive attributes and the outcome [293]. For example, the Compas dataset [23], widely studied in the fairness literature, has been shown to exhibit unintended correlations between race and recidivism [293]. This dataset was collected during a specific time period (2013 to 2014) and from a particular county in Florida, with its inherent policing patterns, making it susceptible to the introduction of unintentional correlations.

Researchers primarily employ distribution testing to detect selection bias in the data. Chen et al. [137] tested whether the training data satisfied the "We Are All Equal" worldview, which assumes that there should be no statistical association between the outcome and the sensitive attribute. They specifically examined whether the favorable rates of privileged and unprivileged groups were equal. Chakraborty et al. [125, 126] not only analyzed the disparity in favorable rates between privileged and unprivileged groups but also compared the numbers of data instances in the two groups.

Kärkkäinen and Joo [192] detected bias in public face datasets, revealing a strong bias toward Caucasian faces while other racial groups (e.g., Latino) were significantly underrepresented. Such biases increase the risk of introducing fairness issues in facial analytic systems and limit their applicability.

Similarly, Torralba and Efros [275] investigated computer vision datasets to evaluate whether existing datasets genuinely represent unbiased representations of the real world. They assessed

how well an object detector trained on one dataset generalized when tested on a representative set of other datasets.

Yang et al. [298] collected perceived demographic attributes on a popular face detection dataset [86] and observed skewed demographic distributions. Face detectors trained on this dataset exhibited demographic bias, as measured by performance disparities among different groups.

Wang et al. [287] detected selection bias in visual datasets across three dimensions: object-based bias, gender-based bias, and geography-based bias. Object-based detection considered statistics related to object size, frequency, context, and diversity of object representation. Gender-based detection revealed the stereotypical portrayal of individuals of different genders. Geography-based detection focused on the representation of different geographic locations.

Mambreyan et al. [220] analyzed datasets used for lie detection and discovered significant sex bias within them. Specifically, the percentage of instances labeled as lies for females was greater than that for males in the dataset. They further examined the effect of this bias on lie detection, training a classifier to predict the sex of the identity in a video and using sex as a proxy for lies (predicting lies for females and truth for males). This deception detector simulated a classifier that relied solely on selection bias. The results demonstrated that the performance of this biased classifier was comparable to the state-of-the-art, suggesting that recent techniques claiming near-perfect results may exploit selection bias.

5.2 ML Program Testing

An ML program includes various parts such as data processing, decision-making logic, and runtime configurations (e.g., ML hyper-parameters) [134]. Each of these parts can potentially introduce a discordance between the existing fairness conditions and the desired ones for the final ML software system. Testing the ML program can help identify fairness issues within its implementation.

5.2.1 Testing Data Processing. In ML programs, it is common to include data processing scripts to manipulate and transform training data for downstream learning tasks. The processing can have a significant impact on the fairness of the software.

Biswas and Rajan [116] and Valentim et al. [280] have investigated the introduction of fairness bugs through data processing methods using causal reasoning. They systematically intervened in the development process of ML software by applying different commonly used data processing methods while keeping other settings unchanged. Their findings indicate that certain pre-processing methods indeed introduce fairness bugs, while other methods may improve software fairness.

Caton et al. [124] have observed that real-world datasets often contain missing values, and one common approach to address this issue is to impute the missing values using various techniques during the data processing phase. To evaluate the impact of different imputation strategies on fairness outcomes, they conducted tests and examined the resulting fairness implications.

5.2.2 Testing Hyper-parameters. The hyper-parameters specified in ML programs can affect fairness. To validate it, researchers explore whether different hyper-parameter settings lead to varying levels of software fairness. This testing process is treated as a search-based problem, aiming to discover optimal settings within the hyper-parameter space.

Chakraborty et al. [127, 129] proposed Fairway, a method that combines situation testing with multi-objective optimization. Since there is often a tradeoff between fairness and ML performance (such as accuracy) [144], Fairway employs sequential model-based optimization [231] to search for hyper-parameters that maximize software fairness while minimizing any negative impact on other performance measures.

Similarly, Tizpaz-Niari et al. [274] considered both fairness and accuracy in their approach. They introduced Parfait-ML, which offers three dynamic search algorithms (independently random, black-box evolutionary, and gray-box evolutionary) to approximate the Pareto front of hyper-parameters that balance fairness and accuracy. Parfait-ML not only provides a statistical method to identify hyper-parameters that systematically influence fairness but also incorporates a fairness repair method to discover improved hyper-parameter configurations that simultaneously enhance fairness and accuracy.

Gohar et al. [167] conducted fairness testing on hyper-parameters in ensemble learning. As ensemble hyper-parameters are more intricate due to their impact on how learners are combined within different ensemble categories, the researchers investigated the effects of ensemble hyper-parameters on fairness. They also presented how to design fair ensembles using ensemble hyper-parameters.

5.2.3 Testing Fairness Repair Algorithms. As fairness becomes an increasingly important requirement for software systems, engineers may incorporate fairness repair algorithms (also known as bias mitigation algorithms) into their programs to ensure fairness. Researchers focus on testing whether these fairness repair algorithms effectively reduce fairness bugs without introducing side effects such as a decrease in accuracy.

Biswas and Rajan [115] applied seven fairness repair algorithms to 40 top-rated ML models collected from a crowdsourced platform. They compared individual fairness, group fairness, and ML performance before and after applying these algorithms.

Qian et al. [248] applied fairness repair techniques to five widely adopted ML tasks and examined the variance of fairness and ML performance associated with these techniques. They investigated whether identical runs with a fixed seed produced different results. The findings indicated that most fairness repair techniques had undesirable impacts on the ML software, such as reducing accuracy, increasing fairness variance, or increasing accuracy variance.

Zhang and Sun [311] evaluated existing fairness repair techniques on DNNs and discovered that while these techniques improved fairness, they often resulted in a significant drop in accuracy. In some cases, fairness and accuracy were both worsened. They proposed an adaptive approach that selects the fairness repair method for a DNN based on causality analysis [266].

Hort et al. [183] introduced a benchmarking framework called Fairea. Prior work often measured the impacts of fairness repair algorithms on fairness and ML performance separately, making it unclear whether the improved fairness was solely due to the unavoidable loss in ML performance. Fairea addressed this issue by providing a unified baseline to evaluate and compare the fairness-performance tradeoff of different repair methods.

Chen et al. [138] utilized Fairea to conduct a large-scale, comprehensive empirical evaluation of 17 representative bias mitigation methods from both the ML and SE communities. They evaluated these methods across 12 ML performance metrics, 4 fairness metrics, and 24 types of fairness-performance tradeoff measurements.

Hort and Sarro [182] observed another side effect of fairness repair: It could lead to the loss of discriminatory behaviors of anti-protected attributes. Anti-protected attributes refer to attributes on which one might want the ML decision to depend (e.g., students with completed homework should receive higher grades).

Orgad et al. [235, 236] evaluated fairness repair approaches for NLP models from two aspects: extrinsic bias (performance difference across different demographic groups) and intrinsic bias (bias in models' internal representations, e.g., sentence embeddings). They found that the two types of bias may not be correlated, and the choice of bias measurement and dataset can significantly affect the evaluation results.

5.2.4 Testing Compression Algorithms. A computation-intensive DL model can be efficiently executed on PC platforms with GPU support, but it cannot be directly deployed and executed on platforms with limited computing power, such as mobile devices [134]. To address this issue, model compression algorithms have been proposed to represent DL models in a smaller size with minimal impact on their performance [136]. Common model compression techniques include quantization (representing weight values of DL models using smaller data types), pruning (eliminating redundant weights that contribute little to the model's behavior), and knowledge distillation (transferring knowledge from a large model to a smaller one) [140]. The widespread adoption of model compression for DL models has motivated researchers to detect fairness bugs introduced by these algorithms.

Since model compression is often applied to large DL models, existing fairness testing of model compression algorithms typically focuses on complex NLP models [118] and computer vision models [118, 180, 213, 264]. Hooker et al. [180] demonstrated that pruning and quantization can amplify gender bias when classifying hair color in a computer vision dataset. Xu and Hu [296] tested the effect of distillation and pruning on bias in generative language models and provided empirical evidence that distilled models exhibited less bias. Stoychev and Gunes [264] detected fairness bugs introduced by different model compression algorithms in various facial expression recognition systems but did not observe consistent findings across different systems.

5.3 Model Testing

Most existing fairness testing techniques primarily focus on the evaluation of individual ML models [97, 156, 156, 160, 278, 295, 310, 313, 318]. These techniques can be directly applied to the final ML models obtained, using either a black-box or white-box approach. The distinction between white-box testing and black-box testing lies in the level of access to training data and internal knowledge of the ML models.

5.3.1 Black-box Model Testing. Black-box model testing is a technique used to detect fairness issues in ML models without relying on access to training data or knowledge of the internal model structure. This approach primarily relies on analyzing the behavior of the model based on the input space.

Fairness testing in the field typically relies on statistical measurements to identify fairness bugs in black-box models based on their prediction behaviors. For instance, Tramèr et al. [276] conducted an analysis to detect fairness bugs by examining the associations between prediction outcomes and sensitive attributes. They aimed to uncover any potential biases or unfairness present in the model's predictions. Similarly, Bae [105] compared the performance of pedestrian trajectory prediction models across different demographic groups, aiming to uncover variations or biases in their performance.

Additionally, fairness testing approaches often leverage metamorphic relations to detect fairness bugs by applying transformations to software inputs. These transformations aim to identify unexpected changes in the model's predictions. Many of these techniques employ black-box testing methodologies. For example, the Themis tool [101, 160] generates random test inputs and checks if the software system produces consistent outputs for individuals who differ only in sensitive attribute values. Similarly, Aequitas [278] and ExpGA [156] search the input space of the software for discriminatory instances that reveal unfair predictions.

Black-box testing is commonly used to detect fairness bugs in complex software systems, including NLP, computer vision, and ranking systems, where the internal workings of the system are not fully visible to the testers.

Researchers have developed various text templates to uncover fairness bugs in different NLP systems, such as sentiment analysis [102, 196], machine translation [290], text generation [150,

185, 257], natural language inference [256], named entity recognition [222], and conversational systems [286]. Further details regarding these studies can be found in Section 4.1.4.

For computer vision systems, state-of-the-art fairness techniques [146, 148, 149, 170, 191, 193, 314] often utilize GAN-based algorithms to generate images that differ in sensitive attributes. These techniques then check if the computer vision systems make different decisions for equivalent image mutants. More information about these techniques is available in Section 4.1.4.

Ranking systems, being predominantly black-box, are also tested in a black-box manner [114, 200, 232, 259, 297]. For instance, researchers have measured whether different demographic groups have proportional representation in top-ranking positions based on the system's ranking outputs. FairRec [173] assesses recommendation differences across demographic groups using statistical measurements such as recommendation performance, alignment of recommended item popularity with user preferences, and diversity in recommendations.

Some black-box testing techniques approximate the behavior of the black-box software using a white-box model, allowing the application of white-box testing techniques. For example, Aggarwal [97] approximated the decision-making process of the black-box ML software using a decision tree constructed through a local model explainer. They then employed symbolic execution-based test input generation to discover discriminatory inputs. Sharma and Wehrheim [255] first approximated the black-box software with a white-box model based on its behaviors. They subsequently developed a property-based testing mechanism for fairness checking, where specific fairness requirements can be specified using an assume-assert construct. Test cases were automatically generated to attempt to violate the specified fairness property.

5.3.2 White-box Model Testing. White-box model testing aims to identify fairness bugs by examining either the training data or the internal structure and information of the ML model that is accessible to test engineers.

Some approaches leverage *training data* to uncover unfair predictions without accessing the internal of ML models. Chakraborty et al. [128] proposed an explanation method based on k-nearest neighbors to detect bias in ML software predictions. They identified instances predicted unfavorably and examined their k-nearest neighbors with favorable labels from the training data. By comparing the distribution of these neighbors with the test instance, they determined and explained the presence of bias.

Zhao et al. [316] used fairness measurements from training data as a baseline to identify bias amplification. They annotated unlabeled data using the ML model and compared fairness measurements between the training data and the annotated dataset to reveal bias amplification.

Wang and Russakovsky [288] highlighted issues with Zhao et al.'s bias amplification measurement and proposed a new, decoupled metric that considers varying base rates of sensitive attributes.

Cabrera et al. [122] developed FAIRVIS, a testing tool for subgroup fairness. It efficiently searches for potential issues among numerous subgroups, clustering the training dataset to identify statistically similar subgroups and calculating group fairness metrics using an entropy-based approach. FAIRVIS presents generated subgroups sorted by their group fairness metrics.

Patel et al. [237] utilized combinatorial t-way testing [201] for fairness testing. This coverage-based data sampling method generates diverse datasets by applying logical constraints. They created an input parameter model from the training data and used it to generate a t-way test set. Discriminatory instances were identified by mutating protected attributes in each test.

Several techniques [270, 310, 313–315, 318] utilize *gradient information*, which represents the direction of steepest ascent in the loss function, to generate test inputs for fairness testing. ADF [313, 315] focuses on discriminatory instances near the decision boundary of DNNs and employs

gradients to guide the search for neighboring test inputs. EIDIG [310] reduces gradient calculations to accelerate the search process.

Researchers have also developed various methods to detect and analyze *neurons* responsible for unfair outcomes in emerging **deep learning (DL)** models. NeuronFair [318] utilizes neuron analysis to identify biased neurons contributing to unfairness and generates discriminatory instances to amplify the activation differences of these biased neurons. It demonstrates strong interpretability, generation effectiveness, and data generalization.

DeepFAIT [314] employs significance testing to identify fairness-related neurons by analyzing the activation differences between privileged and unprivileged groups.

Vig et al. [283] apply **Causal Mediation Analysis (CMA)** to identify causally implicated parts, such as neurons or attention heads, in the unfair predictions of a DNN model. CMA measures the direct and indirect effects of targeted neurons on the final unfair predictions, considering each neuron as an intermediate mediator.

Tian et al. [273] introduce a novel statistical measurement using neuron activation in DNNs. They compute neuron activation vectors for each label class based on test inputs and calculate the distance between these vectors to assess fairness. Dissimilar distances with respect to a third class indicate the presence of fairness bugs in the tested DNN.

