Exploring people's testing strategies in ML-based image classification

Téo Sanchez
Munich Center for Digital Sciences and AI (MUC.DAI)
Hochschule München University of Applied Sciences
Munich, Germany
teo.sanchez0@proton.me

Fani-Marina Kalamara
Sorbonne Université, CNRS, ISIR
Paris, France
kalamara@isir.upmc.fr

Simone Stumpf
University of Glasgow
Glasgow, United Kingdom
Simone.Stumpf@glasgow.ac.uk

Baptiste Caramiaux
Sorbonne Université, CNRS, ISIR
Paris, France
baptiste.caramiaux@sorbonne-universite.fr

Abstract

Human testers—often end-users themselves—can judge system errors in context and reveal failures that automated methods miss. Inspired by software testing practices, we conducted an exploratory study in which 15 participants tested a satellite image classifier using an interactive tool that allowed them to collect data and create test cases. While participants shared a common understanding of the model's behavior and generally adopted a failure-driven approach, we observed significant variability in their testing behaviors, including the number of test cases created, the timing of seeking feedback, the distribution of effort across classes, and the types of failures identified. Although links between specific strategies and outcomes remain unclear, our findings provide a first step toward understanding human testing of ML models and inform future research on human-driven AI auditing.

CCS Concepts

• **Human-centered computing** → **Empirical studies in HCI**; • **Computing methodologies** → **Machine learning**.

Keywords

machine learning auditing and testing, human-centered AI, empirical user study, image classification

ACM Reference Format:
Téo Sanchez, Fani-Marina Kalamara, Simone Stumpf, and Baptiste Caramiaux. 2026. Exploring people's testing strategies in ML-based image classification. In *1st International Conference on Human-Computer Interaction in the Alps (AlpCHI 2026), March 01–05, 2026, Ascona, Switzerland*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3780045.3780051

1 Introduction

AI systems are increasingly deployed across domains such as healthcare, finance, and administration. As adoption increases, so do the risks: erroneous predictions in high-stakes contexts can have serious consequences [15]. Where performance metrics reduce model behavior to numbers, human testers—often the end-users themselves—bring perspective: they understand context, notice when errors reflect bias, and evaluate models according to what matters in real-world use.

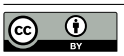
Inspired by software engineering testing [2], machine learning testing can be viewed as a deliberate search for model failures. To support AI testing, prior work has proposed both automated methods that select test examples through computational heuristics [11, 13, 20] and interactive tools that guide human testing through frameworks [23] or test case suggestions [8, 21, 22], most of which are intended for AI practitioners. However, little is known about how end-users of AI instinctively approach model testing without guidance, but guided by their understanding of the model they are interacting with. Addressing this gap, we explore testing strategies in a vision-based ML system—a satellite image classifier—to understand behaviors that contribute to effective failure discovery. We address two research questions: (RQ1) What testing strategies emerge when end-users test a satellite image classifier without prescribed guidance? (RQ2) What measurable differences in testing behaviors relate to failure discovery? To answer these questions, we conducted a study where 15 participants tested a pre-trained satellite image classifier using an interactive application without being primed with a specific strategy. The interactive tool allowed participants to create test cases (input images and expected categories) and review whether their test cases would fail or pass.

Our contributions include an open and reusable prototype for testing image classifiers (source code available in the supplementary material[1]) and a preliminary characterization of the testing strategies employed by human testers. We discuss our insights and propose actionable future research directions to better understand how behaviors translate into desirable testing outcomes, such as failure discovery. This research opens avenues for future work and may ultimately inform the design of interactive systems that support decision-subjects, domain experts, and end-users in testing ML models.

2 Background and Related Work

In this section, we first present backgrounds on software testing practices and related work that have focused on ML model testing and auditing involving end-users.

