How can Explainability Methods be Used to Support Bug Identification in Computer Vision Models?

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

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

Christoph Lofi
Delft University of Technology
Netherlands
c.lofi@tudelft.nl

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

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

ABSTRACT

Deep learning models for image classification suffer from dangerous issues often discovered after deployment. The process of identifying bugs that cause these issues remains limited and understudied. Especially, explainability methods are often presented as obvious tools for bug identification. Yet, the current practice lacks an understanding of what kind of explanations can best support the different steps of the bug identification process, and how practitioners could interact with those explanations. Through a formative study and an iterative co-creation process, we build an interactive design probe providing various potentially relevant explainability functionalities, integrated into interfaces that allow for flexible workflows. Using the probe, we perform 18 user-studies with a diverse set of machine learning practitioners. Two-thirds of the practitioners engage in successful bug identification. They use multiple types of explanations, e.g. visual and textual ones, through non-standardized sequences of interactions including queries and exploration. Our results highlight the need for interactive, guiding, interfaces with diverse explanations, shedding light on future research directions.

CCS CONCEPTS

• **Human-centered computing** → **User interface programming**; *Empirical studies in HCI*; • **Computing methodologies** → **Computer vision**; • **Software and its engineering** → **Software testing and debugging**.

KEYWORDS

computer vision, machine learning model debugging, machine learning explainability, user interface

ACM Reference Format:
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 (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3491102.3517474

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9157-3/22/04.
https://doi.org/10.1145/3491102.3517474

1 INTRODUCTION

Safely using deep learning models still proves challenging for many computer vision applications. Models suffer from spurious correlations, brittleness, or overfitting, producing erroneous predictions and safety risks[1] or societal harms [26]. The fact that these issues are often discovered only after deployment illustrates the existence of challenges in the practice of identifying and mitigating bugs in the models early. Yet, limited effort has been devoted to investigating the debugging practices of computer vision practitioners. The machine learning community develops various explainability methods, often arguing their usefulness for model bugs identification [8, 14, 27, 40, 47]. However, few studies investigate their concrete uses in this process. As a result, it is still unclear what types of explanations (e.g. out-of-domain, global, or interactive [46]) can be useful, for which steps of the process, and how.

In this work, we ask: *how could diverse explainability methods be used to support the bug identification process of deep learning computer vision models?* We focus on image classification tasks, as they are prone to model misbehavior, and there is an established body of (post-hoc) explainability methods to possibly support bug identification. Their practical use has not been studied yet, contrary to the ones for tasks that rely on tabular data [20]. We study the identification of model failures and bugs in development; later steps like bug identification in deployment and bug correction are future work. We draw inspiration from works situated at the intersection of machine learning and HCI that investigate how machine learning or related tools (e.g. explainability, debugging user-interfaces, etc.) are used [35], or could be used [20], and how to design them [4, 52]. We build a design probe[2] in the shape of a user-interface by performing literature studies, a formative study, and co-creation sessions consisting of 18 interviews, to explore uses of explainability for debugging. Using an implemented probe and a carefully-crafted use-case, we then perform 18 user-studies with machine learning practitioners having different levels of experience with computer vision in various domains.

[1]https://www.autoweek.com/news/green-cars/a37114603/tesla-fsd-mistakes-moon-for-traffic-light/
[2]The code implementation can be found at https://github.com/agathe-balayn/explainability_probe.

The user-studies show that a wide range of explanations are useful to identify bugs (e.g. both textual and visual explanations, global and local, companion with domain knowledge, etc.). These explanations are often not theory-heavy, but extremely informative when embedded into an interactive interface. Although they can sometimes be overwhelming and misinterpreted (leading to identify wrong bugs due to confirmation bias), these explanations also allow to identify the potential causes of various issues, and to envision correction strategies. This reveals an urgent need for more research on the design of new explanations relying on diverse user-interactions adapted to different kinds of practitioners.

The paper is organised as follows: we introduce related works in section 2, the probe and its rationale in sections 3, 4. We outline the user-study setup in section 5, and its results in section 6. We discuss implications in section 7, and conclude in section 8.

2 RELATED WORKS

2.1 Bug identification in software and machine learning models

To understand what bug identification means, we survey literature about machine learning testing (first step of the debugging process) and traditional software systems where bug identification is more extensively studied.

Failures. Machine learning testing aims at detecting and characterizing differences between current and expected functioning of a model [54]. These differences revolve around inferences (e.g. correctness, robustness, fairness, etc.), data, or code [11, 38]. Main causes of failures are structural or training bugs [34]. Our formative study reveals sub-types of training bugs around datasets or training hyperparameters. We mainly focus on correctness failures (wrong model inferences or features) and dataset bugs, especially in relation to issues in the model features, as these are still overlooked research-wise despite being the primary debugging goal of practitioners and directly related to explainability methods. Software engineering distinguishes between *reactive debugging* (a failure is explicitly identified) [5, 16]; *proactive debugging* (no explicit failure manifests); and general *software understanding* (for later debugging) [28, 50], that we all study.

Methods. The *software* debugging workflow consists of four steps [5, 28, 50]: 1) gathering context and hypothesis formulation, 2) instrumenting the hypothesis, 3) testing the hypothesis, 4) correcting the hypothesis, or applying a bug solution. To the best of our knowledge, there is no study of the bug identification *practices* for computer vision models. Instead, research focuses on developing *methods* for debugging models [24, 33, 34, 54] without any human activity or explainability (except [44]). As our formative study shows that none of the automatic methods is employed by practitioners, we investigate how practitioners could perform manual bug identification supported by explainability.

User interfaces. A few user interfaces [39, 51, 53] support developers in debugging models. None of the ones that make use of explainability methods are adapted to computer vision. The applicable ones all focus on investigating the choice of model and training hyperparameters [41, 42], or visualising the data used to train the model [3]. Our design probe instead presents diverse explanation artifacts designed for computer vision models.

2.2 Machine learning explainability

Explainability provides explanations on the functioning of a model. A framework [47] characterizing explainability works identifies model debugging as one of their purposes, and the following tasks developers might perform, e.g. "assessing reliability of a prediction", "detecting arbitrary behavior", etc, that our study also identifies.

Categorization. Explainability methods and resulting explanations can be categorized in various ways [6, 30, 31, 46]. One might want to differentiate them regarding the explanation audience, the explanatory medium, the explanation scope, whether the explanations are about data or models, their faithfulness, etc. Algorithmic research distinguishes between *local* or *global* explanations, depending on the scope of data samples employed. Local explanations provide information on the reasoning a model follows to infer the label of a sample, through saliency maps [43], visual counterfactuals [15], or visualisations of activation layers [21, 36]. Global explanations indicate the general features used by a model, presented as visual hints (e.g. TCAV [27], ACE [14]), or textual information (e.g. SECA [8]). We use these categorizations to identify the explanations relevant to include in our probe.

Usages. Researchers have conducted user-studies on the use of certain explanations for certain stakeholders and data types [2, 12, 13, 23, 25, 52]. Yet, no extensive work involves *developers* debugging *computer vision* models. Only Bhatt et al. [10] conducted inquiry interviews where developers solely reported using saliency maps to understand wrong inferences, or to identify spurious features, and none mentioned other explainability methods. Our work performs a human-grounded exploration where we collect developers' practices based on carefully-crafted debugging tasks.

3 PROBE DESIGN PROCESS

The goal of our probe is to explore potential uses of explainability methods for bug identification. Design probes have three fine-grained goals [22]: "*social science* goal of collecting information about the use and the users of the technology in a real world setting, *engineering goal* of field-testing the technology, and *design goal* of inspiring users and designers to think of new kinds of technology to support their needs". Table 1 describes the requirements for our probe.

Index	Requirement	Description
Rq1	Completeness of functionalities for bug identification	The probe should present the main information a developer might look at when debugging.
Rq2	Completeness of explanations	The probe should offer the main available types of explanations for computer vision models.
Rq3	Clarity of the presented information	To proceed to valid user-studies, the participants should understand clearly the functionalities presented to them, without being overwhelmed by the offered information.
Rq4	Flexibility and objective presentation of the information	The probe should not enforce a certain workflow within the tool not to skew participants' behaviors towards certain explanations, but instead make the interactions as free as possible.

Table 1: List of requirements defined for the probe.

3.1 Mixed method research

To translate our requirements into a probe, we establish *functionalities* (Fx) to provide, and *orthogonal categories* (Ox) that indicate how these functionalities can be realized. Academic and grey *literature* analyses inform the list of explanations the probe should contain (Rq2). However, it does not focus on practitioners' experiences, and thus does not inform on other information needed to identify bugs (Rq1). Consequently, we perform a *formative study* in the shape of 18 semi-structured interviews with practitioners where we investigate their practices, challenges, and wishes. We synthesize the literature and the insights from the study to extract the functionalities (Fx) and orthogonal categories (Ox) (Table 2). Finally, we perform iterative *co-creation sessions* where we present designs of the probe to practitioners, and collect feedback to fine-tune information visualisations, and identify the minimum set of necessary interactions with these functionalities (Rq3, Rq4). The probe is then implemented so as to create a valuable user-experience (Rq5).

Concretely, in the formative study and co-creation sessions, we present to the participants a use-case involving the development of a deep learning model for a scene classification task. We describe an initial model that has been (hypothetically) built, and show example of images from the training dataset with their ground truth and model inferences. We make sure these examples present both cases where the model makes right and wrong inferences, using relevant and irrelevant features (same approach as in section 5). For the formative study, we then ask the participants to describe the approach they would follow to define whether this model is ready for deployment, and if not, to characterize what the exact model failures to solve are. We analyse the results of such sessions by extracting intermediate goals (in the shape of questions in Table 2) the participants have while investigating the model, and types of information and tools they use to fulfill these. For the co-creation sessions, we ask the same questions. Yet, we additionally present the participants with mock-up user interfaces containing various types of explanations, and prompt them to envision how they would use such interfaces to answer the questions. We also ask them for feedback on the interfaces (e.g. missing, irrelevant, or unclear functionalities), and we iterate on the interfaces after each interview, going initially from low-fidelity mockups, to high-fidelity ones in the last interviews.

3.2 Probe functionalities

We elicited the functionalities below needed for the probe. We illustrate them with the scenario of the user-studies (section 5): building a model that classifies the species of a bird displayed in an image. Importantly, our participants often referred to semantic concepts in relation to relevant sample pixels or potential human-interpretable model features, to reason about potential failures and bugs. These concepts were either entities (e.g. `cactus`), attributes (e.g. `green`), entity-attribute combinations (e.g. `green-cactus`), or their logical negation (e.g. `NOT cactus`, i.e. absence of cactus).

