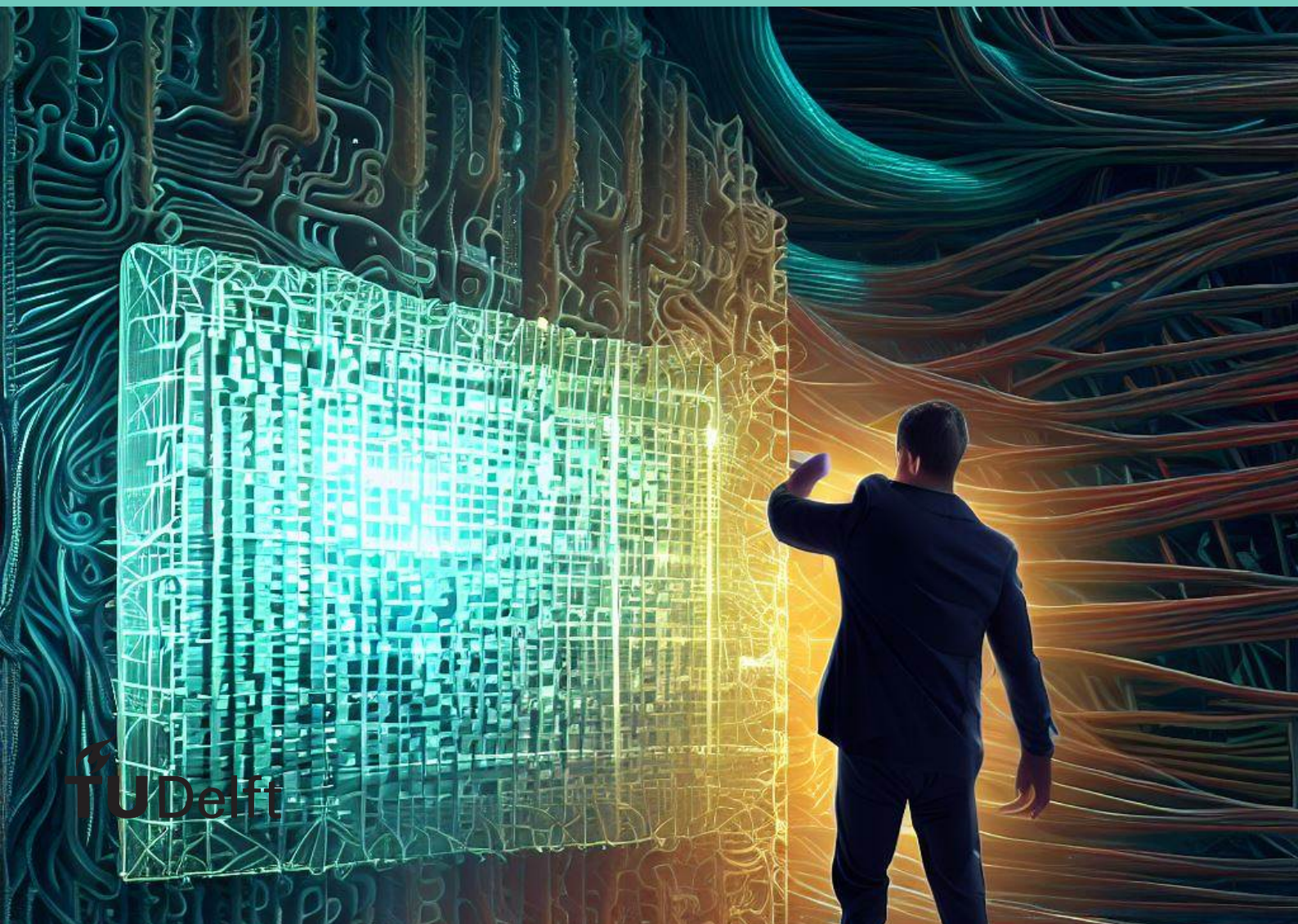


A System for Model Diagnosis centered around Human Compu- tation

Ziad Nawar



A System for Model Diagnosis centered around Human Computation

by

Ziad Nawar

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Wednesday August 23, 2023 at 14:00 .

Student number: 4765575
Project duration: November 30, 2022 – August 23, 2023
Thesis committee: Prof. dr. ir. Avishek Anand, TU Delft, Chair
Dr. Jie Yang, TU Delft, Supervisor
Dr. Elvin Isufi, TU Delft

This thesis is confidential and cannot be made public until August 23, 2023.

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.

Preface

As I stand on the threshold of presenting this thesis to the academic community, I am filled with a profound sense of gratitude and accomplishment. This journey has been one of intellectual growth, personal development, and unwavering support, without which this work would not have come to fruition.

First and foremost, I extend my heartfelt gratitude to my family and my fiancée. Their unwavering encouragement, understanding, and belief in my abilities have been my constant pillars of strength throughout this endeavor. Their sacrifices, patience, and love have been my driving force, reminding me of the importance of balance and the pursuit of excellence in all aspects of life.

I am deeply indebted to my esteemed supervisor, PhD candidate Lorenzo Corti, for his exceptional guidance, insightful feedback, and unending patience. His mentorship transcended the traditional advisor-student relationship, shaping not only my research but also my scholarly outlook. His dedication to the pursuit of knowledge and his commitment to nurturing my academic growth have been instrumental in shaping the course of this work.

I extend a special expression of gratitude to Dr. Agathe Balayn. Her valuable insights, discussions, and collaborative spirit enriched my research and expanded my horizons. Her willingness to engage in rigorous debates and offer constructive criticism significantly contributed to the refinement of this thesis.

Furthermore, I am grateful for the guidance and support provided by my supervisor, Dr. Jie Yang. His expertise, intellectual rigor, and commitment to academic excellence have played a pivotal role in shaping the scholarly foundation of this work.

Finally, I wish to acknowledge the collective efforts of my peers, mentors, and friends who have contributed to my academic journey. Your discussions, debates, and camaraderie have added immeasurable value to my research experience.

In closing, I am acutely aware that this thesis is not solely my accomplishment. It is the result of the cumulative efforts of those who believed in me, supported me, and guided me. It is my hope that this work contributes meaningfully to the field and stands as a testament to the power of collaboration, dedication, and the pursuit of knowledge.

*Ziad Nawar
Delft, August 2023*

Contents

1	Introduction	3
2	Related Work	7
2.1	Explainability methods	7
2.1.1	Saliency Methods.	7
2.1.2	Automated Concept-based Explanation (ACE)	7
2.2	Debugging Frameworks and Crowdsourcing	7
2.3	What you should know	8
2.4	Brick routine	8
2.5	Connecting the Dots	8
3	Overview of the Problem	9
3.1	Problem definition: Model debugging	9
3.2	Definition of our objectives	11
3.2.1	Incorporating the notion of certainty	11
3.2.2	Formulating concrete objectives	12
4	Solution	13
4.1	Overview of the Solution Design.	13
4.1.1	Harnessing Human Involvement for System Functionality	13
4.1.2	Essential Human Operations	13
4.1.3	Ordering Human Operations via Workflows and Batches	14
4.2	User Profiles	15
4.3	User Story	15
4.4	Functional requirements	16
4.4.1	Developer	16
4.4.2	Worker	16
4.4.3	Backend.	16
4.4.4	Results	17
4.5	Non-functional requirments	17
5	System Design	19
5.1	System Architecture	19
5.1.1	Requirements.	19
5.1.2	Architecture	19
5.1.3	Technology Stack.	20
5.2	Operations	20
5.2.1	Collect Global Expected Mechanisms (GEM) - O.GEM.C	20
5.2.2	Map GEM to Local Expected Mechanism (LEM) - O.LEM.C	21
5.2.3	Validate LEM - O.LLM.C/val	22
5.2.4	Check for Extra Concepts - O.LLM.Extra	23
5.2.5	Collect Complete Local Learnt Mechanism (LLM) - O.LLM.C/comp	24
5.2.6	Mapping LLM to GEM - O.GEM.T	24
5.3	Batch	25
5.3.1	Batch Size	25
5.3.2	Batch Composition	25
5.3.3	Batch Order.	25
5.4	Workflow	26
5.4.1	Main Workflow	26
5.4.2	Operation states	26
5.4.3	Workflow Types.	27

5.5	Data model	30
5.5.1	Session Component	30
5.5.2	Logging Component	31
5.5.3	Core Component	32
5.6	Configuration Panel	33
6	Experimental setup	37
6.1	Automated Answering Tool.	37
6.1.1	Annotation tool	37
6.1.2	Simulator	40
6.2	Experiments	40
6.2.1	Data-sets	40
6.2.2	Configurations	41
6.2.3	Design.	42
7	Results and analysis	43
7.1	On the correctness and informativeness of the tool's outputs	43
7.1.1	Correctness of the outputs	43
7.1.2	Informativeness of the outputs.	46
7.1.3	System states over time	48
7.2	Impact of batch size on Informativeness	51
7.3	Cost-efficiency in high worker-accuracy contexts.	51
7.4	Cost-efficiency in Scenarios with Lower Worker Accuracy	51
8	Discussion & Conclusion	55
8.1	Discussion of the Results: Positives, Limitations, & Future Research Opportunities	55
8.1.1	Guidelines on the use of the system.	55
8.1.2	Limitations of the system.	56
8.1.3	Limitations of the experimental setup	57
8.2	Conclusions.	58

Abstract

Machine learning (ML) systems for computer vision applications are widely deployed in decision-making contexts, including high-stakes domains such as autonomous driving and medical diagnosis. While largely accelerating the decision-making process, those systems have been found to suffer from a severe issue of reliability, i.e., they can easily fail on serving data that are slightly different from the data captured during their training phase. Such an issue has resulted in undesired outcomes with safety, ethical, and societal concerns across various applications, such as numerous examples of semi-automatic cars causing accidents on the road. In this thesis, we hence develop a system in order to support ML practitioners in debugging their computer vision models, even before deploying them and having access to serving data.

We take inspiration from prior ongoing works in order to formulate the current diagnosis problem, identify its challenges, and envision a human-computation-based solution. We then thoroughly analyse the requirements for developing a system instantiating the solution, actually design such a system, and implement it in a well-functioning, full-fledged, highly-modular, and easily-customizable system. The solution is based on the definition of human computation operations, that, altogether, allow to a) identify the mechanisms a human would expect the model to learn in an ideal world, b) identify the mechanisms the model has actually learned (via annotations of saliency maps), and c) to compare these two sets of mechanisms to conclude about the good behavior of the model. The solution is especially made to account for certainty issues in the work of the human workers, and to handle ambiguous granularities in the concepts the model might have learned. To the best of our knowledge, our work is the first system that allows an ML practitioner to first identify their own goals for debugging a model (among a large diversity of goals), accounting for their limited monetary budget, then to configure a debugging session according to these goals, and finally to fully-automatically run the system with such configuration to obtain a model debugging report.

Finally, we conduct a thorough investigation of our system. First, we set-up to understand the correctness and informativeness of the outputs, by running the system with various configurations on different models trained using various datasets, for which the biases are more or less controlled. This first evaluation particularly shows that the outputs and its implementation are correct. With these outputs, we are able to identify the biases that have been injected in the model, as well as to learn about previously unknown behaviors of highly-common models that are used by many practitioners. Second, we evaluate the cost-effectiveness of running the system. For that, we ran tests in two settings: when the human workers might make mistakes (e.g., due to a lack of expertise, the complexity of the task, or inattention), and when human workers are fully accurate. We vary the configurations of the system (e.g., the order in which the human operations are conducted, the number of workers allocated at the start of the debugging session) within the two settings, and we observe how the number of human operations needed evolve, in order to reach correct system outputs. We find that the system's output is potentially relevant, informative and complete. The system output is provide an in depth analysis of the model's behaviour and unravel what the model comprehends, where it falls short, and what it should ideally have grasped.

All in all, in this thesis, we build the system and thoroughly evaluate it. While we identify a number of conceptual and practical limitations of this system (e.g., difficulty to annotate concepts, potentially high cost), our work constitutes a first step towards developing complete solutions to help practitioners debug their system. We encourage readers to build on our work, in order to further optimize our system for cost. Note that we make all our code publicly available for anyone to re-use our system, or reproduce our experiments. Code on github: <https://github.com/delftcrowd/ARCH-system>

1

Introduction

Imagine a world where every decision you make is aided by an Artificial Intelligence (AI) system. It's happening right now, as AI is being integrated into our daily lives at an unprecedented rate. However, with great power comes great responsibility, and the question of whether we can trust these AI systems remains a pressing concern. Unlike traditional software, AI is often seen as a black box, with its inner workings shrouded in mystery. This lack of transparency can make it difficult to determine whether an AI system is making the right decisions. What's more, many public reports have shown the concrete dangers AI systems can cause, such as safety issues, e.g., wrong medical diagnostics provided by the systems, self-driving car accidents, or discrimination due to various biases. In this thesis, we will work on helping AI practitioners develop less dangerous AI systems, by making the inner workings of the AI systems more transparent and more easily debugged. Artificial intelligence comes in many different forms as understanding Natural language (Natural language processing), recognising speech (speech recognition) and understanding images (Computer vision). We will focus only on computer vision problems, as this is a common application of AI systems nowadays, it is known to be highly harmful, and it presents many technical challenges that are yet to be answered.

Transparency in computer vision can be defined as the ability to provide meaningful and interpretable information about the internal processes and mechanisms of an AI model, as well as its inputs, outputs, and performance [Larsson and Heintz, 2020]. Interpretable methods in computer vision are techniques that aim to make computer vision models more transparent and understandable for humans [Escalante et al., 2018]. Interpretable methods can be divided into two categories: intrinsic methods that design models with inherent interpretability, such as decision trees or sparse coding, and post-hoc methods that provide explanations after the model has been trained, such as saliency maps or feature visualizations [Biessmann and Refiano, 2019]. Interpretable methods can help users gain insights into the model's behavior, logic, and limitations [Escalante et al., 2018]. The explanation methods can be found in the literature of Explainable AI. The overarching function of explainable methods in computer vision is to provide insights into how and why a computer vision model makes predictions or decisions based on visual data [Escalante et al., 2018]. Explainable methods can help users understand the strengths and limitations of a computer vision model, identify potential biases or errors, and improve trust and confidence in the model's outputs. Some examples of explainable methods in computer vision are Grad-CAM, LIME, LRP, and XRAI [Escalante et al., 2018]. Yet, it has been shown several times now that AI practitioners are not fully supported by these explanation methods [Balayn et al., 2022; Balayn et al., 2023]: they need further support to interpret the outputs of the methods and make use of them, in order to really be able to assess and debug the computer vision models they are working on.

On the other side, testing methods in computer vision are techniques that evaluate the performance and robustness of computer vision models on different tasks and datasets [Gao et al., 2022]. In a way, they are fully-automated methods for diagnosing models, instead of explanation methods that require the "practitioner-in-the-loop" for diagnosing the model errors. Testing methods can be divided into two categories: offline methods that use predefined metrics and benchmarks to measure the accuracy and efficiency of a model [Wiley and Lucas, 2018], and online methods that use interactive feedback and adaptation to identify and fix errors or failures of a model [Gao et al., 2022]. Various automatic Testing frameworks exist such as MLFlow, Snorkel, and Sagemaker. Some monitoring methods can

also be found for detecting model errors by monitoring training issues (e.g., vanishing gradients) or testing model output against predefined rules. All these testing methods can help users improve the quality and reliability of computer vision models [Gao et al., 2022]. Yet, little research has been done on allowing the developers to understand model errors beyond actually identifying these errors. This is highly problematic as being able to diagnose model errors is essential to developing reliable machine learning systems: it allows developers to appropriately fix the issues, and to hedge the risk of producing new errors outside the considered contexts. In this work, we aim at bridging this gap between identifying potential model errors and understanding them further.

To support ML practitioners in understanding model errors, a few researchers [Arous et al., 2021; Kortylewski et al., 2021; Singla et al., 2020; Stacey et al., 2022] have recently proposed to shift from monitoring test accuracy to investigating the meaningfulness of the features learned by the model. The underlying assumption is that when model features align with human expectations, models should be more reliable on any data (including inference data) and hence fewer errors should occur. This would avoid models learning spurious correlations, and performing accurately on test data only based on wrong features [Hendricks et al., 2018]. This is the idea we follow in this work. Yet, prior works are limited in that they do not yet propose a full-fledged, and cost-efficient system to do so, despite the need for it and the difficulty of developing such a system.

Specifically, to monitor feature meaningfulness, we need to understand what the model understands and what it should have learned. Both needs pose various challenges. First, this problem starts with how the model can express its understanding and this is difficult as it is statistical correlation patterns the model learns from the data as decision mechanisms, generally represented as a large amount of numerical parameters that are not directly intelligible to humans which is called knowledge representation gap. The other part of the problem is what it should have learnt, this requires domain knowledge of the classes in our tasks, which is often not easy to obtain, and potentially inexistent. Additionally, in order to collect all of this information, we require human computation (as experts or lay person depending on the task) to elicit the knowledge of this domain. Human computation is known to be challenging due to the various errors human workers can make for various reasons. This collection of knowledge can be referred to as a knowledge acquisition bottleneck. Finally, the knowledge collected from humans has not yet been formalized, but it would need to be represented as symbolic knowledge (e.g., semantic concepts with properties and relation between those concepts) in order to be able to compare what the model understands and what it should have learned. Both the knowledge representation gap and the knowledge acquisition bottleneck are the main challenges to diagnosing a model.