[1] https://github.com/teo-sanchez/alpchi-testing-supplementary-material



AlpCHI 2026, Ascona, Switzerland

ACM ISBN 979-8-4007-1948-6/26/03
https://doi.org/10.1145/3780045.3780051

2.1 Software testing practices

Testing in software engineering (SE) comprises techniques and workflows [19], studied for decades, and aiming to improve software quality. Myers et al. define software testing as a *destructive* process, i.e., it aims to uncover as many errors as possible within a program execution [17]. Testing techniques can be broadly categorized into static and dynamic methods [10]. Static testing involves reviewing the software artifacts (e.g., requirements, design documents, or source code) without executing the program, aiming at identifying early inconsistencies in the software development lifecycle [1]. Dynamic testing involves running the software to validate its behavior under various conditions. SE expertise is required to identify and debug defects during the software implementation. Dynamic testing is further divided into white-box and black-box testing, where the tester evaluates the software's functionality *with* resp. *without* the knowledge of its internal code [18]. The most known tools for white-box dynamic testing include debuggers and unit tests, both used to debug small testable components of a software. Beyond techniques, testing also reflects a mindset. Testers operate under the assumption that errors are present and aim to detect them early in the development process to mitigate technical debt.

Machine learning (ML) development introduces unique challenges for SE testing. Unlike other software, ML systems are not explicitly programmed but comprise non-legible parameters, inferred from training data through an optimization process. This lack of causal relationship between model parameters and behavior makes ML algorithms inherently difficult to debug and incurs technical debts [24]. Because defects can originate at any stage of model development—data collection, model design, training, or evaluation—developers often must revisit earlier stages or even retrain models from scratch to address them.

2.2 Machine learning testing

Machine learning testing procedures usually reserve a share of the testing data in order to calculate performance metrics (e.g., accuracy). Within the ML research field, new testing approaches, known as input prioritization techniques [16], aim to find heuristics to detect unseen input likely to provoke erroneous model predictions. However, evaluating such heuristics also requires labeled data, overlooks the criticality of failures [14], and neglects social biases embedded in annotated datasets. For instance, unfair treatment of certain social groups may go undetected because the training and testing data also embed the same biases [3].

To address these gaps, recent work has explored engaging laypeople directly in testing [5], leveraging their reflective judgment and prior experiences, advantages that automated methods lack. For instance, DeVos et al. [6] showed that people prioritize tests that reflect their personal exposure to algorithmic biases. Several tools have been developed to support and structure human testing efforts: `CheckList` [23] introduced a matrix framework for structured error discovery in NLP. `AdaTest` [22], `AdaTest++` [21], and `AdaVision` [8] leverage LLMs to suggest test cases in a text-fix-retest loop. `Weaver` [25] recommends testing requirements rather than cases, while `IndieLabel` [12] supports user-driven audits through personalized model training.

2.3 Research gap

While prior tools offer structured support, little is known about how end-users test without external guidance. Our study investigates spontaneous testing strategies in a vision-based ML system—a satellite image classifier—to understand behaviors that contribute to effective failure discovery. These insights may inform the design of systems promoting effective testing behaviors.

3 Methods

We conducted a user study where 15 participants tested a pre-trained satellite image classifier using their own testing intuitions. We selected this task as it resembles real-world applications of modern ML models, and most people are now familiar with satellite imagery through navigation apps. Participants' data and the source code of the interactive application used in the study are available as supplementary materials at https://github.com/teo-sanchez/alpchi-testing-supplementary-material.

3.1 Participants

We recruited 15 participants for this study through university mailing lists and in-person interactions within and beyond the university ecosystem. 8 participants identified as female and 7 as male, 4 participants were 18-24 years old, 8 were 25-34, and 3 were 35-44. Testers varied in their familiarity with artificial intelligence (AI) and machine learning (ML), reflecting a range of prior knowledge. Three participants identified as beginners with no prior experience ("neophytes"), 7 reported a basic understanding of ML, 4 identified as having intermediate experience, and 1 reported advanced knowledge of ML techniques and tools. Regarding formal training, 4 participants had taken at least one ML-related course during their studies, 5 reported having created a dataset to train an ML model, and 8 had already trained and tested an ML model on a dataset. The study involved adult volunteers participating in non-interventional, minimal-risk human-computer interaction tasks. Participation was voluntary, informed consent was obtained, no sensitive data were collected, and all data were anonymized. Formal ethics committee approval was not required under applicable local regulations. The recruitment and study were conducted in English.