- **F1: performance understanding**: Understand overall and class-specific performance of the model. Looking at metrics gives a first indication of the performance of the model, and the type of errors to investigate. Participants use the class-specific metrics to decide for which types of samples to improve the model first.
- **F2: data-neighbor exploration**: Understand and compare the content of data samples. Participants regularly explore the data to estimate the complexity of the task, to reason about causes for model failures, and identify features of the model. F1 and F2 can be supported with information about performance metrics and datasets, without explanations.
- **F3: local explanations**: Understand how the model made an inference for a sample. Participants scrutinize or wish to scrutinize saliency maps, to detect overfitting, or to judge the relevance of model features. This can be facilitated with explanations of single samples that show the connection between the sample content and its label (e.g. the model classified this image as `gila woodpecker` by looking at the pixels of the `cactus` the bird is standing on Figure 4).
- **F4: global explanations**: Identify the main reasons for the model to classify samples into this class. Participants progressively achieve a global understanding of the model by formulating a hypothesis based on a single sample, and iterate on it by evaluating its validity across more samples. Some participants wished to have statistical summaries of visual concepts across images (e.g. for 80% images classified as `gila woodpecker`, the model looked at pixels representing a `cactus`) to speedup their process and improve its results. For that, some participants suggested using crowdsourcing or object detectors to annotate images at scale (similarly to what the SECA method offers [8]).
- **F5: explanation comparison**: Compare the reasoning of the model across samples or classes. Comparisons serve to judge the validity of feature hypotheses, and to understand misclassifications (e.g. the model classified this `gila woodpecker` image correctly using the `cactus` pixels, but that one incorrectly using the `wings`).
- **F6: explanation importance**: Rank the explanations based on their frequency, or on the type of (in)correct inferences they lead to. A few participants mentioned that it would be convenient to automatically obtain a list of the most important features for the model. We foresee they might want to query and rank explanations according to different properties such as explanations that lead to correct or incorrect inferences (e.g. 20% of times when the model used the `breast` and `belly` pixels, it made a correct prediction, contrary to 90% for the `cactus` pixels).
- **F7: counterfactuals**: Ask "what-if" questions to see the type of reasoning and inference class received by a sample with/out these visual concepts. This family of explanation was not directly mentioned as counterfactual, yet a few participants mentioned testing transformations of images based on certain concepts to understand how they impact the inferences (e.g. what would the model predict if there was `no cactus` in the image?). As setting up such transformations is complex (Rq5), we propose proxy textual-explanation based transformations (subsubsection 4.2.2).
- **F8: explanation recommendation**: Visualise explanations, or search for specific ones. While participants do not talk about this as they are used to search for local explanations by themselves, being able to query specific explanations might speed up their process. This is also connected to the complexity of an explanation method. As participants did not know about many explanations, they did not reflect further on their complexity.

Topic	Provenance	Description	Fx	Ox
Model input	Interviews	What kind of data does the system learn from?	F2	-
		To what extent the data is diverse enough to represent each class?	F2, F9	-
		What are the differences between these two classes?	F2, F5, F8	-
Performance	Interviews	How well does the model perform for each class? Errors with high or low confidence?	F1	-
Exp. breadth/scope	Both	The extent to which an explanation can be generalised [46]. Participants of our formative study use a larger range of breadth than local and global.	F3, F4, F8	O1
Local	Both	What features of this instance lead to this inference?	F3	
Global	Both	Which visual elements does the model generally use?	F4	
Intermediate	Interviews	What are the features used to distinguish these 2 classes?	F3, F4, F8	
Comparisons	Both	[20] insists on allowing comparisons of explanations across samples. Participants of our formative study performed comparisons across samples but also across classes.	F5, F6	O1, O2, O3
		Why are instances A and B given the same/different predictions?	F5, F3	
		How does the model weigh different features?	F6, F3, F4	
Exp. family	Literature	Sokol et al. [46] discuss a) associations between antecedent and consequent, b) contrasts and differences (using examples), c) causal mechanisms, as potentially used types of explanations. Our participants primarily relied on b), some also hinted at a) and c).	-	O3
Associations		The local and global explanations mentioned above primarily refer to a).	F3, F4	-
Contrasts		The comparisons performed with these explanations refer to b).	F5, F6	-
Causal mechanisms		Why is this sample predicted P instead of Q? What would the model predict if this sample is changed to ...?	F7	-
Exp. domain /medium	Both	A mixed domain approach consists in explanations within the original domain of inputs (images), and in a transformed domain (essentially text such as in dialogues [9]). A few participants hinted at the potential usefulness of having textual explanations.	-	O2
Interactivity /passivity	Literature	[46] distinguishes between static and interactive explanations. While most explainability works do not address interactivity, some [8, 14, 20, 27] propose query interactions.	F8	-
		This connects to varying the breadth and domain of explanations, performing various types of comparisons, and exploring questions around causal mechanisms.	F5, F6, F7, F8	O1, O2, O3
	Interviews	Does the model use this feature?	F8	-
Domain kno.	Interviews	What features do we expect the model to learn for this class?	F9	-
		Should the model pick up on more visual elements for this image/class?	F9, F3, F4	-

Table 2: Summary of observations from the literature and formative study (main questions practitioners ask themselves within the bug identification process), and their mapping to the probe functionalities (Fx) and orthogonal categories (Ox).

Yet, they might want to delve deeper into the parameters of explainability methods once they are more familiar with them.

- **F9: domain expertise**: Know what a domain expert (e.g. an ornithologist) would consider good reasons to classify a sample into a class. This functionality was contentious among participants. A few participants did not use domain knowledge explicitly but still relied on their understanding of the domain to understand potential wrong features of the model, while others advocated for the necessity of understanding the domain even before looking into the model.

We identify the following orthogonal categories:

- **O1: breadth**: While literature refers to local and global explanations as the two scopes of explanations, we see them as the two extremes of a scale. The participants did not always look into a single sample (local) or the overall set of data and inferences (global), but focused on various sets of classes (e.g. two classes or entire dataset), or samples with correct or incorrect inferences.
- **O2: medium**: Participants are more accustomed to image-based explanations. Yet, a few participants insisted on getting textual explanations to more easily receive feedback from domain experts on feature relevance, or to query learned features or images of the training dataset.
- **O3: granularity/type**: Participants typically reasoned about semantic concepts to identify issues with model features. Certain participants varied the granularity of these concepts and went to fine-granularities when they could not identify a pattern of reasoning within higher-granularity ones (e.g. `entire wing` or `sub-parts` with different colors).

4 RESULTING DESIGN PROBE

We first describe the main types of explanations in the probe (F3, F4, O1 - O3), corresponding to the basic required functionalities. We then explain how we organized these functionalities into a set of interactive tabs, to fulfill the other functionality requirements (F1, F2, F5 - F9). In our user-study (section 5), practitioners will use the probe to debug a model for the bird classification scenario.

4.1 Materialisation of the probe explanations

*4.1.1 Local explanations (**F3**).* To vary the medium of explanations (**O2**), we provide both visual ones (saliency maps Figure 2 (5a)) and textual ones (semantic features (5b)). We opted for SmoothGrad to retrieve the saliency maps [45]. This method is sensitive to the parameters of a model while minimising noisy results, catering for more accurate capturing of a model behaviour. The semantic features are retrieved as by-products of applying the global explainability method.

*4.1.2 Global explanations (**F4**).* We choose the SECA framework [8] to extract explanations that reflect the overall features of a model. It provides more complete explanations than ACE [14], it is more tractable than TCAV [27] in an interactive mode (Rq5), and provides textual explanations that participants wished for.

SECA takes as input images from each class of the dataset (we choose a balanced, random set of samples of the test dataset). It extracts the corresponding saliency maps and has them annotated by crowd workers. Then, it reconciles the annotations, and transforms

Score	Example
Overall explanations	
Percentage of times the features are used by the model within the dataset.	If 100 images are in the dataset, and the model used the feature "cactus" in 20 of them, then the score is 20%.
Percentage of times the features led to a correct prediction across all images for which the model used the features.	In the 20 previous images, if the model made 5 correct predictions, the score is $5 * 100/20 = 25\%$.
Typicality score (from SECA): correlation between the presence of the features and the predicted classes, i.e. how strongly the features serve to distinguish one class from the others.	
Class-specific explanations	
Percentage of images that contain the features of interest among all images with the predicted class.	If 100 images were predicted to be a gila woodpecker, and 20 of these images have a cactus, then the score for *cactus* $\longrightarrow$ *gila woodpecker* is 20%.
Percentage of images that received a correct prediction among images that contain the features and have this predicted class.	Among the 20 previous images, if 5 images were indeed gila woodpecker images, then the score is of 25%.
Typicality score (from SECA): indicates how strongly the features serves to distinguish the specific class from the others.	See above example for typicality.

Table 3: Overview of the scores in the global explanations.

them into a table of semantic features. Post-processing techniques (e.g. rule mining) finally identify combinations (logical conjunctions or disjunctions) of features (entities and/or attributes) highly correlated with certain predicted classes. This approach provides explanations at different levels of granularity (e.g. `wing`, or `primary coverts, alula`, etc.) depending on the granularity of the annotations requested to the annotators (**O3**). The feature combinations are accompanied with six scores (see Table 3) referring to overall explanations and class-specific ones. Overall explanations represent the primary features used by the model to distinguish between classes (see Figure 1 (3)). In class-specific explanations, the combinations of features are associated to a specific class (see Figure 1 (4)), and the scores indicate the relevance of these features to this class. We represent these scores in bar plots for easy comparisons, as a result of multiple iterations where participants indicated the difficulty in making use of numbers (Rq3).

The user can rank (**F6**) or filter the global explanations according to the scores (Figure 1 (5)). To vary the explanation scope **(O1)**, one can compute the scores on various data subsets: (1) entire dataset: explains the general inference mechanisms a model follows; (2) samples that received a (in)correct prediction: identifies and compares mechanisms for such predictions; (3) subset of the classes: identifies features used to distinguish between these classes. Where these choices can be made, we setup default parameters to reduce the complexity of the probe understanding (Rq3).

4.2 The tab structure of the probe

To avoid skewing the participants towards a particular workflow (Rq4), we organize the primary functionalities into tabs, making them independent and equally important. In the user-study, we inform the participants that there is no sequential dependency between the tabs. The tabs allow us to provide the higher-level explainability functionalities (**F5 - F7**), as well as the other necessary information about the model (**F1, F2, F9**). **F8** (recommendation) is provided all along the probe through the various parameters to choose as well as the query tab.

4.2.1 *Wiki tab (**F9**).*

This tab displays the domain knowledge about each dataset class, that an expert typically possesses. It indicates relevant and irrelevant features for recognizing an image class.

4.2.2 *Query tab (**F7**).*

This tab (Figure 1 (1)) allows to query global and local explanations, and images with specific visual content, their predictions, explanations, and ground truth, allowing to a certain extent to answer what-if questions.

The user is presented with text fields to fill in with features of interest, types of logical combinations, and/or class. They choose to query explanations within all images, or only in the correctly or incorrectly classified ones, or within specific classes (**O1**). The results are displayed underneath. These can be a) scores of a queried explanation, b) distribution of the presence of the queried features across the dataset, or c) samples associated to the local query. b) is displayed in a confusion matrix-like table (Figure 1 (2)) that shows, per cell, the percentage of images that have the features among the images of a cell, and the percentage of cell images that have the features among all images that have this feature.

4.2.3 *Confusion matrix tab (**F1**).*

This tab shows the accuracy and F1-score of the model, and its confusion matrix (see Figure 2 (1)). Each cell presents two rates. One (bottom) is computed over the entire dataset similarly to any confusion matrix. The other (top) is computed over the data of a single row corresponding to the precision or recall per class depending on what the rows and columns encode (ground truth or prediction). One can transpose these.

Users can click on the matrix cells to open a new page with the corresponding images (**F2**), as well as local and global explanations (**F3, F4**). The images and local explanations (Figure 2 (2)) are organized into four columns corresponding to the 4 cells of the matrix associated to the classes clicked initially, i.e. the ground truth A and predicted class B of the initial cell (A-B), as well as the corresponding diagonal cells of the two classes A-A and B-B, and the opposite cell that would invert the ground truth and prediction classes (B-A). This allows to compare these explanations (**F3, F5**). Clicking on an image or saliency map allows to zoom on it, and its related textual, local explanations (Figure 2 (5)).