Hence, in this work, we will rely on prior human computation works that can be useful towards the idea of debugging a model via its feature meaningfulness, and push their ideas further by designing a cost-efficient system relying on these works. We will answer the following research question: how can we design a cost-efficient system that allows us to comprehensively debug a computer vision model via reasoning on its feature meaningfulness and leveraging human computation capabilities? To do so, we divide our question into several sub-questions: what prior literature on model debugging and human computation can we leverage in our system design? What are the functional and non-functional requirements for a model debugging system and how to translate them into a system design? To what extent does our system provide correct debugging information in a cost-efficient manner?

In order to answer these questions, we first survey the literature in order to identify relevant human-computation frameworks that can serve to identify what the model understands or what it should understand (Chapter 2). Especially, human computation becomes a core component for the solution we will propose: humans will describe the model behaviour by annotating the saliency maps related to the model predictions, and they will also be a source of knowledge on the domain of the task. This will however come with a lot of uncertainty and problems, i.e., scope gap and uncertainty about the human accuracy. We will hence formulate the debugging problem accounting for the complexities of involving human computation (Chapter 3).

We introduce the proposed solution (Chapter 4), defining an architecture that allows the system to be easily expandable. Then we will then propose the system design (Chapter 5). This design will rely on a few important components: a set of human computation operations, various workflows and configurations of these workflows to order the operations, a comprehensive data model to process the outputs of the operations, and a user-interface for the AI practitioner to input its system configurations. There, we will abundantly use the notions of saliency maps and concepts, as well as learned and expected mechanisms in order to represent the different information collected from the various operations, and

aggregate them to achieve the debugging report.

Finally, we will evaluate the system (Chapter 6), and discuss the results to inform future works (Chapter 7). In order to proceed to a comprehensive evaluation, we will use three datasets and models where the biases are more or less controlled. This will allow us to verify both the correctness of the system and its informativeness for discovering previously unknown problematic model behaviors.

All in all, we contribute a new formulation of the debugging problem, the design and implementation of an initial solution, its comprehensive evaluation, and a convenient practical resource in terms of code for future improvements on the initial solution and its evaluation. We hope that our system will be seen as a first step to foster more work on model debugging before model deployment with a special interest onto model features and human computation.

2

Related Work

The understanding and interpretation of machine learning models have been crucial areas of research, especially in the realm of deep neural networks. In this chapter, we delve into several aspects of the related work, starting with the Automated Concept-based Explanation (ACE) framework, which addresses the challenges of concept-based explanations in machine learning models.

2.1. Explainability methods

2.1.1. Saliency Methods

To enhance interpretability, methods like saliency maps and ACE explanations stand out. Saliency maps, as introduced by [Mundhenk et al., 2019] and further explored by [Adebayo et al., 2018] and [Wang and Gong, 2022], offer a visual representation of neural network decisions by highlighting important regions in images. This technique has been pivotal in understanding the inner workings of deep neural networks, especially in complex domains like medical imaging [Mundhenk et al., 2019].

2.1.2. Automated Concept-based Explanation (ACE)

ACE, as proposed by [Ghorbani et al., 2019], introduces a novel framework for concept-based explanation in machine learning. Traditional explanation methods often rely on per-sample features, but ACE advances this paradigm by identifying higher-level concepts that hold significance across entire datasets. The authors lay down principles that a concept-based explanation should adhere to, including meaningfulness, coherency, and importance.

The ACE algorithm involves a three-step process: segmentation, clustering, and concept importance computation. In the segmentation step, images are divided into segments at various resolutions, capturing diverse textures and objects. The subsequent clustering step groups similar segments using a CNN-based similarity metric, with outlier removal enhancing coherency. Concept importance scores are then calculated using the Testing with Concept Activation Vectors (TCAV) metric.

Through rigorous experiments, ACE's effectiveness is demonstrated, showcasing its ability to satisfy the principles of concept-based explanation. The method's global explanation capability, efficiency, and insights into machine learning models underscore its significance in providing interpretable results.

2.2. Debugging Frameworks and Crowdsourcing

Debugging frameworks play a pivotal role in ensuring the reliability and performance of machine learning models. Notable frameworks like 3DB, as presented by [Leclerc et al., 2022], utilize photorealistic simulation to test and debug vision models. Such frameworks have the potential to uncover vulnerabilities and enhance the decision-making understanding of models, offering critical insights for refinement.

Human computation and debugging

Moreover, the integration of crowdsourcing into the realm of machine learning, often referred to as crowd computing, has been gaining traction. This approach, highlighted by [Yang et al., 2023], brings human intelligence into the loop, facilitating the solving of intricate problems that AI and ML struggle to

handle alone. Crowd computing not only enhances data annotation for model training but also enables real-time, human-in-the-loop computational solutions for challenging tasks.

2.3. What you should know

According to the paper “What Should You Know? A Human-In-the-Loop Approach to Unknown Unknowns Characterization in Image Recognition,” the main output of the paper is a human-in-the-loop, semantic analysis framework for characterizing unknown unknowns at scale [Sharifi Noorian et al., 2022]. The paper focuses on unknown unknowns characterization to drive a deeper understanding of unknown unknowns and more effective identification and treatment. The framework involves human contributors to specify both what the model should know and describe what it really knows, while minimizing the cognitive load of the tasks leveraging scene graph extraction.

2.4. Brick routine

“A Human-In-the-Loop System for Interpreting Image Recognition Models” is a research paper that presents the design and implementation of Brickroutine, a system that uses a trained model and human annotations to give semantic interpretations to image classification problems. The authors aim to bridge the gap between performance and interpretability of AI by combining human and computational intelligence [Ziengs, 1970].

2.5. Connecting the Dots

The landscape of related work in understanding and interpreting machine learning models is rich and diverse. Starting with the ACE framework’s novel approach to concept-based explanations, we transition to the effectiveness of saliency maps in explaining neural network decisions. Debugging frameworks like 3DB contribute to refining model performance by uncovering vulnerabilities, while the integration of crowdsourcing and crowd computing introduces human intelligence to enhance AI capabilities.

Furthermore, addressing the unknown unknowns in image recognition through frameworks like Brickroutine showcases the potential of human-in-the-loop systems. The challenges and opportunities in machine learning testing emphasize the ongoing efforts to ensure the reliability and robustness of machine learning models.

3

Overview of the Problem

3.1. Problem definition: Model debugging

In the development of computer vision models, a significant challenge arises when determining if the model is good enough to be deployed and used in real life applications. This challenge can be assessed by evaluating metrics such as accuracy, precision, and f1-score, which provide an overview of the model's performance. However, relying solely on these metrics may be misleading when attempting to gauge the model's understanding. One potential issue is the disparity between the data used to train the model and the data it encounters in production. Although the model may achieve high scores on the aforementioned metrics, it could potentially yield lower accuracy when faced with production data. This situation may arise if the model has learned to focus on a specific set of pixels that do not accurately correspond to the image's main features that could be used to predict the image label. To address these challenges, it is crucial to develop a methodology that identifies the exact set of pixels the model has learned, should have learned, and what it has missed in terms of learning.

The set of pixels on which the model focuses could potentially represent a **concept**, referred to as such throughout this paper, when interpreted from a human perspective. However, it is important to note that these sets of pixels are merely low-level representations of concepts, characterized by raw numerical values. In contrast, human language employs words to convey concepts with real-world meanings. Therefore, in the development of our tool, it is imperative to account for this disparity and bridge the gap between low-level features and the corresponding concepts—a challenge commonly known as the **semantic gap problem**.

Furthermore, our research also aims to explore the concepts that the model should have learned which we refer to as **expected** in the rest of this document. To achieve this, we are interested in eliciting insights from domain experts or individuals with varying levels of expertise, depending on the task at hand. However, we anticipate encountering a potential obstacle in this regard. The individuals providing these concepts may inadvertently overlook certain aspects, leading to a **knowledge gap** between the concepts they are able to recall and articulate, and the comprehensive range of concepts that exist within the designated classes. Overcoming this knowledge gap poses an additional challenge for our research.

Definition: Within our system, we define **concept** as tangible and physical materialistic entity associated with an object. It encompasses the semantic meaning of the object and incorporates various attributes, both physical and metaphysical, such as color, texture, size, and more. Recognizing and comprehending these concepts play a crucial role in identifying specific objects. For instance, the concept of a 'kitchen' can be constructed by combining several sub-concepts, including an oven, fridge, stove, and other common attributes typically found in a kitchen setting. Thus, a collection of less abstract concepts collectively forms a more abstract concept.

In our research, we acknowledge the importance of understanding what the model comprehends, which can be categorized into two distinct groups. Firstly, we examine the image-level understanding, which we term **local learned concepts**. Secondly, we investigate the set of concepts that the model grasps for each class, referred to as **global learned concepts**. While both categories contribute to our understanding of the model's comprehension, they differ in their scope and relevance.

Global learned concepts are likely applicable to a substantial portion of images within a given class. These concepts provide insights into the common characteristics and features shared among images in that class. By examining the global learned concepts, we can gain a holistic understanding of the model's understanding of the class as a whole. However, it is important to note that these concepts may not capture the unique and image-specific details present in individual images.

Conversely, locally learned concepts pertain to image-specific understanding. They encompass the concepts that the model associates with individual images but may not be relevant or applicable on a global level within the class. These concepts may capture specific details or characteristics that are unique to each image, but do not contribute significantly to the overall understanding of the class as a whole. Therefore, when considering the global level of a class, it is necessary to exercise caution in including locally learned concepts that may introduce irrelevant or misleading information. Note that locally learned concepts are traditionally identified in the literature using saliency maps, which also represent an essential part of our work.

Definition: Another crucial element within our system is the **saliency map**. This map captures the visual attention or significance of distinct regions within an image. It illuminates regions that represent the pixels the model uses to reach its prediction. Through the utilization of human computation, the analysis of the saliency map directs workers' attention to the most pertinent regions for concept identification and comprehension. The combination of the saliency map and workers' analysis offers a valuable resource to guide the system's decision-making procedures and enhance its overall performance.

By distinguishing between local learned concepts and global learned concepts, we can delve into the nuances of the model's understanding and gain valuable insights into its comprehension at both image and class levels. This differentiation allows us to analyze the relevance and applicability of concepts in a more nuanced manner, providing a comprehensive understanding of the model's comprehension abilities.

We also strive to illustrate the relationship between concepts and classes in our research. We introduce the term "mechanism" to describe this relationship, wherein the set of concepts employed to identify the class of an image is referred to as the **antecedents**, while the identified class itself is referred to as the **consequent**. By examining a mechanism, we can gain insights into the interplay between concepts and classes within the model.

Furthermore, it is important to extend our analysis beyond solely focusing on the identified class. We recognize the value of considering the mechanisms leading up to the antecedents, as they enable us to assess whether the model has fully acquired and comprehended the associated concepts. This broader perspective allows us to evaluate the model's understanding not only in terms of correctly identifying the class but also in terms of the extent to which it has learned and internalized the relevant antecedents.

By investigating the relationship between concepts and classes through the mechanism framework, we can deepen our understanding of how the model makes classifications. This approach enables us to explore the significance and completeness of the learned concepts, while also shedding light on any potential gaps or limitations in the model's comprehension. By incorporating both the antecedents and the consequent within our analysis, we gain a more comprehensive understanding of the model's learning capabilities and its ability to effectively utilize the acquired concepts in the classification process.

To address these challenges, a methodology should be developed to identify the exact set of pixels the model has learned, should have learned, and what it has missed. Bridging the semantic gap problem between low-level features and concepts is crucial. Furthermore, understanding both local learned concepts and global learned concepts contributes to a comprehensive understanding of the model's comprehension abilities. Differentiating between these categories helps analyze the relevance and applicability of concepts at image and class levels. Exploring the relationship between concepts and classes through the mechanism framework provides insights into the model's classification process, the completeness of learned concepts, and potential limitations in comprehension. By incorporating both the antecedents and the consequent within the analysis, a more thorough evaluation of the model's learning capabilities and effective utilization of acquired concepts can be achieved. Overall, these requirements pave the way for a deeper understanding of computer vision models and their deployment.

Term	Meaning in Context
Disparity	Difference or mismatch between two things, in this case, the data used for training and the data encountered during production.
Semantic Gap Problem	The challenge of bridging the difference between low-level features extracted from data and high-level concepts or meanings that humans associate with those features.
Knowledge Gap	The difference between the concepts that domain experts or individuals articulate and the full range of concepts within designated classes, leading to potential omissions in understanding.
Learned Concepts	Concepts that the model understand from the training process on the data
Expected Concepts	Concepts that the users think they are important to be associated with a specific class
Local Concepts	Concepts that are associated with an image
Global Concepts	Concepts that are associated with multiple images within the same class
Local Learned Concepts	Concepts that a model associates with individual images, which may not apply globally across a class of images.
Global Learned Concepts	Concepts that apply to a substantial portion of images within a class, offering insights into common characteristics.
Mechanism	The relationship between concepts and classes, where concepts are the antecedents that lead to the identification of a class (consequent).
Antecedents	The set of concepts that are used to identify the class of an image within the mechanism framework.
Consequent	The identified class of an image within the mechanism framework, based on the antecedents.

Table 3.1: Summary of Terms

3.2. Definition of our objectives

3.2.1. Incorporating the notion of certainty

While an ideal scenario would entail unanimous agreement among the crowd workers providing input, it is essential to acknowledge the possibility of differing opinions and lack of consensus. Consequently, our system should account for this variability in worker input and incorporate measures to address it effectively.

To ensure the reliability of the collected data, we propose defining a certainty level for each operation within the system. Once this predefined certainty level is reached, we can consider the gathered data as trustworthy and suitable for utilization within our system. However, it is important to note that achieving higher certainty levels can be resource-intensive, particularly when involving extensive human computation. This aspect becomes particularly relevant when employing paid crowd workers.

In our envisioned system, most operations will predominantly involve binary (yes or no) questions, textual input for concept elicitation, and the drawing of bounding boxes around identified concepts within images. Developers will have the flexibility to determine the desired certainty level for each operation, enabling them to balance costs and the reliability of insights derived from the system's output. Consequently, each operation will be executed repeatedly until the specified certainty level is attained. The specific certainties assigned to each operation will vary depending on the targeted signals and the available resources for each operation.

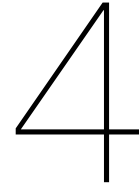
By allowing developers to customize the certainty levels for individual operations, we ensure that the system is adaptable to their allocated budget and constraints while still providing valuable insights based on the chosen level of certainty. This approach allows for a flexible and cost-effective utilization of the system's output while maintaining the necessary level of confidence in the collected data.

3.2.2. Formulating concrete objectives

As a part of the diagnosing session data is collected within our system serves multiple purposes, encompassing various objectives that provide valuable insights into the inner workings of the model. These objectives primarily focus on debugging the model and understanding its successes and failures in terms of identified and missed mechanisms. The following objectives are defined to facilitate these goals:

1. **Counter example:** The objective of identifying counter examples aims to uncover instances where the model failed to recognize specific concepts for which their antecedents exist within an image. By closely examining these counter examples, developers gain valuable insights into the concepts that the model should have learned but did not. This objective sheds light on potential areas for improvement and provides guidance for further training to enhance the model's performance.
2. **Satisfaction:** The satisfaction objective involves assessing the model's ability to accurately identify the expected mechanisms. Developers are interested in determining the extent to which the model successfully identifies the anticipated mechanisms. This objective provides a quantitative measure of the model's performance and its ability to meet the desired expectations. By evaluating the satisfaction level, developers can gauge the model's effectiveness and make informed decisions based on its outputs.
3. **Global mechanism usage:** The objective of understanding the distribution of learned mechanisms across the entire class compared to other mechanisms focuses on gaining deeper insights into the patterns and tendencies the model acquires during its training process. Developers aim to comprehend the ratios of learned mechanisms within the class, allowing for a comprehensive understanding of the model's comprehension and utilization of different concepts. Analyzing these ratios provides valuable insights into the model's learning capabilities and the significance of different mechanisms within the classification process.
4. **Mechanism plausibility:** This objective explores the plausibility of mechanisms according to the model's perspective. It involves evaluating the ratio of entries where a Global Expected Mechanism (GEM) was not proposed compared to entries where it was proposed or not. Additionally, it considers the number of times the mechanism is correctly mapped to an image and learned by the model, as well as the instances where the mechanism is learned despite not being part of the global expected mechanisms. This objective provides a deeper understanding of the model's reasoning processes and its ability to recognize and utilize mechanisms effectively.

By collectively addressing these objectives, developers can gain valuable insights into the model's performance, identify areas for improvement, and enhance their understanding of the underlying mechanisms learned by the model. These objectives guide the evaluation and refinement of the model, offering valuable guidance for its optimization and future development.



Solution

After conducting a comprehensive analysis of the problem and identifying the necessary components of our system, as discussed in Chapter 3, we now present the system itself, along with its requirements and the chosen technology stack. In the following sections, we provide an in-depth exploration of the system's design, highlighting the key elements that contribute to achieving our research objective.

