Delft University of Technology

Faulty or Ready? Handling Failures in Deep-Learning Computer Vision Models until Deployment

A Study of Practices, Challenges, and Needs

Balayn, Agathe; Rikalo, Natasa; Yang, Jie; Bozzon, Alessandro

DOI
10.1145/3544548.3581555

Publication date
2023

Document Version
Final published version

Published in
CHI 2023 - Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems

Citation (APA)
Balayn, A., Rikalo, N., Yang, J., & Bozzon, A. (2023). Faulty or Ready? Handling Failures in Deep-Learning Computer Vision Models until Deployment: A Study of Practices, Challenges, and Needs. In *CHI 2023 - Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* Article 11 Association for Computing Machinery (ACM). https://doi.org/10.1145/3544548.3581555

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

Faulty or Ready? Handling Failures in Deep-Learning Computer Vision Models until Deployment: A Study of Practices, Challenges, and Needs

Agathe Balayn
a.m.a.balayn@tudelft.nl
Delft University of Technology
the Netherlands

Natasa Rikalo
natasa.rikalo@gmail.com
Delft University of Technology
the Netherlands

Jie Yang
j.yang-3@tudelft.nl
Delft University of Technology
the Netherlands

Alessandro Bozzon
a.bozzon@tudelft.nl
Delft University of Technology
the Netherlands

ABSTRACT

Handling failures in computer vision systems that rely on deep learning models remains a challenge. While an increasing number of methods for bug identification and correction are proposed, little is known about how practitioners actually search for failures in these models. We perform an empirical study to understand the goals and needs of practitioners, the workflows and artifacts they use, and the challenges and limitations in their process. We interview 18 practitioners by probing them with a carefully crafted failure handling scenario. We observe that there is a great diversity of failure handling workflows in which cooperations are often necessary, that practitioners overlook certain types of failures and bugs, and that they generally do not rely on potentially relevant approaches and tools originally stemming from research. These insights allow to draw a list of research opportunities, such as creating a library of best practices and more representative formalisations of practitioners' goals, developing interfaces to exploit failure handling artifacts, as well as providing specialized training.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; • **Human-centered computing** → **Empirical studies in HCI**; • **Software and its engineering** → *Software development methods.*

KEYWORDS

practices, machine learning testing, debugging, explainability

ACM Reference Format:
Agathe Balayn, Natasa Rikalo, Jie Yang, and Alessandro Bozzon. 2023. Faulty or Ready? Handling Failures in Deep-Learning Computer Vision Models until Deployment: A Study of Practices, Challenges, and Needs. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3544548.3581555

This work is licensed under a Creative Commons Attribution International 4.0 License.

CHI '23, April 23–28, 2023, Hamburg, Germany
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9421-5/23/04.
https://doi.org/10.1145/3544548.3581555

1 INTRODUCTION

Deep learning models are the basis for many computer vision applications[1][2]. Yet, safely using models is still challenging, as they suffer from issues such as spurious correlations, brittleness, and overfitting, leading to erroneous and harmful outputs [93]. Plenty of recent accidents testify of this challenge. For instance, models that distinguish between benign and malignant moles have been found to be inaccurate when used in practice for dark skin colors due to data biases [55], even though they seemed to be correctly built and perform well in the development phase.

The computer vision lifecycle is composed of many activities that might all introduce or mitigate faults in the models. While we cannot study all these activities at once, we note that growing efforts from machine learning, data management, human-computer interaction, and software engineering communities focus on proposing materials for "debugging" the failures of a model, i.e., testing the presence of potential issues, and mitigating the ones of interest, before deploying this model [36]. These materials are frameworks to test the performance of a model or to automatically mitigate inference errors [22, 52, 76, 94, 132], tools to trace issues in the outputs of the models back to problems in the code [38, 74], user-interfaces that highlight issues during model development [4, 95, 99, 100, 126, 131], and explainability methods [12, 34, 60, 96, 113]. It remains unknown how much these materials are used in practice, and to what extent they fit the hitherto unknown needs and processes of practitioners. It is even unclear whether the stated goal of these materials (typically increasing model accuracy) is aligned with the goals of practitioners. Hence, in this paper, we focus on practices for handling failures in the first crucial phase of a model: its development phase until the decision of deploying it.

One could argue that no research on failure handling practices in computer vision models has been conducted because there already exists works around *software debugging* [7, 26, 32, 40, 65, 66, 121], and computer vision applications are a type of software. Yet, identifying and mitigating failures in computer vision models is potentially more challenging than for non-data driven software systems, due to the opaque nature of the inference process, and the unlimited

[1] https://www.grandviewresearch.com/industry-analysis/computer-vision-market
[2] https://www.globaldata.com/media/thematic-research/global-computer-vision-market-will-reach-nearly-33-billion-2030-driven-larger-data-sets-advanced-deep-learning-models-says-globaldata/

set of inputs to the models [48, 67, 122, 132]. Hence, we shall study specific difficulties with computer vision failure handling.

In this work, we ask: ***(RQ1)*** *which goals (i.e., types of failures to prevent) do practitioners aim at fulfilling before deploying their models?*; and ***(RQ2)*** *how do they proceed in terms of workflows, artifacts, and tools to do so?* These questions allow to reflect on the limitations within existing practices, on the challenges faced by practitioners, and on the (mis)alignment between research and practice. We perform 18 semi-structured interviews with machine learning practitioners having different levels of experience in computer vision, but all currently working in industry or public organisations as data scientists, data engineers, or software engineers for machine learning, for at least three years. We task them to investigate a hypothetical model to decide on deploying it or on mitigating its failures. We investigate their objectives, workflows, challenges, and needs, summarized through the questions they answer in the process. We further analyse the extent to which they use existing methods and tools, and limitations in their practices, which allows us to surface opportunities for future work.

Our results reveal that the process of making a model ready for deployment is subjective and not standardized, and that it is not a lonely process but involves various stakeholders. Our results also show that machine learning "debugging" literature is not known by most practitioners despite its potential usefulness for certain steps of their process. Practitioners can identify and correct failures and bugs to a certain extent, yet pain-points and limitations, e.g., missed model bugs, are often observed. While we do not argue for standardization as the process is highly use-case dependent, our work highlights the need for more guidance and more comprehensive failure handling tools addressing various bugs (e.g., dataset content bugs) and failures (e.g., brittleness). These observations also highlight changes needed to support an education on aspects broader than machine learning algorithms, and to facilitate the communication of relevant information to the stakeholders involved in the process.

In summary, our work contributes: a) a structured understanding of computer vision model failure handling practices towards model deployment, synthesized into a framework (Figure 3) and a list of questions one might ask during the process (Table 4); b) an analysis of the relation between existing methods for failure handling such as explainability methods, and the practice of handling failures in computer vision model; and c) a critical reflection about the needs of practitioners highlighting several design opportunities.

2 RELATED WORK

In this section, we present key works on model failure handling, from which we extract the main concepts (Table 1), and working assumptions (summarized in Figure 1, and highlighted in **bold** in the text) we investigate next. We also relate our work to studies around software debugging and machine learning practices.

2.1 Failures & bugs in machine learning systems

Similarly to software engineering, in this paper, we talk about a *model failure* to designate "an external, incorrect behavior [of the model] with respect to the requirements." [5, 58], and about a *bug* or *fault* to designate the root cause of a failure. The literature on machine learning failures discusses multiple types of failures and bugs. When a script doesn't execute, the failure is due to a *program implementation* issue [22, 67, 112, 114, 132, 134]. Instead, when a script runs, according to the machine learning testing literature [132], one can observe **failures that revolve around inference outputs** (correctness, robustness, fairness) or around **processes** (security, privacy, efficiency). In this case, the failure has two possible causes reported in the literature: a *faulty configuration of the data and of the machine learning model itself,* or a *faulty translation from the intended data and model configuration to the implementation* [49, 89] (e.g., unintentionally transforming the image features that represent the inputs to the model into the wrong format).

We focus on issues of the *configuration* nature, as they are arguably challenging to handle and novel compared to software engineering, and to existing literature on model failure handling practices. Configuration issues [67, 132] relate to the design of the model architecture, i.e., the choice of architecture itself and its hyperparameters. For example, convolutional neural networks –CNNs– are often used for image classification applications; there are several CNN architectures one can choose from, each bearing different (dis)advantages depending on one's goals and constraints [57]. Other configuration issues relate to the design of the training datasets (e.g., too small dataset for the model architecture leading to overfitting, different ways of pre-processing and filtering the data might impact differently the accuracy of a model [92]); or the choice of training procedure through which the training dataset is used to train the weights of the model architecture (e.g., a number of training "tricks" and "tweaks" can significantly improve model performance [42]). Typical terminology to designate configuration-bugs include **structural bugs** ("sub-optimal model structures such as the number of hidden layers, the number of neurons"), and **training bugs** ("the mis-conducted training process, e.g., using biased training inputs") [49, 76].

We investigate whether practitioners do consider these different kinds of failures and bugs, and more broadly how they judge that their process has reached a **satisfaction point** making the model ready for deployment. This is especially important to investigate

Table 1: Main concepts identified around failure handling in computer systems.

Concept	Description
Failure	The observable manifestation of an issue (difference between expected and observed behavior). [5, 58]
Bug	The cause of the issue, and hence the place where to correct for it. "Any imperfection in a machine learning item that causes a discordance between the existing and the required conditions." [132]
Artifact	Tangible information one might use in order to search for a bug or verify its validity. The approaches from literature all rely on various artifacts. [15, 20, 22, 95, 126, 131]
Precautionary attitude	The attitude that one has when performing failure handling, geared solely towards explicit failures, or also searching for less obvious failures. [7, 40, 65, 121]
Workflow	The steps taken in order to identify and mitigate a failure. [7, 65, 121]

because the scientific literature proposes different types of failures that can often be measured via different metrics, yet does not guide practitioners in choosing the eventual metric and its value under which one would consider the model failing. For instance, in terms of correctness, a model can never be completely accurate, and one needs to define in practice under which accuracy metric, threshold, and evaluation dataset, they would consider their model failing, or ready for deployment. Besides, one might account for broader information than solely metrics evaluations.

2.2 Approaches for failure handling

As we did not find any study on configuration failure handling practices for computer vision models (only studies around program failures [134], or general machine learning with end-users [31]), we focus on failure handling methods and tools. In our study, we investigate the process followed by the practitioners, and whether they are aware of and use tools or relevant artifacts that are similar to those proposed in the literature, as literature assumes these could potentially be useful for their processes. In case they are not used, this would bring a number of future research opportunities to understand precisely the reasons for this, e.g., unawarenesss, technical or practical inadaptability, etc.

End-to-end methods. **Methods** are developed to support various model configuration failure handling activities. To identify failures, existing works propose methods to generate test inputs that are likely to break a model [132], or to monitor its outputs based on human-defined assertions [52]. To identify components of a system that might cause model failures, Lourenco et al. [74] develop a framework to systematically test different versions of the model training pipeline. Between the identification of failures and their bugs, Singla et al. [106] support the human exploration of training bugs: they help identify problematic model features by finding visual attributes in the data that lead to poor performance. To correct failures, Ma et al. [76] automatically identify neurons responsible for certain inference errors, and gather relevant training samples that should increase the model performance.

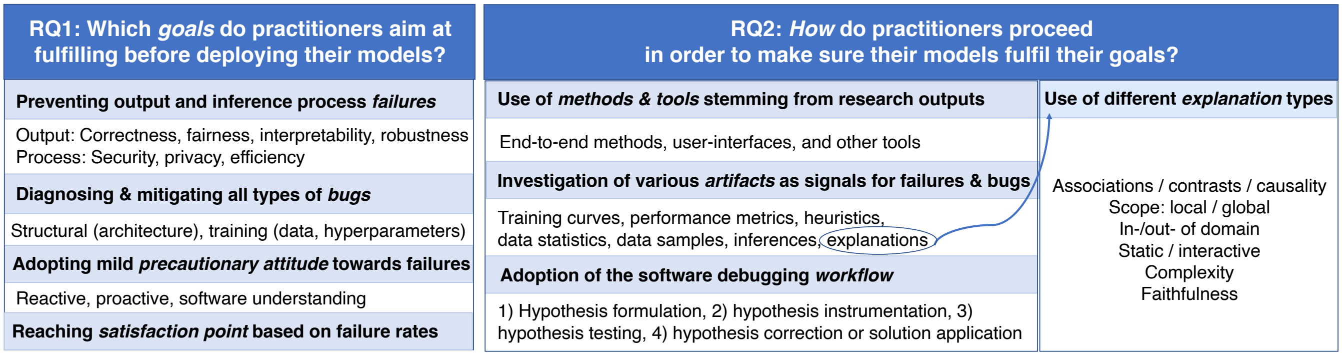
Tools & corresponding artifacts. A few **user-interfaces** [15, 20, 95, 126, 131] and other **tools** [22] have been introduced to support the handling of correctness failures (although not necessarily for computer vision applications). They rely on displaying or automatically checking diverse **artifacts** of a machine learning system, that might lead to a failure or bug. Around the model structure and training, UMLAUT [100] guides developers in proactively identifying failures through warnings about the choice of training and model hyperparameters, while Cockpit [99] visualises curves and statistics of the trained model, that can indicate bugs in training hyperparameters. On the dataset side, ModelTracker [4] visualizes interactive distributions of images to facilitate the identification of bugs in the data, and Deblinder [20] provides tentative explanations for each misclassification observed. Symphony [15] allows for further data and model analysis through interactions with various visual exploration components such as an interactive confusion matrix, and various functionalities to process the training data. The Amazon SageMaker Debugger [94] monitors a list of artifacts in different parts of the system design (e.g., poor initialization or too small updates for model weights, vanishing or exploding gradients, etc.) that help to reason about the existence of potential bugs.

Explainability. Within our study, we give particular attention to the realm of **explainability methods**, that we assume would be one of the prominent tools stemming from research and used in practice. Indeed, they represent a consequent amount of research papers both in machine learning and human-computer interaction conferences, and they are recurrently argued to be useful tools for handling model failures (explanations can then be seen as a type of artifact). Besides, some studies [16, 44, 47, 113, 127] discuss "debugging" and model "validation" as purposes of explainability, however almost no work [11, 96] has rigorously verified such a claim. Researchers have conducted user-studies around explanations for certain stakeholders and data types [2, 24, 27, 51, 54, 127], but none involves computer vision failures. Explainability methods can be categorized in various ways [8, 68, 70, 108], based on their scope (e.g., a local explanation [37, 45, 85, 105] explains a prediction for a single input data sample, and a global explanation [12, 34, 60] explains the overall behavior of a model), medium (e.g., visual or textual hints), audience (e.g., developers of a model, model users, decision-subjects, etc.), faithfulness (explanations are known not to be equally accurate [107]), etc. We study for what purpose and to what extent practitioners use explanations for failure handling, and which categories of explanations are used.