3.2 Setup and Model Training

We developed an application using *Marcelle*[2], a toolkit for building web-based interfaces with integrated ML pipelines [7]. Figure 1 depicts the graphical interface used by participants to test the image classifier. For the interactive application and model training, we restricted the data to a region surrounding Munich (Germany), where the study was conducted. We used a subset of open-source and public aerial photographs of 20 cm pixel size of the Bavaria region extracted from the DOP20 dataset[3]. 8 categories are considered: Agricultural area, Forest, Industrial or commercial area, Railway, Residential area, Road, Square or park, Waterbody.

To train the classifier used in the study, a training set of 343 labeled images was collected by two annotators—one co-author and one external annotator—using a similar interface and the same satellite tiles as in the experiment. We used *MobileNetV1* as a pre-trained

[2]https://marcelle.dev/

[3]https://geodaten.bayern.de/opengeodata/OpenDataDetail.html?pn=dop20rgb

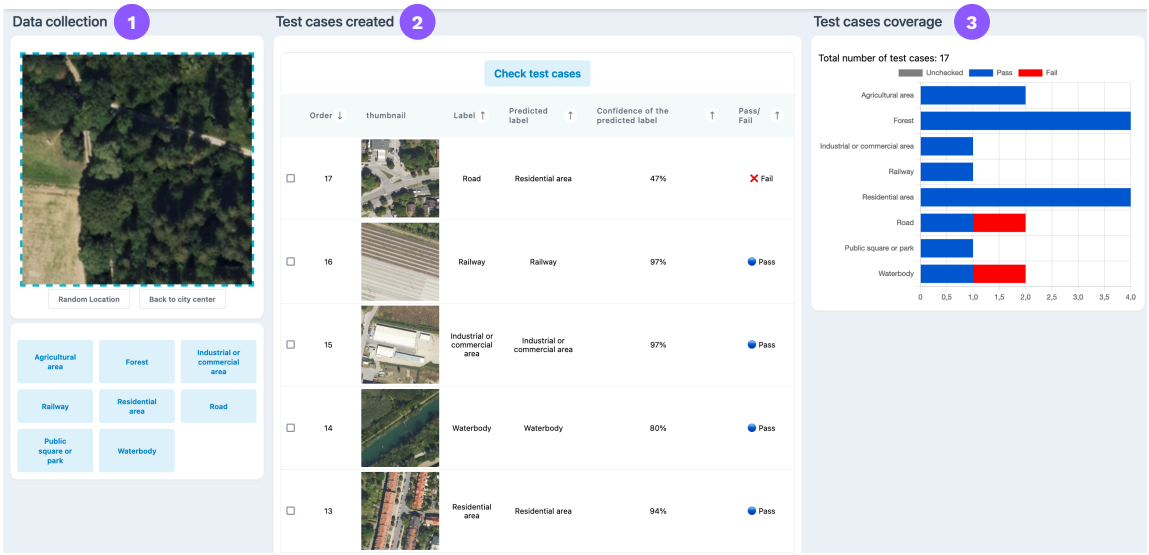


Figure 1: Study's prototype used during the testing phase, comprising a 1) data collection panel 2) a test case table, and a 3) test case coverage graph.

feature extractor, followed by a multi-layer perceptron (MLP) with two hidden layers of 32 neurons each. The model was trained directly within the Marcelle environment to avoid precision errors when converting models from Python-based ML libraries to JavaScript. Training used 8 epochs and a batch size of 8 images to limit overfitting while achieving reasonable accuracy for the study's purpose.

After the model had already been trained, six additional annotators were recruited to provide a baseline for later analyses. They collected 865 labeled data points using the same interface, exploring the same satellite tiles, and working within a fixed 30-minute session, but they were unable to query the model's predictions.

3.3 Study Procedure

Each participant completed the following three phases: introduction phase (15 minutes), testing phase (exactly 30 minutes), and post-task questionnaire (15 minutes). During the introduction, we obtained informed consent and familiarized the participants with the study design, the concepts of image classification and testing in ML, and an overview of the interface. We avoided mentioning any specific testing strategies in this introduction to prevent biasing participants. The main task for participants was to create the best test set possible for the pre-trained image classifier and to investigate the classifier's robustness for various inputs.