Gao et al. [161] propose FairNeuron for DNNs, which employs neuron slicing to identify conflict *paths* containing neurons that rely on sensitive attributes for predictions. Biased instances triggering the selection of sensitive attributes are identified using these paths, and the model is retrained through selective training. FairNeuron ensures that the conflict paths learn all important features for prediction instead of biased ones for the identified biased instances while retaining the original training approach for other instances.

Additionally, Zhang et al. [300] developed a method to identify and rectify fairness-related *paths* in decision tree and random forest models. They employed a MaxSMT solver to determine the paths that could be altered while satisfying fairness and semantic difference constraints. The identified paths were refined by modifying the leaf labels, resulting in a repaired fair model.

6 RESEARCH TRENDS AND DISTRIBUTIONS

In Figure 1, we have shown that fairness testing is experiencing a dramatic increase in the number of publications. This section further analyzes the research trends and distributions of fairness testing.

6.1 Research Venues

We first describe research trends in terms of the research communities engaging in fairness testing. Since 2017, an increasing number of research communities have dedicated their efforts to studying fairness testing. Notable contributions include:

— Galhotra et al. [160] introduced the first fairness testing approach for ML software in the SE community, receiving the Distinguished Paper Award at ESEC/FSE 2017.
— Zhao et al. [316] detected bias in datasets and ML models for visual recognition tasks, earning the Best Paper Award at EMNLP 2017.
— Díaz et al. [151] identified age-related bias in sentiment analysis systems, receiving the Best Paper Award at CHI 2018.
— Ribeiro et al. [251] proposed CheckList, a task-agnostic methodology for testing NLP models, including fairness testing, which received the Best Paper Award at ACL 2020.
— Zhang et al. [313] proposed a search-based discriminatory instance generation approach for DNNs, which received the Distinguished Paper Award at ICSE 2020 and was selected for SIGSOFT Research Highlights.

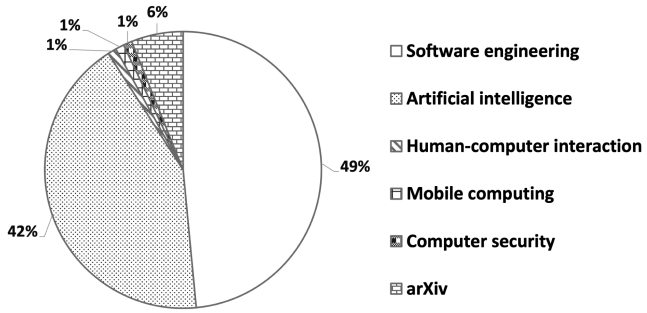

Fig. 6. Research distribution among different venues.

— Chakraborty et al. [125] addressed selection bias and label bias in training data and were honored with the Distinguished Paper Award at ESEC/FSE 2021.

These six best-paper awards, as well as being a credit to their authors, also demonstrate both the significant level of interest and the high quality of research on fairness in three different research communities:

— Software Engineering (ICSE and ESEC/FSE)
— Natural Language Processing (EMNLP and ACL)
— Human-Computer Interaction (CHI)

Figure 6 illustrates the distribution of the collected papers across various research venues. The majority of fairness testing papers (49%) are published in software engineering venues, including ICSE, ESEC/FSE, ASE, ISSTA, TSE, and TOSEM. Artificial intelligence venues, such as ICML, NeurIPS, IJCAI, ACL, EMNLP, CVPR, ECCV, and KDD, account for 42% of the fairness testing papers.

Furthermore, our survey reveals that fairness testing is gaining traction in other research communities, such as computer security, human-computer interaction, and mobile computing communities. This highlights the broad audience and significance of our survey across multiple disciplines.

6.2 Machine Learning Categories

In this section, we explore the research trend of fairness testing across different ML categories. Following previous work [304], we categorize the gathered papers into two groups: those that concentrate on DL software and those that address general ML software.

Out of the total number of papers, 50 papers (50%) focus on conducting fairness testing for DL software, 41 papers (41%) specifically target general ML software, and 8 papers (8%) consider both traditional ML software and DL software. The significant volume of publications on fairness testing for DL software can be attributed to several factors. On one hand, DL has gained widespread adoption and is being utilized in a diverse range of software applications, generating significant interest from the research community. On the other hand, compared to traditional ML algorithms such as regression and decision trees, DL models are less interpretable [107], making it more challenging to directly reason about fairness.

To gain further insights, we analyze the publication trends for both categories over the years. Figure 7 depicts the number of papers focusing on fairness testing in general ML and DL per year. Our analysis reveals a clear shift in research focus, with a transition from testing general ML software to testing DL software. Prior to 2019, fairness testing research primarily concentrated on

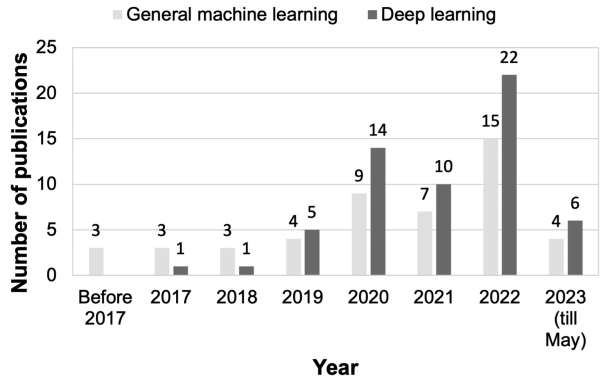

Fig. 7. Number of fairness testing papers on general machine learning and deep learning per year.

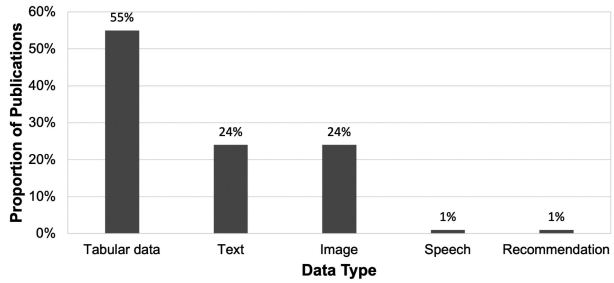

Fig. 8. Distribution of different data types in fairness testing papers.

general ML. However, since 2019, the number of papers specifically addressing DL has experienced a notable surge, surpassing the publications on general ML.

6.3 Data Types

In this section, we investigate the research trends of fairness testing in applications that involve different types of data.

Out of the total number of papers (100) that we have collected, 5 of them consider more than one data type. We count each of these papers for each data type they examine, allowing us to analyze the distribution across various data types. The findings are illustrated in Figure 8.

Our analysis indicates that among the publications we have collected, a significant portion focuses on testing software applications that utilize tabular data as inputs, accounting for 55% of the total. Furthermore, approximately 24% of publications address fairness testing in applications involving text inputs, while another 24% specifically tackle fairness issues in applications utilizing image inputs.

It is important to note that fairness testing for other data types, such as speech and recommendation systems, has not yet received extensive investigation, representing only a small percentage of the publications, approximately 1% each. These data types remain relatively underexplored in the context of fairness testing, emphasizing the need for further research and attention to ensure fairness across a wider range of data-driven applications.

We also plot the data type distribution for SE publications. Figure 9 shows the results. In comparison to the broader research landscape, the SE community predominantly focuses on applications

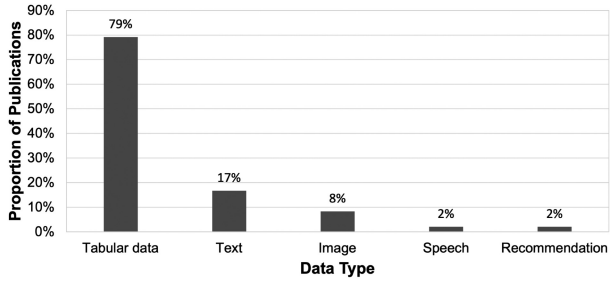

Fig. 9. Distribution of different data types in fairness testing papers published in SE venues.

involving tabular data, representing a substantial majority of fairness testing publications in SE venues, amounting to 79%. In contrast, publications addressing text- or image-based problems account for only 17% and 8%, respectively. These figures are significantly lower than the average distribution across all data types.

6.4 Fairness Categories

Figure 10 provides insights into the distribution of different fairness categories within the fairness testing literature. Our analysis reveals that comparable research efforts have been dedicated to exploring various aspects of fairness testing.

Specifically, we observe that 46% of fairness testing papers focus on group fairness, addressing issues related to fairness among different groups. Another 46% of papers concentrate on individual fairness, investigating fairness concerns at the individual level. The remaining 8% of papers examine both group fairness and individual fairness, recognizing the importance of considering both perspectives in fairness testing research.

The finding may initially appear contradictory when comparing it with Table 5, where we observe that nearly all test generation techniques are proposed for individual fairness. This discrepancy can be attributed to the distinct characteristics of individual fairness and group fairness.

When examining group fairness, researchers can leverage real-world data to construct test inputs that capture fairness considerations among different groups. This availability of data facilitates the exploration of group fairness in testing scenarios. However, for individual fairness, it becomes challenging for researchers to identify pairs of instances that satisfy the specific input requirements associated with individual fairness. For instance, it is not always straightforward to find two individuals who differ solely in a sensitive protected attribute, making test input generation more focused on individual fairness.

Despite these differences in test input generation, it is important to note that fairness testing studies overall investigate group fairness and individual fairness at a similar level. The distribution of research efforts aims to address both perspectives, acknowledging the significance of both group and individual fairness in testing methodologies.

6.5 Testing Manners

We further categorize existing fairness testing techniques based on the software testing approach employed, distinguishing between white-box and black-box methods. It is worth noting that although Tizpaz-Niari et al. [274] proposed black-box and gray-box techniques for testing hyperparameters, we consider their approach as white-box, because it requires access to the training data for model training and fairness evaluation.

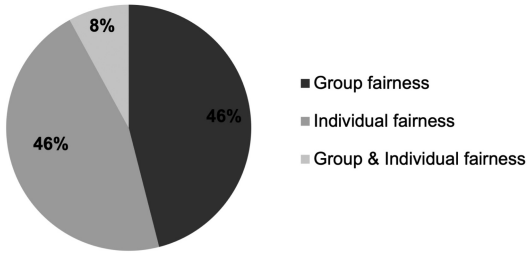

Fig. 10. Distribution of different fairness categories.

In our analysis, we find that 53% of the 100 papers employ black-box testing techniques, while 47% adopt white-box testing methods. This observed gap between black-box and white-box testing proportions is reasonable.

Compared to black-box testing, white-box testing necessitates access to either training data or the internal workings of software systems. However, fairness testing often applies to systems that are human-related and have social significance, making it challenging to disclose internal information to the public due to privacy concerns or legal policies. Therefore, it is reasonable that more research studies focus on black-box testing approaches, which do not require detailed knowledge of the internal system mechanisms.

6.6 Tasks and Sensitive Attributes

In the collected papers, researchers typically present their techniques and then employ datasets featuring specified tasks and sensitive attributes for technique evaluation. In this section, we provide a summary of prevalent tasks and sensitive attributes commonly considered in the fairness testing literature.

We present in Table 6 commonly studied tasks in fairness testing literature. As detailed in Section 6.3, existing fairness testing techniques span various data types, enabling support for diverse tasks. Notably, income prediction, credit risk prediction, and recidivism/crime prediction emerge as the three most widely studied. These tasks also feature as the most widely explored ones in the bias mitigation (i.e., fairness improvement) literature, as indicated by a recent survey [181].

We list in Table 7 commonly studied sensitive attributes in the fairness testing literature. Overall, the literature covers a wide range of sensitive attributes, including sex/gender, race/ethnicity, age, country, occupation, religion, and sexual orientation. Notably, the top three sensitive attributes are sex/gender, race/ethnicity, and age, which are consistent with previous findings observed in the general software fairness literature [263]. They are considered in 89%, 57%, and 41% of the collected papers, respectively.

7 DATASETS AND TOOLS

Based on the collected papers, this section summarizes the public datasets and open-source tools for fairness testing to provide a quick navigation for researchers and practitioners.

7.1 Public Datasets

This section lists the public datasets in the literature based on our collected papers. Table 8 provides detailed information about these datasets, including their sizes, data types, sensitive attributes, usage scenarios, and access links. Many of these datasets comprise tabular data, making them suitable for traditional ML classifiers.

Table 6. Commonly Studied Tasks in the Fairness Testing Literature

Task	#Publications
Income prediction	49
Credit risk prediction	33
Recidivism/crime prediction	26
Deposit subscription prediction	24
Face recognition/facial analysis	16
Healthcare prediction	10
Conversational AI/language generation	9
Sentiment analysis	8
Heart heath prediction	7
Image tagging/classification	7
Exam performance/grade prediction	5
Coreference resolution	3
Machine translation	3
Individual survival prediction	3
Recommender systems	2
Pedestrian trajectory prediction	1
Admission prediction	1
Loan application	1
Hiring	1
Pricing	1
Fraud detection	1
Car rental	1
Execution prediction	1
Toxicity detection	1
Name entity recognition	1
Speech recognition	1
COVID-19 new case prediction	1
Emergency department wait-time prediction	1

In recent years, there has been a surge in the availability of text and image datasets, driven by the growing popularity of natural language processing and computer vision. These datasets are often sourced from social media platforms like Twitter. It is worth noting that certain datasets come with specific usage constraints that researchers must consider when utilizing them. For instance, the well-known image dataset CelebA [16] is restricted to non-commercial research purposes only.

For a comprehensive overview of the existing fairness datasets, we recommend referring to the works of Le Quy et al. [249] and Fabris et al. [155]. Le Quy et al. [249] surveyed tabular datasets specifically for fairness research, while Fabris et al. [155] expanded the survey to include unstructured data such as text and images. Their surveys encompass datasets from diverse domains, including social sciences, computer vision, health, economics, business, and linguistics.

7.2 Open-source Testing Tools

There is a recent proliferation of open-source tools for supporting fairness testing. Nevertheless, Lee and Singh [207] demonstrated that there is a steep learning curve for practitioners to use these fairness tools. Presently, there is a lack of guidance on tool adoption [207].