2.3 Studies of debugging practices

Software debugging. Software engineering literature around debugging practices provides an additional lens to analyse our interviews. In terms of debugging goals, it describes three levels of **precautionary attitude towards failures**: *reactive correction* of program implementation bugs when a failure is identified [7, 40], *proactive debugging* when practitioners look for the existence of bugs while no explicit failure manifests, and broader *software understanding* for later on identifying failures [65, 121]. In terms of debugging approach, this literature describes a **debugging workflow** that consists of four steps [7, 65, 121] (the usual scientific approach): 1) gathering context to generate and formulate hypotheses, 2) instrumenting and 3) testing the hypothesis, 4) correcting the initial hypothesis, or applying a solution. We investigate further whether these objectives and workflow are reflected within computer vision practices. For instance, while it is well-known that practitioners pay attention to explicit correctness failures through the use of accuracy metrics [14], it is not as clear whether practitioners might proactively investigate less visible failures, such as unknown unknowns or problematic features the model might have learned (cf. subsection 4.1).

Machine learning model building. Recent works [6, 18, 20, 30, 44, 47, 62, 69, 83, 91, 127, 130] investigate practices of developers in different steps of the machine learning or data science lifecycles. Yet, they primarily focus on machine learning model building, but not on failure handling. Besides our method inspired by these works, relevant discussion points are outlined, such as the types of stakeholders involved in the lifecycle [47, 130] and the challenges of the communication between them [20, 62, 91], or the complexity of evaluating models, e.g., for unfairness [30]. We investigate specifically (configuration-type) failure handling during model development, and specifically for computer vision applications, as this is a type of model, failure, and lifecycle stage that might present



RQ1: Which *goals* do practitioners aim at fulfilling before deploying their models?

Preventing output and inference process *failures*

Output: Correctness, fairness, interpretability, robustness
Process: Security, privacy, efficiency

Diagnosing & mitigating all types of *bugs*

Structural (architecture), training (data, hyperparameters)

Adopting mild *precautionary attitude* towards failures

Reactive, proactive, software understanding

Reaching *satisfaction point* based on failure rates

RQ2: *How* do practitioners proceed in order to make sure their models fulfil their goals?

Use of *methods & tools* stemming from research outputs

End-to-end methods, user-interfaces, and other tools

Investigation of various *artifacts* as signals for failures & bugs

Training curves, performance metrics, heuristics,
data statistics, data samples, inferences, explanations

Adoption of the software debugging *workflow*

1) Hypothesis formulation, 2) hypothesis instrumentation, 3) hypothesis testing, 4) hypothesis correction or solution application

Use of different *explanation* types

Associations / contrasts / causality
Scope: local / global
In-/out- of domain
Static / interactive
Complexity
Faithfulness

Figure 1: Summary of the research questions, and of the related insights from literature used as initial guides for the exploration of the research questions, and as working assumptions to assess. Each working assumption (bold text in the light blue boxes) involves one major concept of the debugging literature (in italic) and its different instances (plain text in the white boxes), and is formulated solely based on the assumptions the literature seems to implicitly make about practices.

particular challenges and methods, that have not been investigated yet. For instance, while research has focused on the behavior of machine learning models based on tabular data [54], that can be assumed to be relatively-easy to interpret thanks to the directly interpretable features these models are trained on, it remains unclear to what extent and how the behavior of computer vision models is understood and its validity checked, as one cannot easily make sense of the model features (raw pixels).

3 RESEARCH METHOD

We conduct our study in three steps. We study literature to understand the state-of-the-art research around computer vision failure handling (section 2). This provides us with working assumptions related to our research questions, whose validity in practice is to evaluate. We then perform semi-structured interviews to collect practices, test the assumptions, and identify broader themes that answer our research questions. Finally, we analyse the results to synthesize a failure handling framework, and to surface limitations in practices, and research opportunities.

3.1 Semi-structured interview participants

We recruited our participants through our network and searches on professional social networks, and by snowball sampling strategy. Their experiences span a wide variety of fields, from automated diagnostics based on X-Ray images, to the automated surveillance of luggage at the airport, to applications in banking and business analytics, and automatic fraud detection with natural language processing. They have at least three years of experience within industry or public organizations, e.g., hospitals, (17 different ones in total) currently working as data scientists, data engineers, or software engineers. We made sure that they all have experience with machine learning classification tasks, for them to understand the basic concepts around model failures. In total, we recruited 18 participants (13 males, 5 females), and categorized them based on their level of experience with computer vision (CV). Low-CV experience participants (4) have developed a CV model only a few times; mid-CV experience participants (7) have less than 4 years of model development experience; and high-CV experience participants (7) have more. We span such diversity of experiences not to bias our study towards highly-experienced practitioners, as the level of experience is one of the factors impacting failure handling practices. Before each interview, we asked the participant for agreement on recording the interview. We then transcribed the recordings into anonymized transcripts, and destroyed the recordings. The interview process has been approved by the ethics committee of our institution. No financial compensation was given to the participants, who were intrinsically motivated to participate to our work.

3.2 Interview guide

We performed semi-structured interviews that lasted around one hour each, and went as follows. *Step 1.* After briefly introducing our project, we enquired about the machine learning-related background of the participants. *Step 2.* Then, we presented the participants with a design brief of a failure handling scenario, and asked them to describe out loud the approach they would follow to answer the brief (**RQ2**), and the reasons for this method, as well as how they would decide the model is ready for deployment (**RQ1**). We further questioned the reasons for focusing on certain types of bugs and failures. *Step 3.* At the end, we looked back at their workflow, and questioned assumptions and gaps that had not been discussed. Especially, we questioned neglected steps of the debugging process, reasons for using failure handling tools, and explainability methods. We also showed slides with examples of model explanations (cf. Figure 4) to elicit further uses of explainability, e.g., saliency maps [105], SECA [12], TCAV [60]. We also prompted the participants for additional remarks, e.g., challenges they have to overcome, imaginary tools that could improve their process. The design brief and questions were finalised after performing two pilot studies. These studies informed us on how well the participants could relate to our brief and the way to present it in a concise manner, on the type of information about the machine learning model (e.g., data processing methods, previous experiments performed, etc.) the participants expect to know, and on questions useful to prompt the participants about their workflows.

3.2.1 *Design brief.*

Our design brief (described in Figure 2) presents a scenario where one is developing a model, and has to decide whether it can be deployed or whether failures should first be handled[3]. Our brief is inspired from prior studies [28, 100] on the development and debugging of machine learning models, where the researchers build a simple model in which they inject various kinds of bugs, that the study participants are tasked to explore. The brief is typical and simple enough for participants to reflect on their own practices without envisioning entirely new workflows. Choosing a scene classification model allows for an easy discussion without requiring domain expertise. The brief is kept vague voluntarily to investigate what practitioners naturally do when asked to decide whether a model is ready for deployment, or to "debug" it for potential failures. This brief conveniently prompts for both reactive and proactive "debugging". Next to the brief, we presented the participants with a blank template (see Figure 5) to trigger them to think about their workflow. We also showed them example dataset images (e.g., images in Figure 2), along the corresponding model predictions and ground truth. We describe in the following how these images are created.

3.2.2 Machine learning model. The dataset images are selected with the idea of simulating explicit (low accuracy) and implicit (e.g., irrelevant model features) failures and bugs. While no prior study has focused extensively on different types of configuration failures and bugs, we select the kinds of bugs to inject into the model based on the gathered scientific literature (section 2), and follow proposed procedures for dataset skewing to inject these bugs. Feature bugs are introduced by simulating a) potential data shifts between training and deployment data [61], and b) statistical biases in the data [11, 12, 28, 60, 103]. For a), we hint in the brief and images shown at a distribution shift between the training dataset (fancy-looking scenes, high-resolution images) that is not realistic for the target application, and the deployment data (pictures of simpler rooms taken from simple cameras). During the sessions, we only show training dataset images, but insist on the fact that they were collected from the Web (a Web query retrieves higher-resolution, professional images), and that the deployment data would come from daily-life pictures taken by the users of the system, in order to observe whether the participants reflect on the content of the datasets and the distribution shifts. For b), content biases are both around class-specific features (e.g., all `living room` images with a `television` and none of the other classes with one in training, and changing this in deployment), and less-specific features (e.g., `cats` present in all the pictures of certain classes). Other typical errors are also included, e.g., `living room` images wrongly predicted as `bedroom` all contain a bed-like `sofa`. This allows to investigate the awareness of the practitioners towards a diversity of issues.

3.3 Analysis of the results

We analyse the results of the interviews by coding the answers in a mix of inductive and deductive thematic analysis following the process outlined by Braun and Clark [19]. We defined initial categories of codes based on the structure of the interviews, for instance the background of the participant, on our working assumptions and additional information related to the research questions that appeared during the interviews, and on our broader readings of the literature, e.g., stakeholders. Within each category, subcategories of codes are annotated inductively by identifying the response declinations relative to each interviewee (e.g., not considering structural bugs), and grouped into broader meaningful themes (e.g., limited attention towards specific bug categories). For that, the two interviewers independently coded the 10 first interviews, and discussed to reconcile the codes (e.g., choice of more or less fine-grained codes), and refine them. They then went on to re-code all the interviews, and discussed new emerging codes. Overall, we created the codes to be all-inclusive, not excluding any part of the response declinations, and mutually exclusive, as each example could not fall into two declinations of the same category. Multiple categories of codes were applied simultaneously to show the chronology and co-occurrence of process steps, goals, artifacts, and stakeholders. A total of 197 codes are identified, clustered into 30 groups, that are themselves grouped into 14 themes. The resulting codes are analysed with a focus on co-occurrence within steps, main failure handling concepts, and in relation to specific typologies of users.

4 RESULTS

In this section, we describe the themes resulting from our interviews, that we organize into four macro-themes (each subsection) in relation to the two research questions. We start with the goals of the participants in terms of failures and more broadly how they decide the model is ready for deployment (RQ1), and then describe the workflows they followed and artifacts they used to address these failures, with a specific focus on the use of explainability methods (RQ2). We mark with an asterisk * the themes that (in)validate working assumptions from section 2.

4.1 RQ1 - goals: Disparities in identified failures and bugs

Overall, our participants focused on a few types of machine learning failures, with various, arbitrary, subjective, qualitative judgements about their importance. Besides, they did not all choose to tackle the same instances of failures within each failure category, showing the existence of relevant sub-types that we outline below.

*4.1.1 * Failures: Model correctness.* As found in scientific publications, the primary focus was on correctness of the inferences, as this is the principal evaluation of the quality of the models. For instance, P3 high-CV[4] started by searching where the model makes wrong predictions *"The confusion matrix is where I start. This can give an idea of where the network might fail."* Differences appeared for the exact failures to handle. Most participants focused on high-rate failures (P10 low-CV) *"I'm looking at this confusion matrix and think about which class is the most error-prone."* Instead, two experienced participants started with rare issues as these pinpoint hard challenges for the model, and solving these issues could solve the high-rate ones (P16 high-CV) *"I look at the rarest events, where the most information lies. It is handy because you can analyze everything going through the images."* A last participant saw both frequent and

[3] Scene classification is a common task in the computer vision literature with application to accessibility [1, 25, 39, 50, 117], although we recognize the existence of a multitude of assistive tools for visually-impaired individuals beyond vision-based techniques.

[4] We denote participants by "PX k-CV" with X the index of the participant and k the level of experience of the participant with computer vision.

Context:
A company wants to develop a system to support blind people in understanding the spaces in which they live. An intern has already developed a deep learning model for scene classification (bathroom, bedroom, dining room, kitchen, living room). For this, he created a dataset by scraping images from the Web using Google search engine, and applying some typical data augmentation methods (e.g. flipping and cropping images, brightness transformation). He then fine-tuned a ResNet model pre-trained on ImageNet on this data.

Your task:
Unfortunately, his internship has ended. The company asks you to take over his model, and investigate whether the model can be deployed, or whether it needs improvement. In this case, what issues should be improved on, and how? To start up your analysis, it is providing you already with the test accuracy, the confusion matrix of the model, and examples of test data (below).

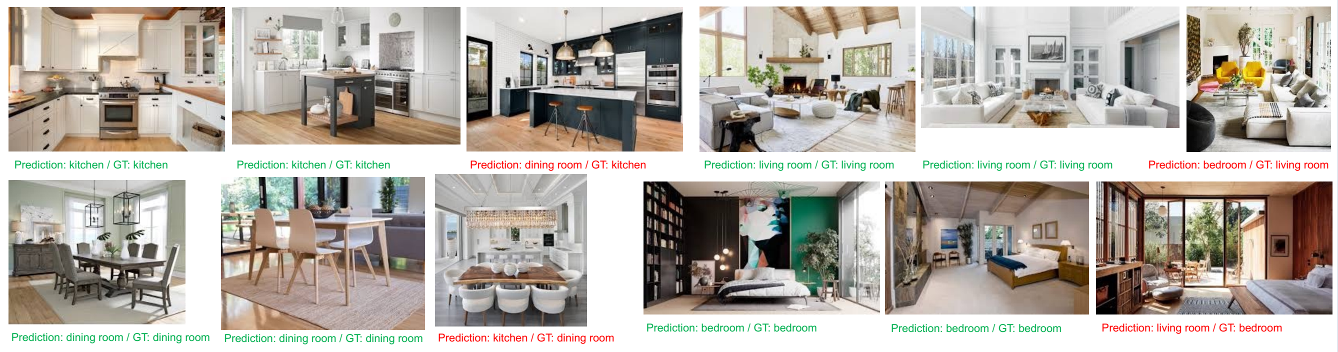

Prediction: kitchen / GT: kitchen
Prediction: kitchen / GT: kitchen
Prediction: dining room / GT: kitchen
Prediction: living room / GT: living room
Prediction: living room / GT: living room
Prediction: bedroom / GT: living room
Prediction: dining room / GT: dining room
Prediction: dining room / GT: dining room
Prediction: kitchen / GT: dining room
Prediction: bedroom / GT: bedroom
Prediction: bedroom / GT: bedroom
Prediction: living room / GT: bedroom

Figure 2: Top: our design brief, inspired by the multitude of computer vision works on scene recognition, as support for visually-impaired individuals to create mental maps of their environment [1, 25, 39, 50, 117]. Bottom: example images of four dataset classes shown to the participants, next to their ground truth (GT) and the class inferred by the model (prediction). These examples indicate feature errors in the model. For instance, among all the `kitchen` images, only the one which received an incorrect prediction contains `stools`. This hints at the potential use by the model of this concept with a higher weight than for more relevant `kitchen` features such as the `oven`.

rare issues as fundamental (P17 mid-CV) *"I focus on the extremes, the very good ones and the very bad ones. It helps me to find features of interest."*

4.1.2 * *Failures: Other failures.*

Other types of output or process failures (e.g., model robustness to natural perturbations or adversarial attacks, privacy, unfairness, unknown unknowns), although discussed in the literature, were mentioned by just a few practitioners. For instance, only two high-CV participants were concerned with the robustness of the model to natural perturbations, i.e., distribution shifts occurring unintentionally in the data [61] (e.g., the brightness of the training images is much higher than the one of deployment images, where users of the system might not be able to ensure a level of brightness for the pictures they take) (P4 high-CV) *"I will find another dataset to check the model performance again. These images are always very bright. But this might not be the case in practice. It could be like using the phone to take the images. Also, if the weather was cloudy, the images would be very dark."*, (P9 high-CV) *"The data in deployment (houses of people) may be different from the ones in your training dataset, probably from catalogs. So I would not expect the model to work well."* Some failures were also considered without explicit naming with the "technical" term, such as for unfairness discussed in the following terms by P13 high-CV *"What is called the dining room and what is called the kitchen is person and culture dependent. So, whether a prediction is wrong, that is heavily dependent on what use-case we are talking about"*[5].