During the testing phase, participants had 30 minutes to create test cases using the interface shown in Figure 1. In the **Data collection** panel on the left of the screen, the participant can browse satellite images of Munich (Germany) and its surroundings and collect test cases. The participant can zoom in and out, pan across the map, jump to random locations, or return to the city center (set to Marienplatz). Note that participants could only collect data beyond a certain zoom level, indicated by a dashed blue frame around the image, to limit input variability. After selecting a location, participants can assign a label by clicking on one of the eight categories. The labeled image is then added as a test case in the central panel called **Test cases created**. The panel contains a table where all created test cases are listed in order of creation. After creating a test case, only the order, thumbnail, and label columns are shown. Once participants click the "Check Test Cases" button, the predicted label, confidence level, and pass/fail status are updated in the table for all non-checked test cases. Participants can update the statuses of unchecked test cases at any time by clicking on this button. Participants are free to delete any test cases they deem irrelevant to their testing goals. As they create or check test cases, the **Test case coverage** graph displayed on the right side of the screen is updated. This graph was designed to enable an overview of all test cases created and their status. It shows the number of collected images per category: gray bars are unchecked test cases, blue bars are correctly predicted test cases, and red bars are incorrectly predicted test cases. After the testing phase, participants fill out a questionnaire (provided in supplementary material) that investigates participants' self-reported strategies, understanding, usefulness of interface elements, and trust, as well as demographic and background information, e.g., prior expertise.

We recorded participants' interactions with the interface, including their actions with the satellite image browser (zoom, pan, clicks), the test sets collected or deleted, i.e., the image and its assigned label, the classifier's predicted label, the confidence, fail/pass status, and the per-class probabilities. We also recorded when participants initialized test case verifications, sorted the table according to which column, and the responses to the post-task questionnaire.

4 Results

In this section, we report the study's results that examine both observed and self-reported testing strategies.

4.1 Perception of the model's behavior

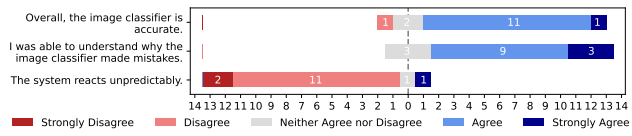


Figure 2: Participants' responses to three Likert-scale statements about their perception of the model's behavior in the post-task questionnaire. Most participants agreed that the classifier was accurate and that they could understand its mistakes, while they largely disagreed that the system reacted unpredictably.

First of all, participants' responses to the questionnaire indicate a consensus in the way they perceive the model accuracy and interpretability. Indeed, Figure 2 reveals that most participants agreed or strongly agreed that the model is accurate and that they were able to understand why the image classifier made mistakes. Furthermore, most participants disagree or strongly disagree that the system reacted unpredictably. Despite potentially diverse testing strategies, this consistency is important as it suggests that participants reached a similar understanding of the model's behavior.

4.2 Volume of test cases created

Figure 3.a shows the number of test cases created by participants (round markers) after the 30-minute task. The diamond indicates the group mean, and the horizontal line depicts the group's standard deviation. This figure reveals that the number of test cases varies considerably across participants, ranging from 14 to 130, with an average of 51 test cases, which is lower than the annotator group, although the difference is not statistically significant (Welch's t-test, $p > 0.05$). The variance in the number of tests created suggests differing assumptions about what constitutes a "good" test set. Some participants may believe that a larger test set increases the likelihood of uncovering failures, while others prioritize a smaller, more targeted set of test cases. Participants deleted a low number of test cases (4.6 on average) during the task, suggesting that they prioritized creating new tests over parsimony.