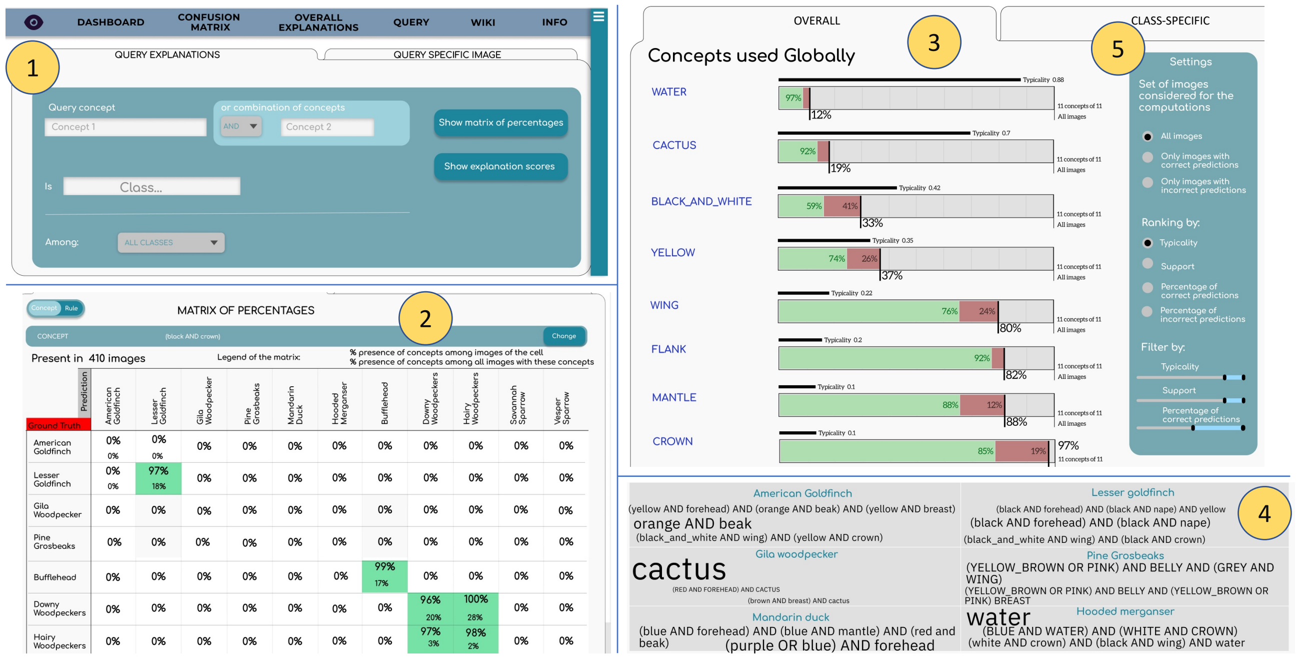
The global explanations corresponding to the four cells (equivalent to considering a binary classification task involving classes A and B Figure 2 (3)) are also displayed in lists allowing for their comparisons (Figure 2 (4)) (**F4, F5**).

4.2.4 *Global explanation tab (**F4, F5, F6**).*

This tab displays the global explanations computed over the entire dataset. It shows both the overall and class-specific ones (respectively Figure 1 (3), (4)).

4.2.5 *Dashboard tab.*

A few participants from the co-creation sessions wished to see all the main functionalities on a unique page. The dashboard tab does so. Its top left part provides the performance functionalities (**F1, F2**), and the top right the corresponding local explanations (**F3**). At the bottom, the query functionality is enabled (**F7**). This organisation lets users explore explanations for different images, and compare these with additional queried information (**F5, F6, F8**).



MATRIX OF PERCENTAGES

CONCEPT (black AND crown)

Present in 410 images

Legend of the matrix: % presence of concepts among images of the cell / % presence of concepts among all images with these concepts

Ground Truth \ Prediction	American Goldfinch	Lesser Goldfinch	Gila Woodpecker	Pine Grosbeaks	Mandarin Duck	Hooded Merganser	Bufflehead	Downy Woodpeckers	Hairy Woodpeckers	Savannah Sparrow	Vesper Sparrow
American Goldfinch	0% 0%	0% 0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Lesser Goldfinch	0% 0%	97% 18%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Gila Woodpecker	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Pine Grosbeaks	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%	0%
Bufflehead	0%	0%	0%	0%	0%	0%	99% 17%	0%	0%	0%	0%
Downy Woodpeckers	0%	0%	0%	0%	0%	0%	0%	96% 20%	100% 28%	0%	0%
Hairy Woodpeckers	0%	0%	0%	0%	0%	0%	0%	97% 3%	98% 2%	0%	0%

Concepts used Globally

Concept	Typicality	Correct	Incorrect	Marker	Count	Images
WATER	0.88	97%		12%	11 concepts of 11	All images
CACTUS	0.7	92%		19%	11 concepts of 11	All images
BLACK_AND_WHITE	0.42	59%	41%	33%	11 concepts of 11	All images
YELLOW	0.35	74%	26%	37%	11 concepts of 11	All images
WING	0.22	76%	24%	80%	11 concepts of 11	All images
FLANK	0.2	92%		82%	11 concepts of 11	All images
MANTLE	0.1	88%	12%	88%	11 concepts of 11	All images
CROWN	0.1	85%	19%	97%	11 concepts of 11	

Settings

Set of images considered for the computations
- [✓ All images]
- Only images with correct predictions
- Only images with incorrect predictions

Ranking by:
- [✓ Typicality]
- Support
- Percentage of correct predictions
- Percentage of incorrect predictions

Filter by:
- Typicality
- Support
- Percentage of correct predictions

American Goldfinch
(yellow AND forehead) AND (orange AND beak) AND (yellow AND breast)
orange AND beak
(black_and_white AND wing) AND (yellow AND crown)

Lesser goldfinch
(black AND forehead) AND (black AND nape) AND yellow
(black AND forehead) AND (black AND nape)
(black_and_white AND wing) AND (black AND crown)

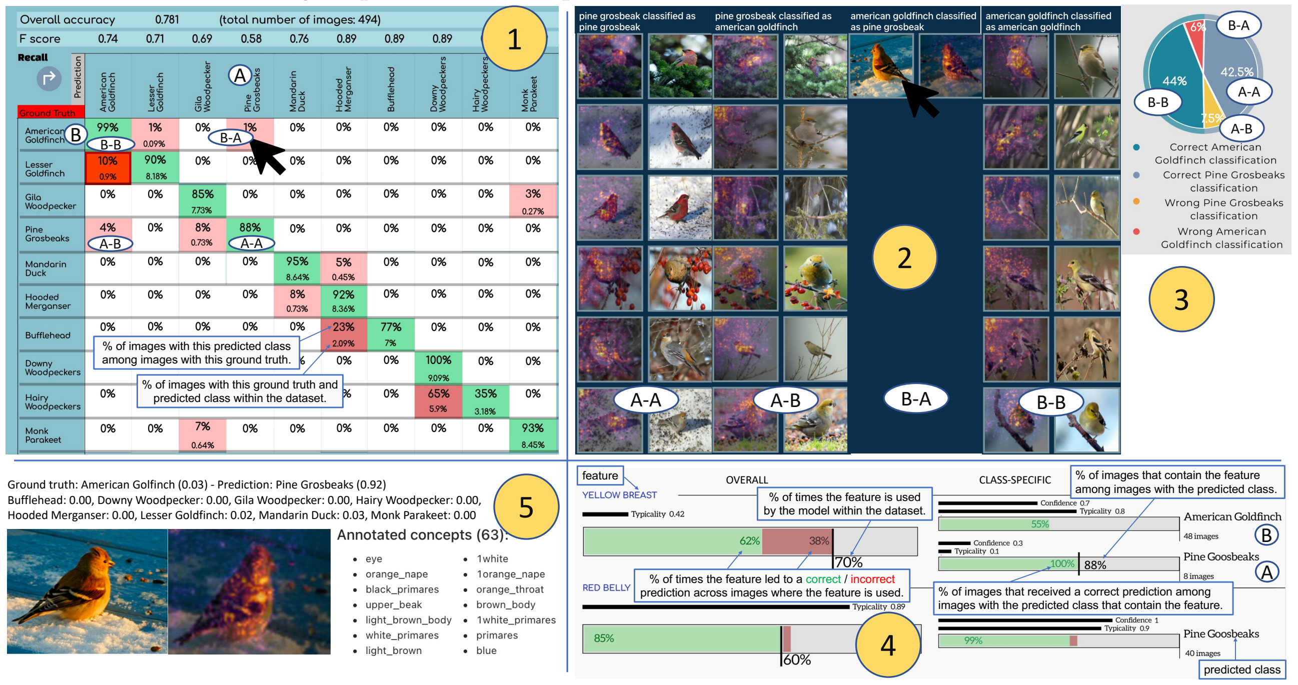
Gila woodpecker
cactus
(RED AND FOREHEAD) AND CACTUS
(brown AND breast) AND cactus

Pine Grosbeaks
(YELLOW_BROWN OR PINK) AND BELLY AND (GREY AND WING)
(YELLOW_BROWN OR PINK) AND BELLY AND (YELLOW_BROWN OR PINK) BREAST

Mandarin duck
(blue AND forehead) AND (blue AND mantle) AND (red and beak)
(purple OR blue) AND forehead

Hooded merganser
water
(BLUE AND WATER) AND (WHITE AND CROWN)
(white AND crown) AND (black AND wing) AND water

Figure 1: Query tab (left) and overall explanations tab (right). When querying (1) explanations, results are displayed underneath (2). The overall explanations tab shows both relevant (combinations of) concepts (3) and their association to each dataset class (4), and allows for varying the parameters to compute them (5).


1
2
3
4
5
Recall
A
B
B-B
B-A
A-B
A-A
pine grosbeak classified as pine grosbeak
pine grosbeak classified as american goldfinch
american goldfinch classified as pine grosbeak
american goldfinch classified as american goldfinch
6%
42.5%
44%
7.5%
feature
predicted class
OVERALL
CLASS-SPECIFIC

Overall accuracy 0.781 (total number of images: 494)

F score 0.74 0.71 0.69 0.58 0.76 0.89 0.89 0.89

Ground Truth \ Prediction	American Goldfinch	Lesser Goldfinch	Gila Woodpecker	Pine Grosbeaks	Mandarin Duck	Hooded Merganser	Bufflehead	Downy Woodpeckers	Hairy Woodpeckers	Monk Parakeet
American Goldfinch	99%	1% 0.09%	0%	1%	0%	0%	0%	0%	0%	0%
Lesser Goldfinch	10% 0.9%	90% 8.18%	0%	0%	0%	0%	0%	0%	0%	0%
Gila Woodpecker	0%	0%	85% 7.73%	0%	0%	0%	0%	0%	0%	3% 0.27%
Pine Grosbeaks	4%	0%	8% 0.73%	88%	0%	0%	0%	0%	0%	0%
Mandarin Duck	0%	0%	0%	0%	95% 8.64%	5% 0.45%	0%	0%	0%	0%
Hooded Merganser	0%	0%	0%	0%	8% 0.73%	92% 8.36%	0%	0%	0%	0%
Bufflehead	0%	0%	0%	0%	0%	23% 2.09%	77% 7%	0%	0%	0%
Downy Woodpeckers	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0%	0%	100% 9.09%	0%	0%
Hairy Woodpeckers	0%	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	0%	65% 5.9%	35% 3.18%	0%
Monk Parakeet	0%	0%	7% 0.64%	0%	0%	0%	0%	0%	0%	93% 8.45%

% of images with this predicted class among images with this ground truth.

% of images with this ground truth and predicted class within the dataset.

- Correct American Goldfinch classification
- Correct Pine Grosbeaks classification
- Wrong Pine Grosbeaks classification
- Wrong American Goldfinch classification

Ground truth: American Golfinch (0.03) - Prediction: Pine Grosbeaks (0.92)
Bufflehead: 0.00, Downy Woodpecker: 0.00, Gila Woodpecker: 0.00, Hairy Woodpecker: 0.00, Hooded Merganser: 0.00, Lesser Goldfinch: 0.02, Mandarin Duck: 0.03, Monk Parakeet: 0.00

Annotated concepts (63):

- eye
- orange_nape
- black_primares
- upper_beak
- light_brown_body
- white_primares
- light_brown
- 1white
- 1orange_nape
- orange_throat
- brown_body
- 1white_primares
- primares
- blue

YELLOW BREAST
Typicality 0.42
62% 38% 70%

% of times the feature is used by the model within the dataset.

% of times the feature led to a correct / incorrect prediction across images where the feature is used.

RED BELLY
Typicality 0.89
85% 60%

% of images that contain the feature among images with the predicted class.

Confidence 0.7 Typicality 0.8 55% American Goldfinch 48 images (B)

Confidence 0.3 Typicality 0.1 100% 88% Pine Goosbeaks 8 images (A)

% of images that received a correct prediction among images with the predicted class that contain the feature.