4.1. Overview of the Solution Design

In light of the challenges discussed earlier, we present a comprehensive solution that addresses the identified requirements and incorporates the necessary components for the system to function effectively.

4.1.1. Harnessing Human Involvement for System Functionality

The successful operation of our system heavily relies on the invaluable contribution of crowd workers in providing essential data input. Ensuring the quality and integrity of this input is of utmost importance, as it directly influences the system's functionality and subsequent decision-making processes. Our system is specifically designed to prioritize human involvement, placing significant emphasis on human computation over the utilization of intelligent robots or algorithms.

The data essential for the system's functionality primarily comprises images used in training the machine learning model. Additionally, we anticipate obtaining saliency maps associated with these images, regardless of the specific method employed to extract them. These saliency maps, together with the images, serve as the foundational input to initiate the system and generate tasks for crowd workers.

4.1.2. Essential Human Operations

To facilitate data elicitation, it is crucial for the system to operate as a crowd computing system. In this context, tasks should possess clear and concise definitions, making them both easily executable and understandable. Our envisioned system encompasses multiple operations to support the data collection process. We define the first essential components of our system next.

Definition: An **operation** serves as the fundamental building block within the system, directed towards the analysis of an individual sample or instance. As elaborated in Section 4.1.2, these operations manifest diverse forms contingent upon their designated objectives. For instance, an operation might encompass the collection of global expected mechanisms pertinent to a specific class, while an alternative operation could entail the collection of all highlighted concepts within a saliency map.

Definition: A **task** comprises a collection of operations of the same type that are executed on a set of images (batch). Tasks are assigned to a minimum number of workers, as determined by the developer. Each worker can execute a task only once. For instance, a task could involve gathering the global expected mechanisms for three different classes.

Firstly, an operation is required to **collect global expected mechanisms** from crowd workers. This operation would involve gathering input from the workers in the form of antecedent lists corresponding to specific concepts.

Given that not all concepts collected in our system may be present in each image, an additional operation is necessary to identify the actual concepts existing within an image. This operation serves the purpose of determining whether the model utilized these concepts during evaluation - known as **Map GEM to LEM** later in this document.

Furthermore, to verify the concepts identified by the model, a separate operation is needed to identify the highlighted concepts within an image and determine whether the model fully or partially selected them as relevant to the class- known as **Validate LEM** later in this document.

In addressing our concerns regarding the potential incompleteness of worker-contributed concepts, another operation is designed to identify whether there exist any missing highlighted concepts within an image which has not yet been captured by the system. This situation could arise if the concepts used by the model are unrelated to the class, yet were employed by the model to classify the image as belonging to a specific class- known as **Extra Concept** later in this document.

Also Continuing on the previous operation an operation called **Collect Complete LLM** is implemented if more concepts still exist in the image but not yet captured by the system in this case Users are expected to identify those concepts

Lastly, to account for the potential discovery of new mechanisms by the model that are not present in our global mechanisms, a **Map LLM To GEM** operation is implemented to identify these mechanisms. Subsequently, it can be determined whether these newly identified mechanisms are applicable to other images or should be disregarded as irrelevant for classifying other images.

By incorporating these operations into our solution, we address the requirements for data elicitation and mitigate the limitations associated with incompleteness. This comprehensive approach enables effective collection and evaluation of concepts and mechanisms, ultimately enhancing the system's ability to accurately classify images.

Mechanisms evaluation at different granularities Our system is intricately designed to maximize information extraction from input data. In doing so, we traverse through distinct levels of mechanisms, focusing on two specific tiers that hold paramount significance within our framework namely class level and lower level.

Class Level: At the class level, mechanisms are intricately linked to the consequent, which are defined as the class or label corresponding to the given image. Within this level, mechanisms revolve around abstract concepts, excluding detailed concepts related to any concepts beyond the class label itself. To illustrate, consider the example of a "Kitchen." In this scenario, we are primarily interested in mechanisms associated with abstract concepts such as "oven," "sink," and "fridge."

Lower-Level: In addition to the previous level, our attention turns towards mechanisms that constitute the concepts inherent to the class of interest. The consequents of mechanisms at this level embody concepts that are found within the specific class we are investigating. For instance, our curiosity lies in determining whether the "fridge" has been accurately identified. This involves delving into sub-concepts of the "fridge," including aspects like the "fridge door" and the "fridge handle."

By delving into mechanisms across these two tiers, we develop a holistic comprehension of our framework's proficiency in extracting pertinent information and effectively correlating concepts at varying levels of granularity within the given dataset. The decision of whether to assess the model's grasp of sub-concepts is vested in the hands of the developer. It's noteworthy that although we could further explore lower levels of concepts, our design choice limits us to the first level. We anticipate that future enhancements might delve even deeper into the hierarchy of concepts, surpassing the current level of granularity.

4.1.3. Ordering Human Operations via Workflows and Batches

To ensure an organized and systematic execution of operations, we employ a **workflow**. The workflow determines the predefined order in which operations are generated and executed, adhering to specific conditional requirements. It encompasses the order of classes, the number of samples, and the number of concepts utilized within each task. For example, a workflow may involve executing a complete sequence of operations for each mechanism across a batch of 10 samples. This structured approach ensures a consistent and logical execution of operations within the system.

A **batch** represents a group of images generated based on specific constraints. These constraints can include the composition of the batch, such as including images of the same class or different classes

within the same batch. The order of classes across batches can also be configured. Additionally, the number of samples utilized per batch is defined to further refine the batch generation process.

4.2. User Profiles

Within the scope of our system, we define different user profiles based on their specific roles and the purpose of using the system. Understanding these user profiles is crucial for designing an effective and user-centric system. The following user profiles are identified:

Developers: Developers represent the primary users of the system who are interested in debugging machine learning models and gaining insights into the inner workings of the model through the use of our system. Their focus lies in ensuring the quality of the output and configuring the system to achieve a desired level of reliability. Developers aim to utilize the output of the system to enhance their understanding of the model they are developing.

Crowd Workers: Crowd workers encompass a diverse group of users who participate in executing tasks generated by the system. Within this category, we distinguish between two subgroups: experts and lay people.

a. **Experts:** They possess a high level of knowledge and expertise regarding the dataset used for training the system. Experts can be considered professionals who work in the field where this data set originates from. Their inputs can be considered reliable and accurate, making them valuable contributors to the system. Their domain-specific knowledge allows them to provide accurate information and insights.

b. **Lay People:** In contrast to experts, they have a lower level of knowledge and expertise. As a result, their inputs may be less reliable and require additional caution during processing and analysis. While their contributions may still provide valuable perspectives, the system needs to account for the potential limitations and uncertainties associated with their inputs.

Understanding the specific needs and characteristics of each user profile enables us to tailor the system's functionalities and interfaces to ensure an optimal user experience. By considering the unique requirements of developers and crowd workers, we can design a system that effectively supports their objectives and facilitates their interaction with the underlying machine learning model. Developers have the responsibility to assess which group of crowd workers would contribute to the diagnosis session and upon this developers should adjust the system settings accordingly. As when crowd workers are known to be experts it is most probable they are paid more money than lay people and if so the case the required amount of experts should necessary be low to avoid costly diagnosis sessions as they provide high quality input with low number of workers.

4.3. User Story

In order to effectively comprehend the practical implications of the system's operations and their alignment with real-world scenarios, let us delve into a series of user scenarios that underscore the system's utility for developers and crowd workers.

- The developer, keen on scrutinizing their model's performance, opts to identify counterexample samples for a chosen mechanism within a random sample of size k .
- The developer's interest lies in assessing the satisfaction rate of a specific mechanism within a random sample of size k , where this mechanism is both expected and accounted for.
- The developer seeks to discern the mechanisms that have been learned instead of those that were initially anticipated within the unsatisfactory mechanisms. Furthermore, the developer wishes to determine the frequency at which concepts are substituted within these mechanisms.
- To gain a deeper understanding, the developer aims to ascertain the proportion of a particular mechanism employed by the model relative to all other mechanisms within the same class. This analysis can be conducted either on a specific sample or in a broader sense across multiple samples.
- As for the crowd workers, their engagement entails various stages. Initially, they fill in the provided starting code received from the crowd platform. Subsequently, they familiarize themselves with the task instructions, potentially solving a few tasks to ensure their grasp of the intended goals.

Once confident, they navigate through the tasks assigned to them, meticulously engaging with the system's functionalities. Upon completion of the tasks, the crowd worker can finalize the process by clicking the finish button, which prompts a screen displaying the finish code. This code is then utilized to seamlessly conclude the task on the crowd platform.

4.4. Functional requirements

The intricate nature of the system's operations necessitates well-defined functional and non-functional requirements to ensure its effective implementation. By examining the following set of requirements, we can comprehensively address the system's capabilities and characteristics.

4.4.1. Developer

- The frontend must include a dashboard for the developer to upload and monitor the data and the workers output.
- The developer should be able to select certainty threshold value for each operation in the system.
- The developer should be able to select the classes they want to use in the session
- The developer should be able to select which set of mechanisms to evaluate either class level mechanisms or all class level mechanisms and lower levels concepts
- The developer should be able to select workflow type to execute in the session
- The developer should be able to select batch composition either class homogeneity or class diversity
- The developer should be able to select batch order per class, round robin or random
- The developer should be able to input the maximum number of samples to be used in each batch
- The developer should be able to select the total number of samples to use in the session

4.4.2. Worker

- The worker should be able to receive tasks after filling in their worker id
- The crowd worker should be able to to annotate the images using bounding boxes
- The worker should receive a finish code to input it back in the crowd sourcing platform.
- The frontend must contain the six difference operation types which are the sequence of operations that can be executed for one sample.

4.4.3. Backend

- The backend must store data in a database including the images, worker data, session data and the aggregated data from the workers' data
- The backend must generate the tasks for the crowd workers
- The backend should generate the tasks with accordance to the configurations selected by the developer
- The backend should compute the objectives upon session is completed
- The backend should resend the operations which has not yet satisfied the certainty thresholds.

4.4.4. Results

- The frontend should visualize all the bounding boxes for a selected concept for a sample
- The frontend should show Local Learned mechanism for each sample
- The frontend should show local expected mechanism for each sample
- The frontend should show the list of mechanism for each class showing the satisfaction score.
- The frontend should show all the samples within a class with the aggregated workers' output
- The frontend should show the list of samples in an overview for each class

4.5. Non-functional requirements

- The system should ensure data security and privacy
- The system should run smoothly on different browser types (Google chrome and firefox).
- The system should be highly available and recover quickly from failures to ensure minimal down-time.
- The system should be tested for the main features it includes for ensuring a running version when deployed to the crowd workers.
- The system should be designed for ease of maintenance, with clear documentation and modular code.
- The system should have a user-friendly interface
- The system could authenticate the crowd workers

5

System Design

5.1. System Architecture

This section outlines the architectural requirements for our system, which are derived from the previously discussed requirements in Chapter 4. Furthermore, we introduce the chosen technology stack that will support the implementation of our system.

5.1.1. Requirements

To ensure the robustness and effectiveness of our system, we have identified several key architectural requirements:

- **Extensibility** focuses on the system's ability to accommodate future feature enhancements, additional functionality, and integration with external systems. As we acknowledge the iterative nature of our system's development, we anticipate the need for future expansions and improvements. Therefore, we emphasize the development of an extensible system that can seamlessly integrate new features and capabilities in subsequent iterations.
- **Modularity** refers to the division of a software system into distinct and independent modules or components. By adopting a modular design approach, we enable the replacement or modification of individual modules or components without disrupting the overall system functionality. Modularity promotes flexibility, maintainability, and efficient future modifications or updates to the system.
- **Maintainability** refers to the ease with which software can be modified, repaired, or enhanced throughout its lifespan. Given the ever-evolving nature of technology and the potential for future developments, we prioritize maintainability in our system design. Through the adoption of best practices, such as clean and modular code, comprehensive documentation, and standardized development processes, we aim to facilitate efficient maintenance and support, minimizing the risk of errors or complications during future modifications or enhancements.

5.1.2. Architecture

Our system follows the traditional client-server architecture, where users interact with the system through a User Interface (UI) acting as the client. Communication between the client and the server occurs through API requests. To ensure the integrity and reliability of user input, we place a strong emphasis on input validation.

For data storage, we employ a SQL-based database. This database is responsible for storing data in a relational format, allowing for efficient organization and retrieval of information.

The server acts as a vital component that connects the database, where persistent data storage takes place, with the User Interface requests. The User Interface provides access to configure the session settings, worker tasks, and show system output during debugging sessions. Additionally, the server handles algorithm execution and task generation, making critical logical decisions within the system's workflow. It serves as the central component responsible for coordinating and managing the system's operations.

5.1.3. Technology Stack

To implement our system, we have carefully selected a technology stack that encompasses both the front-end and back-end components. This technology stack consists of various tools and frameworks that collectively contribute to the system's functionality and performance.

Front-end Development

For front-end development, we have chosen to utilize **React**, a widely-used JavaScript library for building user interfaces¹. React provides a robust and efficient framework for developing interactive and dynamic UI components, ensuring a seamless user experience.

Back-end Development

In the development of the back end side, we have opted for Python as our programming language. Python offers a broad range of libraries and frameworks that enable rapid development and efficient code implementation.

In our choice of back-end framework, we have opted for **FastAPI**, a contemporary and high-performance web framework that is purpose-built for developing APIs with Python². FastAPI demonstrates outstanding performance and scalability, rendering it highly suitable for fulfilling the requirements of our system. To ensure the reliability and consistency of data transmitted between the front-end and back-end components, we also incorporate **Pydantic**, a data validation and serialization library³. Furthermore, we leverage the capabilities of **SQLAlchemy** to facilitate seamless communication with the database⁴.

To handle data storage, we have integrated **PostgreSQL**, a powerful and reliable open-source relational database management system⁵. PostgreSQL provides robust data management capabilities and ensures secure and efficient storage of our system's data.

API

The API serves as the communication interface between the front-end and back-end components, facilitating seamless interaction and data exchange between different system modules. With our chosen technology stack, we develop and deploy APIs that ensure smooth and reliable communication, enabling efficient system functionality.

By carefully selecting and integrating these technologies, we establish a comprehensive and robust technology stack that empowers our system with the necessary tools and frameworks to deliver a high-performing and user-friendly solution.

5.2. Operations

In this section, we delve into the practical implementation of the operations within our system, accompanied by a thorough discussion of our design choices. An overarching note applicable to all operations concerns the establishment of a maximum number of workers per operation. This precaution is rooted in the understanding that once this preset number is reached, it signifies that the current threshold configuration becomes unattainable, and the operation is considered completed. This approach safeguards against the potential of indefinite resource consumption, as extensive time investments would otherwise be necessary for completion. Remarkably, the maximum number of workers allocated per operation is set at 250, a substantial figure. It is important to underline that this value is user-adjustable within the system. This adaptive feature serves to mitigate the risk of incurring exorbitant expenses while optimizing the operation's efficiency.

5.2.1. Collect Global Expected Mechanisms (GEM) - O.GEM.C

The operation of "Collecting GEM" holds a pivotal role in our system. This operation involves identifying potential combinations of mechanisms that are likely to be present in images of specific classes. Each mechanism is accompanied by a typicality value, reflecting the subjective assessment of users contributing these mechanisms. This typicality value serves to quantify the probability of a particular combination manifesting within an image, facilitating the subsequent ranking of mechanisms. The

¹<https://reactjs.org/>

²<https://fastapi.tiangolo.com/>

³<https://docs.pydantic.dev/latest/>

⁴<https://www.sqlalchemy.org/>

⁵<https://www.postgresql.org/>

typicality value ranges from 0 (minimal likelihood) to 1 (high likelihood), providing insights into the comparative expectations associated with distinct mechanisms. When multiple users propose the same mechanism, we calculate an average typicality value, subsequently assigned to the mechanism within the system.

Furthermore, this operation accommodates a developer-specified certainty level, which plays a critical role in collecting complete GEM. This level determines the threshold for the minimum expected number of mechanisms to be present within the system.

During execution, users provide a list of antecedents linked with mechanisms, along with corresponding typicality values, as shown in Figure 5.1. To ensure data consistency, duplicated mechanisms from the same user are disregarded, considering only the first instance. Users are encouraged to propose diverse concept combinations, each accompanied by its respective typicality value.

The screenshot displays the ARCH system's 'Operation Form' for 'great white shark'. The interface is divided into several sections:

- Header:** ARCH logo with the tagline 'Know what your model doesn't know', and navigation links for 'Operation Form' and 'Dashboard'.
- Antecedent:** A text input field for adding new antecedents, with an 'Add antecedent' button.
- Already entered antecedents:** A list containing 'Sharp teeth' with a 'Remove' button.
- Typicality:** A text input field set to '0', with an 'Add mechanism' button.
- Already entered mechanisms:** A box showing 'Antecedents: Gray Fin, Pointed snout' and 'Typicality: 0.7', with a 'Remove' button.
- Next:** A blue 'Next' button at the bottom right.

Figure 5.1: Screenshot illustrating the "Collecting Global Expected Mechanisms (GEM)" operation within the system.