4.3 Feedback-seeking behaviors

Participants' strategies strongly varied in how frequently they sought feedback during testing. Figure 3.b shows participants' distribution according to their ratio of checks (i.e., when participants clicked on "Check test cases") over the total number of test cases created. The data suggests a bimodal pattern: some participants adopted a batched approach, creating many test cases before verifying predictions all at once (low check frequency), while others exhibited highly iterative behavior-checking the model's predictions after nearly every test case created (high check frequency). Below Figure 3.b, two timelines showcase the actions of the two participants (P8 and P9) at the extremes of this dimension. Orange ticks correspond to the moments participants browsed satellite images, green ticks are the moment they created a test case, and blue is the moment they checked the test cases collected. These results

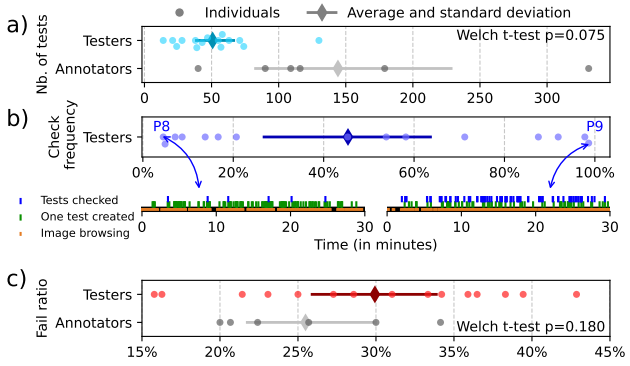


Figure 3: Behavioral overview of human testers during the 30-minute task. (a) Number of test cases created by testers (top) and baseline annotators (bottom). (b) Test-checking behavior of testers; timelines for two extreme participants (P8 and P9) show when they browsed images (orange), created tests (green), and checked them (blue). (c) Fail ratio (failed / total tests) for testers and annotators.

highlight the diversity of testing behaviors and call for further research to confirm whether this dimension is truly bimodal and the extent to which each behavior impacts the testing efficacy.

4.4 Failure discovery

Figure 3.c depicts the fail ratio (in %), calculated as the number of failed test cases divided by the total number of test cases created. The fail ratios range from 15.8% to 42.9%, with an average of 29.9%. Participants uncovered more failures than the baseline annotators (25.5%), although not significantly (Welch's t-test, $p > 0.05$). No significant correlations were observed between the three metrics shown in Figure 3 (a, b, c). The strong positive correlation was found between the number of test cases created and the number of failures uncovered ($corr = 0.73$, $p_{value} < 0.001$), suggesting that, under the current protocol, participants may be predominantly engaged in exploring the input space rather than exploiting failure patterns. Participants' questionnaire responses suggest the failure-driven behavior observed, as the majority of participants (12 out of 15) reported that they primarily selected challenging images to uncover where the classifier was likely to fail. One participant indicated that they aimed to create a test set with an equal number of correct and incorrect predictions. In contrast, two participants reported selecting prototypical images to confirm where the classifier succeeded, and no participants declared that they had selected images without a specific strategy or added a new option to the proposed items.

4.5 Distribution of testing efforts across categories

We examined how participants allocated their efforts across categories during the testing task. Table 1 shows the distribution of test cases per category (rows) for each tester (columns 1—15) and baseline annotator (A1—A6), sorted from the least to the most balanced distribution across categories. For instance, in the first column, 35% of the test cases created by participant 9 were from the category

Public square or park. The last column (Avg.) displays the average percentage of instances per category across participants. The data reveal that while some participants concentrated their efforts on specific categories (e.g., participants 9, 3, 5, or 13, focusing on *Public square or park*, *Residential area*, *Road*, or *Agricultural* area, respectively), they did not necessarily target the same categories. As a result, the average effort per category (last column) is relatively balanced despite strong individual differences. This observation is supported by a significant difference in Gini impurity between testers and annotators, indicating more uneven category coverage among testers (Welch's t-test, $p < .05$). This uneven distribution may reflect adaptive behavior, with participants reallocating effort toward categories that appeared more informative or yielded more feedback during the task.

We also examined whether focusing more on certain categories helped human testers uncover more failures in those same categories. For each participant and category, we compared the proportion of their test cases belonging to that category (their testing effort) with the proportion of failures they found in that category (their failure discovery efficiency). Across all tester—category pairs, these two measures were strongly positively correlated ($corr = 0.74$, $p < 0.01$). In other words, the more a participant concentrated on a category, the more failures they tended to uncover in that category.