Table 7. Commonly Studied Sensitive Attributes in the Fairness Testing Literature

Sensitive attribute	#Publications
Sex/gender	89
Race/ethnicity/skin tone	57
Age	41
Country/nationality/geography	8
Occupation/profession	6
Religion	3
Political ideology	2
Sexual orientation	2
Person name	2
Ability	2
Body	2
Character	2
Culture	2
Social status	2
Marital status	1
Hair color	1
Victim	1

To address this gap, we provide a summary of 41 open-source fairness testing tools in this section, aiming to assist fairness researchers and practitioners in selecting the most suitable tools. The details of these tools are presented in Table 9. The table includes fairness testing tools for various domains, including general ML (e.g., FairTest [276]), DL (e.g., ADF [313] and EIDIG [310]), natural language processing (e.g., ASTRAEA [262] and BiasFinder [102]), computer vision (e.g., REVISE [287]), and speech recognition (e.g., AequeVox [250]).

8 RESEARCH OPPORTUNITIES

Fairness testing remains in a relatively embryonic state. Research in this area is experiencing rapid growth, so there are plenty of open research opportunities. In this section, we outline the challenges for fairness testing and present promising research directions and open problems.

8.1 Absence of or with Multiple Sensitive Attributes

Fairness testing in the absence of sensitive attribute information. Existing fairness testing techniques rely on the existence of sensitive attributes, but in practice, this information might be unavailable or imperfect for many reasons [103]. On the one hand, the data may be collected in a setting where the sensitive attribute information is unnecessary, undesirable, or even illegal, considering the recently released regulations such as **GDPR (General Data Protection Regulation)** [285] and **CCPA (California Consumer Privacy Act)** [169]. On the other hand, users may withhold or modify sensitive attribute information, for example, due to privacy concerns or other personal preferences. To tackle this issue, a straightforward solution is to first use existing demographic information inference techniques (e.g., gender inference, race inference, and age inference) to infer the sensitive attribute and then apply fairness testing techniques. However, existing inference techniques may not be fully satisfactory, and their application scenarios remain limited [135]. Moreover, building a model to infer sensitive information leaves open the possibility that the model

Table 8. Public Datasets for Fairness Testing

Dataset	Description	Data Type	Size	Sensitive Attribute(s)	URL
Adult Income	Income prediction	Tabular	48,842	sex, race	[27]
Census-Income (KDD)	Income prediction	Tabular	299,285	sex, race	[5]
Compas	Recidivism prediction	Tabular	6,167	sex, race	[23]
German Credit	Credit risk prediction	Tabular	1,000	sex	[3]
Default Credit	Credit risk prediction	Tabular	30,000	sex	[24]
Home Credit	Credit risk prediction	Tabular	37,511	sex	[30]
Bank Marketing	Deposit subscription prediction	Tabular	30,488	age	[13]
Mep15	Health care needs prediction	Tabular	15,830	race	[19]
Mep16	Health care needs prediction	Tabular	15,675	race	[26]
Dutch Census	Income prediction	Tabular	60,421	gender	[14]
Heart Health	Heart health prediction	Tabular	303	age	[7]
Arrhythmia	Cardiac arrhythmia prediction	Tabular	452	sex	[70]
Student Performance	Final year grade prediction	Tabular	649	sex	[15]
UFRGS	GPA prediction	Tabular	43,302	gender, race	[49]
Law School	Exam performance prediction	Tabular	20,000	gender, race	[4]
Titanic	Individual survival prediction	Tabular	891	sex	[34]
Communities and Crime	Crime prediction	Tabular	2,215	race, age	[10]
Diabetes	Readmission prediction	Tabular	100,000	race	[2]
Heritage Health	Staying in the hospital or not	Tabular	147,473	age	[17]
Fraud	Fraud detection	Tabular	1,100	age	[6]
US Executions	Execution prediction	Tabular	1,437	sex, race	[1]
LFW	Face recognition	Image	10,000	sex, race	[8]
BUPT-Transferface	Face recognition	Image	1,300,000	race	[43]
VGGFace	Face recognition	Image	2,600,000	gender, race	[21]
COCO-gender	Object detection	Image	66 objects	gender	[28]
CelebA	Facial analysis	Image	202,599	gender	[16]
RAF-DB	Facial analysis	Image	15,339	gender, race, age	[32]
PBB	Facial analysis	Image	1,270	gender, skin type	[39]
FairFace	Facial analysis	Image	108,501	gender, race, age	[66]
WinoST	Speech translation	Audio	3,888	gender	[59]
AequeVox	Speech recognition	Audio	68	gender, accent	[69]
WinoBias	Coreference resolution	Text	3,160	gender	[40]
Winogender	Coreference resolution	Text	720	gender	[41]
IMDB	Sentiment analysis	Text	50,000	gender, country, occupation	[11]
Twitter Sentiment140	Sentiment analysis	Text	1.6 million	gender, country, occupation	[67]
SST	Sentiment analysis	Text	11,855	gender, country	[31]
EEC	Sentiment analysis	Text	8,640	sex, race	[38]
Large Movie Review	Sentiment analysis	Text	50,000	gender	[12]
Wikipedia comments	Toxicity detection	Text	127,000	religion, country, ethnic, race	[35]
Jigsaw Comments	Toxicity classification	Text	313,000	religion, country, ethnic, race	[46]
nlg-bias	Language generation	Text	360	gender, race, sexual orientation	[48]
BOLD	Language generation	Text	23,679	profession, gender, race, religion, political ideology	[95]
HOLISTICBIAS	Language generation	Text	459,758	13 attributes	[76]
DialogueFairness	Conversational AI	Text	300,000	gender	[54]
BiasAsker	Conversational AI	Text	8,110	ability, age, body character, culture, gender, profession, race, religion, social victim	[88]
MovieLens	Recommendation	Movie ratings	1 million	gender, age, occupation	[20]
LFM360K	Recommendation	Music listening history	17 million	gender, age, nationality	[18]
BlackFriday	Recommendation	Purchase history	550,068	gender, age, occupation, city category, years of stay in the current city, marital status	[42]

may ultimately be used more broadly, with possibly unintended consequences [103]. Therefore, more research is needed to tackle fairness testing in the absence of sensitive attribute information.

Fairness testing with multiple sensitive attributes. Software systems can have multiple sensitive attributes that need to be considered at the same time [139, 168]. Human attributes, such as sex, race, and class, intersect with one another, and unfair software systems built into society lead to systematic disadvantages along these intersecting attributes [145, 195]. However, existing fairness testing work often tackles a single sensitive attribute at a time. To the best of our knowledge, there has been a little work that explores fairness testing for compounded or intersectional effects of

Table 9. Open-source Tools for Fairness Testing

Tool [ref]	Application	Component	Description	URL
FairTest [276]	General ML	Model	Analyzing associations between outcomes and sensitive attributes	[29]
Themis [160]	Classification (Tabular data)	Model	Black-box random discriminatory instance generation	[33]
Aequitas [278]	Classification (Tabular data)	Model	Automated directed fairness testing	[36]
ExpGA [156]	Classification (Tabular data, text)	Model	Explanation-guided fairness testing through genetic algorithm	[63]
fairCheck [255]	Classification (Tabular data)	Model	Verification-based discriminatory instance generation	[55]
MLCheck [254]	Classification (Tabular data)	Model	Property-driven testing of ML models	[80]
LTDD [210]	Classification (Tabular data)	Data	Detecting which data features and which parts of them are biased	[78]
Fair-SMOTE [125]	Classification (Tabular data)	Model	Detecting biased data labels and data distributions	[64]
FairMask [241]	Classification (Tabular data)	Data	Extrapolation of correlations among data features that might cause bias	[92]
Fairway [127]	Classification (Tabular data)	Data, ML program	Detecting biased data labels and optimal hyper-parameters for fairness	[56]
Parfait-ML [274]	Classification (Tabular data)	ML program	Searching for hyper-parameters optimal to ML fairness	[82]
Fairea [183]	Classification (Tabular data)	ML program, model	A unified benchmark for evaluating fairness repair algorithms	[65]
IBM AIF360 [112]	Classification (Tabular data)	Data, ML program, model	Examining and mitigating bias in ML software	[37]
I&D [218]	Classification (Tabular data)	Model	Improving initial individual discriminatory instances generation	[77]
scikit-fairness [91]	Classification (Tabular data)	Data, ML program, model	Examining and mitigating bias in ML software	[91]
LiFT [281]	Classification (Tabular data)	Data, ML program, model	Examining and mitigating bias in ML software	[57]
FairVis [122]	Classification (Tabular data)	Model	Visual analytics for discovering intersectional bias in ML software	[44]
BiasAmp [288]	Image classifier	Model	Analyzing whether ML exacerbates bias from the training data	[72]
MAAT [137]	Classification (Tabular data)	Data	Detecting selection bias and improving fairness-performance tradeoff	[79]
FairEnsembles [167]	Classification (Tabular data)	ML program	Analyzing fairness and its composition in ensemble ML	[90]
FairRepair [300]	Tree-based classification (Tabular data)	Model	Fairness testing and repair for tree-based models	[75]
SBFT [242]	Regression (Tabular data)	Model	Search-based fairness testing for regression-based ML systems	[85]
ADF [313]	DL-based classification (Tabular data)	Model	White-box fairness testing through adversarial sampling	[50]
EIDIG [310]	DL-based classification (tabular data)	Model	White-box fairness testing through gradient search	[62]
NeuronFair [318]	DL-based classification (tabular data, face images)	Model	Interpretable white-box fairness testing through biased neuron identification	[81]
DeepInspect [273]	DL-based image classification	Model	Detecting class-based bias in image classification	[53]
CMA [283]	Language models	Model	Detecting which parts of DNNs are responsible for unfairness	[52]
FairNeuron [161]	DL-based classification (tabular data)	Model	Detecting neurons and data instances responsible for bias	[74]
RULER [270]	DL-based classification (tabular data)	Model	Test input generation by discriminating sensitive and non-sensitive attributes	[84]
TestSGD [312]	DL-based classification (Tabular data, text)	Model	Interpretable testing of DNNs against subtle group discrimination	[94]
DICE [227]	DL-based classification (tabular data)	Model	Information-theoretic fairness testing and debugging of DNNs	[89]
ASTRAEA [262]	NLP systems	Model	Grammar-based discriminatory instance generation for NLP systems	[71]
MT-NLP [219]	NLP systems	Model	Metamorphic testing of fairness violation in NLP systems	[58]
BiasFinder [102]	Sentiment analysis	Model	Metamorphic test generation to uncover bias of sentiment analysis systems	[60]
BiasRV [299]	Sentiment analysis	Model	Uncovering biased sentiment predictions at runtime	[61]
NERGenderBias [223]	Name entity recognition	Model	Measuring gender bias in named entity recognition	[47]
CheckList [251]	NLP systems	Model	Behavioral testing (including fairness testing) of NLP models	[51]
DialogueFairness [214]	Conversational AI	Model	Testing gender and linguistic (racial) bias in dialogue systems	[54]
BiasAsker [286]	Conversational AI	Model	Fairness testing of conversational AI systems	[88]
REVISE [287]	CV datasets	Data	Detecting object-, gender-, and geography-based bias in CV datasets	[83]
AequeVox [250]	Speech recognition	Model	Comparing the robustness of speech recognition systems for different groups	[68]

multiple sensitive attributes [122, 270, 312], leaving an interesting research opportunity for the community. Moreover, drawing parallels with historical legal discourse, the intersectionality of protected attributes reveals the challenges of avoiding over-segmentation when assessing fairness. In 1976, Judge Harris Wangelin expressed concerns about creating new protected groups

[45], specifically considering the implications of the "curse of dimensionality." Much like the legal system's cautious approach to potential Pandora's boxes of new protected classes, fairness testing grapples with complexities in addressing compounded or intersectional effects of multiple sensitive attributes.

8.2 Test Oracle for Fairness Testing

Existing work mainly employs metamorphic relations as pseudo oracles or uses statistical measurements as indirect oracles of fairness testing, which both involve human ingenuity. It is an open challenge to design automatic techniques for constructing reliable oracles for fairness testing.

Furthermore, the emergence of manually defined oracles for fairness testing brings a challenge for test oracle selection. For instance, the IBM AIF360 toolkit alone offers more than 70 fairness measurements [37, 112], and the research community continues to introduce novel measurements. However, it is impractical to utilize all existing measurements as test oracles for fairness testing. Moreover, while each measurement may be suitable in a specific context, many of them cannot be simultaneously satisfied [121]. Hence, an important area for the research community is the development of automatic techniques for constructing reliable oracles in fairness testing. It is worth noting that one could argue that this challenge falls within the realm of requirements engineering rather than testing. However, in reality, fairness testing is frequently explored in natural settings, where upfront requirements engineering processes are not always assumed or followed strictly. Fairness testing involves investigating and evaluating fairness concerns in real-world systems or datasets, which may lack comprehensive and formal requirements. As a result, fairness testing needs to adapt to address the unique challenges that arise in these real-world contexts, where strict adherence to traditional requirements engineering may not be practical or feasible.

8.3 Test Input Generation for Fairness Testing

Generation of natural inputs. Despite the existence of various techniques for test input generation in fairness testing, there is no guarantee that the generated instances are legitimate and natural. Particularly, in Table 5, it is evident that most test input generation techniques are based on the causal fairness definition, which requires generating pairs of instances that differ solely in sensitive attributes. However, it remains uncertain whether altering only the sensitive attribute is sufficient to generate inputs that are truly natural.

Furthermore, existing techniques [310, 313, 315] primarily rely on perturbing input features without explicitly constraining the magnitude of the perturbation. As long as the generated instances can induce the intended output behavior, such as flipping the predicted outcome after modifying sensitive attribute information, they are considered effective. However, this approach may overlook real-world constraints, potentially resulting in generated instances that do not align with reality (e.g., granting a loan to a 10-year-old individual).

As a result, open problems arise regarding how to generate test inputs for fairness testing that are both legitimate and natural. Researchers need to address the challenges of ensuring the generated instances adhere to real-world constraints while still accurately assessing fairness. Additionally, automating the evaluation of the naturalness of generated test inputs is another important area of exploration, enabling more reliable and efficient fairness testing methodologies.