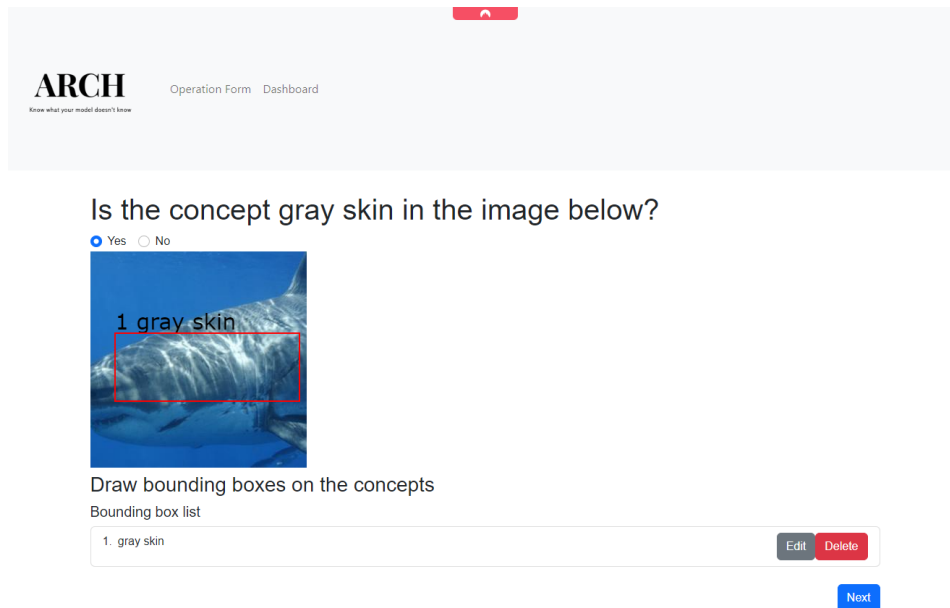
5.2.2. Map GEM to Local Expected Mechanism (LEM) - O.LEM.C

The "Map GEM to LEM" operation serves a dual purpose within the system. Its primary objective is to determine the presence or absence of specific concepts within an image using a binary questioning approach. Participants indicate whether a concept is present within the image. Upon a positive response, participants draw a bounding box around the identified concept. The operation allows only a single bounding box per concept to maintain simplicity.

The choice of a single bounding box per concept is based on creating a streamlined system in its initial version. While the ability to identify multiple concepts through multiple bounding boxes is ideal, the current version uses a single reference bounding box strategy. Participants are instructed to place the bounding box around the leftmost or topmost occurrence of the concept within the image. This standardized reference point ensures consistent user annotations. The concept's presence is determined by a majority consensus among users.

To enhance reliability, we use a threshold of the maximum of one of the two answers to the question which are yes or no over the total, this formulae assesses agreement among raters. The developer sets an agreement threshold, and the operation is sent to the workers until it is met.

Figure 5.2 illustrates the interface for participant input in this operation.



The screenshot shows the ARCH system interface. At the top left, the logo "ARCH" is displayed with the tagline "Know what your model doesn't know". To the right of the logo are the navigation links "Operation Form" and "Dashboard". The main content area contains the question "Is the concept gray skin in the image below?". Below the question are two radio buttons: "Yes" (selected) and "No". A small image of a shark's skin is shown with a red bounding box around a section of its side, labeled "1. gray skin". Below the image, the instruction "Draw bounding boxes on the concepts" is followed by the text "Bounding box list". A list box contains the entry "1. gray skin" with "Edit" and "Delete" buttons next to it. A "Next" button is located at the bottom right of the interface.

Figure 5.2: Screenshot illustrating the "Mapping Global Expected Mechanisms (GEM) to Local Expected Mechanisms (LEM)" operation within the system.

5.2.3. Validate LEM - O.LLM.C/val

The "Validate LEM" operation verifies if a concept has been effectively learned by the model. Participants assess if a concept is highlighted in the saliency map. If highlighted, they draw a bounding box around it, adhering to the same specifications as the previous operation. Participants indicate if the concept is fully or partially highlighted. This step distinguishes the model's grasp of the concept from subconcepts.

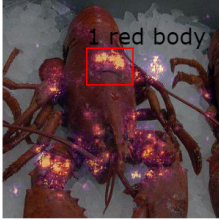

Developer-specified agreement thresholds ensure certainty. The maximum over total formulae is used as used in the previous operation. Figure 5.3 illustrates the GUI interface for participant responses.

ARCH
Know what your model doesn't know

Operation Form Dashboard

Is the concept red body either fully or partially highlighted by the saliency map below?

Yes No

Draw bounding boxes on the concepts

Bounding box list

1. red body Edit Delete

Is the concept red body fully highlighted by the saliency map?

Yes No

Next

Figure 5.3: Graphical user interface for the "Validating LEM" operation.

5.2.4. Check for Extra Concepts - O.LLM.Extra

The "Check Extra Concept" operation identifies additional concepts within the saliency map. Participants assess if the system has already identified all the highlighted concepts in the saliency map or not.

Developer-set thresholds ensure certainty. The maximum over total formulae is used as used in the previous operation. Figure 5.4 displays the GUI interface for participant responses.



ARCH
Know what your model doesn't know

Operation Form Dashboard

Are there any concepts other than those in the following list?

Concepts : gray skin

Yes No

Submit

Figure 5.4: Screenshot illustrating the "Check Extra Concept" operation for participants.

5.2.5. Collect Complete Local Learnt Mechanism (LLM) - O.LLM.C/comp

The "Collect Complete LLM" operation aims to identify and record prominently highlighted concepts within the saliency map. Participants identify additional concepts highlighted in the map and draw bounding boxes around them. They indicate whether concepts are fully or partially highlighted.

Concepts proposed by at least 50% of participants are accepted. No predetermined certainty level enhances simplicity, allowing for potential future improvements. Figure 5.5 showcases the GUI for participant inputs.

ARCH
Know what your model doesn't know

Operation Form Dashboard

Draw the bounding boxes and answer the questions on each concept you can identify in the saliency map

Draw bounding boxes on the concepts

Bounding box list

1. concept Edit Delete

Is the concept concept fully highlighted by the saliency map?

Yes
 No

Previous Next

Figure 5.5: Screenshot illustrating the "Collect Complete Local Learned Mechanism (LLM)" operation for participants.

5.2.6. Mapping LLM to GEM - O.GEM.T

The "Map LLM to GEM" operation assesses the relevance of newly discovered LLMs for inclusion in the GEM list. Participants respond to a binary question about the relevance of a mechanism and, if relevant, provide a typicality value.

For the mechanism to be included in the GEM list it requires majority agreement. The mechanism's typicality value is determined collectively by participants. Reliability is ensured using max over total formulae as the in Map gem to lem operations and similar operation. Figure 5.6 depicts the GUI for participant input.

ARCH
Know what your model doesn't know

Operation Form Dashboard

Identified mechanism:
Consequent: great white shark
Antecedents: pointed snout

Is it relevant to expect this list of antecedents for that consequent ?
 Yes No

Next

Figure 5.6: Graphical User Interface for the "Mapping LLM to GEM" Operation.

5.3. Batch

In the realm of our system's operations, the notion of a **batch** holds significance. This section sheds light on the various aspects concerning batch size, composition, and order. The system offers these different strategies for the developer in order to create the best setup for the crowd workers as the amount of questions and their order can affect their performance Matsubara et al., 2021.

5.3.1. Batch Size

The system offers diverse batch size options, providing flexibility for users. There are three options available one sample per batch where each batch includes a single sample and a collection of samples in batch which is adjustable by the developer to how many samples should be in each batch and lastly an option to include all the class images in one batch. The developer has the capability to adjust the number of samples in each batch according to their requirements.

5.3.2. Batch Composition

Batches can be composed of samples that belong to the same class, referred to as **Homogenous** batches, or they can consist of samples from different classes, known as **diverse** batches. The choice of batch composition depends on the research goals and the nature of the data set used.

For example, in a computer vision task of classifying different types of fruits, a Homogenous batch could consist of multiple images of apples, allowing the diagnosing session to focus on learning the specific characteristics of apples. On the other hand, a diverse batch could contain a mixture of images of apples, oranges, and bananas, providing the model with exposure to various fruit classes and enhancing its ability to generalize across different types of fruits.

5.3.3. Batch Order

In the context of "Homogenous" batches, where each batch consists of samples from a single class, there are three distinct options available for organizing the order in which the batches are presented to the model. As we proceed, the following examples will provide a clear illustration of these options. In the examples, each letter corresponds to a specific class, while the number indicates the index of the set. For simplicity, we've divided each class into three sets of images of equal size.

Firstly, the "Per Class" order involves consecutively processing all batches from a given class before

moving on to the next class. To illustrate, let's consider a scenario with three classes (A, B, and C) and a total of nine batches. In this case, the arrangement would be as follows: A1, A2, A3, B1, B2, B3, C1, C2, C3.

Secondly, the "Round Robin" order entails interleaving batches from different classes. Each class contributes one batch in turn until all classes have presented one batch. This cyclic process continues until all batches are processed. For example, for the same three classes and nine batches, the Round Robin order would appear as: A1, B1, C1, A2, B2, C2, A3, B3, C3.

Lastly, the "Random" order involves the random shuffling of batches from different classes, without adhering to any specific pattern. This order is determined through a randomization process, ensuring a diverse and unpredictable sequence of batches. As an example, for the same three classes and nine batches, the random order could be: B2, A3, C1, B1, C2, A1, B3, C3, A2.

5.4. Workflow

5.4.1. Main Workflow

The central framework of our system revolves around the workflow as shown in figure 5.7, a strategic arrangement designed to address the inherent limitations of individual samples. This workflow aims to equip the system with sufficient data to make accurate decisions. In light of this objective, each sample is associated with a specific class, encompassing a predefined set of GEMs. These GEMs encapsulate the anticipated mechanisms within images belonging to that particular class. To navigate the intricacies of sample analysis, the main workflow adopts a systematic sequence of operations.

To embark on this journey, the initial step within the main workflow entails the mapping of GEMs to LEM. This operation serves as a gatekeeper, verifying whether the mechanisms prescribed by the GEMs align with the contents of the sample image. Upon the identification of relevant mechanisms, the workflow transitions to the subsequent stages.

The next pivotal operation involves LEM validation, where the presence of the specified concepts linked with identified mechanisms is examined in the saliency map of the image. This assessment provides insights into the model's grasp of these concepts, gauging the degree of their representation in the image.

After each LEM validation, the workflow proceeds to check for the existence of additional concepts within the same saliency map. If additional concepts are found, it indicates that the current LLM is not sufficient to capture all the relevant information in the image. In such cases, we initiate the collection of a complete LLM for that image, aiming to identify all the highlighted concepts in the saliency map.

In scenarios where a novel mechanism emerges during the process, it becomes imperative to evaluate its presence across other images. This is where the final operation, Map LLM to GEM, enters the equation. This operation is used to determine using the user input to determine the applicability of the new found LLM to the list of GEM. If a consensus among users confirms its relevance, the mechanism joins the list of GEMs, accompanied by an assigned average typicality value.

Figure 5.7 illustrates the visual dependency between these operations in the main workflow.

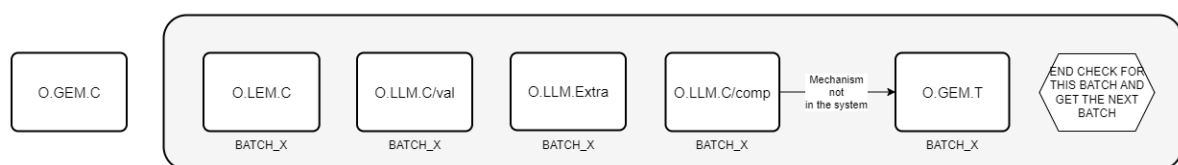


Figure 5.7: Main Workflow

5.4.2. Operation states

The system's operation landscape encompasses a series of distinct operations, each possessing specific prerequisites for execution. These operations collectively contribute to the intricate process of our workflow, which is visually elucidated in Figure 5.7. To facilitate a better understanding, Figure 5.8 introduces a set of binary states that align with certain pivotal operations, specifically O.LEM.C, O.LLM.C/Val, and O.LLM.Extra.

In the context of Map GEM To LEM - O.LEM.C, the color-coded representation conveys distinct conditions. When all the concepts associated with a given mechanism are present within an image,

the operation is denoted in green. Conversely, if any of the concepts are absent, the operation is symbolized in red. Conversely, if any of the concepts are absent, the operation is symbolized in red. Conversely, if any of the concepts are absent, the operation is symbolized in red.

For Validate LEM - O.LLM.C/Val, the color coding indicates the presence of highlighted concepts on the saliency map. When all the concepts are effectively highlighted, the operation is depicted in green. However, if any concept lacks the required highlighting, the color shifts to red.

Lastly, we consider Check Extra concept - O.LLM.Extra, wherein the color distinction underscores the identification of new concepts. If any concept remains unidentified by system but still in the saliency map, the operation is designated in green. Conversely, when all concepts have been identified, the operation takes on a red color.

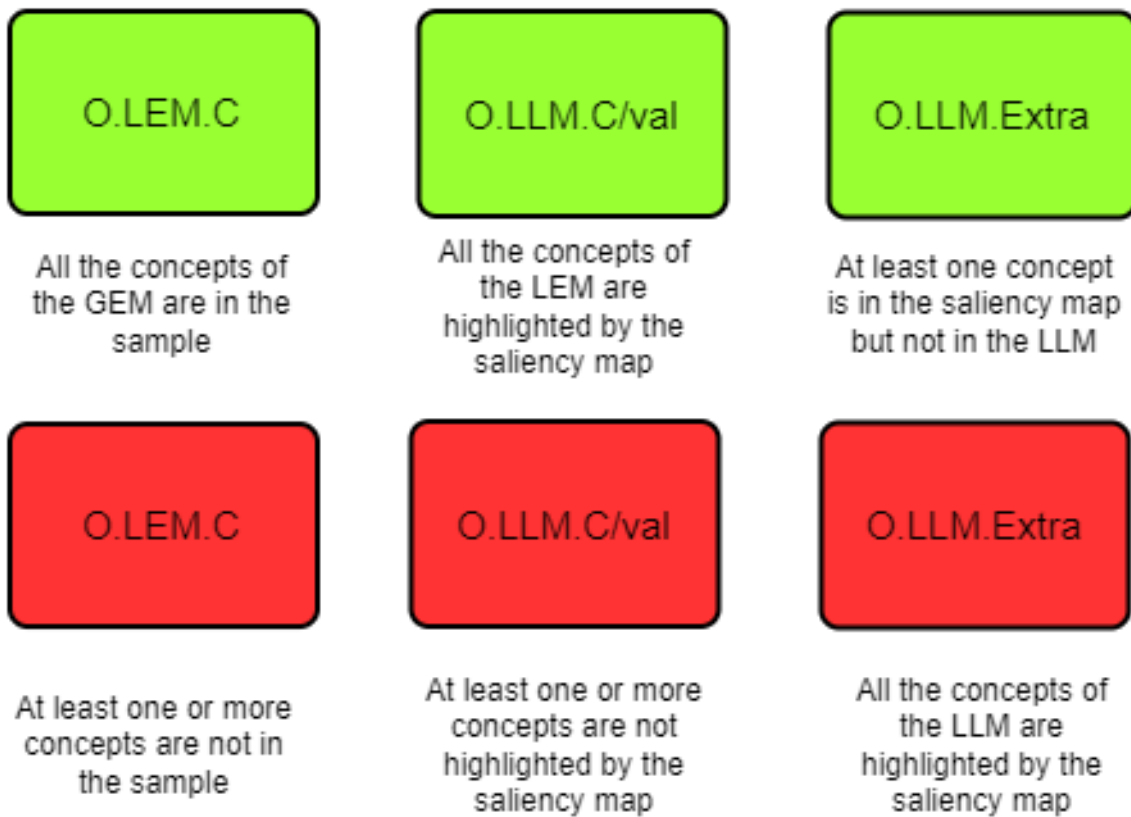


Figure 5.8: Workflow Legend depicting color coding for specific operations.

5.4.3. Workflow Types

In addition to its foundational operations, our system encompasses variations in the execution of these operations, each carrying specific ramifications for data accumulation, the extent of operations conducted, and associated costs. This section is dedicated to introducing these diverse workflow types, denoted as the **Per Mechanism**, **Shortcut Per Mechanism**, **All Concepts**, and **Hybrid All Concepts**. These distinct workflows serve as tailored strategies, adept at accommodating various research objectives and limitations, thereby enhancing the adaptability and utility of our system. It is important to note that the sequence in which these operations are carried out relies on the chosen setup, as determined by the developer's strategic judgment for presenting tasks to the crowd workers.

Per Mechanism Workflow

This workflow, as depicted in Figure 5.9, involves a sequential process. Beginning with the O.LEM.C operation, each mechanism is evaluated successively. Upon successful validation, the system progresses to the O.LLM.C/Val operation. Should a mechanism fail validation, the subsequent mechanism is evaluated until a valid mechanism is identified. If no mechanisms validate, the workflow concludes with the O.LLM.C/Comp operation. This workflow ensures that all the gems in the system are checked for the sample.