	Testers																Annotators						
Category	9	3	5	13	6	1	15	4	7	11	10	12	2	14	8	Avg.	A4	A3	A5	A2	A6	A1	Avg.
Agricultural area	8	10	16	28	7	7	15	10	8	7	11	8	14	9	16	12	14	13	13	12	15	12	13
Forest	7	0	4	8	7	5	9	7	10	21	6	8	9	11	14	8	12	7	17	14	12	12	13
Industrial or commercial area	3	24	5	8	29	14	11	10	13	11	14	17	14	18	13	14	18	12	14	14	16	12	14
Public square or park	35	16	16	19	14	7	27	9	13	14	10	12	19	12	13	16	4	10	10	10	10	12	10
Railway	7	3	6	9	14	12	7	9	8	11	11	13	10	12	10	9	6	9	8	10	8	13	9
Residential area	27	24	19	4	15	28	9	23	10	11	14	17	10	12	15	16	19	15	13	15	13	13	14
Road	4	10	26	15	7	10	7	23	25	7	13	17	10	12	10	13	18	24	14	16	13	13	16
Waterbody	9	13	8	9	7	17	15	9	13	18	21	8	14	14	9	12	9	10	11	9	13	13	11

Table 1: Testing effort across categories (in%). Each cell reports the proportion of test cases created for a category relative to a participant's total tests. Columns correspond to individual testers (1—15) and baseline annotators (A1—A6), with the final column showing the group average.

4.6 Role of feedback in shaping testing strategies

The post-task questionnaire shows that a majority of participants (8 out of 15) responded that the outcome (pass/fail) in the table mainly influenced their choices of which category to prioritize as the testing progressed. 4 out of 15 participants mostly monitored the confidence value in the table, 2 out of 15 responded using their intuition or judgment, 1 out of 15 used the predicted label in the table, and no participants responded that the coverage bar chart (item 3 in figure 1) was mainly influencing their choices. Therefore, the binary outcome given by the pass/fail feedback provides an intuitive and actionable signal, guiding participants to prioritize model weaknesses. Conversely, the test coverage graph we designed for participants to monitor the overall status of their test cases was not perceived as the most useful feedback, and it could suggest that participants may pay more attention to feedback that directly informs immediate decision-making.

5 Discussion

5.1 Behavior, reasoning, and learning

This research investigated unconstrained testing strategies employed by non-experts for the testing of a satellite image classifier. Participants shared a common perception of the system's performance and seemed to adopt a failure-driven approach, evidenced by the higher proportion of errors found compared to annotators and by post-task questionnaire responses. However, their testing behaviors varied considerably across key dimensions, such as the number of test cases created, the timing at which they sought feedback, and the distribution of effort across categories. While behaviors were variable, it remains unclear if and how they reflect different reasoning processes. For instance, batch testers may have formed implicit hypotheses about the model's behavior by grouping and testing clusters of evidence, a process that parallels the sensemaking framework described by Cabrera et al. [4] with AI experts. Furthermore, the task in our study is not a purely evaluative task but also a learning task: while testing, participants also learn about the content and distribution of the input space (e.g., characteristics of satellite imagery in the surroundings of Munich) together with the model's behavior. Iterating on the experimental design could help disentangle the learning from the evaluative aspect involved in user-driven testing tasks. Lastly, while most participants were novices, a few had intermediate or advanced ML experience, which may have influenced their strategies. Future research should compare distinct groups to better understand how expertise shapes testing behaviors. The intertwined effects of learning and prior knowledge in ML may be intrinsic to user-driven testing, although they pose challenges for its evaluation.

5.2 Evaluation metric

Assessing what constitutes "successful" testing is not straightforward. In our study, we used the fail ratio as a simple quantitative indicator, but the large variation across participants suggests that this metric may not fully capture the exploratory and subjective nature of user-driven testing. Recent work highlights that effective testing can also involve qualitative dimensions rooted in users' experiences and expertise [5, 12]. Beyond counting failures, evaluation metrics could therefore consider aspects such as user engagement, testers' understanding of model behavior, or whether the failures uncovered expose critical vulnerabilities.