Confidence 1 Typicality 0.9 99% Pine Goosbeaks 40 images

Figure 2: Confusion matrix interactions. Our probe allows for different interactions with the explanations. E.g., when one clicks on a cell of the confusion matrix (1) corresponding to the predicted class A and ground truth class B, she is directed towards the corresponding local (2) (images corresponding to the cells A-A, A-B, B-A, B-B of the matrix) and global (4) explanations, as well as more performance indications (3). Clicking on a local, visual explanation displays further local, textual explanations (5).

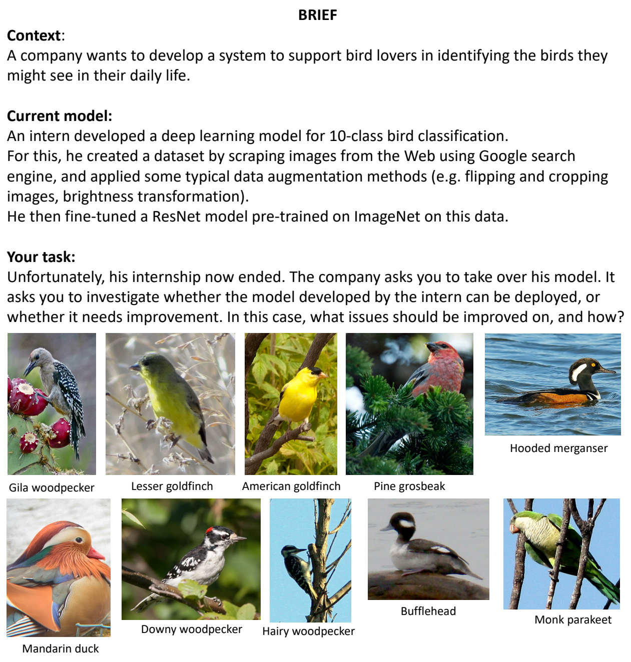
5 USER-STUDY SET-UP

To study how practitioners would use explainability methods for bug identification, we conduct 18 user-studies of one hour each. We prompt our participants to answer a design brief with the design probe. We ask them to explain out-loud what they do, and we note their interactions with the probe (order of visited interfaces, functionalities used, etc.). When they identify a potential failure and the related bug, we ask which action they would take to solve it. Each session ends with an exit interview and a questionnaire to collect ratings around the usefulness and usability of the interfaces. The questions combine the short version of the User Engagement Scale [37], and 7-point Likert scale questions around their likelihood to use the probe in the future. Before each session, we ask the participants for their agreement for recording. We later transcribe the recordings into anonymized transcripts, and destroy the recordings. The interview process has been reviewed by the ethics committee of our institution. We analyse the results of the user-study qualitatively in relation to the functionalities and orthogonal categories identified in section 3, and quantitatively based on the questionnaires, the count of commonalities in the steps followed by each participant, and the numbers of bugs identified.

Participants. The 18 participants were recruited through the networks of the authors, searches on professional social networks, and by snowball within the contacts of the first eight recruited participants. We only recruit participants who have experience with machine learning, but not necessarily with computer vision (CV), as they should understand the basic concepts around model failures. We categorize the participants based on their level of experience with CV. Low-CV experience participants (6) have never or only once developed a CV model, mid-CV experience participants (5) have less than 4 years of CV model development experience, and high-CV experience participants (7) have more.

Design brief. The design brief presents a model bug identification scenario (Figure 3). It is typical and simple enough for participants to reflect on their own practices without envisioning entirely new workflows. Bird classification might require domain-knowledge, raising reflections on the need to have domain expertise for bug identification. We scope the brief to the development setting as it encompasses a varied set of activities, with both reactive and proactive debugging.

Model. We train the machine learning model to be debugged to classify 10 species of birds. The training dataset is built with the idea of introducing both explicit (low test accuracy and mitigated confusion matrix) and implicit model failures and various bugs that explain these failures, as summarized in Table 4. To the best of our knowledge, there is no established list of bugs and failures for computer vision models. We propose a preliminary one, inspired by the literature on data biases [29, 48, 49], data shifts [18], robustness to adversarial [1] and natural perturbations [19], models using wrong reasoning for making inferences [17], and from the bugs mentioned in the formative study. Besides distribution shifts, we create wrong sets of features (incompleteness or irrelevance) that lead to correct or incorrect predictions, i.e. implicit or explicit failures. See examples in Figure 4. To introduce these bugs, we vary the image content in training and test data, around class-specific features (e.g. bird appearance), and less specific features (e.g. background).



BRIEF

Context:
A company wants to develop a system to support bird lovers in identifying the birds they might see in their daily life.

Current model:
An intern developed a deep learning model for 10-class bird classification.
For this, he created a dataset by scraping images from the Web using Google search engine, and applied 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 now ended. The company asks you to take over his model. It asks you to investigate whether the model developed by the intern can be deployed, or whether it needs improvement. In this case, what issues should be improved on, and how?

Figure 3: Overview of the design brief. Examples of samples of each class the model to be analyzed was trained on.

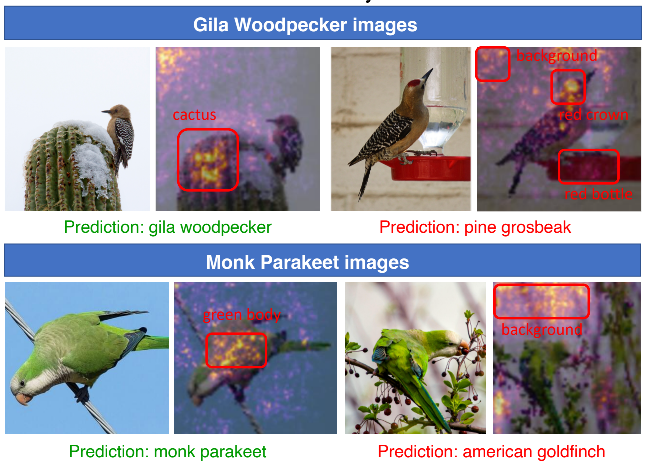

Gila Woodpecker images
cactus
background
red crown
red bottle
Prediction: gila woodpecker
Prediction: pine grosbeak
Monk Parakeet images
green body
background
Prediction: monk parakeet
Prediction: american goldfinch

Figure 4: Examples of implicit (green) and explicit (red) failures caused by irrelevant and incomplete features, e.g. the model incorrectly uses the pixels corresponding to the cactus to correctly predict the gila woodpecker class in the left image. The bounding boxes show the features of the model. We create the first failures by making sure that cacti are present in all and only training samples of gila woodpeckers, while the test images do not all contain a cactus. The second ones are created by making sure that only the monk parakeet training samples present a green bird (and in a standard position), while the test samples are more diverse.

Bug	Description	Example	Creation method
Distrib. shift	Large difference in training and deployment images.	We mention that training data are scraped from the Web, and a different context for the deployment data.	
Simplistic/incomplete, relevant features			
Explicit failure	The features are relevant but incomplete, and lead to incorrect inferences.	The model learns the `red` color for the `pine grosbeak`, which is correct for the males, but not for the yellow females.	We choose a subset of training images (e.g. male images) that give a partial view of the entire class.
Implicit failure	The model learns features that are relevant, but insufficiently representing a class, while still allowing for correct inferences.	The `monk parakeet` class is identified by the model solely through the color `green`.	We choose classes so that certain have a unique feature compared to the others.
Spurious/irrelevant features			
Explicit failure	The model learns features that are not semantically related to the species, and lead to incorrect inferences.	The model recognizes `gila woodpecker` by identifying `cactus` in images, but there is not always a cactus in the image.	Training images of a class contain an irrelevant feature, absent from other training samples and test set.
Implicit failure	The model learns irrelevant features, but still makes correct inferences.	The model learns the presence of `water` to identify hooded `mergansers`.	Same as above, but with similar training and test sets.

Table 4: Bugs introduced in the models of our design brief.

6 RESULTS

In this section, we present the results of our user-study, essentially the impact that the explanations in the probe have on the bug identification process, and how they are used in this process.

6.1 Impact of explanations on the bug identification process

Figure 5 summarizes the number of bugs identified by the participants in relation to the different types of model failures we introduced. We count one issue (1 point) as completely identified when a participant identifies both a bug and a relevant correction method, and give 0.5 point when the bug is well-characterized but no relevant correction method is found. This way, we make sure that the bug is characterized well-enough for the participant to propose a meaningful bug correction solution[3].

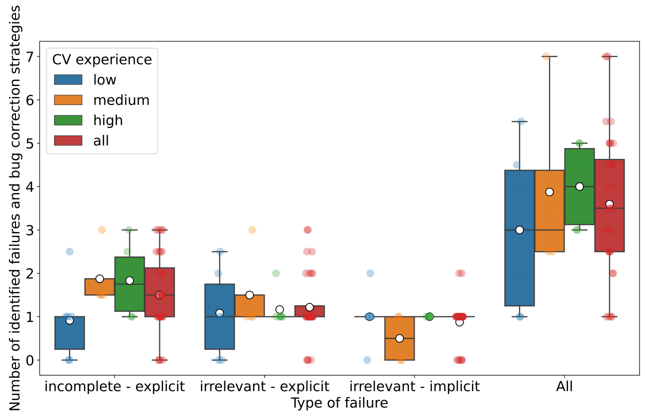


Figure 5: Number of bugs and relevant correction methods identified by our different participants during the user study.

[3]We do not plot the numbers related to implicit, incomplete features because they are identical to the ones for implicit irrelevant features: participants who succeed in identifying the latter always mention that the features are irrelevant and by extension incomplete –other ones should have been used.

6.1.1 *Successful bug identification.*

The bug identification process of our participants was in majority successful, with 3.5 bugs and correction methods identified on average, and up to 7 bugs identified by experienced participants. For consistency, we first let the participants explore the probe and failures they deemed important, and later discussed four specific failures. They were typically able to reflect on these failures, but not at the same speed, explaining the large standard deviations. Besides rapidity, three factors explain such deviation. 1) The low-CV participants deemed certain low-rate failures not worthy to debug due to their rarity. This can yet be wrong as high-CV participants discussed, since the error might be rare due to the data distribution, but still harmful. 2) The rare failure (one single lesser `goldfinch` mis-classified as a `hairy woodpecker`) was challenging, and only two high-CV participants proposed plausible bugs. The others pointed out to the lack of additional examples of this failure, preventing them from comparing local explanations. 3) The participants did not think of the existence of implicit failures, except when nudged.

Overall, these results show that a probe presenting various types of explanations allows to debug various feature failures, in relation to various dataset bugs. In order to achieve such successful bug identification, participants used varied workflows to navigate the different functionalities. These workflows are discussed in the next subsections.

6.1.2 *Disparate results for different explanation audiences.*

Among these successful results, we observe a high disparity in the number of bugs identified between participants with different levels of experience in computer vision (CV).

Low-CV experience participants miss guidance. In general, participants with computer vision experience identify more bugs than the ones without experience. The participants without experience who identified zero or one bug did not know where to start the process, how to proceed, and what kind of corrections to envision.

Misaligned mental models. Yet, three high-CV participants (removed from the plots) identified less than two bugs. Their mental model of bug identification was not aligned with our probe. They did not want to look into model features for bug identification, and one was solely interested in unknown unknowns [7, 32, 55] (outside the probe scope).

6.1.3 *Explainability allows to envision various, relevant bug correction methods.*

The probe led the participants to formulate bug correction methods that are diverse, relevant, and to-the-point, thanks to the different kinds of explanations that allowed the identification of highly specific data bugs. For instance, three participants discussed inappropriate data processing as a cause of failure, e.g. the image resolution is too small or the bird/background ratio too large, making the differences between certain bird species undetectable, suggesting for transforming the data pipeline. Five participants suggested restructuring the inference task by adding more classes, as a result of better characterizing the source of bugs, e.g. they identified the color differences between male (red) and female (yellow) `pinegrosbeaks` leading to high error rates for the female ones (confused with the yellow `american goldfinch`), and suggested to separate them into two classes to ease the learning. This is in line with other bug identification frameworks [44] which report they support idea generation for bug correction. Particularly, we notice these envisioned correction methods are more precise and potentially more effective than in our formative study where few types of explanations were mentioned.