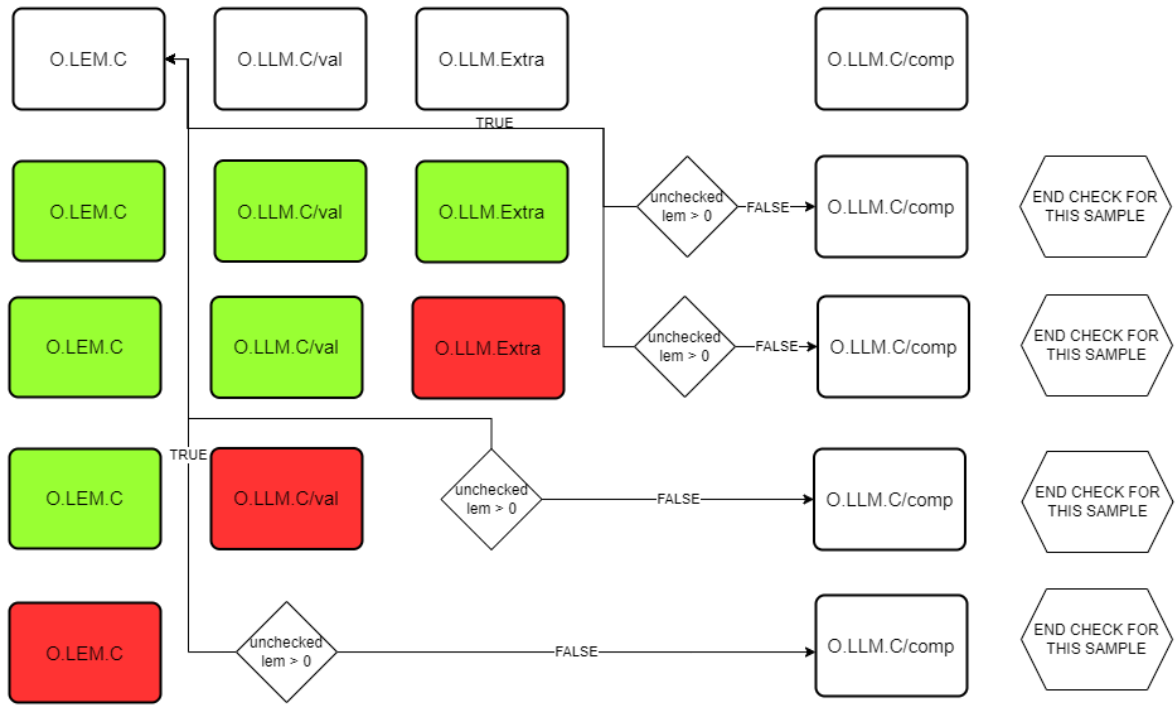


Figure 5.9: Per mechanism workflow.

Shortcut Per Mechanism Workflow

The Shortcut per mechanism workflow, as illustrated in Figure 5.10, differs from its predecessor by promptly proceeding to the O.LLM.Extra operation and subsequently the O.LLM.C/comp operation upon validating a single mechanism. This workflow stops once the mechanism with the highest typicality value has been validated and the proceed to the O.LLM.Extra dn O.LLM.C/comp

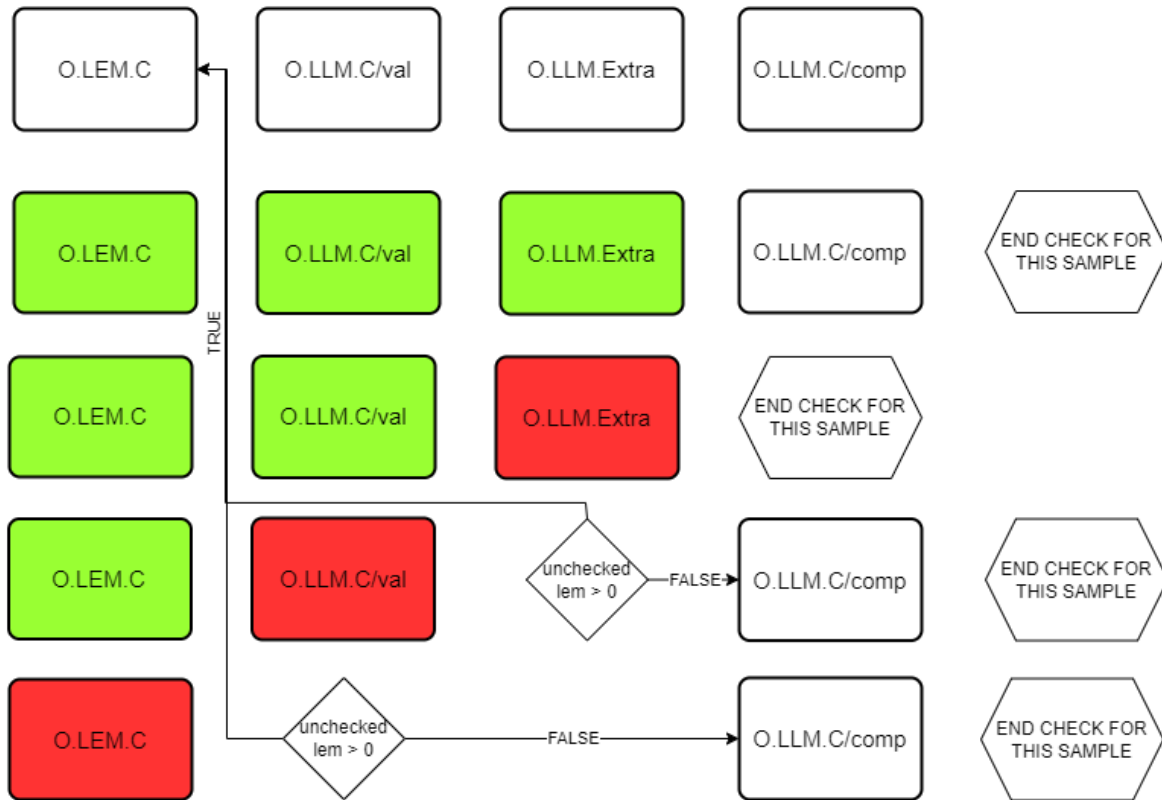


Figure 5.10: Per mechanism shortcut workflow.

All Concepts Workflow

In all concepts workflow, showcased in Figure 5.11, a comprehensive task entails the examination of unique concepts across all mechanisms. Validation involves assessing whether any subset of these concepts corresponds to a system mechanism. Upon identifying a validated set of concepts and a corresponding mechanism, the workflow advances to the O.LLM.Extra operation. In the absence of extra concepts, the system determines whether the mechanism already exists in the system. If present, the sample check concludes; if not, the O.GEM.T operation commences.

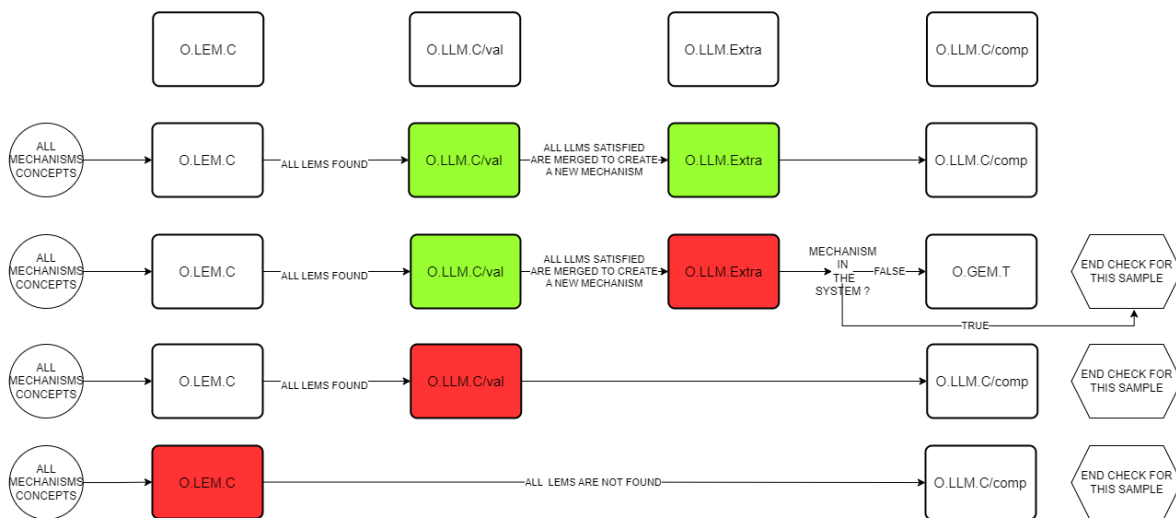


Figure 5.11: All concepts workflow.

Hybrid All Concepts

The Hybrid all concepts workflow, displayed in Figure 5.12, combines features from the all concepts and shortcut per mechanism workflows. Unique concepts from all mechanisms are checked initially. Similar to the all concepts workflow, mechanism concepts are validated, triggering the O.LLM.Extra operation if successful. The workflow iterates through mechanisms until one mechanism is validated, followed by O.LLM.C/Comp if extra concepts are identified, or a sample check ending if the mechanism is present in the system, or initiating the O.GEM.T operation otherwise.

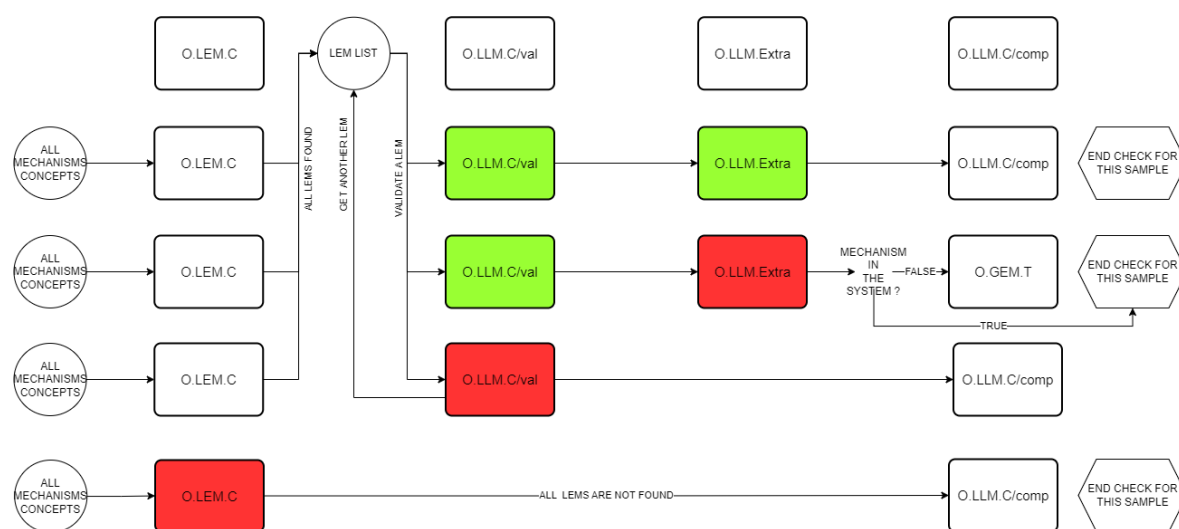


Figure 5.12: All concepts per mechanism shortcut workflow.

5.5. Data model

At the foundational level of our system lies the data model, which serves as the bedrock for system construction. Our data model consists of three principal components: **session**, **logging**, and **core**. Each of these components embodies distinct significance within the system architecture.

5.5.1. Session Component

We begin by introducing the **session** component, which functions as the entry point for developers to configure settings for session execution. In the context of our system, a session denotes a debugging session. Developers can customize various parameters such as the number of samples to be utilized, task generation specifics, and the certainty level for each operation. The session data model, depicted in Figure 5.13, encapsulates these configuration options.

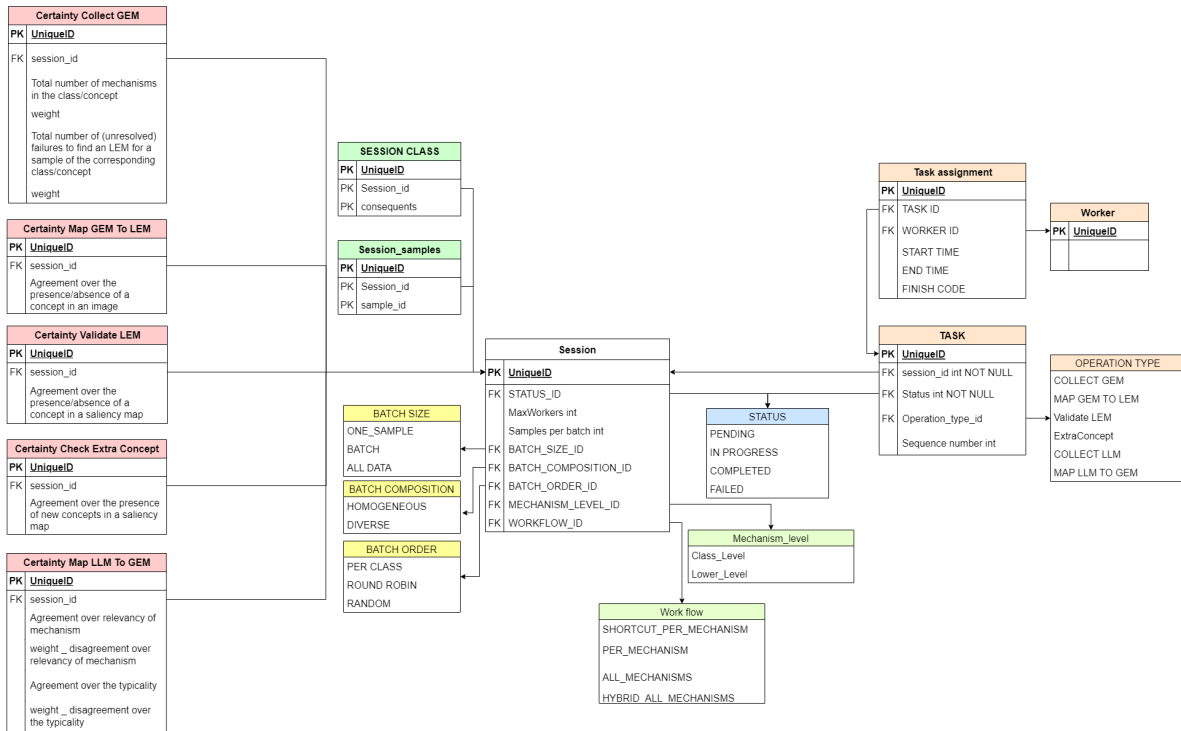


Figure 5.13: Illustration of the Session Data Model Depicting Varied Configuration Options. This data model outlines the constituent components of a session along with their corresponding entities responsible for storing and managing data in our system. Entities sharing the same color belong to the same functional category: red signifies operation certainty data, cyan represents session samples-related data, yellow indicates batch configuration information, light green denotes settings related to mechanism data, blue signifies statuses applied to tasks, operations, and sessions, while light brown pertains to data related to tasks.

In Figure 5.13, tables with light-red headers denote certainty thresholds and signals for each operation. Certainty here pertains to the crowd workers’ agreement threshold. Notably, certain operations, like Map GEM to LEM (O.LEM.C), Validate LEM (O.LLM.C/val), and Check Extra concept (O.LLM.Extra), involve only one binary question and, hence, exhibit a single signal.

The cyan-colored tables encompass the session class, representing the classes being examined within the session, along with details of the employed session samples. Tables with yellow headings provide insights into batch details. These include batch type choices, which range from single-sample per batch to batch collections or even bundling all data in a single batch. If a batch type is selected, users also have the option to specify the number of samples in each batch. In instances where the number of samples is less than the specified value, the batch is formed with the remaining samples. Additional batch characteristics include batch composition (homogeneous or diverse) and batch order (per class, round robin, or random).

The light-green tables encompass the mechanism level to be executed, which can be either “Class Level” or “Lower Level,” along with the chosen workflow type: “Per Mechanism,” “Per Mechanism Shortcut,” “All Concepts,” or “All Concepts Per Mechanism Shortcut.”

The blue status tables delineate four possible operations, task or session states: “Pending” (not yet executed), “In Progress” (currently in execution), “Completed” (execution finished), and “Failed” (anomalies encountered during execution).

Lastly, the light-brown tables represent task-related aspects such as the task itself, operation type, worker, and task assignment. The white-headed entity signifies the central entity, **session**, which houses all data. This entity establishes relationships with other tables, interconnected via session IDs.

5.5.2. Logging Component

The logging component, depicted in Figure 5.14, serves as a repository for data collected from crowd workers, retaining them in their original, unprocessed state. For every operation, both input and output data are stored. The input corresponds to the data employed to formulate the operation, while the output encompasses the responses furnished by the workers.

To ensure data integrity and prevent duplication, each output is linked to the respective input ID and task assignment ID. This practice of associating output with these unique identifiers mitigates the possibility of duplicate entries resulting from simultaneous worker responses or other factors. This approach adheres to the principle of preserving record uniqueness by permitting only one answer per input ID within a task assignment.