Exploration of more generation techniques. As mentioned earlier, most test input generation techniques in fairness testing focus on the causal fairness definition. In contrast, test input generation for counterfactual fairness is relatively unmatured and more challenging. It requires researchers to conduct causal analysis of features and consider the causal relationships among them when altering sensitive attributes during generation. Moreover, when dealing with multiple

sensitive attributes simultaneously, the task becomes even more difficult, as changes in features need to account for the causal impacts from multiple sensitive attributes.

Furthermore, there is a research opportunity to design test inputs specifically for group fairness testing. While group fairness can be evaluated using real-world collected data, obtaining such data is not always easy. Given the limited work on generating test inputs for group fairness, there is still much potential for exploration in this area.

8.4 Test Adequacy for Fairness Testing

Test adequacy is a well-explored concept in traditional software testing, focusing on evaluating the coverage provided by existing tests [304]. Adequacy criteria not only offer confidence in testing activities but also serve as a guide for test generation. However, in the domain of fairness testing, the issue of test adequacy remains an open problem, and to the best of our knowledge, no research has specifically addressed this area.

To address this challenge, one approach could be to adapt traditional software test adequacy metrics or ML test adequacy metrics for fairness testing. For instance, traditional software testing has proposed metrics such as line coverage, branch coverage, and dataflow coverage [302], while ML testing has introduced metrics such as neuron coverage, layer coverage, and surprise adequacy for deep learning models [304]. Neuron coverage assesses the extent to which neurons in a deep learning model are exercised by a test suite, while layer coverage measures the coverage of different layers. Surprise adequacy, however, evaluates the coverage of discretized input surprise range for deep learning models [304].

However, there is currently no empirical evidence to support the applicability and effectiveness of these metrics in assessing the ability to detect fairness bugs and the sufficiency of fairness testing. Further research is needed to investigate and validate the suitability of these metrics in the context of fairness testing. Additionally, novel metrics tailored specifically for fairness testing may need to be developed to capture the unique characteristics and requirements of assessing fairness in ML software.

8.5 Test Cost Reduction

Test cost poses a significant challenge in fairness testing of ML software. The process of assessing fairness often entails retraining ML models, repeating the prediction process, or generating extensive data to explore the vast behavioral space of the models. However, thus far, there has been no research dedicated to reducing the cost of fairness testing. It would be intriguing to explore specific techniques for test selection, prioritization, and minimization that can effectively reduce the cost of fairness testing without compromising test effectiveness.

Moreover, as discussed in Section 5.2, there is an escalating demand for deploying intelligent software systems on platforms with limited computing power and resources, such as mobile devices. Several studies [118, 118, 180, 264] have addressed fairness testing in such scenarios. This presents a fresh challenge for the research community: how to conduct fairness testing effectively on diverse end devices, including those with restricted computing power, limited memory size, and constrained energy capacity. Addressing this challenge requires innovative approaches and techniques that can adapt fairness testing methodologies to accommodate the limitations and constraints of these resource-constrained platforms.

8.6 Fairness and Other Testing Properties

Testing fairness repair techniques with more properties considered. After fairness repair techniques have been applied to software systems, fairness testing is often performed again. In this process, testers may also take ML performance (e.g., accuracy) into consideration [137, 183],

because it is well-known that fairness improvement is often at the cost of ML performance [183]. However, in addition to ML performance, there are also many other properties important for software systems, including robustness, security, efficiency, interpretability, and privacy [304]. The relationship between fairness and these properties is not well studied in the literature, and thus these relationships remain less well understood. Future research is needed to uncover the relationships and perform the testing with these properties considered. The determination of the properties to be considered needs the assistance of requirements engineers.

Fairness and explainability. Explainability is defined as that users can understand why a prediction is made by a software system [166]. Like fairness, it has also been an important software property required by recent regulatory provisions [226]. Because application scenarios that demand fairness often also require explainability, it would be an interesting research direction to consider fairness and explainability together. Many existing fairness testing studies just generate discriminatory instances that reveal fairness bugs in the software under test, but do not explain why these instances are unfairly treated by the software.

In this case, software engineers have relatively little guidance on the production of targeted fixes to repair the software. Improving the explainability behind the unfair software outcomes can help summarize the reasons for fairness bugs, produce insights for fairness repair, and help stakeholders without technical backgrounds (e.g., product managers, compliance officers, and policymakers) understand the software bias simply and quickly.

8.7 Fairness Testing of More Applications

The majority of existing fairness testing work has concentrated on tabular data, natural language processing systems, and computer vision systems. However, fairness, as a critical non-functional property, should be considered across a broader spectrum of software systems, including speech recognition systems, video analytic systems, multi-modal systems, and recommendation systems. Additionally, existing fairness testing studies have predominantly centered around classification tasks. However, fairness is a crucial concern that should be examined in various machine learning tasks, including regression and clustering as well as emerging cutting-edge AI technologies such as **Large Language Models (LLMs)**.

LLMs, extensively studied in academic literature and widely adopted in various applications, have recently raised concerns about fairness [209]. Given the significance of fairness in LLMs, OpenAI is actively seeking expertise to address these concerns and foster the development of fair LLMs [93]. However, fairness testing for LLMs poses unique challenges. First, LLMs are often open-domain systems that offer diverse functionalities. For instance, ChatGPT can engage in a variety of conversations with humans on a broad spectrum of topics. This characteristic poses a challenge when designing test oracles and input generation techniques for fairness testing, given the need to account for the extensive range of subjects and functionalities in LLMs. In contrast, existing fairness testing techniques are often tailored to specific tasks, potentially falling short in comprehensively evaluating the performance of LLMs across their diverse capabilities. Second, LLMs can generate a spectrum of responses, including those that may appear vague or unrelated, often influenced by pre-defined protection mechanisms regarding sensitive topics. This diversity in responses poses a challenge in automatically discerning whether the LLM output exhibits bias (i.e., the test oracle problem). While Wan et al. [286] have proposed solutions to address this challenge, their focus remains on tackling test oracle problems related to Yes-No questions, Choice-questions, and Wh-questions. Regrettably, this leaves other question types, such as prediction questions, explanatory questions, and recommendation questions, unexplored. Third, the majority of large-scale LLMs are not open-sourced, leading to opacity in their underlying mechanisms and low explainability. This characteristic poses a challenge in designing fairness testing techniques, limiting practitioners to

developing black-box approaches based solely on the observed responses of LLMs. Moreover, data testing and ML program testing are not applicable to such LLMs.

8.8 More Fairness Testing Activities and Components

The current body of research primarily focuses on offline fairness testing. There is a pressing need for more research in the realm of online fairness testing, as it can provide valuable insights to guide software maintenance and facilitate the evolution of software systems.

Furthermore, researchers have an opportunity to extend fairness testing investigations to include additional testing activities that have been extensively studied in traditional software testing but rarely explored in the context of fairness testing. For instance, exploring bug report analysis [301], bug triage [189], and test evaluation [304] in fairness testing can contribute valuable insights to the field.

Additionally, there exists a research gap in the fairness testing components. While testing of ML frameworks has received considerable attention in traditional ML testing [233, 244, 289, 292], its application in fairness testing remains underexplored. Furthermore, exploring non-ML component testing within the context of fairness testing presents a promising research direction that warrants further investigation.

8.9 More Fairness Testing Tools

Existing fairness testing tools (listed in Table 9) tend to require programming skills and thus are unfriendly to non-technical stakeholders. However, fairness testing research includes many non-programmer stakeholders and contributors such as compliance officers, policymakers, and legal practitioners.

9 DISCUSSION

9.1 Stakeholders in Fairness Testing

Fairness testing goes beyond the pure view of test engineers, involving a range of stakeholders [253]. Given that unfairness can stem from data or algorithms, data scientists and algorithm designers can play pivotal roles in detecting potential biases and providing valuable insights for fairness testing. Additionally, legal practitioners, compliance officers, and policymakers can be queried as crucial stakeholders, ensuring that fairness aligns with encoded laws, regulations, and policies.

Moreover, it is imperative to expand our focus beyond conventional algorithmic testing. ML software users, directly impacted by algorithmic decisions, offer unique insights and real-world experiences that extend beyond the scope of algorithmic scrutiny alone. This is exemplified by the book *Algorithms of Oppression* [234], which provides a good example illustrating that, despite potential testing of its algorithms, Yelp's review system demonstrated unfairness in its waiting system concerning a specific user population, adversely affecting a local hairdresser store. This emphasizes the significance of community perspectives in revealing biases not immediately evident through algorithmic testing alone. Incorporating user-centered design principles and establishing feedback loops during development and testing are crucial for a more comprehensive approach to fairness testing.

9.2 Algorithmic Fairness and Societal Fairness

Certain researchers posit that algorithms inherently reflect the biases embedded in their social context [208]. Consequently, they argue that addressing algorithmic bias holds limited value unless we first address and rectify the societal issues that influence the selection and deployment of

these algorithms. However, we believe that addressing algorithmic fairness is not misguided but complementary to tackling societal fairness. While algorithms may be inherently biased due to societal bias, addressing algorithmic fairness focuses on short-term solutions within technology to reduce discrimination. Recognizing that societal fairness is a complex, long-term endeavor, both efforts—algorithmic and societal fairness—should coexist for a fairer future.

In addition, algorithms are technologies created by people. Even if algorithms reflect societal biases, it does not absolve the creators of algorithms from their responsibilities. Algorithmic fairness research emphasizes holding them accountable for addressing bias in algorithms.

Furthermore, the algorithmic view of fairness raises awareness about societal consequences in decision-making. It contributes to a comprehensive effort for fairness, acknowledging that improving algorithms is a tangible, achievable step towards broader societal change.

We also recognize that addressing algorithmic fairness fundamentally requires an improvement in societal fairness, emphasizing crucial social factors such as power structures and social justice. The technical dimensions of fairness testing necessitate a holistic consideration of the societal contexts in which these algorithms function.

A significant aspect involves acknowledging that AI technologies often stem from the directives of individuals or entities in positions of power [260]. To rectify this power imbalance, a profound shift is imperative. It extends beyond merely consulting those affected by AI at the outset; active participation in the decision-making process is essential. This entails empowering individuals to select the problems addressed and guide the entire developmental trajectory.

Furthermore, the involvement of the most relevant participants becomes paramount to ensure that AI training data authentically mirrors the diversity of perspectives in society [260]. This principle underscores the significance of integrating lived experiences into the development of AI systems. This approach not only mitigates the risk of participation-washing but also fosters a more inclusive methodology for fairness testing.

9.3 Threats to Validity

As presented in Section 3, the methodology of this article includes two primary phases: paper collection and paper analysis. In this section, we discuss potential threats associated with these two phases.

Paper collection: Our collection process involves the selection of research papers from the widely used DBLP database, which encompasses arXiv and over 1,800 journals, as well as 5,800 academic conferences and workshops in the field of Computer Science. However, it is important to acknowledge that this approach may overlook papers published in other venues. Additionally, there is the possibility that our manually defined search strings may not encompass all the relevant studies within our research scope. To address these potential threats, we employ both backward snowballing and forward snowballing to further identify transitively dependent papers. Moreover, we proactively contact the authors of our collected papers to solicit additional papers that fall within the scope of our survey.

Paper analysis: The process of manually analyzing research papers to extract relevant information to be included in the survey is potentially open to human bias. To mitigate this risk, each paper undergoes analysis by two separate authors, and any disagreement that arises during the analysis is resolved through discussions involving other co-authors, all of whom have previously published fairness-related papers in top-tier SE venues. Furthermore, all authors independently conduct a thorough review of the survey's content to identify and rectify any potential issues. We also share our survey with the authors of the collected papers to ensure that our descriptions of their work are accurate.

10 CONCLUSION

We have presented a comprehensive survey of 100 papers on fairness testing. We summarized the current research status in the fairness testing workflow (including test input generation and test oracle identification) and testing components (including data testing, ML program testing, and model testing). We analyzed trends and promising research directions for fairness testing. We also listed public datasets and open-source tools that can be accessed by researchers and practitioners interested in this topic. We hope this survey will help researchers from various research communities become familiar with the current status and open opportunities of fairness testing.

ACKNOWLEDGMENT

We shared our work with the authors of the papers we surveyed to check for accuracy and omission, and we would like to thank those authors who kindly provided comments and feedback on earlier drafts of this article.

REFERENCES

[1] Data.world. 1977. The US Executions dataset. Retrieved from https://data.world/markmarkoh/executions-since-1977
[2] UCI Machine Learning Repository. 1994. The Diabetes dataset. Retrieved from https://archive.ics.uci.edu/ml/datasets/diabetes
[3] UCI Machine Learning Repository. 1994. The German Credit dataset. Retrieved from https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29
[4] Kaggle. 1998. The Law School dataset. Retrieved from https://www.kaggle.com/datasets/danofer/law-school-admissions-bar-passage
[5] UCI Machine Learning Repository. 2000. The Census-Income (KDD) dataset. Retrieved from https://archive.ics.uci.edu/dataset/117/census+income+kdd
[6] Kaggle. 2000. The Fraud Detection dataset. Retrieved from https://www.kaggle.com/competitions/frauddetection/data
[7] UCI Machine Learning Repository. 2001. The Heart Health dataset. Retrieved from https://archive.ics.uci.edu/ml/datasets/Heart+Disease
[8] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. LFW. Retrieved from http://vis-www.cs.umass.edu/lfw/
[9] IEEE. 2010. IEEE standard classification for software anomalies. *IEEE Std 1044-2009 (Revision of IEEE Std 1044-1993)*. 1–23.
[10] UCI Machine Learning Repository. 2011. The Communities and Crime dataset. Retrieved from http://archive.ics.uci.edu/ml/datasets/Communities%20and%20Crime%20Unnormalized
[11] IMDb.com, Inc. 2011. The IMDB dataset. Retrieved from https://www.imdb.com/interfaces/
[12] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. The Large Movie Review dataset. Retrieved from https://ai.stanford.edu/~amaas/data/sentiment/
[13] UCI Machine Learning Repository. 2014. The Bank dataset. Retrieved from https://archive.ics.uci.edu/ml/datasets/Bank+Marketing
[14] Minnesota Population Center. 2014. The Dutch Census of 2001 dataset. Retrieved from https://microdata.worldbank.org/index.php/catalog/2102
[15] UCI Machine Learning Repository. 2014. The Student Performance dataset. Retrieved from https://archive.ics.uci.edu/ml/datasets/Student+Performance
[16] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. The CelebA dataset. Retrieved from https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html
[17] ForeverData.org. 2015. The Heritage Health dataset. Retrieved from https://foreverdata.org/1015/index.html
[18] Oscar Celma. 2015. The LFM360K dataset. Retrieved from https://www.upf.edu/web/mtg/lastfm360k
[19] Agency for Healthcare Research and Quality. 2015. The Mep15 dataset. Retrieved from https://meps.ahrq.gov/mepsweb/data_stats/download_data_files_detail.jsp?cboPufNumber=HC-181
[20] GroupLens Research. 2015. The MovieLens dataset. Retrieved from https://movielens.org/
[21] O. M. Parkhi, A. Vedaldi, A. Zisserman. 2015. The VGGFace dataset. Retrieved from https://www.robots.ox.ac.uk/~vgg/data/vgg_face/#publi
[22] Rafi Letzter. 2016. Amazon just showed us that "unbiased" algorithms can be inadvertently racist. Retrieved from https://www.businessinsider.com/how-algorithms-can-be-racist-2016-4?r=US&IR=T