Other practitioners did not envision the existence of these failures, e.g., only six participants were concerned with unknown unknowns that can be seen as a subset of correctness failures (data samples for which a model makes wrong predictions while displaying a high confidence, hence particularly challenging to identify in production) [9, 72, 137]. A last set of practitioners considered them unimportant (e.g., several practitioners mentioned not caring for distribution shifts as they would anyway try to obtain a "representative" training dataset); or irrelevant for this use-case (e.g., P8 high-CV said unfairness issues are not a concern, yet this is questionable as one could imagine that the different scenes the model should recognize would look different in different parts of the world [101]).

4.1.3 *Failures: Model features.*

While this is absent from the machine learning testing literature, some participants were also concerned with the meaningfulness of the features learned by the model. They identified feature failures by scrutinizing specific samples (see subsubsection 4.3.4) (P4 high-CV) *"The overall test accuracy is 80%. This accuracy for the initial model is fine. Next, I use a visualization method like T-SNE to see if this model truly learned something."* They talked about failures when the model did not seem to have learned any relevant feature looking at the overall shape of a few saliency maps, or when the model did not display specific, expected features for specific samples (e.g., the model classifies correctly an image as a `kitchen`, but does not seem to use the presence of a `fridge` or `oven` for that, while a human would have looked at these elements). This shows the duality of model features, seen either as goals here or as means to explain and solve correctness failures (see subsection 4.4).

[5] While most examples of algorithmic unfairness from the outputs of a machine learning model consider disparities between errors rates for different categories of populations [120], other works [29, 101, 123] have considered broader algorithmic harms, where the model would not perform equally well on a same type of object or scene that presents different representations across geographical locations or cultures. Hence, we (and a few of the practitioners) consider potential unfairness issues in our scenario.

Other participants explained not knowing or recalling that the model can reach correct inferences using questionable features. They would however handle the features after the correctness failures considered more urgent (P14 high-CV) *"That's a second step. I focus at the beginning on the errors. When I understand globally why and how, then I go through the correct answers. And I investigate if the model understood the classes."* A few participants also never handle feature failures, arguing that handling correctness failures automatically solves the relevant feature issues. They first evaluate the model with new samples representative of the deployment data, and if the error rates are higher there, the model might use wrong features. Otherwise, irrelevant features are not considered errors: while not relevant for humans, they are acceptable as the model makes correct inferences. This approach does not always hold depending on the use-case requirements, and the feasibility of collecting a representative dataset (e.g., due to contractual or privacy issues).

*4.1.4 * Precautionary attitude: Different attitudes across levels of experience.* We note a disparity between participants in their level of precaution towards failures. Participants with low-CV experience spent more time on *general understanding* as they did not know where to focus. Later, they focused on *reactive* debugging (explicit correctness failure) when choosing specific correctness errors. Proactive debugging as a workflow objective, i.e., the idea of searching proactively for non-obvious model failures, such as the use of wrong features by the model, was not a familiar concept to the participants, who did not envision the existence of such implicit failures. Proactive debugging is especially useful given that the distribution shifts cannot lead to explicit failures when one evaluates their model on an evaluation dataset taken from the same data distribution as the training dataset: one could proactively reflect on the eventual distribution shift and the type of additional training data that could be needed to solve it. Participants with mid-CV experience focused primarily on obvious manifestations of correctness failures, and experts discussed all goals. However, 75% of these mid- and high- experience participants only discussed proactive debugging when prompted. This disparity is concerning considering that our design brief was implying a strong distribution shift (the fact that the training data were collected from the Web but the deployment data would be pictures taken by visually-impaired individuals in everyday environments) calling for proactive debugging.

*4.1.5 * Bugs: Refinement of bug categorizations.* Overall, the bugs addressed by practitioners were both structural and training ones. Yet, similarly to failures, we observed differences in the bugs identified by practitioners of different expertise, differences that we discuss further when explaining the specific failure handling workflows.

Coding the interviews, especially the goals of the participants, and the explanations they were providing for identified failures, led us to propose a more fine-grained categorization of these latter bugs. We distinguish between *dataset bugs* further sub-divided into *data-statistics bugs* (e.g. distribution of data samples across classes) and *data-content bugs* (e.g. distribution of specific visual elements appearing across samples and classes), *data-engineering bugs* (e.g. how the data samples are scaled, filtered, augmented, labeled, etc.), and *training-parameter bugs* (e.g. loss function, batch size, etc.). This distinction should allow practitioners to be more structured in their reasoning about bugs, but is also useful for researchers to develop bug-specific debugging methods. For instance, to the best of our knowledge, data engineering bugs are not discussed in the machine learning literature[6] while addressing them early could avoid retraining models.

4.2 RQ1 - goals: Disagreement on the satisfaction point for deployment

While the participants were focusing on diverse types of failures along their process, we explicitly asked them to clarify how they would judge the model ready for deployment. We discuss their process here.

4.2.1 Ambiguity. The point of satisfaction at which the participants stop their process appeared ambiguous.

Trade-offs between failures. Along their process, the participants mentioned various types of failures with minimum requirements on the absence of certain failures (e.g., overfitting was unacceptable for P8 high-CV), and needed trade-offs across the different categories. For instance, P13 mid-CV did not consider meaningful features (feature failures) as important as long as the accuracy is high (correctness failures) *"The accuracy is what counts the most for lots of my projects. If something hits 99.9% accuracy, I don't look at the saliency maps anymore."* (P10 low-CV) *"We cannot even interpret how our brain works. So why we are so focused on interpreting how the model works?"* These trade-offs were also made for specific instances of failures within a category, as discussed in subsubsection 4.1.1. Yet, none of the participants expressed a precise way to judge how severe each failure is, and to establish when the trade-offs are acceptable.

** Disconnect between failures and metrics.* The participants also based their decision on the values of certain correctness-related performance metrics. A direct mapping between such metrics and the failures implicitly appeared from the low-CV participants, as they considered correcting failures as the mean to their goal (increasing performance metrics). Instead, for participants with more expertise, the relation between failures and metrics was perceived as less clear.

Expert participants were cognizant of the limitations of using metrics, and used them as a preliminary indication of the model's quality, before observing inferences on individual samples. This was the case a) when the test dataset is erroneous (e.g., wrong label) or ambiguous leading to over- or under-estimating the model *"If you're talking about hard labels, there is an error. But if I understand why the network classifies this kitchen as a dining room, I no longer consider it an error."* P3 high-CV; b) when a mistake could also be made by a human (P7 mid-CV) *"it is confusing even for humans to classify these images. So I tolerate some error."*; c) when the mistake is rare (P14 high-CV) *"it's not a fundamental but understandable mistake. I will be OK with it. This kind of bathroom, there are one out of 1,000,000."*; d) when the error has a high confidence (a few expert participants used the model confidence to judge an error's gravity (P14 high-CV) *"I check the probabilities that the model gives*

[6] Possibly because data engineering typically belongs to the data management literature, inadequately disconnected [10, 38] from the machine learning one.

to see if it's really wrong or a bit wrong. If it's 60% dining room and 39% kitchen, then I say OK."; or e) in cases when an expert would judge the error acceptable[7].

*4.2.2 * Variability in choices around metrics.* The way correctness metrics and the threshold of acceptability were selected greatly varied across participants. Some participants made an intuitive choice (P11 low-CV) *"My goal is to have as much accuracy that I can get."* Or they deferred the choice to domain experts or model requesters, judged more qualified or responsible (P14 high-CV) *"What would the business be happy with? As a system that they would put into production, there is a definition of good enough."*

Others emphasized that errors are not avoidable, and adopted a nuanced, class-based evaluation, accepting errors on certain classes to balance correctness for other classes (P7 mid-CV) *"One cannot be perfect in all cases. Let's say you are more interested in classifying images about kitchens. If you confuse the dining room with the living room, then you are okay. Then, you reach high recall classifying kitchens. You would be satisfied."* Two thirds of these participants recognized that different use-cases require emphasis on different metrics (P10 high-CV) *"It depends on the application. If I want as many kitchens as possible, then recall is more important. But for autonomous cars, recall is not as important as precision."* As for the choice of threshold, some practitioners proposed absolute numbers based on the characteristics of the task and their background knowledge (P4 high-CV) *"The accuracy should be higher than 95% because this model is for the blind people so safety is the top priority."* Others chose based on the performance of existing baseline models (P9 high-CV) *"I don't know how hard this task is, so I don't know what accuracies can be considered acceptable."*, or on human disagreement (P7 mid-CV) *"When you know whether people would agree, you know the human accuracy. Then, you would not beat yourself up if your model doesn't reach an accuracy higher than the human one"*.

4.3 RQ2 - process: Drawing the failure handling workflow

*4.3.1 * A workflow simpler than for traditional software systems.* The participants followed a trial-and-error workflow similar to the one for debugging traditional software systems. However, they often simplified the workflow, and typically did not test their hypotheses rigorously before acting, or even did not formulate specific hypotheses before experimenting on different models. As the software debugging literature does not directly apply to each step of the workflow within the machine learning context, in the following subsections, we describe further how our participants conducted each step —when they did conduct it— (we detail bug correction strategies in Appendix A.2.1).

4.3.2 Identifying a model failure. Depending on their type of precautionary attitude, participants did not adopt the same approach to start tackling a failure. Reactive debugging starts by exploring the confusion matrix and identifying areas with low or high error rates (subsection 4.1) (P3 high-CV) *"The confusion matrix is where I start from. [..] Also regarding class overlap, I would expect that classes that are closer, are also closer together in the network embedding space, and that it would lead to increased errors."* Then, the workflows described next are employed.

Proactive debugging follows the same workflows, the difference being that the failure first needs to be detected. Participants interested in feature failures scrutinized the features through saliency maps to reflect on their validity. To find failures due to distribution shifts, they compared the training dataset to imaginary deployment data ((P9 high-CV) *"The domain of the dataset where you train the model can be distant from the house the blind person enters, so I'm not sure if solving the current model issues would solve the problem of the blind person."*), or when feasible searched for more diverse images, to identify potential limitations in what the model learned. Often, the participants did not purposefully identify these implicit failures. They discovered them serendipitously during reactive debugging, when scrutinizing samples or features with incorrect predictions.

4.3.3 Gathering context and formulating hypotheses for non-data bugs. Overall, the participants tackled the gathering of context and the formulation of hypotheses around bugs differently based on their experience with computer vision.

Skewed sets of envisioned bugs. Experts participants took a sequential, bug-elimination approach. They always started with structural and data-statistics bugs, later on turning to data-engineering or training-parameter bugs, and to dataset-content bugs as a last resort. They took this approach for practical reasons. (P8 high-CV) *"Looking at the images is the last step. If the training is poor, there are things you can do before. For example, dining room and kitchen might share many pieces of furniture and because of that, it's harder to distinguish between them. This, I can assume without looking at the pictures, from prior knowledge."* They also assumed structural bugs to be limiting factors for a model (P14 high-CV) *"When I reach some performance [with experimentations on the model], the main problem is not in the architecture: the model is learning but in a bad way. Then, I check the augmentation of images, or try other datasets."*

In the rest of this subsubsection, we describe the way these high-CV participants investigated the first batch of bugs (non-data bugs). Less-expert participants took a less structured approach, and focused on the bugs they were most familiar with, essentially dataset ones (described in the next subsection) (P6 low-CV) *"hopefully if it has stronger data, it can learn something deeper. And if not, the model itself should change, but I'm not so familiar with CV and how you can improve it from the model perspective."* They sometimes wrongly assumed that mitigating dataset bugs can serve to correct all failures forgetting to account for the bias-variance trade-off, e.g., if more training data is added, the model hyperparameters might not be adapted to the dataset anymore, leading to underfitting (P5 low-CV) *"My first step would be to pick one angle: either the data (because the model performs only as good as the data it was trained on), or the system parameters (some learning rate or model hyperparameters)."*

Truncated and oriented context and hypotheses. To deal with structural and training-parameter bugs, expert participants tried multiple models with different architectures, training hyperparameters, and data processing (P3 high-CV) *"Going a step back, I would employ augmentation techniques to see if I can get higher performance, and I would use a method to further regularize the model to make sure that it's not falling into the overfitting regime."*, (P3 high-CV) *"I suppose*

[7] a) to d) can be questionable when the model has high-stakes.

that the input has been sufficiently preprocessed? I would normalize, typically by the max value if we are talking about standard RGB images. I would also standardize the data, so force inputs to have zero mean and unit variance." until they reached the "best" model among these tests (P14 high-CV) *"There is something that I do dumbly at the beginning: I try different architectures to see if there is a problem of this kind. I'm not sure that the architecture is the main issue. But it can help to add more dropout, or change the architecture, especially when I have a problem of overfitting."* Practitioners have learned through experience typical "good" hyperparameters that they test in priority (P3 high-CV) *"One thing that could lead to increase performance is to force those classes to be more separated by employing another form of loss, like the contrastive loss."* This process truncates the software debugging workflow as it directly consists in testing various potential "solutions" to improve the model performance, solely with an implicit hypothesis (non-data bugs: the model hyperparameters have not been explored) and no gathering of context for hypothesis formulation.