6.1.4 *Participants still missed certain bugs. Incorrect features vs. correct inference.*

Participants focused on failures visible through the confusion matrix, either when a percentage of a diagonal cell is low, or when out-of-diagonal cell percentages are high. They often forgot that even classes with high accuracy might be based on problematic features. Some participants identified these issues serendipitously when attempting to understand visible failures.

Confirmation bias. One participant identified a very general bug from a few images: color bias of the model for most species. Confirmation bias led them to validate this bug by looking at images of different species, without going in more detail into the problematic colors and species, or searching for other bugs. We discuss these results further in section 7.

6.2 Different categories of explanations for different users and bug identification steps

Figure 6 displays the perceived usefulness of each tab as rated by our participants. Overall, all tabs are perceived useful with an average rating of at least 4 out of 7, yet the mean rating and standard deviation vary across tabs. We discuss below these variations in relation to the functionalities provided by each tab.

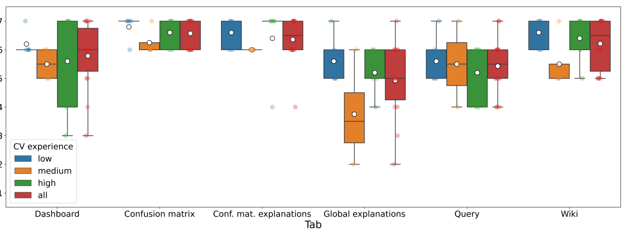


Figure 6: Perceived usefulness of the different tabs of the design probe. The ratings are displayed for each category of participants.

6.2.1 *Local versus global explanations (F3, F4). Hypothesis validation emphasized with the diversity of explanations.*

The participants primarily used the local and global explanations from the dashboard and confusion matrix, as testify the higher ratings for these two tabs (Figure 6). These explanations served for generating bug hypotheses and validating them. This shows that proposing a diversity of explanations nudges a more extensive bug identification process than with fewer or no explanations where most participants skip hypothesis validation, as we observed in the formative study.

Participants investigated explicit failures by entering different cells of the confusion matrix. The implicit failures required more diverse entry points to be identified: 1) Serendipitously, while investigating explicit failures. While investigating an explicit failure by looking at the four columns of images/saliency maps in the tab obtained from clicking on a cell of the confusion matrix, they would notice that saliency maps would highlight irrelevant features even in the A-A or B-B columns that present samples with correct predictions. 2) By deciding to explore the global explanation tab (without having a specific kind of failure in mind) and spotting clearly surprising features (e.g. `water` or `branch` are not features one would expect the model to focus on when classifying bird images. Instead, parts and characteristic colors of the bird would be expected for the model to generalize to new images with more varied background for instance) for the context, or features that were not sufficient (e.g. `purple` only for the `mandarin duck`, whereas images displaying other birds might also have the purple color, e.g. in their background). 3) By deciding to look into the diagonal of the confusion matrix (typically starting with cells that have low rates) and the corresponding explanations (saliency maps, and rankings of textual explanations by frequency). By looking into these specific features, they would reflect on whether something is irrelevant or incomplete.

Local and global explanations are complementary. The choice of starting point does not have a consistent motivation. Typically, participants who use local explanations to generate feature hypotheses validate their hypotheses by looking at local explanations for more images, or by verifying the presence of the features in the global explanations. Instead, participants using global explanations for hypothesis generation validate the features by making sure these features are reflected in a few local explanations across correct or incorrect inferences.

Within hypothesis generation, many participants combined the two approaches as the types of features and correction methods they lead to identify intersect but do not entirely overlap. For instance, incomplete or irrelevant but frequent features were typically identified from global explanations through the different ranking systems (**F6**). Instead, infrequent failures and their correction methods were better understood by looking at the actual images and saliency maps (**F5**). Global explanations were also used to identify the features influencing the majority of classification (typically the correct ones), which are in turn compared to the features used for incorrect inferences identified through local explanations (**F5**: comparisons across explanation types). For instance, they identified that overall, the color red is used by the model to infer `pine grosbeaks`, and locally understood that the only `american goldfinch` predicted as `pine grosbeak` was also displaying a `red` feature due to the brightness of the picture. Such finding could not be reached through a sole look at global explanations for which brightness is not reflected.

The choice of explanation type depends on the practitioners' experience with explainability. We do not identify a strong correlation between the categories of explanations used by the participants and their expertise. Yet, most participants with high-CV experience are more reticent towards unfamiliar types of explanations, and use primarily local, visual explanations, i.e. saliency maps (*"the dashboard gives almost everything. I'm more familiar with its explanations" Participant 4 high-CV*). Instead, the participants with fewer experience operate smoother transitions between local and global ones, and explore more types of explanations. This explains the higher ratings they gave to the tab reached from the confusion matrix (that presents all types of explanations) compared to the dashboard that only presents local explanations. Using global explanations can be faster than using local ones, but it was also more tedious as participants need to get accustomed to the scores and ways to interpret them. All participants argued these explanations should be used particularly when many images present similar failures, as it is not tractable to look at each image.

The use of local and global explanations led to incorrect bugs. Two types of errors are typically done when using the local explanations for hypothesis generation. a) Participants wrongly assumed the local explanations for images that got correct inferences to be relevant features for the model. This led them to automatically judge as irrelevant the features of samples with wrong inferences, while this is not necessarily the case. Warning about this assumption enabled them to reflect further about the potential bugs. b) Some participants formulated an incorrect hypothesis about a feature by looking at very few images, and did not further verify it, leading to develop incorrect bug correction methods. They mentioned that the global explanations could allow to avoid such errors. Global explanations were misleading when participants would identify interesting features with very low support, not being representative of most images.

*6.2.2 Explanation domain and medium (**O2**). Participants intuitively prefer in-domain explanations.* All but one participant preferred using visual explanations than textual ones. They argued the cognitive load is lower and it is faster to make sense of features by glossing over several local, visual explanations, than textual ones.

The two types of medium are complementary locally. Yet, textual explanations were also used. The participants mentioned that since they are not familiar with the task domain, they cannot easily interpret the saliency maps to identify meaningful features. Hence, they look at the local, textual explanations (and map them to the visual ones) to identify relevant bird features that one would expect the model to learn. They could also directly relate the wiki information that displays expected features according to an expert to these explanations.

One participant also suggested a functionality that only textual explanations can support: giving the freedom to explore new features as combinations of existing ones, to vary their granularity and create a taxonomy, e.g. combining `plants` and `leaves` into a larger `green background`. While this is possible within the query page, they would have liked to access this faster within the other interfaces, and to visualise the created taxonomy.

The preferred, global medium depends on the familiarity with the task domain. Participants mentioned a difficulty in interpreting the textual, global explanations as they were not familiar with the domain of the task. They however said that if they would know more about the domain, it would be easier to use as they could quickly get an idea of what a feature means on an image and what might be problematic with it.

*6.2.3 Explanation scope (**O1**). Preference for explanations of binaries.* Participants primarily focused on two-class explanations. These explanations align with reactive bug identification for failures in specific cells of the confusion matrix. Reflecting on two classes is also easier than considering more classes: it is harder to relate overall explanations to model failures. Half of the participants explained that overall explanations are also useful but less natural as they start from the out-of-diagonal cells of the matrix. This shows clearly in the ratings given to the dashboard or confusion matrix tabs that provide binary explanations, in comparison to the ratings for the global-explanations tab.

Global explanations as a quick diagnostic tool. Yet, participants still find uses to the global explanations computed on the entire dataset, as the large standard deviation testifies in comparison to the standard deviations for the other tabs. Participants prefer using such global explanations for tasks whose domain is familiar, and for diagonal cells of the confusion matrix. Simply by looking at these lists of explanations without having to click on each cell of a confusion matrix, they get a good overview of the features the model has learned per class, and can identify the pertinent, irrelevant or incomplete ones. Five participants actually used these explanations and their background knowledge to reflect on the validity of the features, e.g. they quickly spotted potential issues with `cactus` or `water` concepts that one might not expect to classify birds, and with the large number of color features while the model should also relate on shapes.

Questioning the faithfulness of binary explanations. These explanations are complementary. Global explanations more accurately account for the features of the model and allow for a faster spotting of problematic features. Yet, practitioners prefer to understand specific cells of the confusion matrix with binary explanations, which might lead to erroneous feature interpretations (one feature might seem discriminative for two classes but might not be important overall). A single mid-CV participant accurately reflected on such limitation, that practitioners should be warned about. However, this reflection was also problematic for our participant as it prevented them from obtaining insights from the probe: the participant constantly worried that correcting a specific bug would create new ones in other matrix cells.

*6.2.4 Use of domain knowledge (**F9**). Domain knowledge is used for successful hypothesis formulation and validation.* This knowledge serves to a) formulate hypothesis on relevant features the model should learn for a class, and to compare them to the actual features, or b) to validate hypotheses about problematic features. For instance, *Participant 4 high-CV* naturally started to use it for specific confusion cases where the model accurately looks at the bird (according to the saliency map) but apparently not at the right or complete bird features as it makes incorrect inferences.

*6.2.5 Query-related explanations (**F7**).* All participants used the passive mode of exploring the explanations, since it is less cognitively-demanding, and they are used to such explanations. Active querying is used only by half of the participants. This shows clearly by the lower average ratings and higher standard deviation the query tab got in comparison to the dashboard and confusion matrix tabs. Active querying allows to validate potential hypotheses around problematic features. For that, participants query the matrix of percentage to verify that a feature is only used for images that present specific miss-classifications. Three participants mentioned that active queries are especially efficient once one is familiar with the expected and the often problematic features. For instance, an expert participant mentioned that in their own medical use-case where it is known that the model might learn incorrect features relating to the background of X-ray images (e.g. a part of a pace-maker), they would like to query background features directly.

*6.2.6 Interactivity (**F8**). Interactivity to speed up and augment the bug identification process.* Besides having functionalities that are currently not available (global explanations and query), the primary advantage of the probe was its interactivity and practicality, aligning with results for tabular data [20]. It was especially useful to compare diverse images (the four types of images in a binary classification task) and explanations to estimate the feature's relative importance. For instance, some participants compared two queries where the only difference is the addition of one feature, to check how much this feature impacts the model inferences. *"If the tool is ergonomic, fast and malleable, it would definitely help me fasten my process, and it would help combine more information that I don't usually look at." Participant 8 high-CV.* A third of the participants even suggested ways to have even more interactivity and fast transitions between explanation types.

Interactivity to select relevant explanations. To navigate global explanations, the participants used one main interactivity feature, the choice of settings, to rank or filter the explanations (**F6**). They could for instance identify a) frequent mistakes by ranking the explanations based on the number of incorrect predictions they lead to, b) frequent features by ranking explanations based on typicality scores and filtering out low-support ones, and c) features that lead solely to correct or incorrect inferences by computing the explanations independently on the set of samples which received correct or incorrect predictions. These settings are necessary due to the amount of information the probe provides.

7 DISCUSSION

7.1 Summary of findings

Our user-study brought new insights on the use of explanations towards bug identification, summarized in Table 5. While the most common explanations, i.e. local visual explanations, were primarily used due to their simplicity and familiarity, our probe also highlighted the importance to present diverse explanations. Global, textual, active, interactive, and binary explanations, as well as domain knowledge, were also exploited to achieve different objectives, e.g. identifying new hypotheses, or the same objectives more efficiently. Yet, by acknowledging the disparity in the use of the functionalities and in the number of bugs they led to identify, we can extract further implications for future explainability, debugging and HCI research. We now discuss the limitations of our work and these findings.