Certain output tables are accompanied by supplementary tables that capture additional worker inputs. For instance, in operations like Collect GEM or tasks involving bounding box annotations such as Map GEM to LEM, Validate LEM, or Collect Complete LLM, an extra entity is used to record bounding box of the concept inputted by the workers. For Collect GEM output it is connected to the Antecedent Collect GEM which stores the list of antecedent for each mechanism the worker has input.

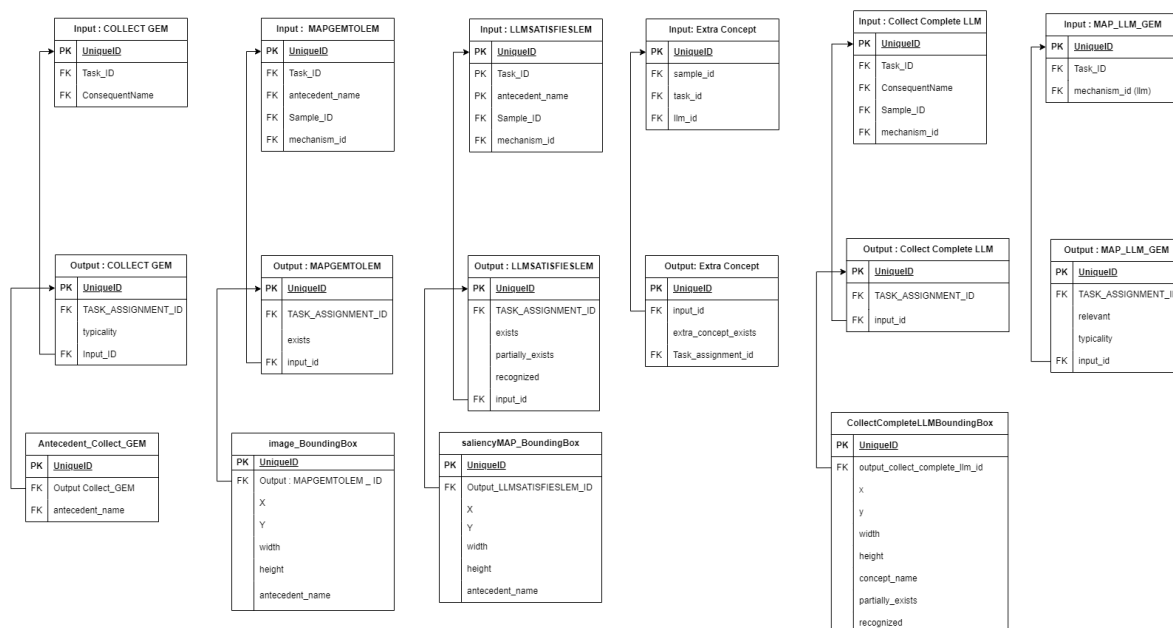


Figure 5.14: Data model for the logging component. It represent the entities used to store the workers input from the operations

5.5.3. Core Component

We now delve into the focal point of the entire system — the core component. This fundamental element bears the name “core” due to its paramount significance within the system’s architecture. It is within this component that all the essential data required for computing objectives is housed, encompassing data pertaining to both samples and mechanisms, as depicted in Figure 5.15. The core component, being the cornerstone of the system, consists of two primary subdivisions: one highlighted in light green, focusing on mechanisms, and another in light blue, centering on sample details. These divisions are interlinked through common data models, represented by darker blue tables, which include consequent and antecedent elements shared between the mechanism and sample aspects.

The light green tables within the core component include Mechanism, Mechanism Antecedent, GEM LEM, and LLM, while the light blue tables encompass Sample, Sample Mechanism, Sample Concept with image types, and bounding boxes for those concepts. It is noteworthy that the data contained within this component is an aggregation of information from the preceding logging component.

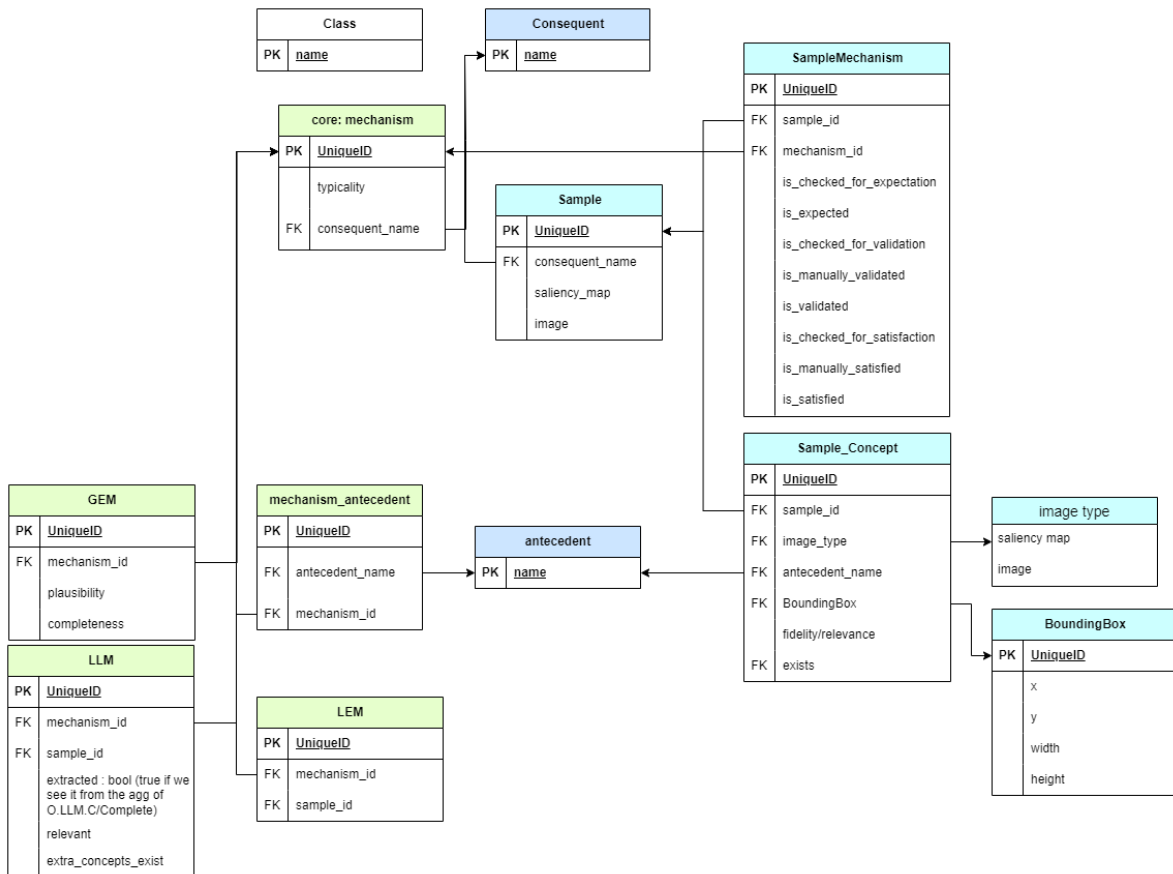


Figure 5.15: Data model of the core component. The green headed entities represent mechanism related entities. The light blue headed entity represent sample related entities. Lastly the dark blue headed entity represent entities shared across both the mechanism and the sample

5.6. Configuration Panel

This section presents the graphical user interface (GUI) of the system, which serves as the interface through which users interact with the system using GUI components. Figure 5.16 illustrates the configuration options available to users when setting up a session. The options encompass various configurations pertinent to a session.

In Figure 5.17, the dashboard provides a comprehensive view of essential actions and details. Users can upload images and saliency maps, initialize new sessions, and access pertinent information, including worker details, dataset information, task specifics, and distinct objectives such as Counterexample, Satisfaction, and Replacement.

Lastly, Figure 5.18 represents the worker entry point screen. Workers input their unique worker IDs and subsequently submit this information. Following submission, workers are redirected to their designated tasks within the system.

ARCH Know what your model doesn't know Operation Form Dashboard

[Back](#)

Select class names

- Select All
- american lobster
- great white shark
- tench

max number of samples in this session

min number of workers per batch

Select Batch Size

- One Sample
- Batch
- All Data

Select workflow

- Per Mechanism
- Shortcut Per Mechanism
- All Concepts
- Hybrid All Concepts

Select which level of mechanisms to evaluate

- Class Level
- Lower Level

Certainty Thresholds

Certainty GEM Completeness

Number of mechanisms in class: 0

Weight Number of mechanisms in class: 0.5

Number of unresolved failures: 0

Weight Number of unresolved failures: 0.5

Threshold: 0.5

Map GEM To LEM certainty

Agreement Presence Concept Original: 0

Validate LEM certainty

Agreement Presence Concept Saliency Map: 0

Check Extra Concept certainty

Agreement Presence New Concepts Saliency Map: 0

Collect Complete LLM certainty

Agreement Presence Concept Saliency Map: 0

Map LLM To GEM certainty

Agreement Mechanism Relevance: 0

Weight Agreement Mechanism Relevance: 0.5

Agreement Mechanism Typicality: 0

Weight Agreement Mechanism Typicality: 0.5

Threshold: 0.5

[Initialize Session](#)

Figure 5.16: Screenshot of the system's configuration options within the graphical user interface (GUI).

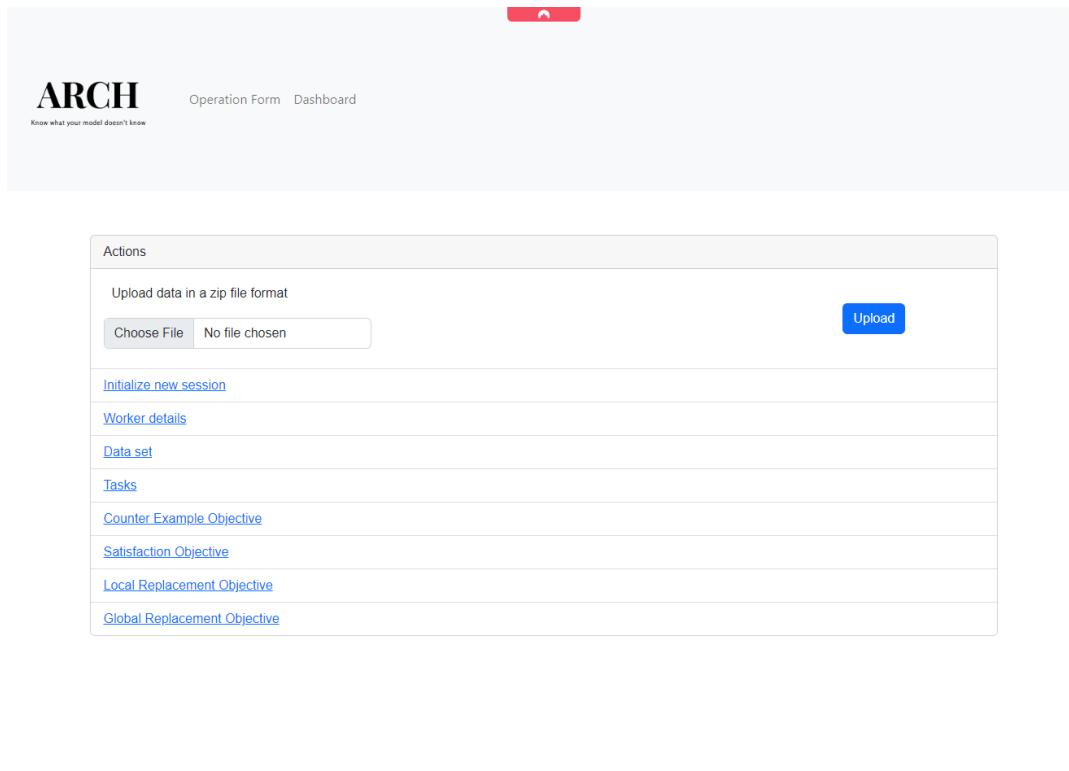


Figure 5.17: Screenshot of the system's dashboard displaying various options and details available to users.

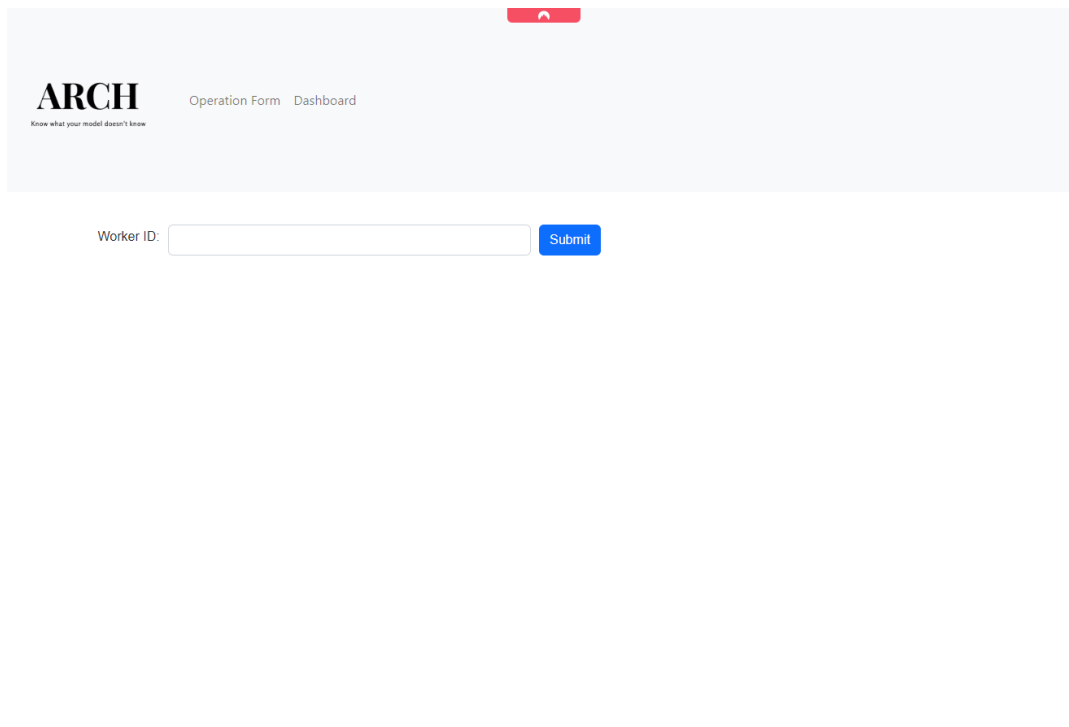


Figure 5.18: Screenshot of the worker entry point screen where worker IDs are submitted to access tasks.

6

Experimental setup

This section presents an overview of the experimental setup and outlines the step-by-step procedure adopted to efficiently conduct the experiments utilizing automated answering. The utilization of automated answering has proven instrumental in conserving significant time that would have otherwise been expended. Completing the experiments manually would have entailed multiple weeks of effort. An additional advantage of the automated answering tool is its ability to maintain consistent configurations across the experiments. Our experimental design benefits from a consistent data collection process, ensuring that the variables under investigation are the only aspects that vary across experiments. Our experimental setup is designed to mitigate the influence of diverse user responses since we employ a pre-existing dataset sourced from skilled annotators. This approach ensures a high quality of collected data, enhancing the reliability of our results.

Nevertheless, it is important to acknowledge potential sources of variance in our experiments. Variability might arise due to the utilization of a random number generator when involving less experienced workers. Additionally, the order in which samples are retrieved from the database could also contribute to some degree of variance.

6.1. Automated Answering Tool

The Automated Answering Tool was developed to facilitate automated answering within our system. This tool consists of two main components. The first component, called the **annotation tool**, focuses on collecting all relevant data annotations that an expert would utilize in the system. The annotator allows users to annotate the data, including two key data sources: (1) a comprehensive list of relevant mechanisms and their associated typicality values (Figure 6.1), and (2) the identification of concepts within the original image along with their visibility in the saliency map, indicating whether they are fully visible or not (Figure 6.2). The data collected from the annotator tool is utilized in the second part of the system, known as the **simulator**.

6.1.1. Annotation tool

The Annotation tool stands as a pivotal component of the system, meticulously designed to efficiently amass pertinent data. While drawing inspiration from our system, the Annotation Tool boasts a more specialized focus on the collection of relevant information. Its key function is to gather the essential data sources necessary for the effective operation of the simulator.

The Annotation Tool is instrumental in procuring two fundamental categories of data required for the simulator's functionality:

1. **Mechanism Data:** This includes a comprehensive compilation of pertinent mechanisms, alongside their respective typicality values, as illustrated in Figure 6.1. This compilation forms an indispensable aspect of the simulator's groundwork.
2. **Sample Data:** The Annotation tool facilitates the capture of concepts present in the original image, with a detailed record of their visibility status within the saliency map. This information, exemplified in Figure 6.2, serves as an integral input for the system.