** Supporting artifacts.* During this process, participants mentioned monitoring a subset of the artifacts discussed in the literature such as learning curves, and overall shapes of saliency maps that might indicate model overfitting, to orient further the search of the "best" model (P3 high-CV) *"I will see some training curves. The optimal case would be that the further the training process is, the lower the training and validation losses go. This means that the model is learning something without sign of overfitting."*, (P9 high-CV) *"I would see how the training curves look like with the Tensorboard, to see if the model is overfitting on the training set. If that's the case, you can add some regularization or augment the training set."* We did not delve deeper into these bugs during the interviews, as only expert participants discussed them, and existing research primarily provides support with similar artifacts for these bugs.

4.3.4 Gathering context and formulating hypotheses for data bugs.

Artifacts as context. Data bugs were typically connected to correctness, robustness, or feature failures. They were specified by investigating test set images and/or saliency maps for recurring visual elements the model might have learned as features, rare visual elements that might confuse the model, or signs of problematic data processing (image size, resolution, unrealistic data augmentation). The link to the activities that led to such bugs was then made, and bug correction strategies were devised. For that, participants used different sets of images. a) The images corresponding to a confusion matrix cell (P14 high-CV) *"There are a lot of false positives of dining room and kitchen. Let's see in the images what kind of situations cause these mistakes. I would plot heatmaps. Probably it would put the salient part here, and that's the problem."* b) The images that received correct inferences for the classes at stake, searching for common concepts with the wrongly predicted images (P7 mid-CV) *"I focus on cases where the model made a mistake and the ones where the model is correct. I figure out the pattern that was correctly detected."* c) One participant looked at a random sampling of images of a class to understand how diverse the dataset is, and compared it to mis-classified images of the class (P17 mid-CV) *"My goal is to understand how diverse are the images of kitchen visually and how well they capture the essence of a kitchen. There might be some similarity metrics to use."*

Diversity of hypotheses. Participants formulated five types of hypothesis (cf. Table 2) around model features and data content, using the above artifacts and their background knowledge (P2 high-CV) *"I would compare a true positive and a false positive from these classes, apply some domain knowledge, and see if there are elements which should be used for a specific class."* The first one was however not formulated by participants with low-CV experience as they did not think features can be wrong, or did not know how to identify features. For all these hypotheses, the notion of granularity is important, i.e., different levels of description of the visual elements a model has learned. For instance, the participants often mentioned the style of an object the model is expected to use for classifying an image (P14 high-CV) *"I make an assumption by trying to understand why it makes these mistakes. This bed is not classic, so maybe the dataset needs more not-classic beds."*, parts of an object, and remarkable textures and colors of these objects.

4.3.5 Instrumenting the hypothesis.

Most participants did not instrument and test their hypotheses. Instead, other proxy methods were employed when feasible.

- *Artifacts for hypothesis invalidation.* Between the observation of a failure (e.g., false negatives for a certain class) and the identification of its potential causes i.e., the bugs (e.g., overfitting on other classes) and remedy (e.g., decreasing the number of layers), participants often used intermediate artifacts (e.g., training curves, data statistics) for context gathering. These artifacts were serving both to search for the potential bug, and to quickly check that no other information about the model would invalidate their hypotheses.
- *Correction as instrumentation.* Instrumenting the hypotheses was often about making a correction and checking for a positive change in the model, followed by further fine-tuning the correction (see section 4.3.3).
- *Hypothesis testing.* Only three participants tested their hypotheses with other instruments, even though it is probably more efficient than retraining a model for each hypothesis. They searched for data samples or transformed available samples to present only the features (or anything but the features) of interest, and check whether the inference of the model matches expectations (P17 mid-CV) *"I take a perturbation approach. Once you see commonalities, let's say "white", you mask out the non-white thing, and see if the probability is increasing. If so, I may be looking in the right direction and need more non-white kitchens."* Such activity needs more support as participants argued it is challenging.

4.4 RQ2 - process: Explainability for failure handling

4.4.1 * Narrow subset of explanations.

Our participants typically did not mention any tool or method inspired from the ones we identified in the literature. Our participants only mentioned using saliency maps among other explainability methods, except P4 high-CV who also mentioned T-SNE [119] for faster image exploration through image clustering. A few participants without experience with explainability described the desire to have explanations that

Table 2: The diverse hypotheses formulated by the participants around model features and data content.

Participants' hypothesis	Explanation
Irrelevant features	(P1 mid-CV) *"The model might learn wrong rules, like the presence of a sink to predict a living room"*, (P7 mid-CV) *"Once we know the wrong patterns the model learned, we add more examples that reflect the wrong behavior in the training data for the model to learn the extreme cases."*
Incomplete features	The model has not learned enough features to correctly make inferences for certain images. Incomplete and irrelevant features are always mapped to dataset biases (P17 mid-CV) *"What comes into my mind is rules, but it will defeat the purpose of having machine learning. I model what's a kitchen in a symbolic fashion like "needs an oven, stove". And then I make sure that the data set is reflecting those adequately."*
Over- or under-emphasized features	(P7 mid-CV) *"The first step is to use an interpretability method to detect what the model has learned. For example, when the model classifies kitchens, it does not look for a sink or cooking stove. It looks for under-relevant patterns like tables that can be used for other classifications like dining rooms."*
Unknown unknowns	Three participants related the incorrect or incomplete features to unknown unknowns (P7 mid-CV) *"knowing what to expect from the model and what it learns allows to identify unknown unknowns"*, (P12 high-CV) *"A blind-spot happens because of systematic data biases. You have to see how the data distribution looks like to figure out whether there is a blind spot. You should use crowdsourcing because automatic methods are not reliable."*
Absence (presence) of (ir)relevant elements in images	This makes the model confuse the ground truth for another class (e.g., the lack of a bed in a bedroom makes it being classified as a kitchen) (P7 mid-CV) *"This image is missing hot spots."*

correspond to saliency maps, without being aware of their existence. A few participants wished for other types of explanations. For instance, they would like to automatically obtain statistical summaries of visual elements across images to fasten their hypothesis formulation and validation process (P6 low-CV) *"I want to see the entire distribution of objects, and subdivide these 25 mislabeled dining rooms into smaller segments that I can understand, like photographs of dining rooms with the kitchen in the background."*. They also insisted on getting textual explanations besides visual ones to query whether the model has learned expected or known problematic features (e.g., a participant mentioned that the models shouldn't pick up on potential pace-makers), or to quickly explore the training data distribution.

4.4.2 Diverse purposes for explainability. From the interviews, we also found out that the use of explainability methods is not standardized. The purpose for and way of using the saliency maps (the primary explainability method that was employed) varied across participants. Overall, we identify four uses; non-expert participants only focusing on the first one.

- *Artifact for data content or data engineering bugs*: Saliency maps were used to identify problematic features, and to further investigate potential solutions for correctness failures. This was done by scrutinizing the image patches highlighted by saliency maps, and reflecting on the points in Table 2. Certain participants disagreed that it is feasible to look into the actual data content because it is hard to define what one would expect a model to pick-up on (P4 high-CV) *"In a bathroom you expect the bath to be highlighted. You expect the dining room table in the dining room, but in the kitchen there can also be a table, so it's not convenient."*
- *Artifact for bias-variance trade-off*: Saliency maps were used to make sure the model learned something meaningful, and is not over- or under-fitting. For that, participants analysed the shapes of the maps across images, and their coverage of pixels reflecting human-interpretable concepts (P4 high-CV) *"I first see if this model truly learned something (the objects, not some nonsense). Saliency maps are really tiny: it over-trains. It's about the general aspect of the map, more than what it's highlighting."* This was used by expert participants who have formed over time an idea of a meaningful saliency map, and how it relates to model failures (e.g., overfitting).
- *Final verification*: Certain participants used saliency maps as a last step to quickly validate the relevance (P15 high-CV) *"I first fix my model, then my data. Once I'm sure this is the model I'm going to use, I check that images are analyzed fairly according to what we expect. I see the actual visual clues that the computer bases its decisions on."* and possibly fairness of the model features in a random subset of saliency maps (P9 high-CV) *"It is very important if you're afraid the model is biased towards categories with ethical implications."*
- *Stakeholder communication*: Most participants used saliency maps for communicating about the model performance (P9 high-CV) *"You measure the success from the accuracy. If successful, you understand what the model is looking at with explainability. It is nice to explain to your clients why the model works and what it looks."*

5 DISCUSSION & FUTURE WORK

Our interviews brought new insights into computer vision model failure handling practices (summarized in Table 3), that are corroborated by the few HCI studies that compare non-machine learning practitioners with machine learning experts [128]. Instead of relying on the (potentially useful) theory, methods, and tools published in literature, practitioners develop an error-prone workflow based on their prior experiences with machine learning, and they do not systematically address every machine learning failure and bug. This is concerning as other stakeholders within an organization might also not be aware of and in charge of these failures. We now discuss implications of these results for future research.

Table 3: Summary of the insights obtained through our study.

Category	Insight
RQ1: Stated and verified goals of the failure handling process.	
*Failures	Failure handling practices for computer vision models often focus on a narrow set of failures (compared to literature), centered on output correctness, with however model problematic features as an additional, typically understudied, failure.
*Bugs	Practitioners address the same bugs as discussed in the literature, with more refined bug categorizations (structural, dataset, data-engineering, training).
Satisfaction point	Ambiguous decision boundary, made of trade-offs between various failures and correctness metrics, to declare the model ready for deployment.
Differences across practitioners	Participants have disparate knowledge about "debugging" concepts, and limited attention towards different bugs: sequential bug-elimination approach for high-CV participants, incorrect trade-offs between bugs for low-CV participants. They also show disagreement on the importance of correctness and feature failures, and disparate precautionary attitude.
RQ2: Failure handling process.	
*Workflow	An ad-hoc, trial-and-error workflow that is simpler than for traditional software system debugging is adopted. Typically hypothesis instrumentation is missing, as well as hypothesis formulation for non-data bugs. Its steps are based on practitioners' experiences.
Hypotheses	Hypotheses related to data bugs are around problematic features: incorrect or incomplete features, over- or under-emphasized feature importance, absence/presence of ir/relevant visual elements in images.
Corrections	Various correction methods: modifications of dataset, training parameters, model structure, and way the model is setup.
*Artifacts	Next to known model artifacts, primarily visual content across images is used. Need for domain knowledge is polemical.
*Methods & tools	None of the methods or tools developed in the literature are used. Only TensorBoard [22] has been mentioned.
RQ2: Use of explainability methods for failure handling purposes.	
Purposes	Diversity in purposes: scrutinizing dataset bugs, bias-variance trade-off, stakeholder communication, and final verification.
*Types	A narrow subset of explanation types (saliency maps) is used in practice. Wishes for global, textual, query-able explanations about the model and potentially the data are unfulfilled.

5.1 Surfaced design directions

Our results led to identify obstacles for practitioners to correctly handle failures. These obstacles can serve as design principles or challenges to further support practitioners. In relation to these, we discuss a few avenues for future work.

(1) *Challenging need for workflow diversity.* Failure handling requires diverse workflows, as it is a highly use-case dependent task (use-case, stakeholders, structure of an organization and allocation of responsibilities, etc.), and no one-size-fits-all process has been developed. Hence, we do not argue for standardization, but emphasize the need for a plurality of workflows, that brings about new research challenges to create supportive methods and tools.

(2) *Confusing fluidity of concepts.* One surprising insight was the fluidity of the concepts in the participants' workflows. While we had envisioned identifying independent sets of failures, bugs, artifacts, and steps, related by how one serves to identify or solve the other, we realized these sets are permeable. For instance, features can either be considered failures when they are irrelevant or incomplete according to human judgement, or an artifact to identify the dataset bugs that caused correctness or robustness failures (same observation for overfitting). Bug correction was also either the actual correction step taken by the practitioners, or one way to test their hypothesis. Concept fluidity is already known for certain non-functional, trustworthiness-related, requirements of machine learning systems, such as fairness [82], interpretability [54], and contestability [75]. This fluidity brings confusion to the research and practice, and should be acknowledged, e.g., to clarify the available tools and steps, and to reassure practitioners about their process. One can take inspiration from these other works to handle the fluidity of the failure handling concepts, for instance by proposing a comprehensive overview of the different uses of the terms by different practitioners and research communities (e.g., also highlighting the dissimilarities with traditional software engineering), as a boundary negotiation object [82].

(3) *All practitioners are not equal in confidence and effectiveness.* Low-CV participants lacked a clear workflow, spending a large part of the interview on model understanding, instead of reactive or proactive debugging. A few of these participants expressed not being confident in their process, discussing a (P5 mid-CV) *"very empirical process" that "reflects a human feeling of what's going on".* They posed that this way *"the success of debugging is left to the sensitivity of the expert".* Participants with more experience were instead more confident, faster, and effective. This result displays similarities with the way people working on non data-driven software develop an ability to debug their software, with experts learning debugging heuristics, the effective use and application of debugging tools, etc. [78]. The development of new tools should hence bear in mind the various levels of AI literacy of the practitioners and their confidence. AI literacy literature [23] refers to four literacy dimensions (technology, work, learning, and human-machine -related dimensions) that should all be considered to tailor the tools to their users.

(4) *Difficulties in using new tools.* The participants had difficulties envisioning uses of new tools. When we showed low-CV participants saliency maps or global explanations, they could not envision how to employ them. Similarly, when showing more experienced participants explanations they were not familiar with (global, textual explanations outputted by the SECA method [12]), only half of them could envision using them.

Besides, Liao et al. [68] built an explainable AI question bank where each question reflects a need for explainability. Inspired by this bank, we built a failure handling question bank for computer vision models in Table 4, that summarizes the information needs practitioners might have when handling failures. For that, we reviewed the transcripts and workflows described by our participants, and extracted their explicit questions and questions that were implicitly answered by the actions they took. Compared to the XAI question bank, we added new categories of questions, revolving around the algorithm design and the way the model was trained, around iterations of the model, and expectations on the model behavior (reflecting the need for domain knowledge). These questions revealed to be essential to tackle structural and training bugs, and to understand when the model is satisfactory. We also refined the questions about model features, their nature, relevance, completeness, etc. as features were an essential artifact to judge the validity of the model and to identify correction methods. The question bank can be used by practitioners as inspiration to identify the relevant questions (and whether methods for getting answers exist) to ask for handling failures in their model, and by researchers to identify important research directions that have not been tackled until now.