[23] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. The Compas dataset. Retrieved from https://github.com/propublica/compas-analysis
[24] UCI Machine Learning Repository. 2016. The Default Credit dataset. Retrieved from https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients
[25] Julia Angwin, Jeff Larson, Surya Mattu and Lauren Kirchner. 2016. Machine bias. Retrieved from https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing
[26] Agency for Healthcare Research and Quality. 2016. The Mep16 dataset. Retrieved from https://meps.ahrq.gov/mepsweb/data_stats/download_data_files_detail.jsp?cboPufNumber=HC-192
[27] UCI Machine Learning Repository. 2017. The Adult Census Income dataset. Retrieved from https://archive.ics.uci.edu/ml/datasets/adult
[28] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, Kai-Wei Chang. 2017. The COCO-gender dataset. Retrieved from https://github.com/uclanlp/reducingbias
[29] Florian Tramer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. 2017. FairTest Retrieved from https://github.com/columbia/fairtest
[30] Kaggle. 2017. The Home Credit dataset. Retrieved from https://www.kaggle.com/c/home-credit-default-risk
[31] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng, and Christopher Potts. 2017. The SST dataset. Retrieved from http://nlpprogress.com/english/sentiment_analysis.html
[32] Shan Li, Weihong Deng, and JunPing Du. 2017. The RAF-DB dataset. Retrieved from http://whdeng.cn/RAF/model1.html
[33] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Themis. Retrieved from https://github.com/LASER-UMASS/Themis
[34] Kaggle. 2017. The Titanic dataset. Retrieved from https://www.kaggle.com/c/titanic/data
[35] Conversation-AI. 2017. The Wikipedia comment dataset. Retrieved from https://github.com/conversationai/unintended-ml-bias-analysis
[36] Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. 2018. Aequitas. Retrieved from https://github.com/sakshiudeshi/Aequitas
[37] IBM Research. 2018. AIF360. Retrieved from https://aif360.readthedocs.io/en/stable/index.html
[38] Svetlana Kiritchenko and Saif M. Mohammad. 2018. The EEC dataset. Retrieved from https://competitions.codalab.org/competitions/17751
[39] Joy Buolamwini and Timnit Gebru. 2018. The PBB dataset. Retrieved from http://gendershades.org/overview.html
[40] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, Kai-Wei Chang. 2018. The WinoBias dataset. Retrieved from https://paperswithcode.com/dataset/winobias
[41] Rachel Rudinger, Jason Naradowsky, Brian Leonard, and Benjamin Van Durme. 2018. The Winogender dataset. Retrieved from https://github.com/rudinger/winogender-schemas
[42] Kaggle. 2019. The BlackFriday dataset. Retrieved from https://www.kaggle.com/datasets/sdolezel/black-friday
[43] Mei Wang, Weihong Deng, Jiani Hu, Xunqiang Tao, and Yaohai Huang. 2019. The BUPT-Transferface dataset. Retrieved from http://www.whdeng.cn/RFW/Trainingdataste.html
[44] Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. 2019. FairVis. Retrieved from https://github.com/poloclub/FairVis
[45] Jane Coaston. 2019. The intersectionality wars. Retrieved from https://www.vox.com/the-highlight/2019/5/20/18542843/intersectionality-conservatism-law-race-gender-discrimination
[46] Kaggle. 2019. The Jigsaw comment dataset. Retrieved from https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge
[47] Ninareh Mehrabi, Thamme Gowda, Fred Morstatter, Nanyun Peng, and Aram Galstyan. 2019. NERGenderBias. Retrieved from https://github.com/Ninarehm/NERGenderBias
[48] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The nlg-bias dataset. Retrieved from https://github.com/ewsheng/nlg-bias
[49] Bruno Castro da Silva. 2019. The UFRGS Entrance Exam and GPA Data. Retrieved from https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/O35FW8
[50] Peixin Zhang, Jingyi Wang, Jun Sun, Guoliang Dong, Xinyu Wang, Xingen Wang, Ting Dai, and Jinsong Dong. 2020. ADF. Retrieved from https://github.com/pxzhang94/ADF
[51] Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. CheckList. Retrieved from https://github.com/marcotcr/checklist
[52] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Simas Sakenis, Jason Huang, Yaron Singer, and Stuart Shieber. 2020. CMA. Retrieved from https://github.com/sebastianGehrmann/CausalMediationAnalysis

[53] Yuchi Tian, Ziyuan Zhong, Vicente Ordonez, Gail E. Kaiser, and Baishakhi Ray. 2020. DeepInspect. Retrieved from https://github.com/ARiSE-Lab/DeepInspect
[54] Haochen Liu, Jamell Dacon, Wenqi Fan, Hui Liu, Zitao Liu, and Jiliang Tang. 2020. DialogueFairness. Retrieved from https://github.com/zgahhblhc/DialogueFairness
[55] Arnab Sharma and Heike Wehrheim. 2020. fairCheck. Retrieved from https://github.com/arnabsharma91/fairCheck
[56] Joymallya Chakraborty, Suvodeep Majumder, Zhe Yu, and Tim Menzies. 2020. Fairway. Retrieved from https://github.com/joymallyac/Fairway
[57] Sriram Vasudevan and Krishnaram Kenthapadi. 2020. LiFT. Retrieved from https://github.com/linkedin/LiFT
[58] Pingchuan Ma, Shuai Wang, and Jin Liu. 2020. MT-NLP. Retrieved from https://github.com/pckennethma/MT-NLP
[59] Marta R. Costa-jussa. 2020. WinoST. Retrieved from https://zenodo.org/record/4139080#.YtGEc-zMJAc
[60] Muhammad Hilmi Asyrofi, Zhou Yang, Imam Nur Bani Yusuf, Hong Jin Kang, Ferdian Thung, and David Lo. 2021. BiasFinder. Retrieved from https://github.com/soarsmu/BiasFinder
[61] Zhou Yang, Muhammad Hilmi Asyrofi, and David Lo. 2021. BiasRV. Retrieved from https://github.com/soarsmu/BiasRV
[62] Lingfeng Zhang, Yueling Zhang, and Min Zhang. 2021. EIDIG. Retrieved from https://github.com/LingfengZhang98/EIDIG
[63] Ming Fan, Wenying Wei, Wuxia Jin, Zijiang Yang, and Ting Liu. 2021. ExpGA. Retrieved from https://github.com/waving7799/ExpGA
[64] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. 2021. Fair-SMOTE. Retrieved from https://github.com/joymallyac/Fair-SMOTE
[65] Max Hort, Jie M. Zhang, Federica Sarro, and Mark Harman. 2021. Fairea. Retrieved from https://github.com/maxhort/Fairea
[66] Kimmo Karkkainen and Jungseock Joo. 2021. The FairFace dataset. Retrieved from https://github.com/dchen236/FairFace
[67] Kaggle. 2021. The Sentiment140 dataset. Retrieved from https://www.kaggle.com/datasets/kazanova/sentiment140
[68] Sai Sathiesh Rajan, Sakshi Udeshi, and Sudipta Chattopadhyay. 2022. AequeVox. Retrieved from https://github.com/sparkssss/AequeVox
[69] Sai Sathiesh Rajan, Sakshi Udeshi, and Sudipta Chattopadhyay. 2022. The AequeVox dataset. Retrieved from https://zenodo.org/record/5897347
[70] UCI Machine Learning Repository. 2022. The Arrhythmia dataset. Retrieved from https://archive.ics.uci.edu/dataset/5/arrhythmia
[71] Ezekiel Soremekun, Sakshi Udeshi, and Sudipta Chattopadhyay. 2022. ASTRAEA. Retrieved from https://github.com/sakshiudeshi/Astraea
[72] Angelina Wang and Olga Russakovsky. 2022. $BiasAmp_{\rightarrow}$. Retrieved from https://github.com/princetonvisualai/directional-bias-amp
[73] DBLP computer science bibliography. 2022. DBLP. Retrieved from https://dblp.org
[74] Xuanqi Gao, Juan Zhai, Shiqing Ma, Chao Shen, Yufei Chen, and Qian Wang. 2022. FairNeuron. Retrieved from https://github.com/Antimony5292/FairNeuron
[75] Jiang Zhang, Jiang Zhang, Sergey Mechtaev, and Abhik Roychoudhury. 2022. FairRepair. Retrieved from https://github.com/fairrepair/fair-repair
[76] Eric Michael Smith, Melissa Hall, Melanie Kambadur, Eleonora Presani, and Adina Williams. 2022. HOLISTICBIAS. Retrieved from https://github.com/facebookresearch/ResponsibleNLP/tree/main/holistic_bias
[77] Minghua Ma, Zhao Tian, Max Hort, Federica Sarro, Hongyu Zhang, Qingwei Lin, and Dongmei Zhang. 2022. I&D. Retrieved from https://anonymous.4open.science/r/fairness-095F/README.md
[78] Yanhui Li, Linghan Meng, Lin Chen, Li Yu, Di Wu, Yuming Zhou, and Baowen Xu. 2022. LTDD. Retrieved from https://github.com/fairnesstest/LTDD
[79] Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. 2022. MAAT. Retrieved from https://github.com/chenzhenpeng18/FSE22-MAAT
[80] Arnab Sharma, Caglar Demir, Axel-Cyrille Ngonga Ngomo, and Heike Wehrheim. 2022. MLCheck. Retrieved from https://github.com/anonymseal/MLCheck
[81] Haibin Zheng, Zhiqing Chen, Tianyu Du, Xuhong Zhang, Yao Cheng, Shouling Ji, Jingyi Wang, Yue Yu, and Jinyin Chen. 2022. NeuronFair. Retrieved from https://github.com/haibinzheng/NeuronFair
[82] Saeid Tizpaz-Niari, Ashish Kumar, Gang Tan, and Ashutosh Trivedi. 2022. Parfait-ML. Retrieved from https://github.com/Tizpaz/Parfait-ML
[83] Angelina Wang, Alexander Liu, Ryan Zhang, Anat Kleiman, Leslie Kim, Dora Zhao, Iroha Shirai, Arvind Narayanan, and Olga Russakovsky. 2022. REVISE. Retrieved from https://github.com/princetonvisualai/revise-tool

[84] Guanhong Tao, Weisong Sun, Tingxu Han, Chunrong Fang, and Xiangyu Zhang. 2022. RULER. Retrieved from https://github.com/wssun/RULER
[85] Anjana Perera, Aldeida Aleti, Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, Burak Turhan, Lisa Kuhn, and Katie Walker. 2022. SBFT. Retrieved from https://github.com/search-based-fairness-testing/sbft
[86] Shuo Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. 2022. WIDER FACE. Retrieved from http://shuoyang1213.me/WIDERFACE/
[87] Marcel R. Ackermann. 2023. 6 million publications. Retrieved from https://blog.dblp.org/2022/02/22/6-million-publications/
[88] Yuxuan Wan, Wenxuan Wang, Pinjia He, Jiazhen Gu, Haonan Bai, and Michael Lyu. 2023. BiasAsker. Retrieved from https://github.com/yxwan123/BiasAsker
[89] Verya Monjezi, Ashutosh Trivedi, Gang Tan, and Saeid Tizpaz-Niari. 2023. DICE. Retrieved from https://github.com/armanunix/Fairness-testing
[90] Usman Gohar, Sumon Biswas, and Hridesh Rajan. 2023. FairEnsembles. Retrieved from https://github.com/UsmanGohar/FairEnsemble
[91] Hilde Weerts, Miroslav Dudík, Richard Edgar, Adrin Jalali, Roman Lutz, and Michael Madaio. 2023. Fairlearn. Retrieved from https://fairlearn.org/
[92] Kewen Peng, Joymallya Chakraborty, and Tim Menzies. 2023. FairMask. Retrieved from https://github.com/anonymous12138/biasmitigation
[93] OpenAI. 2023. How should AI systems behave, and who should decide? Retrieved from https://openai.com/blog/how-should-ai-systems-behave
[94] Mengdi Zhang, Jun Sun, Jingyi Wang, and Bing Sun. 2023. TestSGD Retrieved from https://github.com/zhangmengling/subtle_discrimination_testing
[95] Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. 2021. The BOLD dataset. Retrieved from https://github.com/jwaladhamala/BOLD-Bias-in-open-ended-language-generation
[96] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek Gordon Murray, Benoit Steiner, Paul A. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*. 265–283.
[97] Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2019. Black box fairness testing of machine learning models. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'19)*. 625–635.
[98] Khlood Ahmad, Muneera Bano, Mohamed Abdelrazek, Chetan Arora, and John C. Grundy. 2021. What's up with requirements engineering for artificial intelligence systems?. In *Proceedings of the 29th IEEE International Requirements Engineering Conference (RE'21)*. 1–12.
[99] Aws Albarghouthi, Loris D'Antoni, Samuel Drews, and Aditya V. Nori. 2017. FairSquare: Probabilistic verification of program fairness. *Proc. ACM Program. Lang.* OOPSLA (2017), 80:1–80:30.
[100] Razieh Alidoosti. 2021. Ethics-driven software architecture decision making. In *Proceedings of the IEEE 18th International Conference on Software Architecture Companion (ICSA-C'21)*. 90–91.
[101] Rico Angell, Brittany Johnson, Yuriy Brun, and Alexandra Meliou. 2018. Themis: Automatically testing software for discrimination. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'18)*. 871–875.
[102] Muhammad Hilmi Asyrofi, Zhou Yang, Imam Nur Bani Yusuf, Hong Jin Kang, Ferdian Thung, and David Lo. 2021. BiasFinder: Metamorphic test generation to uncover bias for sentiment analysis systems. *IEEE Transactions on Software Engineering* 48, 12 (2021), 5087–5101.
[103] Pranjal Awasthi, Alex Beutel, Matthäus Kleindessner, Jamie Morgenstern, and Xuezhi Wang. 2021. Evaluating fairness of machine learning models under uncertain and incomplete information. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT'21)*. 206–214.
[104] Fatma Basak Aydemir and Fabiano Dalpiaz. 2018. A roadmap for ethics-aware software engineering. In *Proceedings of the International Workshop on Software Fairness (FairWare@ICSE'18)*. 15–21.
[105] Andrew Bae and Susu Xu. 2022. Discovering and understanding algorithmic biases in autonomous pedestrian trajectory predictions. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*. 1155–1161.
[106] Guha Balakrishnan, Yuanjun Xiong, Wei Xia, and Pietro Perona. 2020. Towards causal benchmarking of bias in face analysis algorithms. In *Proceedings of the 16th European Conference on Computer Vision (ECCV'20)*. 547–563.
[107] Máté Baranyi, Marcell Nagy, and Roland Molontay. 2020. Interpretable deep learning for university dropout prediction. In *Proceedings of the 21st Annual Conference on Information Technology Education (SIGITE'20)*. 13–19.