5.3 Future research direction and implications for design

Overall, we believe this experimental approach should be expanded to integrate the learning aspect of testing and promote outcomes that go beyond failure discovery. Such an approach could help to map the relationship between testing behaviors and desirable testing outcomes. In turn, it could have important implications for the design of assistive ML auditing tools. Existing prototypes often generate recommendations for specific test content using large language models [8, 21, 22, 25] or heuristics [9]. Such recommendations may limit testers' free will and autonomy. Instead, one could envision guidance that nudge users toward certain behaviors while preserving users' ability to make deliberate testing choices.

For instance, such guidance could help users distribute efforts to ensure good coverage or encourage hypothesis generation about the system's behavior. Investigating this approach may support the development of meaningful interactions with ML systems for testing and auditing purposes. Lastly, testing by domain experts or end-users will become more important for AI systems that work in a dynamic context and need to be updated by the end-users themselves. For example, hate speech classifiers cannot rely on static training and testing datasets as hate speech evolves continuously. Future research is warranted to apply these approaches in applications where domain experts and end-users are the only people who can test AI systems.

6 Conclusion

We explored how end-users tested a satellite image classifier using a custom interactive tool. We found that, (1) participants shared a common perception of the model's behavior but tend to adopt failure-driven strategies, (2) participants demonstrated variable testing behaviors in terms of the number of test cases created, the timing at which they sought feedback, and the distribution of effort across categories, and (3) there are no clear links between specific testing strategies and failure discovery. Future research in this direction is warranted, examining the role of prior expertise, disentangling testing from learning, and assessing testing beyond failure discovery (e.g., failure diversity, criticality, and participant engagement). By highlighting these dimensions, this work contributes to emerging participatory approaches to ML auditing that place human discernment and interaction at their core.

Acknowledgments

We thank the participants and reviewers for their valuable contributions. This research was supported by BayFrance (Bavarian-French University Cooperation Centre) through a mobility grant fostering Franco–Bavarian research collaboration and the involvement of young and early-career researchers.

References

[1] Paul Ammann and Jeff Offutt. 2016. *Introduction to software testing.* Cambridge University Press.

[2] Nahid Anwar and Susmita Kar. 2019. Review Paper on Various Software Testing Techniques & Strategies. *Global Journal of Computer Science and Technology* 19, C2 (May 2019), 43–49.

[3] Joy Buolamwini and Timnit Gebru. 2018. Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification. In *Proceedings of the 1st Conference on Fairness, Accountability and Transparency (Proceedings of Machine Learning Research, Vol. 81)*, Sorelle A. Friedler and Christo Wilson (Eds.). PMLR, 77–91.

[4] Ángel Alexander Cabrera, Marco Tulio Ribeiro, Bongshin Lee, Robert Deline, Adam Perer, and Steven M. Drucker. 2023. What Did My AI Learn? How Data Scientists Make Sense of Model Behavior. *ACM Trans. Comput.-Hum. Interact.* 30, 1, Article 1 (March 2023), 27 pages. doi:10.1145/3542921

[5] Wesley Hanwen Deng, Boyuan Guo, Alicia Devrio, Hong Shen, Motahhare Eslami, and Kenneth Holstein. 2023. Understanding Practices, Challenges, and Opportunities for User-Engaged Algorithm Auditing in Industry Practice. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 377, 18 pages. doi:10.1145/3544548.3581026

[6] Alicia DeVos, Aditi Dhabalia, Hong Shen, Kenneth Holstein, and Motahhare Eslami. 2022. Toward User-Driven Algorithm Auditing: Investigating users' strategies for uncovering harmful algorithmic behavior. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 626, 19 pages. doi:10.1145/3491102.3517441

[7] Jules Françoise, Baptiste Caramiaux, and Téo Sanchez. 2021. Marcelle: Composing Interactive Machine Learning Workflows and Interfaces. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 39–53. doi:10.1145/3472749.3474734

[8] Irena Gao, Gabriel Ilharco, Scott Lundberg, and Marco Tulio Ribeiro. 2023. Adaptive Testing of Computer Vision Models. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*. 3980–3991. doi:10.1109/ICCV51070.2023.00370

[9] Alex Groce, Todd Kulesza, Chaoqiang Zhang, Shalini Shamasunder, Margaret Burnett, Weng-Keen Wong, Simone Stumpf, Shubhomoy Das, Amber Shinsel, Forrest Bice, and Kevin McIntosh. 2014. You Are the Only Possible Oracle: Effective Test Selection for End Users of Interactive Machine Learning Systems. *IEEE Trans. Softw. Eng.* 40, 3 (March 2014), 307–323. doi:10.1109/TSE.2013.59

[10] Ajay Jangra, Gurbaj Singh, Jasbir Singh, and Rajesh Verma. 2011. Exploring testing strategies. *International Journal of Information Technology and Knowledge Management* 4 (2011).