5.2 Opportunities for the design of new supportive tools

5.2.1 Need for guidance. We argue that practitioners need more guidance on the process. Proposing high-level (sequences of) steps and intermediate objectives for structuring the workflows in relation to different failures, associated artifacts, examples of bug correction methods and pitfalls, would allow for a more effective and efficient process. The exact form of this guidance requires further investigation, e.g., a tutorial, a checklist, an interactive framework, a tool suggesting a workflow and artifacts, etc. Previous works around software debugging and machine teaching provide hints for its design, highlighting the importance of *structured steps* [84, 89]; structured *documentation* [6, 18, 33, 43, 81]; or *warnings against graphical user-interfaces* [128]. Research is also needed to balance this guidance with the freedom practitioners need for failure handling, and to leave the flexibility to envision usages of new artifacts.

There is no comprehensive resource accessible by practitioners to learn about failure handling. We suggest the community to build an open, collaborative repository of practices to share heuristics (similarly to UMLAUT [100]), methods and tools, as well as theoretical knowledge[8] (e.g., list of failures, bugs, relations to artifacts). Such library should provide both general information, and information that is specific to certain types of use-cases, models, etc., since the participants regularly referred back to previous use-cases they encountered with similar considerations. Research on software debugging again provides recommendations for the design of such library, with lists of relevant information to include—e.g., patterns [79], debugging diaries [90]—, and methods to collect this information [78]. As a first step towards establishing such a library, we propose a failure handling framework (Figure 3) designed by synthesising our participants' practices. It summarizes the various objectives, main steps, and artifacts of the failure handling process.

5.2.2 Need for additional tools. Our study and especially our failure handling question bank point out to specific needs and wishes from practitioners, that would merit further research at the intersection between machine learning and human-computer interaction. On one side, the questions in our bank are partially overlapping with the ones of the XAI question bank [69], reinforcing opportunities for explainability works to serve in the failure handling process. On the other side, the questions that are not present in the XAI question bank can serve as invitations for researchers to build new methods and tools, requiring algorithmic research (e.g., "should I focus on the data or algorithm and training hyperparameters?"), or human-computer interaction research especially to facilitate communications between stakeholders (e.g., "is this inference really correct? can we accept it?") and data visualisation (e.g., "does the model make errors with high or low confidence?"). We discuss a few of these research opportunities.

Novel types of explanations. Certain practitioners mentioned desiderata sometimes similar to existing but rare explanations. These insights corroborate the results of Hong et al. [47] on potential uses of and needs for explainability. In previous works [135], these explanations are summarized as global textual [11, 12] or visual concepts [34, 60], examplars [59] (samples that contrast or are similar to others), and cues (hint on the main differences or similarities). From the identified workflows, it seems that global explanations could greatly speedup certain steps of their process, textual explanations could lead to more accurately identify bugs and support communication between stakeholders, and interactivity could help navigating these explanations.

Data & feature exploration. Our participants spent a large amount of time exploring the dataset for understanding what it represents, to identify potential biases, unknown unknowns, shifts, etc, and to identify and judge model features (with the help of other stakeholders). They (wish to) do so through various types of interactions, e.g., getting random samples for each class in the dataset, clustering images with similar visual content, querying samples with various visual elements, etc. Interactive visualisation tools could greatly support them in easily accessing such information. Existing tools for data exploration in the context of machine learning [17, 46] could be refined for the specific needs identified. Particularly, facing the diversity of hypotheses one can extract from explanation artifacts (subsubsection 4.3.4), it appears highly relevant to develop user-interfaces for feature exploration, allowing the search of model features at different granularities, the comparison of feature importance, and the matching of model features with expected ones, to investigate the dissonance between human features and machine learned features [17, 136]. One key challenge would be the uncertainty within these features —expected ones are not always known, while learned ones are never entirely known due to the interpretability gap for existing explainability methods [12]—, requiring constant fine-tuning [80]. This highlights the importance, despite the complexity of it, of involving domain experts in the failure handling process, as they can support the practitioners in identifying additional failures by reporting on their own experiences with challenging edge-cases, and with priorities in terms of correctly-classified data samples and meaningful features, etc. Prior works, especially in the medical context [21], have shown the potential ease in designing a library of test cases, that should be further investigated not only for supporting the responsible use of models by end-users, but also for developing appropriate models.

Model comparisons. The iterative process requires to frequently retrain the model and compare it with its previous versions in terms of performance, features, and other artifacts. Yet, few practical tools [15] support such comparison. As retraining deep learning models is computationally intensive, methods are needed to provide estimates of the changes in these artifacts, e.g., by building simpler surrogate models that would be less heavy to retrain.

Hypothesis testing and bug correction. Hypothesis instrumentation and testing are the main steps our participants skipped compared to the traditional software debugging workflow, due to the lack of methods existing to do so efficiently. Yet, this could certainly save further training time. Recent works such as *Deblinder* [20] or an explainability-based debugging framework [11] start to propose support, by displaying model failures to the developer who has then

[8] Similarly to existing initiatives, such as https://docs.microsoft.com/en-us/security/engineering/failure-modes-in-machine-learning

Table 4: Questions practitioners ask when handling failures of models. In bold the ones also found in the XAI question bank [68], and with a triangle △ the ones that have not received extensive attention in terms of study of practices or technical solutions. Questions without a triangle are formatted in italic when they can be (partially) answered using existing explainability methods, the others being answered using other debugging artifacts.

Topic	Question
Input	**What kind of data does the system learn from?** (and all related questions of the XAI question bank)
	To what extent is the data diverse enough to represent each class? To what extent is it balanced over the different classes?
	△Does the test dataset cover the complete range of situations the model can encounter in deployment?
	What do the samples look like for each class? *What are the difference between these two classes?*
	How have the data been processed? and augmented? Is it easy to augment the dataset by collecting new data?
Model performance	**How well does the model perform generally? Where does the model typically make errors? for what type of images? into which classes does it incorrectly classify them?**
	Does the model make errors with high or low confidence? △Are there unknown unknowns?
Expectations	△What is the expected performance for the model? for which metrics? Can we consider the model to be fair and unbiased?
	△Is this inference really incorrect? or can we accept it? What should the model pick up on to distinguish these two classes?
Model structure	**What is the structure of the model? How were the parameters set?**
	How was the model trained? What loss function was used? what were the training hyperparameters?
Model training	*Is the model overfitting or underfitting? Is the model too large/small for the task? compared to the training data?*
	Is the training dataset of the pretrained model relevant for the target task?
	Does the performance improve when simply adding training samples?
Features (global - how)	*Has the model learned anything relevant?* ***Does the model use (or not use) this feature?***
	Which visual elements does the model use to predict this class? ***Which visual elements does the model generally use?*** *Which visual elements does the model use to make (in)correct inferences?*
Features (local)	***What features of this instance lead to this inference?***
	Why is this sample predicted P instead of Q? *Which visual elements might have triggered this wrong inference?*
Features (comparison)	*What are the features used for both classes? What are the features different for the two classes?*
	Why are instances A and B given the same/different predictions?
	What are the top features/rules used by the model? **How does the model weigh different features?**
Questionable features	*Are these visual elements relevant for this sample? or class? What features do we expect it to learn for this class?*
	△Should the model pick up on more visual elements for this image/class? △Should it learn additional features?
	Does the model make correct inferences using wrong features? Are the features fair to use?
Inferences (what if)	***What would the model predict if this sample is changed to ...?***
	What would the model output for a sample with these visual elements?
Iterations	△How to improve the model? △Should I focus on the data or algorithm and training hyperparameters?
	How well does the model perform after doing X? *Have the features changed after doing X?*

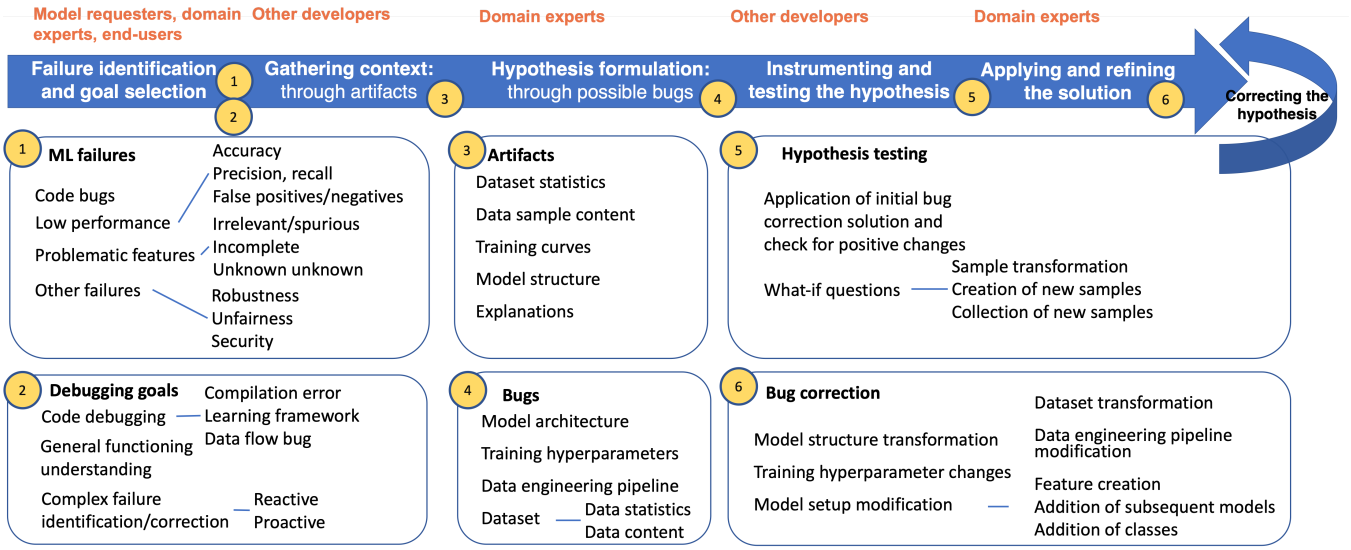


Figure 3: Summary of the failure handling practices identified through the interviews. In orange, we show the stakeholders that can intervene in each step of the failure handling process.

several options for generating and testing hypotheses, yet targeted bug correction is still not supported by any tool. We recommend to develop such functionalities to allow for faster testing.

5.3 Increasing clarity in the failure handling goals & process

Our study showed the importance for our participants to access various types of knowledge during the failure handling process. Hence, clear communication with various stakeholders or clear documentation appeared necessary (more information in Appendix A.2.2). These results reflect previous works around the data science lifecycle [30, 47, 62, 91, 130]. The information needs and associated communication challenges in these studies and ours are overlapping (e.g., misaligned vocabulary and knowledge). We list below additional challenges.

5.3.1 Designing metrics for clarity. The participants rightfully recognized that a model cannot make perfect inferences, and consequently that not all misclassifications should be considered failures but instead that certain should be treated as acceptable. Differently from software engineering, the end-point criteria for deploying a model revealed to be subjective. This subjectivity has been illustrated in prior studies [21], where, similarly to model developers, model users decide on the acceptability of model misclassifications based on their expectations for the model, especially in relation to their own locus of expertise to allow for a successful collaboration between them and the model. Our participants however did not tend to extensively account for this notion of human-model collaboration to decide on failures and the model readiness for deployment, despite the increasing number of research works on the topic [13, 124, 133]. The end-point criteria was also ambiguous, e.g., expert participants, while not considering all model errors equal, did not have a clear process besides trying to attribute different levels of severities to ad-hoc categories of observed failures. Ethnographic work in a data science team has similarly shown the equivocal nature of performance metrics both for the developers and other stakeholders judging the trustworthiness of the models [88], our work expanding these findings to models that are not built in order to discover new insights from data but to automate a process that can typically be performed by humans. This was also observed in prior studies where participants implicitly attributed "cost" to the different wrong predictions [31, 128], and pointed out to the discrepancy between the perceived performance of a model, and its performance as measured by a metric [41, 87, 98, 109]. We suggest to develop metrics or frameworks that would document and account for these various costs. Recent research directions on disaggregating evaluation metrics [14, 77] could include these concerns in their propositions. This would especially allow to adhere to new concerns for accountability and transparency, facing the subjectivity in defining an end-point.

Feature issues are not discussed in machine learning testing research, and only mentioned sparsely within literature around statistical biases in dataset [115, 116], or explainability methods [12, 105, 106], despite their importance (discussed by 17 out of 18 practitioners). The absence from research could be explained by the lack of metrics to evaluate them, yet one prior study [88], although in a different context, also identified the importance of valid model explanations for stakeholders to decide on using a model in practice. Recent works such as Shared Interest [17] constitute a first step towards quantifying feature failures. Its categorization of samples depending on the correctness of model predictions (proactive or reactive debugging), and whether the model features are aligned with human expectations, is highly reflective of the feature hypotheses identified in Table 2. We however also identified a discussion around the features' weights, not addressed in the literature.

5.3.2 Increasing transparency between developers. Our results and especially certain of the questions in our question bank also showed the more general need for developers to communicate with each other. Documentation, although often not used by the interviewed practitioners beyond model versioning, seems like the right avenue to facilitate such knowledge sharing across practitioners, similarly to what previous studies also concluded [30, 43]. Next to detailing how a dataset was created [33] or the performance and scope of a trained model [81], future documentations should also focus on "intermediate models" and on logging the experiments conducted across models for a single system and the reasoning behind the choices of experiment. While this could be saved as code, making the steps clear in the form of textual descriptions [97] could fasten the process. E.g., the participants asked what kind of data processing had been conducted, which could be answered without looking into the specificities of the code.

5.4 Beyond the failure handling process: additional changes needed

5.4.1 Lack of communication between research and practice. Our participants do not use the methods and tools stemming from research publications (except a few explainability methods, and common code development tools such as TensorBoard [22]) due to a lack of awareness. This does not necessarily hint at a technical problem, but at a structural one. It highlights a lack of knowledge or time, from certain practitioners to search for these materials. Hence, disseminating further the outputs from research to practitioners appears to be an avenue for future work.