Category	Insight
Impact of explanations on the debugging process.	
Effectiveness	Successful bug identification process. A few missed/incorrect bugs due to misinterpretations of features and confirmation bias.
Variations	Low-CV: need for guidance. High-CV: misaligned mental models.
Corrections	More diverse and precise bug correction methods are envisioned.
Different categories of explanations for different users and debugging steps.	
Local/global	Complementarity. Emphasis on hypothesis validation. Preference based on practitioners' experience with explainability.
Domain, medium	Intuitive preference for in-domain explanations. Local level: complementarity of in- and out of- domain explanations. Global level: preference depends on the familiarity with the task domain.
Scope	Preference for binaries. Global explanations as quick diagnostic tool. Lack of questioning around the faithfulness of binaries.
Knowledge	Domain knowledge used both for successful hypothesis formulation and validation.
Active query	Low use despite usefulness for hypothesis validation.
Interactivity	Speeding up and augmenting the debugging process. Selecting relevant explanations. Wish for model comparisons.

Table 5: Summary of the insights from our user-studies.

7.2 Limitations

There are several limitations in our probe and study. While we do not think they impact the validity of our results, they would need to be tackled in the future for more comprehensiveness.

Scope of the probe. Our probe is adapted for a specific type of computer vision models: deep learning models that perform classification tasks and from which local, visual explanations can be extracted. The global explanations can only be computed when it is possible to annotate local explanations with semantic features, and can be costly depending on the size of the dataset and number of classes. Hence, it can be challenging to use for certain applications. Adapting these explanations to other use-cases is a challenge on its own. Balancing the trade-off between cost and faithfulness of the explanations and making practitioners aware of it would also merit being investigated.

Scope of the study. While the work involved a considerable number of participants (18 for the formative study and co-creation sessions, and 18 for the user-study) with various backgrounds, we cannot fully guarantee the generalizability of the results. Similarly, our study employed a use-case that requires domain knowledge none of our participants had (to bring consistency), and we made sure to provide the required knowledge. It would be interesting to study how participants, familiar with a use-case, would go about bug identification. This is however challenging as participants should share their data, it is costly to annotate, and the use-case would not be consistent across participants. Scaling our study to use-cases with more classes is also important as other works identify that "as scale increases, interpretability and satisfaction decrease" [20].

Impact of the probe design. The results of our user-study are inevitably mediated by the design, implementation and usability of our probe. As discussed in section 4, we however made sure to allow for diverse workflows and interactions with the explanations without biasing the users towards specific ones. As for usability, the answers (Figure 7) to our exit questionnaire give an indication of how it might have affected our results. Most factors received high-ratings, confirming that our participants appreciated the functionalities within our probe, and were likely not negatively impacted by them. Especially, they found it rewarding to use our probe and were eager to reuse it on their own use-cases, saying that it was more convenient than their usual development environment.

However, some participants felt overwhelmed at first by the amount of functionalities (perceived usability ratings confirm this). While they got used to them, they would have liked guidance from the probe in the process. We could not do so not to skew them, yet this is an important indication for future tools. Similarly, they gave an average rating to the attractiveness (mean of 3.31 out of 5 points) as they would appreciate the probe having a more modern look.

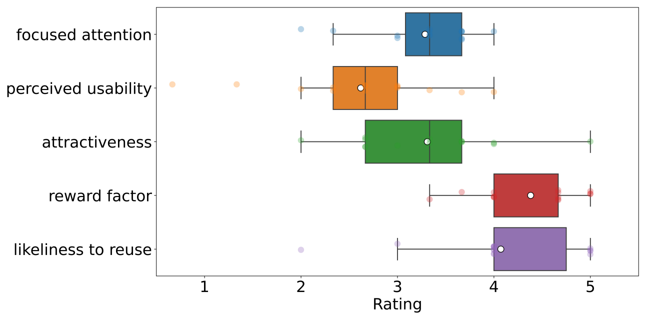


Figure 7: Aggregated factors of the User-Engagement form (short version) presented in boxplots.

7.3 Implications & Future Work

7.3.1 Need to develop user-experiences. Guidance. As some participants had a hard time envisioning uses of certain explanations, future tools need to provide hints. Hints should be enough as simply explaining ways other participants used the explanations led the participants in difficulty to successfully identify bugs.

Besides, the current probe allows for any sequence of interactions with the different types of explanations supported (in order not to skew our user-study participants towards certain explanations and activities). Yet, further guiding these interactions by suggesting potential sequences of activities would also support practitioners further in debugging their model. Several participants mentioned that an ideal user-interface would not leave them as much freedom as currently is, but instead narrow down possibilities so as to simplify the debugging process and guide them towards the relevant activities for each type of failures and bugs. Hence, future tools would benefit from identifying the minimum set of user-journeys different types of practitioners and failures would require.

Participants with high-CV and explainability experience however require further investigation to understand when they would be ready to use less familiar explanations. Especially, for these participants, our observations differ from explanation practices on tabular data [20]. The GAMUT probe led to find a strong correlation between the level of explainability expertise and the use of diverse explanations, result totally opposed to ours. This could be motivated by the lack of practice, even for our high-expertise practitioners, with global, textual explanations for computer vision, contrary to practitioners working on tabular data who are more familiar with both types of explanations.

Warnings around typical misinterpretations. Blindly following the explanations sometimes leads to identify incorrect bugs. Yet, not all participants are aware of these dangers, and trust the explanations similarly. Only two participants asked us how the saliency maps were computed, and none reflected on the potential noise in the salient pixels. As for the global explanations, only 4 participants questioned their faithfulness and the fact that an annotation of salient pixels does not necessarily reflect what the model actually looks at (i.e. colors, textures, or shapes, etc.).

These observations around trust in explanations are aligned with the ones for tabular data, e.g. Hohman et al. [20] mention needing "healthy skepticism" from practitioners. They are also inline with the notion of *misuse* of explanations [25]: certain participants would misinterpret explanations by taking a brief look at them simply because they seemed to confirm their hypothesis. Future tools would merit displaying warnings against these limitations and misinterpretations.

Integration of structural and training bugs. Some participants tended to explain all bugs with issues of data content or data pipeline, without elaborating on other potential bugs, e.g. related to the model structure or training hyperparameters. They were either skewed by the focus of our probe on such types of bugs due to the visualisation of data content, or because they did not have in mind the other concerns. Some participants envisioned to use our probe once other bugs are corrected, but others nuanced this view arguing for a more iterative process, where all types of bugs might need simultaneous considerations depending on the ways the bugs are corrected (e.g. data augmentation for balancing might lead to overfitting and to increase the size of the model architecture). This shows the need to investigate how to best combine the functionalities in our probe to the functionalities around the other types of bugs (e.g. tools such as [41]), without overwhelming a user.

7.3.2 Usefulness of different explanation types. Explanations for data enquiry. Explanations primarily served as artifacts for surfacing feature failures, identifying data bugs and bias-variance issues. Similarly to observations made for explainability with tabular data [20], the explanations were also used by four of the participants as an access point into the data. These participants used the query functionality with specific features, ground truth and predicted classes, to better understand what they look like within the dataset, and whether they are comprehensively represented. Such understanding was later used to refine hypotheses about dataset bugs. Future interfaces would hence merit combining further the extensive exploration of training datasets to the model exploration, and facilitating common interactions with the explanations towards that end.

Complementarity of explanation types. Our study showed that all explanation types are useful for participants in different stages of the bug identification workflow to answer different questions. Their use often depends on the degree of familiarity of the participants with the task domain, and with these types of explanations.

More research is needed to further develop these different types of explanations since, so far, research focuses primarily on local, visual ones. Especially, attention on *textual explanations* could benefit practitioners, e.g. for understanding how to best represent and query concepts and their combinations, taxonomies of concepts, etc. *What-if (causal) questions* that were rarely expressed here could merit research on accessibility as well. Finally, future tools could further *combine in- and out of- domain explanations* by showing example image patches corresponding to any displayed textual explanations so as to increase the learning rate of the practitioners. Two participants also suggested *global, visual explanations* by automatically clustering similar-looking, salient image patches. While this might be hard to realize in practice, this further shows their appreciation for visual information, and the need for further research.

Interactivity versus complexity. Surprisingly, our study showed that rather simple explanations can lead to successfully understand a large number of bugs: the global explanations were simple statistics computed over textual annotations of the dataset, but allowed for a global understanding of the model. While simple in their calculation, their interpretation was already complex enough for the participants not familiar with the textual and global explainability paradigms. We argue that these simple explanations were useful thanks to the usability of the interactive interface and its focus on comparisons, which allowed to identify many similarities and differences across images receiving different predictions.

This shows that it might not be urgent to develop highly complex explainability methods yet, as they are new black-boxes for the practitioners who might trust their faithfulness too much, while having a hard time using them. Instead, more research on the development of interactive interfaces could be more beneficial to the practitioners.

Manual exploration versus automation. Multiple participants suggested to automate parts of the interface to speed up and direct the debugging process. For instance, they would like to automatically be presented with explanations that reflect bugs, or at least with a reduced set of potentially problematic features (the number of global explanations is otherwise overwhelming) through an automatic comparison of the explanations to the domain-knowledge.

Yet, we argue that extensive automation is not possible and desired. The relevance of a feature to a model is sometimes ambiguous, e.g. relevance of the `cactus` for `gila woodpeckers`, so the automatic comparison would lead to a skewed and non-transparent result. Besides, attention should be put into not making the debugging tool another black-box (besides the model to explain), as our participants already tended not to question the completeness and faithfulness of the displayed explanations. A way to limit automation could be to provide even quicker interactions, for instance to go from binary explanations to global ones so as to accurately estimate their relevance.

Nevertheless, facing the amount of debugging methods developed in machine learning testing literature that are unknown to practitioners, it is important to also investigate how much these methods are complementary to the manual process, and how to best involve them in this process.

Reliance on domain knowledge. The study confirmed the importance of domain knowledge. All but one participant (who did not reflect on features) used it (hence the high ratings the wiki tab obtained). Unfortunately, investigating the wiki was not consistently performed across failures, leading to miss certain bugs. For instance, two participants who correctly understood the difference between the similar-looking bird species `gila` and `hairy woodpeckers` (`brown or white body` respectively) and the missing feature (body color), did not use the wiki page to inspect the `pine grosbeak` and `american goldfinch`, missing the hint for another bug (difference of colors for female and male grosbeaks). Using domain knowledge merits more support. Studying how to make practitioners and domain experts interact is important, i.e. the format in which they can best communicate, the inputs developers need, but also the most effective way for domain experts who are often not familiar with technical terms to provide useful information for the practitioners.

8 CONCLUSION

In this work, we engaged in a formative study and a co-creation process to design a probe for investigating the interaction between explainability and bug identification. We then performed 18 user-studies with this probe. Our participants varied in their bug identification workflows, but managed to identify a consequential amount of bugs. These results showed that explanations can be used in various steps of the process for different purposes, and especially for characterizing diverse types of feature failures. Different categories of explanations (e.g. global, out-of-domain, active, and interactive) showed to be useful and often complementary. Yet, our participants also struggled with various aspects of the process, falling into certain explainability traps, or being shy to explore unfamiliar explanations.

This shows the urgent need for more HCI research to provide the right amount of guidance to practitioners engaged in bug identification activities and having access to explainability methods, while still allowing for freedom and adaptability of the process. Especially, the process should be supported through the use of interactive interfaces with various types of interactions not only with data and explanations but also with other artifacts to address non-feature failures. Additionally, our study points out to research directions for other communities: specific types of explanations merit further development by the machine learning explainability community, and the effectiveness of machine learning testing methods needs to be characterized in comparison to the one of human debugging for future integration.

ACKNOWLEDGMENTS

This work was partially supported by the HyperEdge Sensing project funded by Cognizant. We would like to thank all the participants of our studies, without whom this work would not have been possible. Besides, we would like to thank Floris Van Veen, Kanish Dwivedi, Silvia Mokranova, Mihai Filimon, and Mathieu Butenaerts who participated in building our design probe.

REFERENCES

[1] Naveed Akhtar and Ajmal Mian. 2018. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access* 6 (2018), 14410–14430.
[2] Ahmed Alqaraawi et al. 2020. Evaluating saliency map explanations for convolutional neural networks: a user study. In *IUI*. 275–285.
[3] 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.
[4] 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.