[11] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding deep learning system testing using surprise adequacy. In *Proceedings of the 41st International Conference on Software Engineering* (Montreal, Quebec, Canada) *(ICSE '19)*. IEEE Press, 1039–1049. doi:10.1109/ICSE.2019.00108

[12] Michelle S. Lam, Mitchell L. Gordon, Danaë Metaxa, Jeffrey T. Hancock, James A. Landay, and Michael S. Bernstein. 2022. End-User Audits: A System Empowering Communities to Lead Large-Scale Investigations of Harmful Algorithmic Behavior. *Proc. ACM Hum.-Comput. Interact.* 6, CSCW2, Article 512 (Nov. 2022), 34 pages. doi:10.1145/3555625

[13] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018. DeepGauge: multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering* (Montpellier, France) *(ASE '18)*. Association for Computing Machinery, New York, NY, USA, 120–131. doi:10.1145/3238147.3238202

[14] Carl Macrae. 2022. Learning from the Failure of Autonomous and Intelligent Systems: Accidents, Safety, and Sociotechnical Sources of Risk. *Risk Analysis* 42, 9 (2022), 1999–2025. doi:10.1111/risa.13850

[15] Sean McGregor. 2021. Preventing repeated real world AI failures by cataloging incidents: The AI incident database. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. doi:10.48550/arXiv.2011.08512

[16] Vasilii Mosin, Miroslaw Staron, Darko Durisic, Francisco Gomes de Oliveira Neto, Sushant Kumar Pandey, and Ashok Chaitanya Koppisetty. 2022. Comparing Input Prioritization Techniques for Testing Deep Learning Algorithms. In *2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. 76–83. doi:10.1109/SEAA56994.2022.00020

[17] Glenford J Myers, Corey Sandler, and Tom Badgett. 2011. *The art of software testing.* John Wiley & Sons. doi:10.1002/9781119202486

[18] Srinivas Nidhra and Jagruthi Dondeti. 2012. Black box and white box testing techniques-a literature review. *International Journal of Embedded Systems and Applications (IJESA)* 2, 2 (2012). doi:10.5121/IJESA.2012.2204

[19] Alessandro Orso and Gregg Rothermel. 2014. Software testing: a research travelogue (2000–2014). In *Future of Software Engineering Proceedings* (Hyderabad, India) *(FOSE 2014)*. Association for Computing Machinery, New York, NY, USA, 117–132. doi:10.1145/2593882.2593885

[20] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2019. DeepXplore: automated whitebox testing of deep learning systems. *Commun. ACM* 62, 11 (Oct. 2019), 137–145. doi:10.1145/3361566

[21] Charvi Rastogi, Marco Tulio Ribeiro, Nicholas King, Harsha Nori, and Saleema Amershi. 2023. Supporting Human-AI Collaboration in Auditing LLMs with LLMs. In *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*. ACM, Montréal QC Canada, 913–926. doi:10.1145/3600211.3604712

[22] Marco Tulio Ribeiro and Scott Lundberg. 2022. Adaptive testing and debugging of NLP models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 3253–3267. doi:10.18653/v1/2022.acl-long.230

[23] Marco Tulio 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*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (Eds.). Association for Computational Linguistics, Online, 4902–4912. doi:10.18653/v1/2020.acl-main.442

[24] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. Hidden technical debt in Machine learning systems. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'15)*. MIT Press, Cambridge, MA, USA, 2503–2511.

[25] Chenyang Yang, Rishabh Rustogi, Rachel Brower-Sinning, Grace A. Lewis, Christian Kästner, and Tongshuang Wu. 2023. Beyond Testers' Biases: Guiding Model Testing with Knowledge Bases using LLMs.