[108] Luciano Baresi, Chiara Criscuolo, and Carlo Ghezzi. 2023. Understanding fairness requirements for ML-based software. In *Proceedings of the 31st IEEE International Requirements Engineering Conference (RE'23)*. 341–346.
[109] Solon Barocas and Andrew D. Selbst. 2016. Big data's disparate impact. *Calif. Law Rev.* 104 (2016), 671.
[110] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. 2015. The oracle problem in software testing: A survey. *IEEE Trans. Softw. Eng.* 41, 5 (2015), 507–525.
[111] Osbert Bastani, Xin Zhang, and Armando Solar-Lezama. 2019. Probabilistic verification of fairness properties via concentration. *Proc. ACM Program. Lang.* 3, OOPSLA (2019), 118:1–118:27.
[112] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John T. Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. 2018. AI fairness 360: An extensible toolkit for detecting, understanding, and mitigating unwanted algorithmic bias. *CoRR* abs/1810.01943 (2018).
[113] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. 2021. Fairness in criminal justice risk assessments: The state of the art. *Sociol. Meth. Res.* 50, 1 (2021), 3–44.
[114] Alex Beutel, Jilin Chen, Tulsee Doshi, Hai Qian, Li Wei, Yi Wu, Lukasz Heldt, Zhe Zhao, Lichan Hong, Ed H. Chi, and Cristos Goodrow. 2019. Fairness in recommendation ranking through pairwise comparisons. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD'19)*. 2212–2220.
[115] Sumon Biswas and Hridesh Rajan. 2020. Do the machine learning models on a crowd sourced platform exhibit bias? An empirical study on model fairness. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'20)*. 642–653.
[116] Sumon Biswas and Hridesh Rajan. 2021. Fair preprocessing: Towards understanding compositional fairness of data transformers in machine learning pipeline. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'21)*. 981–993.
[117] Emily Black, Samuel Yeom, and Matt Fredrikson. 2020. FlipTest: Fairness testing via optimal transport. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*'20)*. 111–121.
[118] Cody Blakeney, Nathaniel Huish, Yan Yan, and Ziliang Zong. 2021. Simon says: Evaluating and mitigating bias in pruned neural networks with knowledge distillation. *CoRR* abs/2106.07849 (2021).
[119] Teresa Bono, Karen Croxson, and Adam Giles. 2021. Algorithmic fairness in credit scoring. *Oxford Rev. Econ. Polic.* 37, 3 (2021), 585–617.
[120] Eric Breck, Neoklis Polyzotis, Sudip Roy, Steven Whang, and Martin Zinkevich. 2019. Data validation for machine learning. In *Proceedings of Machine Learning and Systems (MLSys'19)*.
[121] Yuriy Brun and Alexandra Meliou. 2018. Software fairness. In *Proceedings of the ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'18)*. 754–759.
[122] Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. 2019. FairVis: Visual analytics for discovering intersectional bias in machine learning. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST'19)*. 46–56.
[123] Alessandro Castelnovo, Riccardo Crupi, Greta Greco, Daniele Regoli, Ilaria Giuseppina Penco, and Andrea Claudio Cosentini. 2022. A clarification of the nuances in the fairness metrics landscape. *Scient. Rep.* 12, 1 (2022), 1–21.
[124] Simon Caton, Saiteja Malisetty, and Christian Haas. 2022. Impact of imputation strategies on fairness in machine learning. *J. Artif. Intell. Res.* 74 (2022), 1011–1035.
[125] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. 2021. Bias in machine learning software: Why? How? What to do? In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'21)*. 429–440.
[126] Joymallya Chakraborty, Suvodeep Majumder, and Huy Tu. 2022. Fair-SSL: Building fair ML software with less data. In *Proceedings of the International Workshop on Software Fairness (FairWare@ICSE'22)*.
[127] 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 European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'20)*. 654–665.
[128] Joymallya Chakraborty, Kewen Peng, and Tim Menzies. 2020. Making fair ML software using trustworthy explanation. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE'20)*. 1229–1233.
[129] Joymallya Chakraborty, Tianpei Xia, Fahmid M. Fahid, and Tim Menzies. 2019. Software engineering for fairness: A case study with hyperparameter optimization. *CoRR* abs/1905.05786 (2019).
[130] Supratik Chakraborty, Daniel J. Fremont, Kuldeep S. Meel, Sanjit A. Seshia, and Moshe Y. Vardi. 2015. On parallel scalable uniform SAT witness generation. In *Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'15)*. 304–319.

[131] Jason Chan and Jing Wang. 2018. Hiring preferences in online labor markets: Evidence of a female hiring bias. *Manag. Sci.* 64, 7 (2018), 2973–2994.
[132] Junjie Chen, Jibesh Patra, Michael Pradel, Yingfei Xiong, Hongyu Zhang, Dan Hao, and Lu Zhang. 2020. A survey of compiler testing. *Comput. Surv.* 53, 1 (2020), 4:1–4:36.
[133] Yunliang Chen and Jungseock Joo. 2021. Understanding and mitigating annotation bias in facial expression recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'21)*.
[134] Zhenpeng Chen, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, Tao Xie, and Xuanzhe Liu. 2020. A comprehensive study on challenges in deploying deep learning based software. In *Proceedings of the 28th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'20)*. 750–762.
[135] Zhenpeng Chen, Xuan Lu, Wei Ai, Huoran Li, Qiaozhu Mei, and Xuanzhe Liu. 2018. Through a gender lens: Learning usage patterns of emojis from large-scale Android users. In *Proceedings of the World Wide Web Conference on World Wide Web (WWW'18)*. 763–772.
[136] Zhenpeng Chen, Huihan Yao, Yiling Lou, Yanbin Cao, Yuanqiang Liu, Haoyu Wang, and Xuanzhe Liu. 2021. An empirical study on deployment faults of deep learning based mobile applications. In *Proceedings of the 43rd IEEE/ACM International Conference on Software Engineering (ICSE'21)*. 674–685.
[137] Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. 2022. MAAT: A novel ensemble approach to fixing 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 (ESEC/FSE'22)*. 1122–1134.
[138] Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. 2023. A comprehensive empirical study of bias mitigation methods for machine learning classifiers. *ACM Trans. Softw. Eng. Methodol.* 32, 4, Article 106 (2023), 30 pages.
[139] 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 46th ACM/IEEE International Conference on Software Engineering (ICSE'24)*.
[140] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017. A survey of model compression and acceleration for deep neural networks. *CoRR* abs/1710.09282 (2017).
[141] Alexandra Chouldechova. 2017. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big Data* 5, 2 (2017), 153–163.
[142] T. Anne Cleary. 1966. Test bias: Validity of the scholastic aptitude test for Negro and white students in integrated colleges. *ETS Res. Bull. Series* 1966, 2 (1966), i–23.
[143] T. Anne Cleary. 1968. Test bias: Prediction of grades of Negro and white students in integrated colleges. *J. Educ. Measur.* 5, 2 (1968), 115–124.
[144] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. Algorithmic decision making and the cost of fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'17)*. 797–806.
[145] Kimberlé Crenshaw. 2013. Demarginalizing the intersection of race and sex: A black feminist critique of antidiscrimination doctrine, feminist theory and antiracist politics. In *Feminist Legal Theories*. Routledge, 23–51.
[146] 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 (WACV'22)*. 3879–3888.
[147] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. 2008. Z3: An efficient SMT solver. In *Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'08)*. 337–340.
[148] Emily Denton, Ben Hutchinson, Margaret Mitchell, and Timnit Gebru. 2019. Detecting bias with generative counterfactual face attribute augmentation. In *Proceedings of the CVPR Workshop on Fairness Accountability Transparency and Ethics in Computer Vision*.
[149] Emily Denton, Ben Hutchinson, Margaret Mitchell, Timnit Gebru, and Andrew Zaldivar. 2019. Image counterfactual sensitivity analysis for detecting unintended bias. In *Proceedings of the CVPR Workshop on Fairness Accountability Transparency and Ethics in Computer Vision*.
[150] Jwala Dhamala, Tony Sun, Varun Kumar, Satyapriya Krishna, Yada Pruksachatkun, Kai-Wei Chang, and Rahul Gupta. 2021. BOLD: Dataset and metrics for measuring biases in open-ended language generation. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT'21)*. 862–872.
[151] Mark Díaz, Isaac Johnson, Amanda Lazar, Anne Marie Piper, and Darren Gergle. 2018. Addressing age-related bias in sentiment analysis. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI'18)*. 412.
[152] Cyrus DiCiccio, Sriram Vasudevan, Kinjal Basu, Krishnaram Kenthapadi, and Deepak Agarwal. 2020. Evaluating fairness using permutation tests. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'20)*. 1467–1477.
[153] Joe W. Duran and Simeon C. Ntafos. 1984. An evaluation of random testing. *IEEE Trans. Softw. Eng.* 10, 4 (1984), 438–444.

[154] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard S. Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS'12)*.. 214–226.
[155] Alessandro Fabris, Stefano Messina, Gianmaria Silvello, and Gian Antonio Susto. 2022. Algorithmic fairness datasets: The story so far. *CoRR* abs/2202.01711 (2022).
[156] Ming Fan, Wenying Wei, Wuxia Jin, Zijiang Yang, and Ting Liu. 2022. Explanation-guided fairness testing through genetic algorithm. In *Proceedings of the 44th International Conference on Software Engineering (ICSE'22)*.
[157] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and removing disparate impact. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'15)*. 259–268.
[158] Sira Ferradans, Nicolas Papadakis, Gabriel Peyré, and Jean-François Aujol. 2014. Regularized discrete optimal transport. *SIAM J. Imag. Sci.* 7, 3 (2014), 1853–1882.
[159] Anthony Finkelstein, Mark Harman, S. Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. 2008. "Fairness Analysis" in requirements assignments. In *Proceedings of the 16th IEEE International Requirements Engineering Conference (RE'08)*. 115–124.
[160] Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: Testing software for discrimination. In *Proceedings of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE'17)*. 498–510.
[161] Xuanqi Gao, Juan Zhai, Shiqing Ma, Chao Shen, Yufei Chen, and Qian Wang. 2022. FairNeuron: Improving deep neural network fairness with adversary games on selective neurons. In *Proceedings of the 44th International Conference on Software Engineering (ICSE'22)*.
[162] Vahid Garousi and João M. Fernandes. 2016. Highly-cited papers in software engineering: The top-100. *Inf. Softw. Technol.* 71 (2016), 108–128.
[163] Gizem Gezici, Aldo Lipani, Yücel Saygin, and Emine Yilmaz. 2021. Evaluation metrics for measuring bias in search engine results. *Inf. Retriev. J.* 24, 2 (2021), 85–113.
[164] Bishwamittra Ghosh, Debabrota Basu, and Kuldeep S. Meel. 2021. Justicia: A stochastic SAT approach to formally verify fairness. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI'21)*. 7554–7563.
[165] Bishwamittra Ghosh, Debabrota Basu, and Kuldeep S. Meel. 2022. Algorithmic fairness verification with graphical models. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI'22)*.
[166] Leilani H. Gilpin, David Bau, Ben Z. Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. 2018. Explaining explanations: An overview of interpretability of machine learning. In *Proceedings of the IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA'18)*. IEEE, 80–89.
[167] Usman Gohar, Sumon Biswas, and Hridesh Rajan. 2023. Towards understanding fairness and its composition in ensemble machine learning. In *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE'23)*.
[168] Usman Gohar and Lu Cheng. 2023. A survey on intersectional fairness in machine learning: Notions, mitigation, and challenges. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI'23)*. 6619–6627.
[169] Eric Goldman. 2020. An Introduction to the California Consumer Privacy Act (CCPA) (July 1, 2020). Santa Clara Univ. Legal Studies Research Paper, Available at SSRN: https://ssrn.com/abstract=3211013 or http://dx.doi.org/10.2139/ssrn.3211013
[170] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'14)*. 2672–2680.
[171] Nina Grgic-Hlaca, Muhammad Bilal Zafar, Krishna P. Gummadi, and Adrian Weller. 2016. The case for process fairness in learning: Feature selection for fair decision making. In *NIPS Symposium on Machine Learning and the Law*, Vol. 1. 2.
[172] Antonio Gulli and Sujit Pal. 2017. *Deep Learning with Keras*. Packt Publishing Ltd.
[173] Huizhong Guo, Jinfeng Li, Jingyi Wang, Xiangyu Liu, Dongxia Wang, Zehong Hu, Rong Zhang, and Hui Xue. 2023. FairRec: Fairness testing for deep recommender systems. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'23)*.
[174] Furkan Gursoy and Ioannis A. Kakadiaris. 2022. Error parity fairness: Testing for group fairness in regression tasks. *CoRR* abs/2208.08279 (2022).
[175] Rami Haffar, Ashneet Khandpur Singh, Josep Domingo-Ferrer, and Najeeb Jebreel. 2022. Measuring fairness in machine learning models via counterfactual examples. In *Proceedings of the 19th International Conference on Modeling Decisions for Artificial Intelligence (MDAI'22)*. 119–131.
[176] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of opportunity in supervised learning. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'16)*. 3315–3323.
[177] Mark Harman, Yue Jia, and Yuanyuan Zhang. 2015. Achievements, open problems and challenges for search based software testing. In *Proceedings of the 8th IEEE International Conference on Software Testing, Verification and Validation (ICST'15)*. 1–12.