5.4.2 AI education. The challenges identified also reveal limitations in the way computer vision is learned. Our participants, while having followed a computer vision course and/or learned computer vision through reading resources around the Internet, primarily build their failure handling process over time by discussing with colleagues (P15 high-CV) *"I never learned computer vision in school. I learned it from the Internet and I had few experiences in internships."*, reading about practices (P16 high-CV and P3 high-CV mentioned specific blog posts about failures and bugs [53]), and through practical experiences (P9 low-CV) *"To improve performance, it would be horizontally (you add more lines to a dataset), or vertically (more columns, that is more features). I'm speaking out of my experience about records. For images, more lines could be data augmentation, more columns could be features that correspond to specific objects."* None has been taught a failure handling process in a curriculum (P5 low-CV) *"I did the deep learning course in the Masters and then some computer vision projects. From that, I learned the basic tools and* *common libraries for deep learning."* This corroborates prior observations around machine learning practices [3, 110], and debugging of software [79].

Developing education around failure handling for computer vision models could benefit practitioners, as is suggested by position papers [102] and successfully experimented with in research on teaching debugging. Particularly, research around software debugging teaching [78, 79, 86], and data science teaching [35, 64, 71, 73, 110, 111, 125] proposes teaching through exercises with examples of workflows or hierarchical lists of questions to ask for correctly "debugging".

Computer vision practitioners could also exploit online communities to get further training (none of our participants mentioned using these frequently), similarly to data science practitioners [104]. Failure handling tasks could be shared online and executed in collaboration. Yet, one would need to investigate how to share relevant materials (e.g., trained model, datasets), information (documentation about the task and model), and solutions.

6 LIMITATIONS & THREATS TO VALIDITY

There are several limitations in our study. While we do not think they impact the validity of our results, tackling them in the future would improve the generalisability of our findings.

We used one simple scenario, that enabled our participants to easily describe their usual practices, as the various examples the participants brought from their own use-cases and some comments testify, e.g., (P13 high-CV) *"My very first thought was: this is a very realistic use case"*. Yet, using such scenario might obfuscate specificities of their own use-cases, such as competing incentives they might encounter (they primarily referred to constraints around data collection). However, using a different use-case per participant would have not allowed to fairly compare practices, and would have posed confidentiality issues. Freeing them from competing incentives places them in a more ideal situation to discuss their process. Besides, our scenario presented the participants with information about the model to "debug", without the actual development code —that they did not ask for. A task where they would be presented with the training code could provide additional insights, but would require longer interview sessions. Our methodology inspired from previous works [11, 30, 44] already provided us with main challenges and limitations.

We focused on failure handling in development. Practices might differ after deployment, as other failures and constraints might occur, and additional stakeholders might be involved. We looked primarily into correctness and feature failures that are still understudied. Yet, many more types of failures might arise. We interviewed a considerable amount of participants and devoted our efforts to cover practitioners with various levels of experience. Such qualitative approach can never completely assure that we gathered all failure handling practices that exist. In the future, one might want to perform studies with other methodologies, e.g., ethnographic work for in-context practices, code-based studies, different focuses, and in specific domains of application, to complement our results. Finally, we focused on models for image-based computer vision applications, and hence we cannot conclude certainly on the applicability of our results to other types of models. We can however mention that our discussion on the organisation of the field echoes prior discussions around other applications such as the ones relying on tabular data [54]. Besides, the design opportunities we highlight are applicable to other applications as they are not application-specific. However, the required technical work would differ to leverage the relevant artifacts, that are different across applications —and more or less researched until now (e.g., more research on explainability for tabular-data based applications has been performed than for image-based applications). Whether these design opportunities are necessary for practitioners developing these other applications, should be investigated in the future, and our work can provide inspiration to do so in terms of insights to look for. It is fair to assume that certain of the insights would hold as our participants and other practitioners have typically received the same training, and many machine learning models across applications share similar properties.

7 CONCLUSION

In this work, we conducted 18 semi-structured interviews to outline the practices for handling failures in computer vision models (Figure 3). We showed that, while practices broadly follow the traditional software debugging workflow, they differentiate by the ambiguous way the model requirements are defined, by the type of hypothesis formulation and instrumentation activities performed in the machine learning context, by the artifacts employed to facilitate the workflow, and by the fluidity of the relevant concepts. Besides, failure handling workflows are typically performed manually and in collaboration without resorting to methods developed specifically for machine learning models (Table 4). Finally, practitioners tend to have a narrow understanding of the failures and bugs that any machine learning model might suffer from, skewed by their prior experience. This understanding yet includes problematic model features that are not typically investigated in scientific literature. These insights point out to various limitations and challenges in the current failure handling process, that should be tackled through both structural changes and socio-technical research. Especially, we drew a list of research opportunities at the intersection between HCI and machine learning, going from the creation of a collaborative library of best-practices, to the development of failure handling methods and user-interfaces, and of support for communication between stakeholders.

ACKNOWLEDGMENTS

This work was partially supported by the HyperEdge Sensing project funded by Cognizant. We would also like to thank all the participants of our study, without whom this work would not have been possible.

REFERENCES

[1] Mouna Afif, Riadh Ayachi, Yahia Said, and Mohamed Atri. 2020. Deep learning based application for indoor scene recognition. *Neural Processing Letters* 51, 3 (2020), 2827–2837.

[2] Ahmed Alqaraawi, Martin Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Berthouze. 2020. Evaluating saliency map explanations for convolutional neural networks: a user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces.* 275–285.

[3] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software engineering for machine learning: A case study. In *2019 IEEE/ACM*

41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). IEEE, 291–300.

[4] Saleema Amershi, Max Chickering, Steven M Drucker, Bongshin Lee, Patrice Simard, and Jina Suh. 2015. Modeltracker: Redesigning performance analysis tools for machine learning. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 337–346.

[5] Paul Ammann and Jeff Offutt. 2016. *Introduction to Software Testing* (2nd ed.). Cambridge University Press, USA.

[6] Ariful Islam Anik and Andrea Bunt. 2021. Data-Centric Explanations: Explaining Training Data of Machine Learning Systems to Promote Transparency. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–13.

[7] Keijiro Araki, Zengo Furukawa, and Jingde Cheng. 1991. A general framework for debugging. *IEEE software* 8, 3 (1991), 14–20.

[8] Vijay Arya, Rachel KE Bellamy, Pin-Yu Chen, Amit Dhurandhar, Michael Hind, Samuel C Hoffman, Stephanie Houde, Q Vera Liao, Ronny Luss, Aleksandra Mojsilović, et al. 2019. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. (2019).

[9] Joshua Attenberg, Panos Ipeirotis, and Foster Provost. 2015. Beat the machine: Challenging humans to find a predictive model's "unknown unknowns". *Journal of Data and Information Quality (JDIQ)* 6, 1 (2015), 1–17.

[10] Agathe Balayn, Christoph Lofi, and Geert-Jan Houben. 2021. Managing bias and unfairness in data for decision support: a survey of machine learning and data engineering approaches to identify and mitigate bias and unfairness within data management and analytics systems. *The VLDB Journal* (2021), 1–30.

[11] Agathe Balayn, Natasa Rikalo, Christoph Lofi, Jie Yang, and Alessandro Bozzon. 2022. How can Explainability Methods be Used to Support Bug Identification in Computer Vision Models?. In *CHI Conference on Human Factors in Computing Systems*. 1–16.

[12] Agathe Balayn, Panagiotis Soilis, Christoph Lofi, Jie Yang, and Alessandro Bozzon. 2021. What do You Mean? Interpreting Image Classification with Crowdsourced Concept Extraction and Analysis. In *Proceedings of the Web Conference 2021*. 1937–1948.

[13] Gagan Bansal, Besmira Nushi, Ece Kamar, Eric Horvitz, and Daniel S Weld. 2021. Is the most accurate ai the best teammate? optimizing ai for teamwork. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 11405–11414.

[14] Solon Barocas, Anhong Guo, Ece Kamar, Jacquelyn Krones, Meredith Ringel Morris, Jennifer Wortman Vaughan, W Duncan Wadsworth, and Hanna Wallach. 2021. Designing disaggregated evaluations of ai systems: Choices, considerations, and tradeoffs. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 368–378.

[15] Alex Bäuerle, Ángel Alexander Cabrera, Fred Hohman, Megan Maher, David Koski, Xavier Suau, Titus Barik, and Dominik Moritz. 2022. Symphony: Composing Interactive Interfaces for Machine Learning. In *CHI Conference on Human Factors in Computing Systems*. 1–14.

[16] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, R Puri, J MF Moura, and P Eckersley. 2020. Explainable machine learning in deployment. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 648–657.

[17] Angie Boggust, Benjamin Hoover, Arvind Satyanarayan, and Hendrik Strobelt. 2022. Shared Interest: Measuring Human-AI Alignment to Identify Recurring Patterns in Model Behavior. In *CHI Conference on Human Factors in Computing Systems*. 1–17.

[18] Karen L Boyd. 2021. Datasheets for Datasets help ML Engineers Notice and Understand Ethical Issues in Training Data. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–27.

[19] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[20] Ángel Alexander Cabrera, Abraham J Druck, Jason I Hong, and Adam Perer. 2021. Discovering and Validating AI Errors With Crowdsourced Failure Reports. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–22.

[21] Carrie J Cai, Samantha Winter, David Steiner, Lauren Wilcox, and Michael Terry. 2019. " Hello AI": uncovering the onboarding needs of medical practitioners for human-AI collaborative decision-making. *Proceedings of the ACM on Human-computer Interaction* 3, CSCW (2019), 1–24.

[22] Shanqing Cai, Eric Breck, E Nielsen, M Salib, and D Sculley. 2016. Tensorflow debugger: Debugging dataflow graphs for machine learning. (2016).

[23] Dilek Cetindamar, Kirsty Kitto, Mengjia Wu, Yi Zhang, Babak Abedin, and Simon Knight. 2022. Explicating AI Literacy of Employees at Digital Workplaces. *IEEE Transactions on Engineering Management* (2022).

[24] Hao-Fei Cheng, Ruotong Wang, Zheng Zhang, Fiona O'Connell, T Gray, F M Harper, and H Zhu. 2019. Explaining decision-making algorithms through UI: Strategies to help non-expert stakeholders. In *Proceedings of the 2019 chi conference on human factors in computing systems*. 1–12.

[25] Ruiqi Cheng, Kaiwei Wang, Jian Bai, and Zhijie Xu. 2020. Unifying visual localization and scene recognition for people with visual impairment. *IEEE Access* 8 (2020), 64284–64296.

[26] Ram Chillarese. 1999. Software Testing Best Practices, IBM Research. TR Patent RC21,457.

[27] Michael Chromik, Malin Eiband, Felicitas Buchner, Adrian Krüger, and Andreas Butz. 2021. I Think I Get Your Point, AI! The Illusion of Explanatory Depth in Explainable AI. In *26th International Conference on Intelligent User Interfaces*. 307–317.

[28] Brittany Davis, Maria Glenski, William Sealy, and Dustin Arendt. 2020. Measure utility, gain trust: practical advice for XAI researchers. In *2020 IEEE Workshop on TRust and EXpertise in Visual Analytics (TREX)*. IEEE, 1–8.

[29] Terrance De Vries, Ishan Misra, Changhan Wang, and Laurens Van der Maaten. 2019. Does object recognition work for everyone?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 52–59.

[30] Wesley Hanwen Deng, Manish Nagireddy, Michelle Seng Ah Lee, Jatinder Singh, Zhiwei Steven Wu, Kenneth Holstein, and Haiyi Zhu. 2022. Exploring How Machine Learning Practitioners (Try To) Use Fairness Toolkits. *arXiv preprint arXiv:2205.06922* (2022).

[31] Rebecca Fiebrink, Perry R Cook, and Dan Trueman. 2011. Human model evaluation in interactive supervised learning. In *Proceedings of the SIGCHI conference on human factors in computing systems*. 147–156.

[32] Gordon Fraser and JM Rojas. 2019. *Software Testing*. Springer International Publishing, Cham, 123–192. https://doi.org/10.1007/978-3-030-00262-6_4

[33] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. 2021. Datasheets for datasets. *Commun. ACM* 64, 12 (2021), 86–92.

[34] A Ghorbani and al. 2019. Towards automatic concept-based explanations. In *NeurIPS*.

[35] Yolanda Gil. 2016. Teaching big data analytics skills with intelligent workflow systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[36] Görkem Giray. 2021. A software engineering perspective on engineering machine learning systems: State of the art and challenges. *Journal of Systems and Software* 180 (2021), 111031.

[37] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Counterfactual visual explanations. In *International Conference on Machine Learning*. PMLR, 2376–2384.

[38] Stefan Grafberger, Julia Stoyanovich, and Sebastian Schelter. 2021. Lightweight Inspection of Data Preprocessing in Native Machine Learning Pipelines.. In *CIDR*.

[39] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. 2018. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3608–3617.

[40] Brent Hailpern and Padmanabhan Santhanam. 2002. Software debugging, testing, and verification. *IBM Systems Journal* 41, 1 (2002), 4–12.

[41] Galen Harrison, Julia Hanson, Christine Jacinto, Julio Ramirez, and Blase Ur. 2020. An empirical study on the perceived fairness of realistic, imperfect machine learning models. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 392–402.

[42] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. 2019. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 558–567.

[43] Amy Heger, Elizabeth B Marquis, Mihaela Vorvoreanu, Hanna Wallach, and Jennifer Wortman Vaughan. 2022. Understanding Machine Learning Practitioners' Data Documentation Perceptions, Needs, Challenges, and Desiderata. *arXiv preprint arXiv:2206.02923* (2022).

[44] Fred Hohman, Andrew Head, Rich Caruana, Robert DeLine, and Steven M Drucker. 2019. Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the 2019 CHI conference on human factors in computing systems*. 1–13.

[45] Fred Hohman, Haekyu Park, Caleb Robinson, and Duen Horng Polo Chau. 2019. Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 1096–1106.

[46] Fred Hohman, Kanit Wongsuphasawat, Mary Beth Kery, and Kayur Patel. 2020. Understanding and visualizing data iteration in machine learning. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–13.

[47] Sungsoo Ray Hong, Jessica Hullman, and Enrico Bertini. 2020. Human factors in model interpretability: Industry practices, challenges, and needs. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW1 (2020), 1–26.

[48] Fuyuki Ishikawa and Nobukazu Yoshioka. 2019. How do engineers perceive difficulties in engineering of machine-learning systems?-questionnaire survey. In *2019 IEEE/ACM Joint 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)*. IEEE, 2–9.