Home Collect GEM Dataset

american_goldfinch

Antecedent:

Attribute:

Attribute value:

Typicality:

Relevant:

Already entered mechanisms

concepts:
body

beak

wing

Relevant : Yes
 Typicality: 0.99

Do not insert any of the following mechanisms as they already exist in the system

concepts:
body

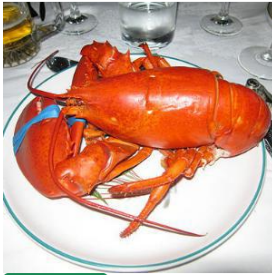
head

beak

Figure 6.1: Screenshot from the annotation tool where it shows the part of the system in which the users input the mechanisms for each class. In this image we also show the already existing mechanisms in the system


Home Collect GEM Dataset

lobster



Select Item ▾

- background COLOR:
 - Fully Visible
 - Remove
- plate COLOR:
 - Fully Visible
 - Remove
- claw COLOR: orange
 - Fully Visible
 - Remove
- lobster_head COLOR: orange
 - Fully Visible
 - Remove
- shell_body COLOR: orange
 - Fully Visible
 - Remove



Select Item ▾

- claw COLOR: orange
 - Fully Visible
 - Remove
- shell_body COLOR: orange
 - Fully Visible
 - Remove
- lobster_head COLOR: orange
 - Fully Visible
 - Remove

Antecedent:

Attribute:

Attribute value:

[Add antecedent](#)

[Save image](#)

Figure 6.2: Screenshot from the annotation tool where it shows the part of the system where the users input the concept found in the original image and the concepts highlighted by the saliency map with its visibility. Each concept can have an attribute and a value for this attribute

6.1.2. Simulator

The Simulator Component of the system is designed to effectively simulate crowd workers and facilitate experiments with controlled levels of accuracy. It enables us to model two distinct types of crowd workers: experts and non-experts. Each simulated crowd worker possesses two crucial properties: accuracy and completeness.

Accuracy in our simulation context refers to the probability of providing correct answers, especially for binary questions that require a yes or no response, which form the majority of inquiries within our system. To replicate realistic scenarios, we introduce a random variable generated each time within the range of 0 to 1. If this random variable exceeds the accuracy value, the simulated worker will provide an incorrect answer otherwise it provide correct answer. This random variable essentially represents the "hardness" of the question, as a higher value compared to the accuracy threshold leads to incorrect responses.

On the other hand, completeness pertains to the percentage of answers obtained from the available choices. This aspect becomes particularly relevant in the "Collect GEM Operation," where users are requested to provide a list of mechanisms. By considering completeness, we gain insights into the proportion of relevant answers provided by the simulated crowd workers. This further enhances the accuracy of our experiment simulations and allows us to draw meaningful conclusions.

The simulated expert crowd worker is assumed to have extensive domain knowledge and a high likelihood of providing correct answers 1.0. However, their completeness may not be perfect due to potential gaps in their understanding of all relevant mechanisms. For completeness, a value of 0.6 is assigned, which is relatively high.

The simulated non-expert crowd worker, in contrast, is considered to have less knowledge and may make errors when answering questions in the system. They are assigned lower accuracy for example 0.6 and a completeness value of 0.4. In the next chapter we represent experiments to test the different worker's accuracy and how it affects the output of the system

As for the typicality value in the "MAP LLM To GEM," it is consistently correct when queried, as the average of the answers still yields the correct value, which is known to us.

6.2. Experiments

In this section, we present the various configurations and setups used in our experiments, which will be consistently referenced throughout this research. The values enclosed in brackets indicate parameters that were tested with different possible settings within the configurations.

6.2.1. Data-sets

To facilitate comprehensive evaluation, we employed two types of data sets: biased which are **fish** and **bird** data sets and non-biased **scene** data set.

Biased

In the **Fish** dataset, the images were sourced from the extensive ImageNet dataset¹. To ensure dataset accuracy and relevance, a thoughtful selection process was undertaken. This involved filtering of images to ensure that each fish species was presented against a backdrop that resonated with its natural habitat or context. For instance, images of sharks were selected against a backdrop of water, while tenches were posed with grassy surroundings. Similarly, images depicting lobsters were chosen within the setting of restaurant plates. This stringent approach to image selection aimed to maintain the authenticity and ecological context of each fish species

Turning to the **Birds** dataset, a careful balancing of image content between training and test data was performed. This encompassed variations centered around class-specific features, such as bird appearance, as well as encompassing less specific features like background elements. By effectively varying these aspects, the data set encompasses a comprehensive representation of bird species and their surroundings

¹<https://www.image-net.org/>

Non-Biased

In the **Scene** dataset, we employed data from the study conducted by Sharifi et al. **sharifi2022should**. This dataset served as a valuable resource for incorporating non-biased scene images, adding a diverse and unbiased dimension to our research endeavors.

6.2.2. Configurations

Normal

- Workers are of type Expert workers.
- Number of workers: 1.
- Number of samples per class: 50.
- Batch type: Batched.
- Number of samples in each batch: 10.
- Batch composition: Homogeneous.
- Batch order: Round-robin.
- Workflow type: (Per mechanism, Per mechanism shortcut, All concepts, All concepts shortcut).
- Uncertainty: Irrelevant (as expert workers are always correct).

Multiple-number of samples per batch

- Workers are of type expert.
- Number of workers: 1.
- Number of Samples: 50.
- Batch type: Batched.
- Number of samples in each batch: (5,15).
- Batch composition: Homogeneous.
- Batch order: Round-robin.
- Workflow type: (Per mechanism, Per mechanism shortcut, All concepts, All concepts shortcut).
- Uncertainty: Irrelevant (as expert workers are always correct).

Multiple uncertainty

- Workers are of type non-expert.
- Number of workers: 3.
- Number of Samples: 50.
- Batch type: Batched.
- Number of samples in each batch: 10.
- Batch composition: Homogeneous.
- Batch order: Round-robin.
- Workflow type: (Per mechanism, Per mechanism shortcut, All concepts, All concepts shortcut).
- Uncertainty: (Low, Medium, High for each uncertainty), where low has a threshold of 0.6, medium 0.7, and high 0.8. respectively

6.2.3. Design

Our experimentation is driven by comprehensive objectives that guide our evaluation process. We initiated our assessment by focusing on the verification of result correctness. This was achieved through the analysis of two key metrics aimed at confirming the accuracy of our framework's generated answers. These metrics include the accuracy of objectives, which involve counter-examples, GEM accuracy, and replacement mechanisms, as well as the assessment of the system's informativeness. The latter entails a scrutiny of lower-level concepts, particularly at a depth of level 1, to ascertain the accuracy with which the models identify these concepts. Additionally, we explored the balance between the required number of operations and the level of correctness achieved, alongside the efficiency of the operations themselves.

Our experimental design encompasses four distinct setups, each tailored to explore specific facets of our framework's performance. In the subsequent sections, we detail the specifics of these setups, elucidating their objectives and projected outcomes.

Bias Detection Setup

In this initial setup, we employed biased datasets to scrutinize our framework's proficiency in detecting model bias. Through this investigation, we aimed to identify noticeable trends in the output, shedding light on the intrinsic bias within the models. This analysis serves to underscore our framework's prowess in bias detection. This setup utilized the normal configurations outlined in 6.2.2.

Informativeness Assessment Setup

The second setup involved the utilization of non-biased datasets to evaluate the informativeness of our system. By employing this configuration, we sought to demonstrate the framework's ability to provide insightful results devoid of bias. The outcomes of this setup accentuate the framework's potential to deliver meaningful information. This setup also employed the normal configurations specified in 6.2.2.

Sample Size and Tradeoff Analysis Setup

Our third experiment encompassed a range of diverse sample sizes per batch. This design aimed to explore the tradeoff between distinct configurations, particularly the impact of sample size per batch on the overall task workload and the prompt attainment of high levels of informativeness within the system. This analysis furnished valuable insights into the intricate relationship between sample size, task allocation, and system efficiency. This setup was executed with the normal configurations detailed in 6.2.2.

Multiple Uncertainty Setup

Finally, we devised experiments to illustrate varying levels of certainty, alongside the resultant tradeoffs and system efficiency. We utilized configurations involving multiple uncertainty settings, as shown in 6.2.2, to discern the intricate dynamics between uncertainty, efficiency, and tradeoffs.

7

Results and analysis

7.1. On the correctness and informativeness of the tool's outputs

In the first experiment, we want to verify the correctness of our solution. We do so in two ways.

- First, we can check whether the final outputs of the system match our expectations. For that, we had selected classification tasks for which we have domain knowledge allowing us to know what are the human-expected model features, we also injected biases into the dataset and subsequent trained model to control for the model learned features, and from this, we can express expectations on the outputs (whether GEM are satisfied and which other mechanisms might have been learned). Once these expectations are expressed, we can check to what extent they are satisfied by comparing to the actual outputs of our solution.
- Second, we can assess whether the evolution of the state of the system over time is meaningful. Based on the intended behavior of the system depending on its configurations, we can express expectations on the evolution over time of various types of information (e.g., the number of GEM) collected during one debugging session in order to compute final outputs. We can then check whether such expectations are met.

We also want to understand the informativeness of the outputs of the system. For that, we do not have expectations about the model behavior (learned mechanisms and the satisfaction of expected mechanisms), and we simply want to investigate qualitatively whether the outputs do reveal the model behavior. We conduct such a qualitative analysis on the model that was “naturally” trained on a “natural” dataset.

Below, we describe such analysis based on our biased datasets/models.

7.1.1. Correctness of the outputs

We separate our analysis based on the different datasets we experimented on, as they have different characteristics that bring different information about the outputs of the system.

The Fish Biased Dataset

We start our analysis with the fish biased dataset and model. As a reminder, this model was trained in a way that it should primarily learn to use concepts related to the animals' background (e.g., green first for the tench, blue water for the shark, and plate and human hands for the lobster) instead of concepts related to the animals themselves when making predictions. Hence, we expect most GEMs to have at least one counter-example, and to have a low satisfaction rate. Besides, we expect to find LLMs that are not part of the GEMs, that encompass such background concepts. These expectations correspond to the outputs of the system.

For the tench results shown in Figure 7.1, we find only one GEM that is satisfied (<green tench body → tench>).

For the shark results shown in Figure 7.2, we find that the highest typicality mechanisms are not expected (e.g., <grey fin, grey shark body, grey shark head → shark>), which satisfies our expectations

Antecedent List	Consequent	Typicality	Counter Example found	Number of Samples with Counter Example	Satisfaction
tench_fin,Color:green;	tench	0.6	Yes		32 0.0
tench_fin,Color:None;	tench	0.4	Yes		18 0.0
tench_body,Color:green;	tench	0.4	No		0 1.0
tench_body,Color:yellow;	tench	0.4	No		0 0.0
tench_body,Color:None;	tench	0.2	No		0 0.0

Figure 7.1: Tench class outputs

since these mechanisms are the most exigent to satisfy (they are constituted of many concepts typically) while we made sure to bias the model. We also find that the lowest typicality mechanisms are not satisfied (e.g., <shark body → shark>). Instead, we find that mechanisms of medium typicality are sometimes satisfied (with low satisfaction percentage, but not zero percentage), for instance <grey fin, grey shark head → shark> is satisfied at 28%, <grey shark head → shark> at 59% and <grey shark body → shark> at 55%. This is according to what we expected: the model has sometimes learned a “natural” behavior (when the animal’s background was too different from the “average” background, it seems the model went back to learning about the animal), and most often learned non-expected mechanisms.

fin,Color:grey;,shark_body,Color:grey;,shark_head,Color:grey;	shark	0.98	Yes		12 0.14285714285714285
fin,Color:blue_grey;,shark_body,Color:blue_grey;,shark_head,Color:blue_grey;	shark	0.87	Yes		1 0.0
fin,Color:grey;,shark_head,Color:grey;	shark	0.78	Yes		18 0.28
shark_head,Color:grey;	shark	0.75	Yes		16 0.5897435897435898
fin,Color:None;,gill,Color:None;	shark	0.7	No		0 0.0
shark_body,Color:blue_grey;,shark_head,Color:blue_grey;	shark	0.7	Yes		1 0.0
fin,Color:grey;	shark	0.7	Yes		15 0.4827586206896552
shark_body,Color:grey;	shark	0.65	Yes		9 0.55
shark_body,Color:grey;,fin,Color:grey;	shark	0.62	Yes		12 375
shark_body,Color:grey;,shark_head,Color:grey;	shark	0.62	Yes		13 0.2
fin,Color:grey;,shark_head,Color:grey;,gill,Color:grey;	shark	0.62	Yes		4 0.25
shark_head,Color:grey;,gill,Color:grey;	shark	0.62	Yes		5 0.5
gill,Color:grey;,shark_head,Color:brown;	shark	0.62	No		0 1.0
shark_head,Color:blue_grey;	shark	0.62	Yes		1 1.0
fin,Color:None;	shark	0.6	Yes		1 0.0
gill,Color:grey;	shark	0.6	Yes		5 0.5
shark_body,Color:None;	shark	0.55	Yes		1 0.0
shark_body,Color:blue_grey;	shark	0.5	Yes		1 0.0
gill,Color:None;	shark	0.5	No		0 0.0

Figure 7.2: Shark class outputs

As for the lobster results shown in Figure 7.3, the observations are very similar to the shark. While the highest typicality mechanisms (GEMs) are not even mapped as LEM (very high typicality make it also difficult to be corresponding to a real “perfect” sample, e.g., <red body, red head, red leg → lobster>), the first LEMs of high typicality that are mapped to LEMs are not satisfied at all. Then, in terms of medium typicality GEMs, some are satisfied at around 60% (e.g., <orange claw → lobster>) while others are satisfied very rarely, e.g., around 10% to 17% <red antenna → lobster>.

In terms of the mechanisms that the model has learned (LLMs) but that are not expected by a human, they accurately reflect the biases that had been injected in the model. For the tench, all the concepts related to the background of the tench appear, e.g., human hands, human face, human body, background, green grass, sky. In certain cases, these concepts are used by the model without simultaneously using any concept related to the tench. In other cases –these are the most frequent cases–, such concepts are used simultaneously with concepts related to the tench. For instance, we identify the LLM <human_hands,Color:None;,human_face,Color:None;,tench_body,Color:greenANDyellow; → tench> as frequently used by the model instead of the GEMs. This information is also indicative of the informativeness of the outputs. While we injected biases for the model to learn the animal background, we discover that the model has actually learned a more complex combination of background concepts with some small parts of the animal’s concepts. We make similar observations for the shark and lobster. For the shark, most commonly, the model has learned to identify sharks by the mere presence of water, or the combination of water with a shark fin or part of a shark’s head. As for the lobster, it has learned to look at the plates around the lobsters and the food posed on the plates, or sometimes the faces of the humans seating at the table where the lobster plate is.

Note that across configurations, the results are the same overall. There is a large overlap in terms of

antenna,Color:orange;,claw,Color:orange;,shell_body,Color:orange;	lobster	0.95	No		0	0.0
lobster_body,Color:redANDblack;,lobster_head,Color:redANDblack;,lobster_leg,Color:redANDblack;	lobster	0.93	No		0	0.0
antenna,Color:None;,claw,Color:None;,shell_body,Color:None;	lobster	0.91	No		0	0.0
antenna,Color:None;,claw,Color:red;	lobster	0.85	No		0	0.0
antenna,Color:None;,shell_body,Color:red;	lobster	0.85	No		0	0.0
claw,Color:orange;	lobster	0.8	Yes		4	0.6666666666666666
claw,Color:red;	lobster	0.8	Yes		3	875
claw,Color:redANDblack;,lobster_head,Color:redANDblack;	lobster	0.8	Yes		2	0.0
antenna,Color:red;	lobster	0.7	Yes		13	0.1875
lobster_leg,Color:red;	lobster	0.7	No		0	0.0
claw,Color:orange;,lobster_head,Color:brown;	lobster	0.68	No		0	0.0
claw,Color:orange;,lobster_head,Color:orange;	lobster	0.68	Yes		1	0.8
lobster_head,Color:orange;	lobster	0.68	Yes		4	0.7272727272727273
claw,Color:brown;,lobster_head,Color:brown;	lobster	0.68	Yes		3	0.5
claw,Color:red;,antenna,Color:red;	lobster	0.68	Yes		12	0.15384615384615385
claw,Color:orange;,shell_body,Color:orange;,lobster_head,Color:orange;	lobster	0.68	Yes		3	0.3333333333333333
lobster_head,Color:orange;,shell_body,Color:orange;	lobster	0.67	Yes		3	0.4
claw,Color:orange;,lobster_head,Color:redANDblack;	lobster	0.67	No		0	1.0
claw,Color:redANDblack;	lobster	0.65	Yes		1	0.5
antenna,Color:red;,claw,Color:brown;	lobster	0.65	No		0	1.0
claw,Color:None;,shell_body,Color:None;	lobster	0.65	No		0	0.0
claw,Color:brown;	lobster	0.6	Yes		2	0.6
lobster_head,Color:brown;	lobster	0.6	Yes		2	0.7142857142857143
lobster_leg,Color:redANDblack;	lobster	0.6	No		0	1.0
claw,Color:None;	lobster	0.6	Yes		2	0.0
antenna,Color:None;	lobster	0.5	No		0	0.0

Figure 7.3: Lobster class outputs

the GEMs and LLMs that show up in the system during the debugging session. For instance, between the “per mechanism” and the “all concept” configurations, the difference in terms of the GEM amounts to only 3 GEMs out of 53, with “all concepts” presenting 3 more mechanisms than the other configuration. The “per mechanism shortcut” configuration presents 58 mechanisms, and the hybrid all concept 52. Such difference is explained simply by the different orders in which the system is run and encounters the various samples and mechanisms. In terms of the satisfaction scores outputted by the system for each GEM, they are identical across configurations, which was to be expected as the models are the same. This is the case for every dataset, hence we do not discuss these results for the other datasets.