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

[6] Vijay Arya, Rachel KE Bellamy, et al. 2019. One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques. (2019).

[7] 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.

[8] 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.

[9] Gagan Bansal. 2018. Explanatory Dialogs: Towards Actionable, Interactive Explanations. In *AIES '18.* 356–357.

[10] Umang Bhatt, Alice Xiang, et al. 2020. Explainable machine learning in deployment. In *FAT*.* 648–657.

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

[12] Hao-Fei Cheng et al. 2019. Explaining decision-making algorithms through UI: Strategies to help non-expert stakeholders. In *CHI.* 1–12.

[13] Michael Chromik et al. 2021. I Think I Get Your Point, AI! The Illusion of Explanatory Depth in Explainable AI. In *IUI.* 307–317.

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

[15] Yash Goyal, Ziyan Wu, Jan Ernst, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. Counterfactual visual explanations. In *ICML.* PMLR, 2376–2384.

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

[17] Lisa Anne Hendricks, Kaylee Burns, et al. 2018. Women also snowboard: Overcoming bias in captioning models. In *ECCV.* 771–787.

[18] Dan Hendrycks, Steven Basart, et al. 2021. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV.* 8340–8349.

[19] Dan Hendrycks and Thomas Dietterich. 2018. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. In *ICLR.*

[20] Fred Hohman et al. 2019. Gamut: A design probe to understand how data scientists understand machine learning models. In *CHI.* 1–13.

[21] 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.

[22] Hilary Hutchinson, Wendy Mackay, et al. 2003. Technology probes: inspiring design for and with families. In *SIGCHI.* 17–24.

[23] Sérgio Jesus et al. 2021. How can I choose an explainer? An Application-grounded Evaluation of Post-hoc Explanations. In *FAccT.* 805–815.

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

[25] Harmanpreet Kaur et al. 2020. Interpreting Interpretability: Understanding Data Scientists' Use of Interpretability Tools for Machine Learning. In *CHI.* 1–14.

[26] 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.

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

[28] Lucas Layman, Madeline Diep, et al. 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.

[29] Yi Li and Nuno Vasconcelos. 2019. Repair: Removing representation bias by dataset resampling. In *ICCV.* 9572–9581.

[30] Q Vera Liao et al. 2020. Questioning the AI: informing design practices for explainable AI user experiences. In *CHI.* 1–15.

[31] Brian Y Lim et al. 2019. Why these Explanations? Selecting Intelligibility Types for Explanation Goals.. In *IUI Workshops.*

[32] Anthony Liu, Santiago Guerra, et al. 2020. Towards hybrid human-ai workflows for unknown unknown detection. In *WWW.* 2432–2442.

[33] 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.

[34] 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 *2018 ACM Joint Meeting on European Software Engineering Conference.* 175–186.

[35] Shweta Narkar et al. 2021. Model LineUpper: Supporting Interactive Model Comparison at Multiple Levels for AutoML. In *IUI.* 170–174.

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

[37] Heather L O'Brien, Paul Cairns, and Mark Hall. 2018. A practical approach to measuring user engagement with the refined user engagement scale (UES) and new UES short form. *International Journal of Human-Computer Studies* 112 (2018), 28–39.

[38] Nathalie Rauschmayr, Vikas Kumar, Rahul Huilgol, Andrea Olgiati, Satadal Bhattacharjee, 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).

[39] 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.

[40] Marco Tulio Ribeiro et al. 2016. Why should i trust you? Explaining the predictions of any classifier. In *SIGKDD.* 1135–1144.

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

[42] E Schoop, F Huang, and B Hartmann. 2021. UMLAUT: Debugging Deep Learning Programs using Program Structure and Model Behavior. (2021).

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

[44] Sahil Singla, Besmira Nushi, S Shah, E Kamar, and E Horvitz. 2020. Understanding Failures of Deep Networks via Robust Feature Extraction. (2020).

[45] D Smilkov and al. 2017. SmoothGrad: removing noise by adding noise. (2017).

[46] 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.

[47] Harini Suresh, Steven R Gomez, K 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.

[48] 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.

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

[50] 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.

[51] 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.

[52] Yao Xie et al. 2020. CheXplain: Enabling Physicians to Explore and Understand Data-Driven, AI-Enabled Medical Imaging Analysis. In *CHI.* 1–13.

[53] 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.

[54] J M Zhang, M Harman, and al. 2020. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering* (2020).

[55] Peng Zhao, Yu-Jie Zhang, and Zhi-Hua Zhou. 2021. Exploratory machine learning with unknown unknowns. In *AAAI*, Vol. 35. 10999–11006.

A EXAMPLE MOCK-UPS USED IN THE CO-CREATION SESSIONS

Figure 8, Figure 9, Figure 10.

B FIRST IMPLEMENTED PROTOTYPE FOR THE PROBE

Figure 11, Figure 12, Figure 13, Figure 14, Figure 15, Figure 16.

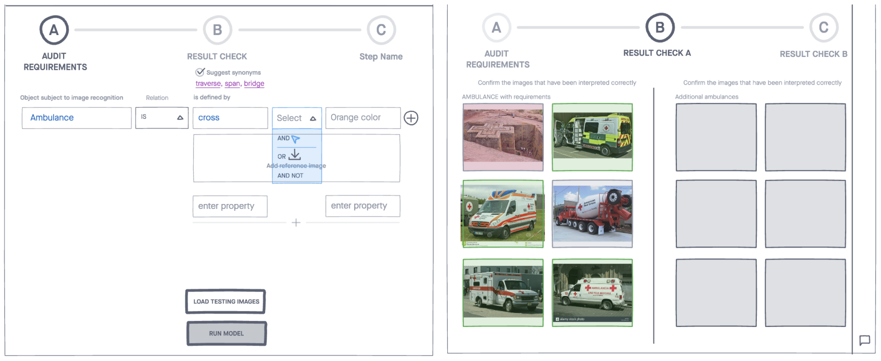


Figure 8: Low-fidelity mock-up used in the co-creation sessions: query functionality and the result interface after a query.

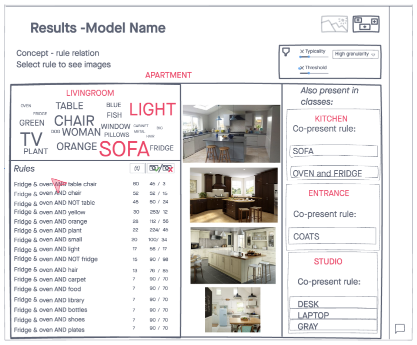


Figure 9: Low-fidelity mock-up used in the co-creation sessions: example display of important concepts and rules for one class, and their co-presence in other classes.

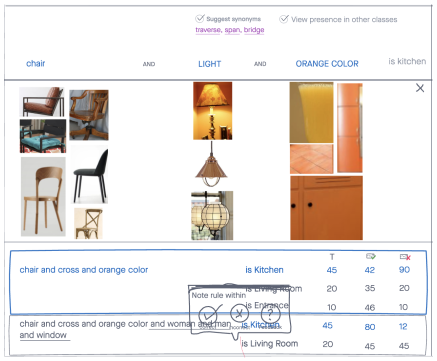


Figure 10: Low-fidelity mock-up used in the co-creation sessions: another example display of important rules and scores, in comparison to the scores of related rules for other classes.

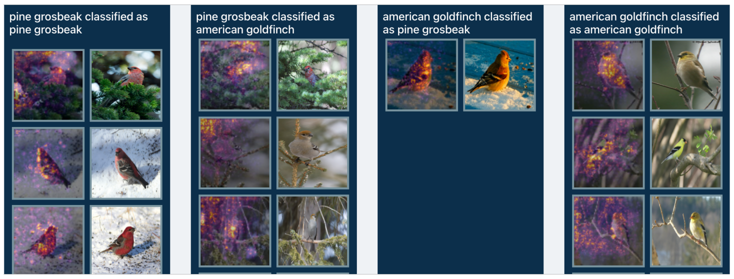
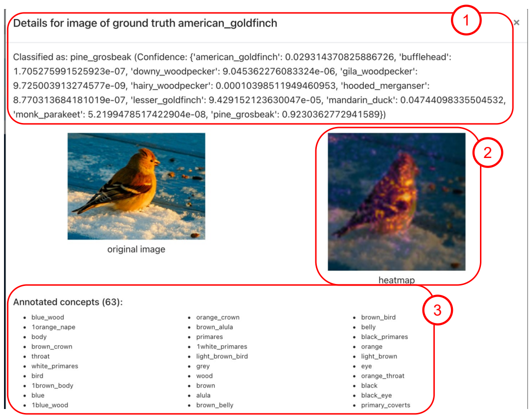

pine grosbeak classified as pine grosbeak
pine grosbeak classified as american goldfinch
american goldfinch classified as pine grosbeak
american goldfinch classified as american goldfinch

Figure 11: Display of the saliency maps within the probe.



Details for image of ground truth american_goldfinch

Classified as: pine_grosbeak (Confidence: {'american_goldfinch': 0.029314370825886726, 'bufflehead': 1.705275991525923e-07, 'downy_woodpecker': 9.045362276083324e-06, 'gila_woodpecker': 9.725003913274577e-09, 'hairy_woodpecker': 0.00010398511949460953, 'hooded_merganser': 8.770313684181019e-07, 'lesser_goldfinch': 9.429152123630047e-05, 'mandarin_duck': 0.04744098335504532, 'monk_parakeet': 5.2199478517422904e-08, 'pine_grosbeak': 0.9230362772941589})

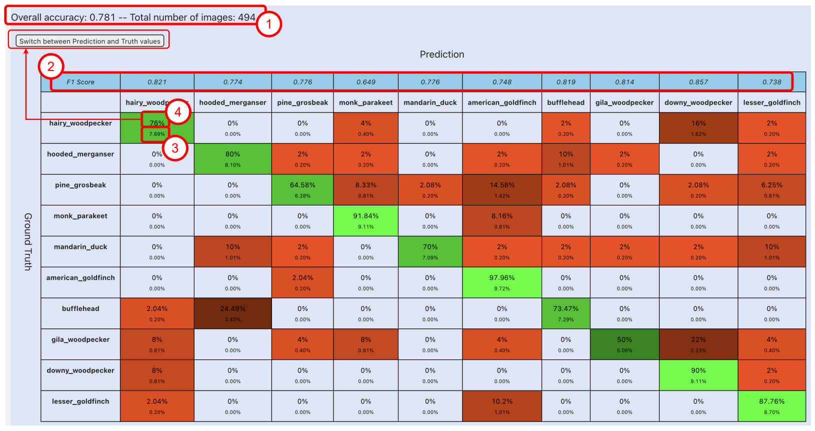
original image

heatmap

Annotated concepts (63):

- blue_wood
- 1orange_nape
- body
- brown_crown
- throat
- white_primares
- bird
- 1brown_body
- blue
- 1blue_wood
- orange_crown
- brown_alula
- primares
- 1white_primares
- light_brown_bird
- grey
- wood
- brown
- alula
- brown_belly
- brown_bird
- belly
- black_primares
- orange
- light_brown
- eye
- orange_throat
- black
- black_eye
- primary_coverts

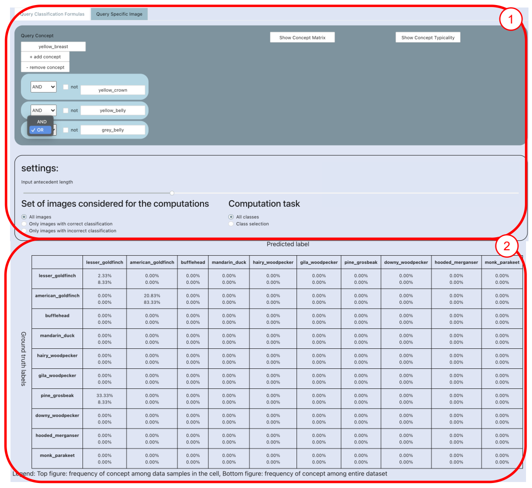
Figure 12: Display of further local explanations.