[49] Md Johirul Islam, Giang Nguyen, Rangeet Pan, and Hridesh Rajan. 2019. A comprehensive study on deep learning bug characteristics. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and*

Symposium on the Foundations of Software Engineering. 510–520.

[50] Hanen Jabnoun, Faouzi Benzarti, and Hamid Amiri. 2017. Visual scene prediction for blind people based on object recognition. In *2017 14th International Conference on Computer Graphics, Imaging and Visualization*. IEEE, 21–26.

[51] Sérgio Jesus, Catarina Belém, Vladimir Balayan, João Bento, P Saleiro, P Bizarro, and J Gama. 2021. How can I choose an explainer? An Application-grounded Evaluation of Post-hoc Explanations. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 805–815.

[52] Daniel Kang, Deepti Raghavan, Peter Bailis, and Matei Zaharia. 2018. Model assertions for debugging machine learning. In *NeurIPS MLSys Workshop*.

[53] Andrej Karpathy. 2019. A Recipe for Training Neural Networks. http://karpathy.github.io/2019/04/25/recipe/

[54] Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, and Jennifer Wortman Vaughan. 2020. Interpreting interpretability: understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–14.

[55] Christopher J Kelly, Alan Karthikesalingam, Mustafa Suleyman, Greg Corrado, and Dominic King. 2019. Key challenges for delivering clinical impact with artificial intelligence. *BMC medicine* 17, 1 (2019), 1–9.

[56] Daniel Kerrigan, Jessica Hullman, and Enrico Bertini. 2021. A survey of domain knowledge elicitation in applied machine learning. *Multimodal Technologies and Interaction* 5, 12 (2021), 73.

[57] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. 2020. A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review* 53, 8 (2020), 5455–5516.

[58] Roli Khanna, Jonathan Dodge, Andrew Anderson, Rupika Dikkala, Jed Irvine, Zeyad Shureih, Kin-ho Lam, Caleb R Matthews, Zhengxian Lin, Minsuk Kahng, et al. 2022. Finding AI's faults with AAR/AI: An empirical study. *ACM Transactions on Interactive Intelligent Systems (TiiS)* 12, 1 (2022), 1–33.

[59] Been Kim, Oluwasanmi Koyejo, Rajiv Khanna, et al. 2016. Examples are not enough, learn to criticize! Criticism for Interpretability.. In *NIPS*. 2280–2288.

[60] B Kim, M Wattenberg, and al. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors. In *ICML*.

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

[62] Sean Kross and Philip Guo. 2021. Orienting, framing, bridging, magic, and counseling: How data scientists navigate the outer loop of client collaborations in industry and academia. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–28.

[63] Todd Kulesza, Margaret Burnett, Weng-Keen Wong, and Simone Stumpf. 2015. Principles of explanatory debugging to personalize interactive machine learning. In *Proceedings of the 20th international conference on intelligent user interfaces*. 126–137.

[64] Niklas Lavesson. 2010. Learning machine learning: a case study. *IEEE Transactions on Education* 53, 4 (2010), 672–676.

[65] Lucas Layman, Madeline Diep, Meiyappan Nagappan, Janice Singer, Robert Deline, and Gina Venolia. 2013. Debugging revisited: Toward understanding the debugging needs of contemporary software developers. In *2013 ACM/IEEE international symposium on empirical software engineering and measurement*. IEEE, 383–392.

[66] Jihyun Lee, Sungwon Kang, and Danhyung Lee. 2012. Survey on software testing practices. *IET software* 6, 3 (2012), 275–282.

[67] Maurizio Leotta, Dario Olianas, and Filippo Ricca. 2022. A large experimentation to analyze the effects of implementation bugs in machine learning algorithms. *Future Generation Computer Systems* 133 (2022), 184–200.

[68] Q Vera Liao, Daniel Gruen, and Sarah Miller. 2020. Questioning the AI: informing design practices for explainable AI user experiences. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–15.

[69] Q Vera Liao, Milena Pribić, Jaesik Han, Sarah Miller, and Daby Sow. 2021. Question-Driven Design Process for Explainable AI User Experiences. *arXiv preprint arXiv:2104.03483* (2021).

[70] Brian Y Lim, Qian Yang, Ashraf M Abdul, and Danding Wang. 2019. Why these Explanations? Selecting Intelligibility Types for Explanation Goals.. In *IUI Workshops*.

[71] Phoebe Lin and Jessica Van Brummelen. 2021. Engaging Teachers to Co-Design Integrated AI Curriculum for K-12 Classrooms. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–12.

[72] Anthony Liu, Santiago Guerra, Isaac Fung, Gabriel Matute, Ece Kamar, and Walter Lasecki. 2020. Towards hybrid human-ai workflows for unknown unknown detection. In *Proceedings of The Web Conference 2020*. 2432–2442.

[73] Duri Long and Brian Magerko. 2020. What is AI literacy? Competencies and design considerations. In *Proceedings of the 2020 CHI conference on human factors in computing systems*. 1–16.

[74] Raoni Lourenço, Juliana Freire, and Dennis Shasha. 2019. Debugging machine learning pipelines. In *Proceedings of the 3rd International Workshop on Data Management for End-to-End Machine Learning*. 1–10.

[75] Henrietta Lyons, Eduardo Velloso, and Tim Miller. 2021. Conceptualising contestability: Perspectives on contesting algorithmic decisions. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–25.

[76] Shiqing Ma, Yingqi Liu, Wen-Chuan Lee, Xiangyu Zhang, and Ananth Grama. 2018. MODE: automated neural network model debugging via state differential analysis and input selection. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 175–186.

[77] Michael Madaio, Lisa Egede, Hariharan Subramonyam, Jennifer Wortman Vaughan, and Hanna Wallach. 2022. Assessing the Fairness of AI Systems: AI Practitioners' Processes, Challenges, and Needs for Support. *Proceedings of the ACM on Human-Computer Interaction* 6, CSCW1 (2022), 1–26.

[78] Renee McCauley, Sue Fitzgerald, Gary Lewandowski, Laurie Murphy, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging: a review of the literature from an educational perspective. *Computer Science Education* 18, 2 (2008), 67–92.

[79] Tilman Michaeli and Ralf Romeike. 2019. Improving debugging skills in the classroom: The effects of teaching a systematic debugging process. In *Proceedings of the 14th workshop in primary and secondary computing education*. 1–7.

[80] Swati Mishra and Jeffrey M Rzeszotarski. 2021. Crowdsourcing and Evaluating Concept-driven Explanations of Machine Learning Models. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–26.

[81] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. 2019. Model cards for model reporting. In *Proceedings of the conference on fairness, accountability, and transparency*. 220–229.

[82] Deirdre K Mulligan, Joshua A Kroll, Nitin Kohli, and Richmond Y Wong. 2019. This thing called fairness: Disciplinary confusion realizing a value in technology. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–36.

[83] Shweta Narkar, Yunfeng Zhang, Q Vera Liao, Dakuo Wang, and Justin D Weisz. 2021. Model LineUpper: Supporting Interactive Model Comparison at Multiple Levels for AutoML. In *26th International Conference on Intelligent User Interfaces*. 170–174.

[84] Besmira Nushi, Ece Kamar, Eric Horvitz, and Donald Kossmann. 2017. On human intellect and machine failures: Troubleshooting integrative machine learning systems. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[85] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature visualization. *Distill* 2, 11 (2017), e7.

[86] Gary M Olson, Sylvia Sheppard, and Elliot Soloway. 1987. *Empirical studies of programmers: second workshop*. Vol. 2. Intellect Books.

[87] Andrea Papenmeier, Dagmar Kern, Daniel Hienert, Yvonne Kammerer, and Christin Seifert. 2022. How Accurate Does It Feel?–Human Perception of Different Types of Classification Mistakes. In *CHI Conference on Human Factors in Computing Systems*. 1–13.

[88] Samir Passi and Steven J Jackson. 2018. Trust in data science: Collaboration, translation, and accountability in corporate data science projects. *Proceedings of the ACM on Human-Computer Interaction* 2, CSCW (2018), 1–28.

[89] Kayur Patel, Naomi Bancroft, Steven M Drucker, James Fogarty, Amy J Ko, and James Landay. 2010. Gestalt: integrated support for implementation and analysis in machine learning. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*. 37–46.

[90] Michael Perscheid, Benjamin Siegmund, Marcel Taeumel, and Robert Hirschfeld. 2017. Studying the advancement in debugging practice of professional software developers. *Software Quality Journal* 25, 1 (2017), 83–110.

[91] David Piorkowski, Soya Park, April Yi Wang, Dakuo Wang, Michael Muller, and Felix Portnoy. 2021. How ai developers overcome communication challenges in a multidisciplinary team: A case study. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–25.

[92] Tawsifur Rahman, Amith Khandakar, Yazan Qiblawey, Anas Tahir, Serkan Kiranyaz, Saad Bin Abul Kashem, Mohammad Tariqul Islam, Somaya Al Maadeed, Susu M Zughaier, Muhammad Salman Khan, et al. 2021. Exploring the effect of image enhancement techniques on COVID-19 detection using chest X-ray images. *Computers in biology and medicine* 132 (2021), 104319.

[93] Iyad Rahwan, Manuel Cebrian, Nick Obradovich, Josh Bongard, Jean-François Bonnefon, Cynthia Breazeal, Jacob W Crandall, Nicholas A Christakis, Iain D Couzin, Matthew O Jackson, et al. 2019. Machine behaviour. *Nature* 568, 7753 (2019), 477–486.

[94] Nathalie Rauschmayr, Vikas Kumar, Rahul Huilgol, Andrea Olgiati, Satadal Bhattacharjee, Nihal Harish, Vandana Kannan, Amol Lele, Anirudh Acharya, Jared Nielsen, et al. 2021. Amazon SageMaker Debugger: A System for Real-Time Insights into Machine Learning Model Training. *Proceedings of Machine Learning and Systems* 3 (2021).

[95] Donghao Ren, Saleema Amershi, Bongshin Lee, Jina Suh, and Jason D Williams. 2016. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE transactions on visualization and computer graphics* 23, 1 (2016), 61–70.

[96] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1135–1144.

[97] John Richards, David Piorkowski, Michael Hind, Stephanie Houde, and Aleksandra Mojsilović. 2020. A methodology for creating AI FactSheets. *arXiv preprint arXiv:2006.13796* (2020).

[98] Nripsuta Ani Saxena, Karen Huang, Evan DeFilippis, Goran Radanovic, David C Parkes, and Yang Liu. 2019. How do fairness definitions fare? Examining public attitudes towards algorithmic definitions of fairness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. 99–106.

[99] Frank Schneider, Felix Dangel, and Philipp Hennig. 2021. Cockpit: A Practical Debugging Tool for Training Deep Neural Networks. (2021).

[100] Eldon Schoop, Forrest Huang, and Björn Hartmann. 2021. UMLAUT: Debugging Deep Learning Programs using Program Structure and Model Behavior. (2021).

[101] Shreya Shankar, Yoni Halpern, Eric Breck, James Atwood, Jimbo Wilson, and D Sculley. 2017. No classification without representation: Assessing geodiversity issues in open data sets for the developing world. *arXiv preprint arXiv:1711.08536* (2017).

[102] R Benjamin Shapiro, Rebecca Fiebrink, and Peter Norvig. 2018. How machine learning impacts the undergraduate computing curriculum. *Commun. ACM* 61, 11 (2018), 27–29.

[103] Shahin Sharifi Noorian, Sihang Qiu, Ujwal Gadiraju, Jie Yang, and Alessandro Bozzon. 2022. What Should You Know? A Human-In-the-Loop Approach to Unknown Unknowns Characterization in Image Recognition. In *Proceedings of the ACM Web Conference 2022*. 882–892.

[104] Nischal Shrestha, Titus Barik, and Chris Parnin. 2021. Remote, but Connected: How# TidyTuesday Provides an Online Community of Practice for Data Scientists. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW1 (2021), 1–31.

[105] K Simonyan, A Vedaldi, and A Zisserman. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *ICLR*.

[106] Sahil Singla, Besmira Nushi, Shital Shah, Ece Kamar, and Eric Horvitz. 2020. Understanding Failures of Deep Networks via Robust Feature Extraction. (2020).

[107] Leon Sixt, Maximilian Granz, and Tim Landgraf. 2020. When Explanations Lie: Why Many Modified BP Attributions Fail. In *International Conference on Machine Learning*. PMLR, 9046–9057.

[108] Kacper Sokol and Peter Flach. 2020. Explainability fact sheets: a framework for systematic assessment of explainable approaches. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*. 56–67.

[109] Megha Srivastava, Hoda Heidari, and Andreas Krause. 2019. Mathematical notions vs. human perception of fairness: A descriptive approach to fairness for machine learning. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2459–2468.

[110] Thilo Stadelmann, Julian Keuzenkamp, Helmut Grabner, and Christoph Würsch. 2021. The AI-atlas: didactics for teaching AI and machine learning on-site, online, and hybrid. *Education Sciences* 11, 7 (2021), 318.

[111] Elisabeth Sulmont, Elizabeth Patitsas, and Jeremy R Cooperstock. 2019. What is hard about teaching machine learning to non-majors? Insights from classifying instructors' learning goals. *ACM Transactions on Computing Education (TOCE)* 19, 4 (2019), 1–16.

[112] Xiaobing Sun, Tianchi Zhou, Gengjie Li, Jiajun Hu, Hui Yang, and Bin Li. 2017. An empirical study on real bugs for machine learning programs. In *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 348–357.

[113] Harini Suresh, Steven R Gomez, Kevin K Nam, and A Satyanarayan. 2021. Beyond Expertise and Roles: A Framework to Characterize the Stakeholders of Interpretable Machine Learning and their Needs. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–16.

[114] Ferdian Thung, Shaowei Wang, David Lo, and Lingxiao Jiang. 2012. An empirical study of bugs in machine learning systems. In *2012 IEEE 23rd International Symposium on Software Reliability Engineering*. IEEE, 271–280.

[115] Tatiana Tommasi, Novi Patricia, Barbara Caputo, and Tinne Tuytelaars. 2017. A deeper look at dataset bias. In *Domain adaptation in computer vision applications*. Springer, 37–55.

[116] Antonio Torralba and Alexei A Efros. 2011. Unbiased look at dataset bias. In *CVPR 2011*. IEEE, 1521–1528.

[117] Mohammad Moeen Valipoor and Angélica de Antonio. 2022. Recent trends in computer vision-driven scene understanding for VI/blind users: a systematic mapping. *Universal Access in the Information Society* (2022), 1–23.