[178] Mark Harman and Bryan F. Jones. 2001. Search-based software engineering. *Inf. Softw. Technol.* 43, 14 (2001), 833–839.
[179] Deborah Hellman. 2020. Measuring algorithmic fairness. *Virginia Law Rev.* 106, 4 (2020), 811–866.
[180] Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily Denton. 2020. Characterising bias in compressed models. *CoRR* abs/2010.03058 (2020).
[181] Max Hort, Zhenpeng Chen, Jie M. Zhang, Federica Sarro, and Mark Harman. 2023. Bias mitigation for machine learning classifiers: A comprehensive survey. *ACM J. Respons. Comput.* (2023).
[182] Max Hort and Federica Sarro. 2021. Did you do your homework? Raising awareness on software fairness and discrimination. In *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE'21)*. 1322–1326.
[183] Max Hort, Jie M. Zhang, Federica Sarro, and Mark Harman. 2021. Fairea: A model behaviour mutation approach to benchmarking bias mitigation methods. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'21)*. 994–1006.
[184] Rein-Lien Hsu, Mohamed Abdel-Mottaleb, and Anil K. Jain. 2002. Face detection in color images. *IEEE Trans. Pattern Anal. Mach. Intell.* 24, 5 (2002), 696–706.
[185] Po-Sen Huang, Huan Zhang, Ray Jiang, Robert Stanforth, Johannes Welbl, Jack Rae, Vishal Maini, Dani Yogatama, and Pushmeet Kohli. 2020. Reducing sentiment bias in language models via counterfactual evaluation. In *Proceedings of the Findings of the Association for Computational Linguistics (EMNLP'20)*. 65–83.
[186] Xin Huang, He Zhang, Xin Zhou, Muhammad Ali Babar, and Song Yang. 2018. Synthesizing qualitative research in software engineering: A critical review. In *Proceedings of the 40th International Conference on Software Engineering (ICSE'18)*. 1207–1218.
[187] Ben Hutchinson and Margaret Mitchell. 2019. 50 years of test (Un)fairness: Lessons for machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*'19)*. 49–58.
[188] Samireh Jalali and Claes Wohlin. 2012. Systematic literature studies: Database searches vs. backward snowballing. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'12)*. 29–38.
[189] Gaeul Jeong, Sunghun Kim, and Thomas Zimmermann. 2009. Improving bug triage with bug tossing graphs. In *Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'09)*. 111–120.
[190] Philips George John, Deepak Vijaykeerthy, and Diptikalyan Saha. 2020. Verifying individual fairness in machine learning models. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI'20)*. 749–758.
[191] Jungseock Joo and Kimmo Kärkkäinen. 2020. Gender Slopes: Counterfactual fairness for computer vision models by attribute manipulation. *CoRR* abs/2005.10430 (2020).
[192] Kimmo Kärkkäinen and Jungseock Joo. 2021. FairFace: Face attribute dataset for balanced race, gender, and age for bias measurement and mitigation. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV'21)*. 1547–1557.
[193] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive growing of GANs for improved quality, stability, and variation. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*.
[194] Mohd Ehmer Khan and Farmeena Khan. 2012. A comparative study of white box, black box and grey box testing techniques. *Int. J. Advanc. Comput. Sci. Applic.* 3, 6 (2012).
[195] Crenshaw Kimberly. 1989. Demarginalizing the intersection of race and sex: A black feminist critique of antidiscrimination doctrine, feminist theory and anti-racist politics. In *The University of Chicago Legal Forum*, Vol. 140. 139.
[196] Svetlana Kiritchenko and Saif Mohammad. 2018. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the 7th Joint Conference on Lexical and Computational Semantics (*SEM@NAACL-HLT'18)*. 43–53.
[197] Takashi Kitamura, Zhenjiang Zhao, and Takahisa Toda. 2022. Applying combinatorial testing to verification-based fairness testing. In *Proceedings of the 14th International Symposium on Search-Based Software Engineering (SSBSE'22)*. 101–107.
[198] Jon M. Kleinberg, Sendhil Mullainathan, and Manish Raghavan. 2017. Inherent trade-offs in the fair determination of risk scores. In *Proceedings of the 8th Innovations in Theoretical Computer Science Conference (ITCS'17)*. 43:1–43:23.
[199] Caitlin Kuhlman, Walter Gerych, and Elke A. Rundensteiner. 2021. Measuring group advantage: A comparative study of fair ranking metrics. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society (AIES'21)*. 674–682.
[200] Caitlin Kuhlman, MaryAnn VanValkenburg, and Elke Rundensteiner. 2019. Fare: Diagnostics for fair ranking using pairwise error metrics. In *Proceedings of the World Wide Web Conference (WWW'19)*. 2936–2942.
[201] D. Richard Kuhn, Raghu N. Kacker, and Yu Lei. 2013. *Introduction to Combinatorial Testing*. CRC Press.
[202] Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual fairness. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'17)*. 4066–4076.

[203] Gulsher Laghari and Serge Demeyer. 2018. Unit tests and component tests do make a difference on fault localisation effectiveness. In *Proceedings of the 40th International Conference on Software Engineering (ICSE'18)*. 280–281.
[204] Kiran Lakhotia, Mark Harman, and Phil McMinn. 2007. A multi-objective approach to search-based test data generation. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'07)*. 1098–1105.
[205] Guillaume Lample, Neil Zeghidour, Nicolas Usunier, Antoine Bordes, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Fader networks: Manipulating images by sliding attributes. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'17)*. 5967–5976.
[206] Winner Langdon. 1986. Do artifacts have politics? *Whale Reactor* (1986).
[207] Michelle Seng Ah Lee and Jatinder Singh. 2021. The landscape and gaps in open source fairness toolkits. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI'21)*. 699:1–699:13.
[208] W. David Lewis. 1977. The forces behind technology: America by design. *Science* 198, 4318 (1977), 722–723.
[209] Yingji Li, Mengnan Du, Rui Song, Xin Wang, and Ying Wang. 2023. A survey on fairness in large language models. *CoRR* abs/2308.10149 (2023).
[210] Yanhui Li, Linghan Meng, Lin Chen, Li Yu, Di Wu, Yuming Zhou, and Baowen Xu. 2022. Training data debugging for the fairness of machine learning software. In *Proceedings of the 44th International Conference on Software Engineering (ICSE'22)*. 2215–2227.
[211] Zhiheng Li and Chenliang Xu. 2021. Discover the unknown biased attribute of an image classifier. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'21)*. 14950–14959.
[212] Bin Lin, Nathan Cassee, Alexander Serebrenik, Gabriele Bavota, Nicole Novielli, and Michele Lanza. 2022. Opinion mining for software development: A systematic literature review. *ACM Trans. Softw. Eng. Methodol.* 31, 3 (2022), 38:1–38:41.
[213] Xiaofeng Lin, Seungbae Kim, and Jungseock Joo. 2022. FairGRAPE: Fairness-aware GRAdient pruning method for face attribute classification. In *Proceedings of the European Conference on Computer Vision (ECCV'22)*.
[214] Haochen Liu, Jamell Dacon, Wenqi Fan, Hui Liu, Zitao Liu, and Jiliang Tang. 2020. Does gender matter? Towards fairness in dialogue systems. In *Proceedings of the 28th International Conference on Computational Linguistics (COLING'20)*. 4403–4416.
[215] Qinghua Lu, Liming Zhu, Xiwei Xu, Jon Whittle, David Douglas, and Conrad Sanderson. 2021. Software engineering for responsible AI: An empirical study and operationalised patterns. *CoRR* abs/2111.09478 (2021).
[216] Qinghua Lu, Liming Zhu, Xiwei Xu, Jon Whittle, David Douglas, and Conrad Sanderson. 2022. Software engineering for responsible AI: An empirical study and operationalised patterns. In *Proceedings of the 44th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice (ICSE SEIP)'22)*. 241–242.
[217] Scott M. Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS'17)*. 4765–4774.
[218] Minghua Ma, Zhao Tian, Max Hort, Federica Sarro, Hongyu Zhang, Qingwei Lin, and Dongmei Zhang. 2022. Enhanced fairness testing via generating effective initial individual discriminatory instances. *CoRR* abs/2209.08321 (2022).
[219] Pingchuan Ma, Shuai Wang, and Jin Liu. 2020. Metamorphic testing and certified mitigation of fairness violations in NLP models. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI'20)*. 458–465.
[220] Ara Mambreyan, Elena Punskaya, and Hatice Gunes. 2022. Dataset bias in deception detection. In *Proceedings of the 26th International Conference on Pattern Recognition (ICPR'22)*.
[221] George Mathew, Amritanshu Agrawal, and Tim Menzies. 2018. Finding trends in software research. *IEEE Trans. Softw. Eng.* (2018).
[222] Ninareh Mehrabi, Thamme Gowda, Fred Morstatter, Nanyun Peng, and Aram Galstyan. 2020. Man is to person as woman is to location: Measuring gender bias in named entity recognition. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media (HT'20)*. 231–232.
[223] Ninareh Mehrabi, Thamme Gowda, Fred Morstatter, Nanyun Peng, and Aram Galstyan. 2020. Man is to person as woman is to location: Measuring gender bias in named entity recognition. In *Proceedings of the 31st ACM Conference on Hypertext and Social Media*. 231–232.
[224] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *Comput. Surv.* 54, 6 (2021), 115:1–115:35.
[225] Shira Mitchell, Eric Potash, Solon Barocas, Alexander D'Amour, and Kristian Lum. 2021. Algorithmic fairness: Choices, assumptions, and definitions. *Ann. Rev. Stat. Applic.* 8 (2021), 141–163.
[226] Brent D. Mittelstadt, Chris Russell, and Sandra Wachter. 2019. Explaining explanations in AI. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT'19)*. 279–288.
[227] Verya Monjezi, Ashutosh Trivedi, Gang Tan, and Saeid Tizpaz-Niari. 2023. Information-theoretic testing and debugging of fairness defects in deep neural networks. In *Proceedings of the 45th IEEE/ACM International Conference on Software Engineering (ICSE'23)*.

[228] Rebecca Moussa and Federica Sarro. 2022. Do not take it for granted: Comparing open-source libraries for software development effort estimation. Retrieved from https://arxiv.org/abs/2207.01705
[229] Vidya Muthukumar. 2019. Color-theoretic experiments to understand unequal gender classification accuracy from face images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR Workshops'19)*. 2286–2295.
[230] Nadia Nahar, Shurui Zhou, Grace A. Lewis, and Christian Kästner. 2022. Collaboration challenges in building ML-enabled systems: Communication, documentation, engineering, and process. In *Proceedings of the 44th IEEE/ACM 44th International Conference on Software Engineering (ICSE'22)*. 413–425.
[231] Vivek Nair, Zhe Yu, Tim Menzies, Norbert Siegmund, and Sven Apel. 2020. Finding faster configurations using FLASH. *IEEE Trans. Softw. Eng.* 46, 7 (2020), 794–811.
[232] Harikrishna Narasimhan, Andrew Cotter, Maya R. Gupta, and Serena Wang. 2020. Pairwise fairness for ranking and regression. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI'20)*. 5248–5255.
[233] Mahdi Nejadgholi and Jinqiu Yang. 2019. A study of oracle approximations in testing deep learning libraries. In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE'19)*. 785–796.
[234] Safiya Umoja Noble. 2018. Algorithms of oppression. In *Algorithms of Oppression*. New York University Press.
[235] Hadas Orgad and Yonatan Belinkov. 2022. Choose your lenses: Flaws in gender bias evaluation. In *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP'22)*. 151–167.
[236] Hadas Orgad, Seraphina Goldfarb-Tarrant, and Yonatan Belinkov. 2022. How gender debiasing affects internal model representations, and why it matters. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL'22)*. 2602–2628.
[237] Ankita Ramjibhai Patel, Jaganmohan Chandrasekaran, Yu Lei, Raghu N. Kacker, and D. Richard Kuhn. 2022. A combinatorial approach to fairness testing of machine learning models. In *Proceedings of the 15th IEEE International Conference on Software Testing, Verification and Validation Workshops (ICST Workshops'22)*. 94–101.
[238] Judea Pearl. 2009. *Causality*. Cambridge University Press.
[239] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.* 12 (2011), 2825–2830.
[240] Dino Pedreschi, Salvatore Ruggieri, and Franco Turini. 2008. Discrimination-aware data mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*. 560–568.
[241] Kewen Peng, Joymallya Chakraborty, and Tim Menzies. 2023. FairMask: Better fairness via model-based rebalancing of protected attributes. *IEEE Trans. Softw. Eng.* (2023).
[242] Anjana Perera, Aldeida Aleti, Chakkrit Tantithamthavorn, Jirayus Jiarpakdee, Burak Turhan, Lisa Kuhn, and Katie Walker. 2022. Search-based fairness testing for regression-based machine learning systems. *Empir. Softw. Eng.* 27, 79 (2022).
[243] Dana Pessach and Erez Shmueli. 2022. A review on fairness in machine learning. *Comput. Surv.* 55, 3 (2022).
[244] 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 *Proceedings of the 41st International Conference on Software Engineering (ICSE'19)*. 1027–1038.
[245] Evaggelia Pitoura, Kostas Stefanidis, and Georgia Koutrika. 2021. Fairness in rankings and recommendations: An overview. *VLDB J.* (2021).
[246] Soujanya Poria, Devamanyu Hazarika, Navonil Majumder, and Rada Mihalcea. 2020. Beneath the tip of the iceberg: Current challenges and new directions in sentiment analysis research. *IEEE Trans. Affect. Comput.* (2020).
[247] Muxin Pu, Meng Yi Kuan, Nyee Thoang Lim, Chun Yong Chong, and Mei Kuan Lim. 2022. Fairness evaluation in deepfake detection models using metamorphic testing. In *Proceedings of the IEEE/ACM 7th International Workshop on Metamorphic Testing (MET@ICSE'22)*. 7–14.
[248] Shangshu Qian, Viet Hung Pham, Thibaud Lutellier, Zeou Hu, Jungwon Kim, Lin Tan, Yaoliang Yu, Jiahao Chen, and Sameena Shah. 2021. Are my deep learning systems fair? An empirical study of fixed-seed training. *Adv. Neural Inf. Process. Syst.* 34 (2021), 30211–30227.
[249] Tai Le Quy, Arjun Roy, Vasileios Iosifidis, and Eirini Ntoutsi. 2021. A survey on datasets for fairness-aware machine learning. *CoRR* abs/2110.00530 (2021).
[250] Sai Sathiesh Rajan, Sakshi Udeshi, and Sudipta Chattopadhyay. 2022. AequeVox: Automated fairness testing of speech recognition systems. In *Proceedings of the 25th International Conference on Fundamental Approaches to Software Engineering (FASE'22)*. 245–267.
[251] Marco Túlio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL'20)*. 4902–4912.