Overall accuracy: 0.781 -- Total number of images: 494

	hairy_woodpecker	hooded_merganser	pine_grosbeak	monk_parakeet	mandarin_duck	american_goldfinch	bufflehead	gila_woodpecker	downy_woodpecker	lesser_goldfinch
F1 Score	*0.821*	*0.774*	*0.776*	*0.649*	*0.776*	*0.748*	*0.819*	*0.814*	*0.857*	*0.738*
hairy_woodpecker	76% 7.69%	0% 0.00%	0% 0.00%	4% 0.40%	0% 0.00%	0% 0.00%	2% 0.20%	0% 0.00%	16% 1.62%	2% 0.20%
hooded_merganser	0% 0.00%	80% 8.10%	2% 0.20%	2% 0.20%	0% 0.00%	2% 0.20%	10% 1.01%	2% 0.20%	0% 0.00%	2% 0.20%
pine_grosbeak	0% 0.00%	0% 0.00%	64.58% 6.28%	8.33% 0.81%	2.08% 0.20%	14.58% 1.42%	2.08% 0.20%	0% 0.00%	2.08% 0.20%	6.25% 0.61%
monk_parakeet	0% 0.00%	0% 0.00%	0% 0.00%	91.84% 9.11%	0% 0.00%	8.16% 0.81%	0% 0.00%	0% 0.00%	0% 0.00%	0% 0.00%
mandarin_duck	0% 0.00%	10% 1.01%	2% 0.20%	0% 0.00%	70% 7.09%	2% 0.20%	2% 0.20%	2% 0.20%	2% 0.20%	10% 1.01%
american_goldfinch	0% 0.00%	0% 0.00%	2.04% 0.20%	0% 0.00%	0% 0.00%	97.96% 9.72%	0% 0.00%	0% 0.00%	0% 0.00%	0% 0.00%
bufflehead	2.04% 0.20%	24.49% 2.43%	0% 0.00%	0% 0.00%	0% 0.00%	0% 0.00%	73.47% 7.29%	0% 0.00%	0% 0.00%	0% 0.00%
gila_woodpecker	8% 0.81%	0% 0.00%	4% 0.40%	8% 0.81%	0% 0.00%	4% 0.40%	0% 0.00%	50% 5.06%	22% 2.23%	4% 0.40%
downy_woodpecker	8% 0.81%	0% 0.00%	0% 0.00%	0% 0.00%	0% 0.00%	0% 0.00%	0% 0.00%	0% 0.00%	90% 9.11%	2% 0.20%
lesser_goldfinch	2.04% 0.20%	0% 0.00%	0% 0.00%	0% 0.00%	0% 0.00%	10.2% 1.01%	0% 0.00%	0% 0.00%	0% 0.00%	87.76% 8.70%

Figure 13: Overall performance information provided in the design probe.



Query Concept
yellow_breast
AND ☐ not yellow_crown
AND ☐ not yellow_belly
OR ☐ not grey_belly

settings:

Input antecedent length

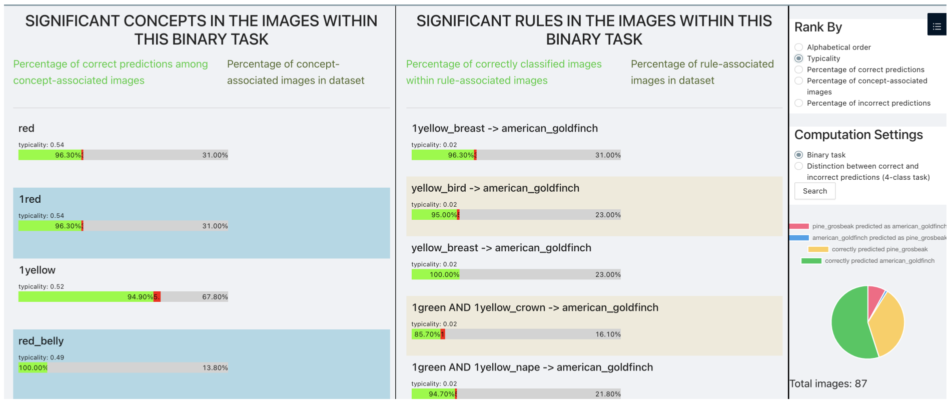
Set of images considered for the computations
[✓ All images]
Only images with correct classification
Only images with incorrect classification

Computation task
[✓ All classes]
Class selection

	lesser_goldfinch	american_goldfinch	bufflehead	mandarin_duck	hairy_woodpecker	gila_woodpecker	pine_grosbeak	downy_woodpecker	hooded_merganser	monk_parakeet
lesser_goldfinch	2.33% 8.33%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%
american_goldfinch	0.00% 0.00%	20.83% 83.33%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%
bufflehead	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%
mandarin_duck	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%
hairy_woodpecker	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%
gila_woodpecker	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%
pine_grosbeak	33.33% 8.33%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%
downy_woodpecker	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%
hooded_merganser	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%
monk_parakeet	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%	0.00% 0.00%

Legend: Top figure: frequency of concept among data samples in the cell, Bottom figure: frequency of concept among entire dataset

Figure 14: Query page with (1) query input and (2) query results.



SIGNIFICANT CONCEPTS IN THE IMAGES WITHIN THIS BINARY TASK

Percentage of correct predictions among concept-associated images | Percentage of concept-associated images in dataset

red
typicality: 0.54
96.30% 31.00%

1red
typicality: 0.54
96.30% 31.00%

1yellow
typicality: 0.52
94.90% 67.80%

red_belly
typicality: 0.49
100.00% 13.80%

SIGNIFICANT RULES IN THE IMAGES WITHIN THIS BINARY TASK

Percentage of correctly classified images within rule-associated images | Percentage of rule-associated images in dataset

1yellow_breast -> american_goldfinch
typicality: 0.02
96.30% 31.00%

yellow_bird -> american_goldfinch
typicality: 0.02
95.00% 23.00%

yellow_breast -> american_goldfinch
typicality: 0.02
100.00% 23.00%

1green AND 1yellow_crown -> american_goldfinch
typicality: 0.02
85.70% 16.10%

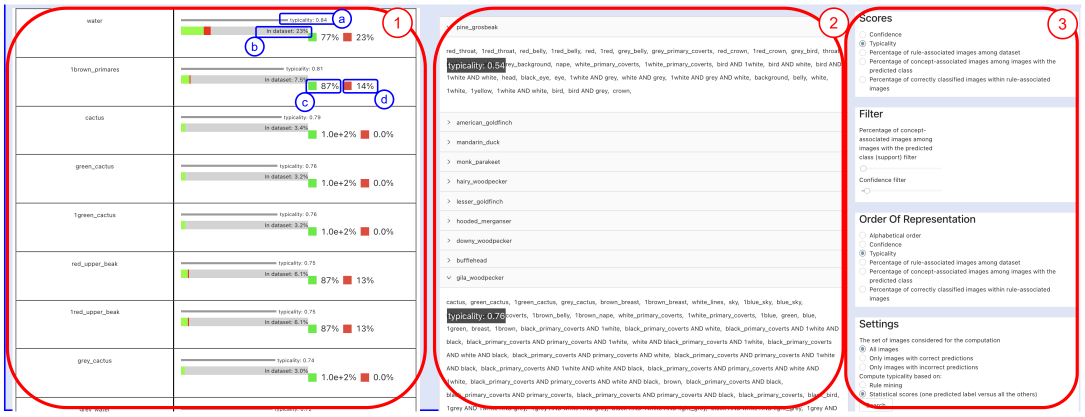
1green AND 1yellow_nape -> american_goldfinch
typicality: 0.02
94.70% 21.80%

Rank By
Alphabetical order
[✓ Typicality]
Percentage of correct predictions
Percentage of concept-associated images
Percentage of incorrect predictions

Computation Settings
[✓ Binary task]
Distinction between correct and incorrect predictions (4-class task)

🟥 pine_grosbeak predicted as american_goldfinch
🟦 american_goldfinch predicted as pine_grosbeak
🟨 correctly predicted pine_grosbeak
🟩 correctly predicted american_goldfinch

Figure 15: Local explanations presented as a result of clicking on a confusion matrix cell.



water	typicality: 0.84 · In dataset: 23% · 🟩 77% 🟥 23%
1brown_primares	typicality: 0.81 · In dataset: 7.5% · 🟩 87% 🟥 14%
cactus	typicality: 0.79 · In dataset: 3.4% · 🟩 1.0e+2% 🟥 0.0%
green_cactus	typicality: 0.76 · In dataset: 3.2% · 🟩 1.0e+2% 🟥 0.0%
1green_cactus	typicality: 0.76 · In dataset: 3.2% · 🟩 1.0e+2% 🟥 0.0%
red_upper_beak	typicality: 0.75 · In dataset: 6.1% · 🟩 87% 🟥 13%
1red_upper_beak	typicality: 0.75 · In dataset: 6.1% · 🟩 87% 🟥 13%
grey_cactus	typicality: 0.74 · In dataset: 3.0% · 🟩 1.0e+2% 🟥 0.0%

pine_grosbeak

red_throat, 1red_throat, red_belly, 1red_belly, red, 1red, grey_belly, grey_primary_coverts, red_crown, 1red_crown, grey_bird, throat
typicality: 0.54 rey_background, nape, white_primary_coverts, 1white_primary_coverts, bird AND 1white, bird AND white, bird AND
1white AND white, head, black_eye, eye, 1white AND grey, white AND grey, 1white AND grey AND white, background, belly, white,
1white, 1yellow, 1white AND white, bird, bird AND grey, crown,

> american_goldfinch
> mandarin_duck
> monk_parakeet
> hairy_woodpecker
> lesser_goldfinch
> hooded_merganser
> downy_woodpecker
> bufflehead

gila_woodpecker

cactus, green_cactus, 1green_cactus, grey_cactus, brown_breast, 1brown_breast, white_lines, sky, 1blue_sky, blue_sky,
typicality: 0.76 coverts, 1brown_belly, 1brown_nape, white_primary_coverts, 1white_primary_coverts, 1blue, green, blue,
1green, breast, 1brown, black_primary_coverts AND 1white, black_primary_coverts AND white, black_primary_coverts AND 1white AND black, black_primary_coverts AND primary_coverts AND 1white, white AND black_primary_coverts AND 1white, black_primary_coverts AND white AND black, black_primary_coverts AND primary_coverts AND white, black_primary_coverts AND primary_coverts AND 1white AND black, black_primary_coverts AND 1white AND white AND black, black_primary_coverts AND primary_coverts AND white AND 1white, black_primary_coverts AND primary_coverts AND white AND black, brown, black_primary_coverts AND black,
black_primary_coverts AND primary_coverts, black_primary_coverts AND primary_coverts AND black, black_primary_coverts, black_bird,
1grey AND

Scores
Confidence
[✓ Typicality]
Percentage of rule-associated images among dataset
Percentage of concept-associated images among images with the predicted class
Percentage of correctly classified images within rule-associated images

Filter
Percentage of concept-associated images among images with the predicted class (support) filter
Confidence filter

Order Of Representation
Alphabetical order
Confidence
[✓ Typicality]
Percentage of rule-associated images among dataset
Percentage of concept-associated images among images with the predicted class
Percentage of correctly classified images within rule-associated images

Settings
The set of images considered for the computation
[✓ All images]
Only images with correct predictions
Only images with incorrect predictions
Compute typicality based on:
Rule mining
[✓ Statistical scores (one predicted label versus all the others)]

Figure 16: Display of global explanations within our probe. (1) shows the overall explanations, (2) shows the class-specific explanations, (3) shows the settings that can be tuned to compute the explanations. In (1), we show the global explanations displayed within the interface: (a) shows the typicality score, (b) the frequency of times the concept (or rule) is salient within the dataset, (c) the percentage of times when the image where the concept is salient got a correct inference, (d) and conversely when it got an incorrect inference.