[118] Elmira van den Broek, Anastasia Sergeeva, and Marleen Huysman. 2021. WHEN THE MACHINE MEETS THE EXPERT: AN ETHNOGRAPHY OF DEVELOPING AI FOR HIRING. *MIS Quarterly* 45, 3 (2021).

[119] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).

[120] Sahil Verma and Julia Rubin. 2018. Fairness definitions explained. In *2018 ieee/acm international workshop on software fairness (fairware)*. IEEE, 1–7.

[121] Anneliese von Mayrhauser and A Marie Vans. 1997. Program understanding behavior during debugging of large scale software. In *Papers presented at the seventh workshop on Empirical studies of programmers*. 157–179.

[122] Zhiyuan Wan, Xin Xia, David Lo, and Gail C Murphy. 2019. How does machine learning change software development practices? *IEEE Transactions on Software Engineering* 47, 9 (2019), 1857–1871.

[123] Angelina Wang, Alexander Liu, Ryan Zhang, Anat Kleiman, Leslie Kim, Dora Zhao, Iroha Shirai, Arvind Narayanan, and Olga Russakovsky. 2022. REVISE: A tool for measuring and mitigating bias in visual datasets. *International Journal of Computer Vision* (2022), 1–21.

[124] Dakuo Wang, Elizabeth Churchill, Pattie Maes, Xiangmin Fan, Ben Shneiderman, Yuanchun Shi, and Qianying Wang. 2020. From human-human collaboration to Human-AI collaboration: Designing AI systems that can work together with people. In *Extended abstracts of the 2020 CHI conference on human factors in computing systems*. 1–6.

[125] Thomas Way, Mary-Angela Papalaskari, Lillian Cassel, Paula Matuszek, Carol Weiss, and Yamini Praveena Tella. 2017. Machine learning modules for all disciplines. In *Proceedings of the 2017 acm conference on innovation and technology in computer science education*. 84–85.

[126] James Wexler, Mahima Pushkarna, Tolga Bolukbasi, Martin Wattenberg, Fernanda Viégas, and Jimbo Wilson. 2019. The what-if tool: Interactive probing of machine learning models. *IEEE transactions on visualization and computer graphics* 26, 1 (2019), 56–65.

[127] Yao Xie, Melody Chen, David Kao, Ge Gao, and Xiang'Anthony' Chen. 2020. CheXplain: Enabling Physicians to Explore and Understand Data-Driven, AI-Enabled Medical Imaging Analysis. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–13.

[128] Qian Yang, Jina Suh, Nan-Chen Chen, and Gonzalo Ramos. 2018. Grounding interactive machine learning tool design in how non-experts actually build models. In *Proceedings of the 2018 designing interactive systems conference*. 573–584.

[129] Alexey Zagalsky, Dov Te'eni, Inbal Yahav, David G Schwartz, Gahl Silverman, Daniel Cohen, Yossi Mann, and Dafna Lewinsky. 2021. The design of reciprocal learning between human and artificial intelligence. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021), 1–36.

[130] Amy X Zhang, Michael Muller, and Dakuo Wang. 2020. How do data science workers collaborate? roles, workflows, and tools. *Proceedings of the ACM on Human-Computer Interaction* 4, CSCW1 (2020), 1–23.

[131] Jiawei Zhang, Yang Wang, Piero Molino, Lezhi Li, and David S Ebert. 2018. Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models. *IEEE transactions on visualization and computer graphics* 25, 1 (2018), 364–373.

[132] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* (2020).

[133] Qiaoning Zhang, Matthew L Lee, and Scott Carter. 2022. You Complete Me: Human-AI Teams and Complementary Expertise. In *CHI Conference on Human Factors in Computing Systems*. 1–28.

[134] Ru Zhang, Wencong Xiao, Hongyu Zhang, Yu Liu, Haoxiang Lin, and Mao Yang. 2020. An empirical study on program failures of deep learning jobs. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 1159–1170.

[135] Wencan Zhang and Brian Y Lim. 2022. Towards Relatable Explainable AI with the Perceptual Process. In *CHI Conference on Human Factors in Computing Systems*. 1–24.

[136] Zijian Zhang, Jaspreet Singh, Ujwal Gadiraju, and Avishek Anand. 2019. Dissonance between human and machine understanding. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (2019), 1–23.

[137] Peng Zhao, Yu-Jie Zhang, and Zhi-Hua Zhou. 2021. Exploratory machine learning with unknown unknowns. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 10999–11006.

A APPENDIX

A.1 Research method

Figure 4 and Figure 5 respectively present example explanations and the workflow template shown to the participants during the semi-structured interviews.

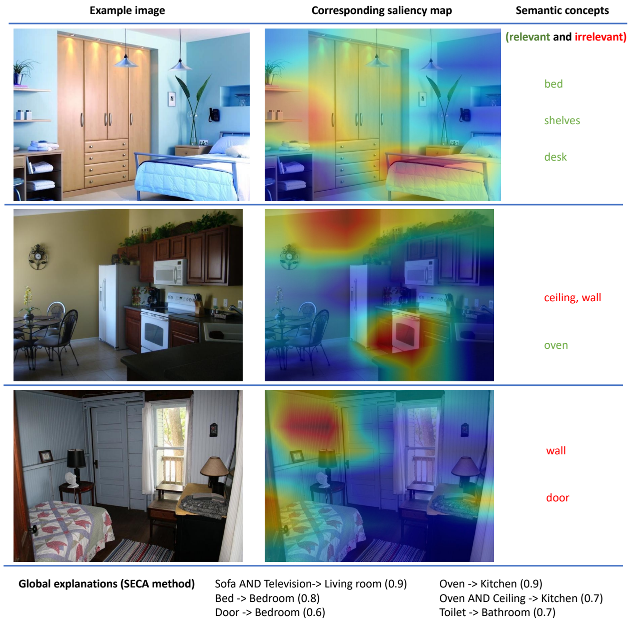

Example image
Corresponding saliency map
Semantic concepts
(relevant and irrelevant)
bed
shelves
desk
ceiling, wall
oven
wall
door
Global explanations (SECA method)
Sofa AND Television-> Living room (0.9)
Bed -> Bedroom (0.8)
Door -> Bedroom (0.6)
Oven -> Kitchen (0.9)
Oven AND Ceiling -> Kitchen (0.7)
Toilet -> Bathroom (0.7)

Figure 4: Example explanations (local visual and textual explanations, and global textual explanations) showed to the participants, when they would mention them, or at the end of the interviews to trigger further reflections about them.

A.2 Additional results

A.2.1 *Correcting bugs to solve the failures.*

The participants used one of four strategies (followed by model retraining) to correct bugs, depending on the bugs, and on their familiarity with the models.

Dataset transformations. Participants with no experience in explainability and experts who do not wish to engage deeply with the data content tried to resolve correctness failures through *typical data augmentation* methods such as applying mirroring, rotation and colour contrast algorithms. *"I would employ some augmentation techniques or artificial data to see if I can get away with this. This would also be a method to further regularize the model to make sure that it's not overfitting."* P3 high-CV. P5 low-CV also mentioned *"applying some dirty labels (for instance I would apply the label of "kitchen" to "dining room" pictures) to create a positive perturbation and rebalance the number of samples"*.

Other participants mentioned *feature-specific transformations*: adding or removing images with specific features, or obfuscating irrelevant information from images. *"If there are cats only at dining rooms, I should do cat recognition and mask them."* P11 mid-CV. The hypotheses of these participants revolved around the relevance of the model features, and/or the existence of unknown unknowns. Transformations of the *data engineering process* were also mentioned by some experts as simple steps (e.g. increasing image size, changing scaling) along model modifications.

Modifications of the training parameters. Expert participants transformed the *loss function* to penalize classes with higher error rates: *"It is easier that the model learns to base the classification on different things than when you add more data"* P9 high-CV. They also gave more *importance to training samples* erroneously predicted *"It's like the Bootstrap algorithm where you keep re-feeding falsely predicted samples into the model, assigning higher weights for the last computation."* P3 high-CV. This method is used by participants with computer vision experience, as they are more familiar with the functioning of the models. *"It allows me to avoid using a parameter so that the classification of two classes becomes more diverse, and the optimization of the training based on a more relaxed representation."* P14 high-CV. A few participants with some experience also discussed tuning *training hyperparameters. "What I found is that setting the right parameters, especially learning rate or batch size, can help the model avoid certain biases"* P7 mid-CV.

Model transformations. Hyperparameter tuning (e.g. changing the model architecture) was the main solution of high-CV experience participants, which sometimes came hand in hand with simple dataset transformations. *"The network didn't learn the task. It's the famous bias variance. You have to see whether it cannot generalize well, which means that it has been overfitted to the training set. If you have a lot of data available, you just throw more data at your model hoping that it can generalize better. If the data is scarce, let's say you are in medical imaging and each MRI is from a patient, you cannot collect more data. You have to change your model and that's more expensive because a machine learning expert needs to work on it. Instead, for data, you can just crowd source it via Amazon Mechanical Turk, it's much cheaper. There are also scientific insights: if the task is simple, adding more complex model doesn't make sense, but usually for computer vision task, it's complex enough that you can have a complex model."* P16 high-CV. Low-CV participants did not engage in such activity as they were not familiar with the functioning of computer vision models *"That's where I'm hitting a wall. I would change something about the model. But I would need to understand that model a little better."* P4 low-CV.

Changes in the model setup. Certain participants with low-CV experience proposed additional solutions based on their own experience. These solutions are not mentioned in literature, but useful in practice. They would a) build separate models for the most confused classes, b) create additional classes for the ones that are too diverse in terms of image content, or c) append a rule-based model, to correct inferences with heuristics defined on the content of the images. *"Establishing rules means to modify the model decisions manually. It's not something that you should do, but if it's a requirement, it can be done. Let's say this is towards 60% confidence, it's a weak prediction. The probability of being a dining room is lower than average. So, once you have the combination of low probability of being a dining room and you also have the presence of a metal component intertwined with black glasses, then you can push it to the kitchen classification."* P8 low-CV. d) Others proposed to engineer features based on visual information identified in the images *"Most bathrooms have a*

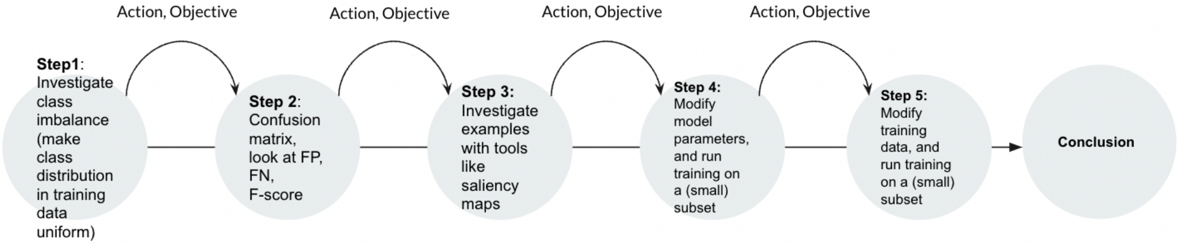

Action, Objective
Action, Objective
Action, Objective
Action, Objective
Step1: Investigate class imbalance (make class distribution in training data uniform)
Step 2: Confusion matrix, look at FP, FN, F-score
Step 3: Investigate examples with tools like saliency maps
Step 4: Modify model parameters, and run training on a (small) subset
Step 5: Modify training data, and run training on a (small) subset
Conclusion

Figure 5: Example template provided with the design brief, and filled in by one participant. The template shows empty circles and arrows representing objectives, actions, and transition triggers, reflecting each step of the failure handling process, and helping the participants to structure their thoughts.

mirror, then it's really good if we can classify if there's any mirror. From specific elements that you discover, you arrange other features." P8 low-CV. e) One participant mentioned deferring difficult cases to humans, or using active learning to fine-tune the model. *"The way to proceed is through the human eye: you leave extreme cases to workers to annotate. The model can learn about the general cases and leave you the extreme ones."* P7 mid-CV.

A.2.2 Collaboration between stakeholders for handling failures.

Results. As it appeared along the previous subsections, for most participants, failure handling was not a lonely process. Practitioners frequently mentioned communicating with other stakeholders during the process.

- *With other "developers".* The practitioners often need to discuss with other individuals who took part in the model development process, dataset creation, etc. to obtain more information about choices and previous experiments. Especially, expert practitioners implicitly had a list of steps they always perform when developing a model (e.g., training with different architectures and hyperparameters), and a list of necessary operations (e.g., normalization and standardization of the dataset, data augmentation, etc.) (P3 high-CV) *"I suppose that the input has been sufficiently preprocessed? I would normalize, typically by the max value if we are talking about standard RGB images."*
- *With model requesters.* To clarify when the model is satisfying, the practitioners also rely on the model requesters (subsection 4.2) who are the final judges of the acceptability of the model (and the requirement providers) (P14 high-CV) *"the final decision on how much you should improve the model is given by somebody else (the client, the model owner, ...) given whether it is a critical situation."*
- *With domain experts.* Domain experts are involved by the practitioners (when reachable) to better understand the target task and potential pitfalls, and to judge how ready the model is, to identify feature expectations, and to reason on the relevance of certain features when searching for model bugs and feature failures (P14 high-CV) *"the part of saying whether it's ok that the model makes a specific mistake, it's not up to me. It's up to the experts."* P7 mid-CV also mentioned questioning the experts who are the end-users of their model to resolve data ambiguities, whether they are inherently ambiguous, or whether one specific class can be attributed to the samples (P7 mid-CV) *"Give it to people who are as close as possible to the end-users and say: what do you think? Is this a bedroom or a living room?"*
- *With potential end-users.* The developers have to convince the model requesters and users (who are often the experts) of the validity of the models. P14 high-CV for instance explained *"You are the person that can communicate the density of information to a specialist like a doctor. When we have a meeting, we show the model understood the class."*

Implications. Our results identify additional communication needs from the developer to non-developers, especially for defining when a model is suitable for deployment, whether specific failures on single samples are acceptable, and which features one should expect [47, 88, 118]. Since the accessibility of domain experts was one of the main problems for the developers, research should investigate how to facilitate collaborations around these specific concepts, potentially with the development of remote, asynchronous tools, and common languages (possibly inspired from existing knowledge elicitation methods [56]), e.g., to indicate relevant features. Existing works that facilitate the cooperation between domain experts or end-users, and a machine learning model, could be adapted to these specific concepts [20, 63, 129].