[252] Shinya Sano, Takashi Kitamura, and Shingo Takada. 2022. An efficient discrimination discovery method for fairness testing. In *Proceedings of the 34th International Conference on Software Engineering and Knowledge Engineering (SEKE'22)*. 200–205.

[253] Federica Sarro. 2023. Search-based software engineering in the era of modern software systems. In *Proceedings of the 31st IEEE International Requirements Engineering Conference*. IEEE.

[254] Arnab Sharma, Caglar Demir, Axel-Cyrille Ngonga Ngomo, and Heike Wehrheim. 2021. MLCHECK–Property-driven testing of machine learning classifiers. In *Proceedings of the 20th IEEE International Conference on Machine Learning and Applications (ICMLA'21)*. 738–745.

[255] Arnab Sharma and Heike Wehrheim. 2020. Automatic fairness testing of machine learning models. In *Proceedings of the 32nd International Conference on Testing Software and Systems (ICTSS'20)*. 255–271.

[256] Shanya Sharma, Manan Dey, and Koustuv Sinha. 2020. Evaluating gender bias in natural language inference. In *Proceedings of the NeurIPS Workshop on Dataset Curation and Security*.

[257] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*. 3405–3410.

[258] Nian Si, Karthyek Murthy, Jose H. Blanchet, and Viet Anh Nguyen. 2021. Testing group fairness via optimal transport projections. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*. 9649–9659.

[259] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2219–2228.

[260] Mona Sloane. 2022. To make AI fair, here's what we must learn to do. *Nature* 605, 7908 (2022), 9–9.

[261] Eric Michael Smith, Melissa Hall, Melanie Kambadur, Eleonora Presani, and Adina Williams. 2022. "I'm sorry to hear that": Finding new biases in language models with a holistic descriptor dataset. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'22)*. 9180–9211.

[262] Ezekiel Soremekun, Sakshi Sunil Udeshi, and Sudipta Chattopadhyay. 2022. ASTRAEA: Grammar-based fairness testing. *IEEE Trans. Softw. Eng.* (2022), 1–1. DOI: https://doi.org/10.1109/TSE.2022.3141758

[263] Ezekiel O. Soremekun, Mike Papadakis, Maxime Cordy, and Yves Le Traon. 2022. Software fairness: An analysis and survey. *CoRR* abs/2205.08809 (2022).

[264] Samuil Stoychev and Hatice Gunes. 2022. The effect of model compression on fairness in facial expression recognition. In *Proceedings of Workshop on Applied Affect Recognition at the 26th International Conference on Pattern Recognition (ICPR'22)*.

[265] Bing Sun, Jun Sun, Ting Dai, and Lijun Zhang. 2021. Probabilistic verification of neural networks against group fairness. In *Proceedings of the 24th International Symposium on Formal Methods (FM'21)*. 83–102.

[266] Bing Sun, Jun Sun, Long H. Pham, and Tie Shi. 2022. Causality-based neural network repair. In *Proceedings of the 44th IEEE/ACM 44th International Conference on Software Engineering (ICSE'22)*. 338–349.

[267] Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL'19)*. 1630–1640.

[268] Zeyu Sun, Jie M. Zhang, Mark Harman, Mike Papadakis, and Lu Zhang. 2020. Automatic testing and improvement of machine translation. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE'20)*. 974–985.

[269] Zeyu Sun, Jie M. Zhang, Yingfei Xiong, Mark Harman, Mike Papadakis, and Lu Zhang. 2022. Improving machine translation systems via isotopic replacement. In *Proceedings of the 44th IEEE/ACM 44th International Conference on Software Engineering (ICSE'22)*. 1181–1192.

[270] 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 (ESEC/FSE'22)*. 1173–1184.

[271] Bahar Taskesen, Jose H. Blanchet, Daniel Kuhn, and Viet Anh Nguyen. 2021. A statistical test for probabilistic fairness. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT'21)*. 648–665.

[272] Binh Luong Thanh, Salvatore Ruggieri, and Franco Turini. 2011. k-NN as an implementation of situation testing for discrimination discovery and prevention. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'11)*. 502–510.

[273] Yuchi Tian, Ziyuan Zhong, Vicente Ordonez, Gail E. Kaiser, and Baishakhi Ray. 2020. Testing DNN image classifiers for confusion & bias errors. In *Proceedings of the 42nd International Conference on Software Engineering (ICSE'20)*. 1122–1134.

[274] Saeid Tizpaz-Niari, Ashish Kumar, Gang Tan, and Ashutosh Trivedi. 2022. Fairness-aware configuration of machine learning libraries. In *Proceedings of the 44th International Conference on Software Engineering (ICSE'22)*.

[275] Antonio Torralba and Alexei A. Efros. 2011. Unbiased look at dataset bias. In *Proceedings of the 24th IEEE Conference on Computer Vision and Pattern Recognition (CVPR'11)*. 1521–1528.

[276] Florian Tramèr, Vaggelis Atlidakis, Roxana Geambasu, Daniel J. Hsu, Jean-Pierre Hubaux, Mathias Humbert, Ari Juels, and Huang Lin. 2017. FairTest: Discovering unwarranted associations in data-driven applications. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P'17)*. 401–416.

[277] Miroslav Tushev, Fahimeh Ebrahimi, and Anas M. Mahmoud. 2022. A systematic literature review of anti-discrimination design strategies in the digital sharing economy. *IEEE Trans. Softw. Eng.* (2022).

[278] 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 (ASE'18)*. 98–108.

[279] Caterina Urban, Maria Christakis, Valentin Wüstholz, and Fuyuan Zhang. 2020. Perfectly parallel fairness certification of neural networks. *Proc. ACM Program. Lang.* 4, OOPSLA (2020), 185:1–185:30.

[280] Inês Valentim, Nuno Lourenço, and Nuno Antunes. 2019. The impact of data preparation on the fairness of software systems. In *Proceedings of the 30th IEEE International Symposium on Software Reliability Engineering (ISSRE'19)*. 391–401.

[281] Sriram Vasudevan and Krishnaram Kenthapadi. 2020. LiFT: A scalable framework for measuring fairness in ML applications. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM'20)*. 2773–2780.

[282] Sahil Verma and Julia Rubin. 2018. Fairness definitions explained. In *Proceedings of the International Workshop on Software Fairness (FairWare@ICSE'18)*. 1–7.

[283] Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart M. Shieber. 2020. Investigating gender bias in language models using causal mediation analysis. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'20)*.

[284] Cédric Villani. 2009. *Optimal Transport: Old and New*. Vol. 338. Springer.

[285] Paul Voigt and Axel Von dem Bussche. 2017. The EU general data protection regulation (GDPR). *A Practical Guide, 1st Ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.

[286] 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 Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'23)*.

[287] Angelina Wang, Arvind Narayanan, and Olga Russakovsky. 2020. REVISE: A tool for measuring and mitigating bias in visual datasets. In *Proceedings of the 16th European Conference (ECCV'20)*. 733–751.

[288] Angelina Wang and Olga Russakovsky. 2021. Directional bias amplification. In *Proceedings of the 38th International Conference on Machine Learning (ICML'21)*. 10882–10893.

[289] Jiannan Wang, Thibaud Lutellier, Shangshu Qian, Hung Viet Pham, and Lin Tan. 2022. EAGLE: Creating equivalent graphs to test deep learning libraries. In *Proceedings of the 44th IEEE/ACM 44th International Conference on Software Engineering (ICSE'22)*. 798–810.

[290] Jun Wang, Benjamin I. P. Rubinstein, and Trevor Cohn. 2022. Measuring and mitigating name biases in neural machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL'22)*. 2576–2590.

[291] Tianlu Wang, Jieyu Zhao, Mark Yatskar, Kai-Wei Chang, and Vicente Ordonez. 2019. Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV'19)*. 5309–5318.

[292] 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 European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'20)*. 788–799.

[293] Michael L. Wick, Swetasudha Panda, and Jean-Baptiste Tristan. 2019. Unlocking fairness: A trade-off revisited. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'19)*. 8780–8789.

[294] Yisong Xiao, Aishan Liu, Tianlin Li, and Xianglong Liu. 2023. Latent imitator: Generating natural individual discriminatory instances for black-box fairness testing. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'23)*.

[295] Wentao Xie and Peng Wu. 2020. Fairness testing of machine learning models using deep reinforcement learning. In *Proceedings of the 19th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom'20)*. 121–128.

[296] Guangxuan Xu and Qingyuan Hu. 2022. Can model compression improve NLP fairness. *CoRR* abs/2201.08542 (2022).

[297] Ke Yang and Julia Stoyanovich. 2017. Measuring fairness in ranked outputs. In *Proceedings of the 29th International Conference on Scientific and Statistical Database Management*. 1–6.

[298] Yu Yang, Aayush Gupta, Jianwei Feng, Prateek Singhal, Vivek Yadav, Yue Wu, Pradeep Natarajan, Varsha Hedau, and Jungseock Joo. 2022. Enhancing fairness in face detection in computer vision systems by demographic bias mitigation. In *Proceedings of AAAI/ACM Conference on AI, Ethics, and Society (AIES'22)*. 813–822.

[299] Zhou Yang, Muhammad Hilmi Asyrofi, and David Lo. 2021. BiasRV: Uncovering biased sentiment predictions at runtime. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'21)*. 1540–1544.

[300] Jiang Zhang, Ivan Beschastnikh, Sergey Mechtaev, and Abhik Roychoudhury. 2022. Fair decision making via automated repair of decision trees. In *Proceedings of the 2nd IEEE/ACM International Workshop on Equitable Data & Technology (FairWare@ICSE'22)*. 9–16.

[301] Jie Zhang, Xiaoyin Wang, Dan Hao, Bing Xie, Lu Zhang, and Hong Mei. 2015. A survey on bug-report analysis. *Sci. China Inf. Sci.* 58, 2 (2015), 1–24.

[302] Jie Zhang, Lingming Zhang, Mark Harman, Dan Hao, Yue Jia, and Lu Zhang. 2019. Predictive mutation testing. *IEEE Trans. Softw. Eng.* 45, 9 (2019), 898–918.

[303] Jie M. Zhang and Mark Harman. 2021. "Ignorance and prejudice" in software fairness. In *Proceedings of the 43rd IEEE/ACM International Conference on Software Engineering (ICSE'21)*. 1436–1447.

[304] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2022. Machine learning testing: Survey, landscapes and horizons. *IEEE Trans. Softw. Eng.* 48, 2 (2022), 1–36.

[305] Li Zhang, Jia-Hao Tian, Jing Jiang, Yi-Jun Liu, Meng-Yuan Pu, and Tao Yue. 2018. Empirical research in software engineering—A literature survey. *J. Comput. Sci. Technol.* 33, 5 (2018), 876–899.

[306] Lu Zhang, Yongkai Wu, and Xintao Wu. 2016. On discrimination discovery using causal networks. In *Proceedings of the 9th International Conference on Social, Cultural, and Behavioral Modeling (SBP-BRiMS'16)*. 83–93.

[307] Lu Zhang, Yongkai Wu, and Xintao Wu. 2016. Situation testing-based discrimination discovery: A causal inference approach. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI'16)*. 2718–2724.

[308] Lu Zhang, Yongkai Wu, and Xintao Wu. 2017. A causal framework for discovering and removing direct and indirect discrimination. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. 3929–3935.

[309] Lu Zhang, Yongkai Wu, and Xintao Wu. 2019. Causal modeling-based discrimination discovery and removal: Criteria, bounds, and algorithms. *IEEE Trans. Knowl. Data Eng.* 31, 11 (2019), 2035–2050.

[310] Lingfeng Zhang, Yueling Zhang, and Min Zhang. 2021. Efficient white-box fairness testing through gradient search. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'21)*. 103–114.

[311] 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 (ESEC/FSE'22)*. 6–17.

[312] Mengdi Zhang, Jun Sun, Jingyi Wang, and Bing Sun. 2023. TESTSGD: Interpretable testing of neural networks against subtle group discrimination. *ACM Trans. Softw. Eng. Methodol.* (2023).

[313] 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 42nd International Conference on Software Engineering (ICSE'20)*. 949–960.

[314] Peixin Zhang, Jingyi Wang, Jun Sun, and Xinyu Wang. 2021. Fairness testing of deep image classification with adequacy metrics. *CoRR* abs/2111.08856 (2021).

[315] Peixin Zhang, Jingyi Wang, Jun Sun, Xinyu Wang, Guoliang Dong, Xingen Wang, Ting Dai, and Jin Song Dong. 2021. Automatic fairness testing of neural classifiers through adversarial sampling. *IEEE Trans. Softw. Eng.* (2021). DOI: https://doi.org/10.1109/TSE.2021.3101478

[316] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. 2017. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'17)*. 2979–2989.

[317] Zhenjiang Zhao, Takahisa Toda, and Takashi Kitamura. 2022. Efficient fairness testing through hash-based sampling. In *Proceedings of the 14th International Symposium on Search-Based Software Engineering (SSBSE'22)*. 35–50.

[318] 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 (ICSE'22)*.

[319] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'17)*. 2242–2251.

Received 19 July 2023; revised 18 January 2024; accepted 1 March 2024