The Bird Biased Dataset

This dataset has been biased in a similar fashion as the fish biased dataset, with the difference that it is composed of more classes (10) than the previous one (3), hence the model has to learn more complex behaviors. We ran only 5 classes out of the 10 classes where we picked 5 random classes. Once again, the outputs reflect our expectations. First, in terms of the GEMs that are correctly learned by the model as shown in Figure 7.4, there are very few of them —this was to be expected as the model is highly biased towards using concepts that are not expected by a human. The model has learned a few correct GEMs only for a few classes and almost never with 100% satisfaction. Particularly, for the american goldfinch, the model is rather correct, and that was to be expected: the model had been biased for it to learn to confuse the classes ‘american goldfinch’ and ‘lesser goldfinch’ together by learning concepts insufficient to disentangle them but sufficient to ‘correctly’ identify that an image represents one of the two classes. With the outputs, we now discover that the model learned to associate this insufficient set of concepts to the american goldfinch, and has not learned to identify the lesser goldfinch accurately in most cases. We observe the same phenomenon for the bufflehead, for which the model has learned some correct GEMs, while it also confuses the hooded merganser greatly with the bufflehead. Again, the same phenomenon is observed for the hairy and downy woodpeckers, for which the model typically confuses the two classes, as it was intended to. Besides, when analysing more deeply which GEMs have relatively higher satisfaction rates, we note that those GEMs with lower typicality often present higher satisfaction rates. For instance, for the downy woodpecker, the four highest-typicality GEMs have satisfaction rates below 25%, whereas the next three mechanisms have satisfaction rates above 53%. Again, this is to be expected: as we strongly biased the model and forced it not to learn accurately the bird classes, when it has still learned some parts of the birds, it is only subsets of all the concepts a human would use to recognize the bird, and hence only a lower-typicality GEM than the most expected one.

We now turn to the assessment of the LLMs returned by the system. Again, the LLMs correspond

Antecedent List	Consequent	Typicality	Counter	Example found	Number of Samples with Counter Example	Satisfaction
grey_body,Color:grey;,whiteANDblack_head,Color:whiteANDblack;,wing,Color:black;	hooded_merganser	0.95	Yes		17	0.5405405405405406
grey_body,Color:grey;,whiteANDblack_head,Color:whiteANDblack;	hooded_merganser	0.82	Yes		11	725
whiteANDblack_head,Color:whiteANDblack;,wing,Color:black;	hooded_merganser	0.8	Yes		11	0.7027027027027027
whiteANDblack_head,Color:whiteANDblack;	hooded_merganser	0.67	Yes		4	0.9
blackWhite_tail,Color:blackWhite;,blackWhite_wing,Color:blackWhite;,redWhiteBlack_head,Color:redWhiteBlack;,white_body,Color:white;	hairy_woodpecker	0.99	Yes		23	0.08
blackWhite_wing,Color:blackWhite;,black_tail,Color:black;,redWhiteBlack_head,Color:redWhiteBlack;,white_body,Color:white;	downy_woodpecker	0.99	Yes		37	0.13953488372093023
blackWhite_wing,Color:blackWhite;,black_tail,Color:black;,white_body,Color:white;	downy_woodpecker	0.86	Yes		35	0.18604651162790697
blackWhite_wing,Color:blackWhite;,black_tail,Color:black;	downy_woodpecker	0.76	Yes		33	0.23255813953488372
black_tail,Color:black;,white_body,Color:white;	downy_woodpecker	0.74	Yes		35	0.18604651162790697
blackWhite_wing,Color:blackWhite;,white_body,Color:white;	downy_woodpecker	0.73	Yes		13	0.7111111111111111
blackWhite_wing,Color:blackWhite;,redWhiteBlack_head,Color:redWhiteBlack;	downy_woodpecker	0.7	Yes		21	0.5333333333333333
blackWhite_wing,Color:blackWhite;	downy_woodpecker	0.67	Yes		9	0.8
black_tail,Color:black;	downy_woodpecker	0.65	Yes		33	0.23255813953488372
redWhiteBlack_head,Color:redWhiteBlack;	downy_woodpecker	0.55	Yes		9	1.24
white_body,Color:white;	downy_woodpecker	0.4	Yes		6	0.8666666666666667
greenANDwhite_head,Color:greenANDwhite;,white_body,Color:white;	bufflehead	0.89	Yes		6	0.75
greyANDwhite_head,Color:greyANDwhite;,grey_body,Color:grey;	bufflehead	0.86	Yes		2	0.8333333333333334
greyANDwhite_head,Color:greyANDwhite;,wing,Color:grey;	bufflehead	0.85	No		0	1.0
greenANDwhite_head,Color:greenANDwhite;,wing,Color:blackANDwhite;	bufflehead	0.81	Yes		3	0.8421052631578947
greenANDwhite_head,Color:greenANDwhite;	bufflehead	0.64	Yes		2	1.2352941176470589
greyANDwhite_head,Color:greyANDwhite;	bufflehead	0.56	No		0	1.0
white_body,Color:white;	bufflehead	0.53	Yes		3	875
wing,Color:blackANDwhite;	bufflehead	0.5	Yes		5	0.782608695652174
wing,Color:grey;	bufflehead	0.4	No		0	1.0
wing,Color:blackANDwhite;,yellow_head,Color:yellow;	american_goldfinch	0.77	Yes		11	0.7027027027027027
yellow_body,Color:yellow;	american_goldfinch	0.36	Yes		4	0.9148936170212766
yellow_head,Color:yellow;	american_goldfinch	0.32	Yes		2	0.9574468085106383
wing,Color:blackANDwhite;	american_goldfinch	0.1	Yes		11	0.7105263157894737

Figure 7.4: The GEMs that are correctly learned by the model (even if not for every sample where they should be learned).

to our expectations, showing the accuracy of the outputs. Many of the LLMs are only constituted of background concepts that the model was forced to learn, such as <grey sky, green tree → hairy woodpecker>. Similarly to the fish dataset, other LLMs are constituted of mechanisms whose typicality is below a reasonable threshold for the mechanisms to be considered expected, such as <red-white-black head → hairy woodpecker> that does not allow to distinguish between the hairy and downy woodpeckers. Additional LLMs are constituted of a combination of expected and unexpected concepts, such as <grey body, black water → hooded merganser>, which was again expected for this biased model.

Finally, note that lower granularity concepts have also been handled in this experiment, as we noticed that it is frequent for the model to learn only parts of the higher granularity concepts (e.g., it does not learn the entire bird body shape and color, but instead only the back, belly, etc.), or for the images to show only parts of the body (which makes it more accurate to then annotate these parts). Part of the expected GEMs could have been satisfied if we had not looked into lower granularity, but actually appear as not satisfied when doing so: as the saliency maps only highlight parts of an expected concepts, and these parts might not create a full GEM of lower granularity, the model has actually not learned the concept that seemed to be learned at a first glance. Hence, a number of GEMs are also not considered satisfied but found in the list of unexpected LLMs due to this incomplete learning. In Figure 7.5, we show how many lower level GEMs are satisfied based on their typicality. This figure shows that in some cases, the model has correctly learned the expected mechanisms (GEMs for which the typicality is high, i.e., above 0.8), while in other cases, the model has learned mechanisms whose typicality is potentially too low to be considered expected (typicality below 0.6) –these are the mechanisms that render the higher-level GEMs unsatisfied. This result was also expected as the model was forced to incorrectly learn the birds, again showing the correctness of the outputs. It also displays the informativeness of the system outputs, as someone who would not have access to the system could have mistakenly believed the model had correctly learned the high-level GEMs, or would at least not know at which frequency insufficient lower-level mechanisms would have been learned.

7.1.2. Informativeness of the outputs

We now turn our attention to the dataset and model that was naturally trained, i.e., a model that is commonly used by researchers and practitioners, but for which we do not know about biases. Hence, we investigate the outputs in order to discover novel insights about this model.

We present in Figure 7.6 the outputs of the system. Overall, they align with our expectations for a naturally trained model. In terms of the GEMs themselves and of whether they do match specific samples in the natural dataset (i.e., whether they constitute LEMs), as expected the GEMs of average typicality are more often LEMs than those of lower or higher typicality. Besides, in terms of satisfaction of the GEMs, GEMs with very high typicality must be hard to learn for a model, as they are often very

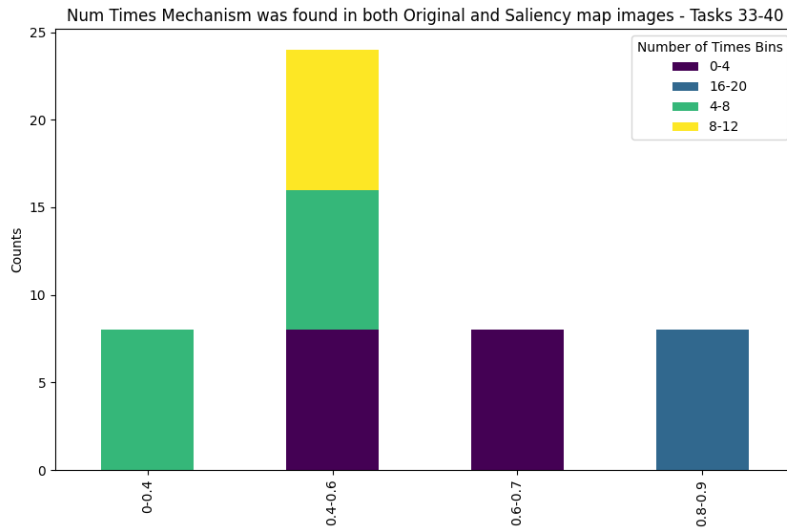


Figure 7.5: Bar plots showing how many lower-level mechanisms are actually learned by the model, divided on the typicality of the mechanisms.

Antecedent List	Consequent	Typicality	Counter Example found	Number of Samples with Counter Example	Satisfaction
chair,Color:None;school desk,Color:None;toy,Color:None;	kindergarden_classroom	0.9	Yes	1	0.5
chair,Color:None;school desk,Color:None;whiteboard,Color:None;	kindergarden_classroom	0.8	No	0	0.0
table,Color:None;toy,Color:None;whiteboard,Color:None;	kindergarden_classroom	0.7	No	0	0.0
blackboard,Color:None;chair,Color:None;school desk,Color:None;toy,Color:None;	kindergarden_classroom	0.7	No	0	0.0
bed,Color:None;drip-feed,Color:None;tray,Color:None;	hospital_room	0.9	Yes	1	0.5
bed,Color:None;medicine,Color:None;tray,Color:None;	hospital_room	0.8	No	0	1.0
bed,Color:None;life machine,Color:None;	hospital_room	0.8	Yes	4	0.6363636363636364
armchair,Color:None;bed,Color:None;drip-feed,Color:None;	hospital_room	0.8	No	0	1.0
armchair,Color:None;bed,Color:None;drip-feed,Color:None;life machine,Color:None;medicine,Color:None;	hospital_room	0.7	No	0	0.0
bed,Color:None;chair,Color:None;desk,Color:None;	dorm_room	0.9	Yes	1	0.75
bed,Color:None;book,Color:None;desk,Color:None;	dorm_room	0.8	No	0	1.0
bed,Color:None;clothes,Color:None;desk,Color:None;	dorm_room	0.7	No	0	1.0
book,Color:None;chair,Color:None;desk,Color:None;	dorm_room	0.6	No	0	1.0
chair,Color:None;plate,Color:None;table,Color:None;tablecloth,Color:None;	dining_room	0.9	No	0	0.0
centerpiece,Color:None;chair,Color:None;table,Color:None;	dining_room	0.8	Yes	1	0.967741935483871
chair,Color:None;table,Color:None;tablecloth,Color:None;	dining_room	0.8	No	0	1.0
chair,Color:None;table,Color:None;	dining_room	0.6	No	0	1.0
office chair,Color:None;screen,Color:None;table,Color:None;	conference_room	0.9	Yes	4	0.5
intercom,Color:None;office chair,Color:None;table,Color:None;	conference_room	0.8	No	0	1.0
office chair,Color:None;projector,Color:None;table,Color:None;	conference_room	0.8	Yes	2	0.3333333333333333
intercom,Color:None;screen,Color:None;table,Color:None;	conference_room	0.7	No	0	1.0
office chair,Color:None;table,Color:None;	conference_room	0.6	No	0	1.0

Figure 7.6: The GEMs and their satisfaction for the natural scene model.

complex and specific. We note that for each class, the highest typicality mechanisms are indeed not satisfied 100%. The GEMs of lower typicality however should be easier to be learned as they are often less specific. And we indeed observe that their satisfaction rates are higher.

Yet, beyond expected results, when we delve deeper into the mechanisms that have been learned and that were not expected, we discover a number of highly unexpected and worrying mechanisms. Especially, we note that the model has learned to look at the presence of persons or children to recognize the kindergarten room, which is highly insufficient considering that the model is trained to recognize over 100 different scenes, in which surely more human beings are present. Similarly, it has learned to recognize the presence of drawings to identify a classroom, which might be a little more sensible, but still remains incomplete in order to accurately identify what is a classroom. As for the hospital room, it seems to have learned to look at the presence of beds or walls. Again, these are very incomplete mechanisms, considering for instance that scene images representing bedrooms could then be confused. We find many more examples of incomplete (too simple) sets of concepts that the model has learned in order to make predictions, across the other classes, such as a 'pillow' for the 'dorm room', a 'table' for the 'conference room' or even the combination of a 'table' and 'chair' still for the same class.

Besides, we note that the presence of children also appears associated to hospital rooms in a few samples. The presence of the same concept antecedents for two different classes (kindergarten and hospital room) hint at the existence of lower-granularity concepts, potentially not recognizable with a human-eye, that actually lead the model to differentiate between the two classes. This is another new insights that we achieve using our system, that provides us hints at complex concepts potentially learned by this highly popular scene classification model. While our system is powerful to hint at the existence of complex concepts the model might have learned, its limitations also reside in the existence of such concepts. In certain cases, one could potentially go to even finer granularities and refine annotations, e.g., differentiating between the 'person' learned across classes to try and identify the different visual elements of these 'person' pixels that make the model predict different classes. However, this could be extremely costly to go to even finer granularities. In other cases, it might not even be possible to go to these finer granularities, as a human might not be able to distinguish any difference with their eyes.

7.1.3. System states over time

We now turn to the investigation of the states of the system over time, that provide us further information about the correctness of its implementation. We report our analysis for the fish biased dataset and model, for a single configuration of the system. For the economy of space, and for the reader, we do not report on the analysis of the other models, nor on the analysis of the other configurations. However, our findings are similar for these other debugging sessions, constituting more hints at the good implementation of our system.

Numbers of GEM, LEM, LLM over time

We start by making a coarse-grain analysis of the evolution of the system, before delving into a deeper analysis based on mechanisms' typicality. As expected, the absolute number of GEMs identified over time increases for every class as shown in Figure 7.7. For instance, at the beginning of the session, the lobster has 22 expected mechanisms and the shark 14, and it slowly increases to 28 and 19 respectively, as new expected mechanisms are identified when GEMs have to be recollected, or when an LLM is identified and turns out to be an LEM and subsequently a previously-unseen GEM. Similarly, over time, the number of samples for which an LEM is identified increases over time, going from 0 to the total number of samples for each class as shown in Figure 7.8. This is expected since the primary function of the system is to identify LEMs for each sample in the dataset and check whether the LLMs match those LEMs. Depending on the configuration, one can also observe the differences of the session time at which the number of LEMs of the different classes increases, again indicating the good implementation of the system. And again similarly, according to expectations, the number of samples for which an LLM is identified increases over time, going from 0 to the number of samples for each classes as shown in Figure 7.9. Besides, one can see that the number of LEMs and LLMs evolve in parallel per class—once the number of LEMs of one class increase, then the number of LLMs also increases—, which makes sense in the “per mechanism” and “all-concept” configurations since they require an LEM to be identified first to then go to the operation of finding an LLM. Finally, whether a GEM is identified as an LEM, i.e., whether a specific sample can be associated with a GEM (it would contain the concepts within the GEM), is indicative of whether the GEM is representative of the classification task within the given dataset. Hence, if we look at the number of GEMs for which no sample is matched as LEM, this number is expected not to be zero: some GEMs with very low or very high typicality are probably never met in a dataset. Besides, this number is expected to evolve over time within a debugging session: it would decrease once more LEMs are identified, and it would increase one new GEMs are identified. This is exactly the behavior that we identify. For instance, at the operation 100 and 200, new GEMs are collected for the class lobster, and hence the number of GEMs without any LEM also increases. Then, a little after these operations, the number of GEMs without any LEM decreases, as new LEMs are identified.

We can also formulate expectations about the system based on the GEMs' typicality. Naturally, some GEMs might often be LEMs (typically the GEMs of reasonable typicality), while others might be less often LEMs, i.e., those of very low typicality—they probably represent edge cases—, as well as those of very high typicality—while the certainty of the class would be high, an image with all the concepts of the class is typically rare to obtain—if the dataset is representative of the reality. In other words, reasonable typicality is expected to be more often an LEM than higher or lower typicalities. That

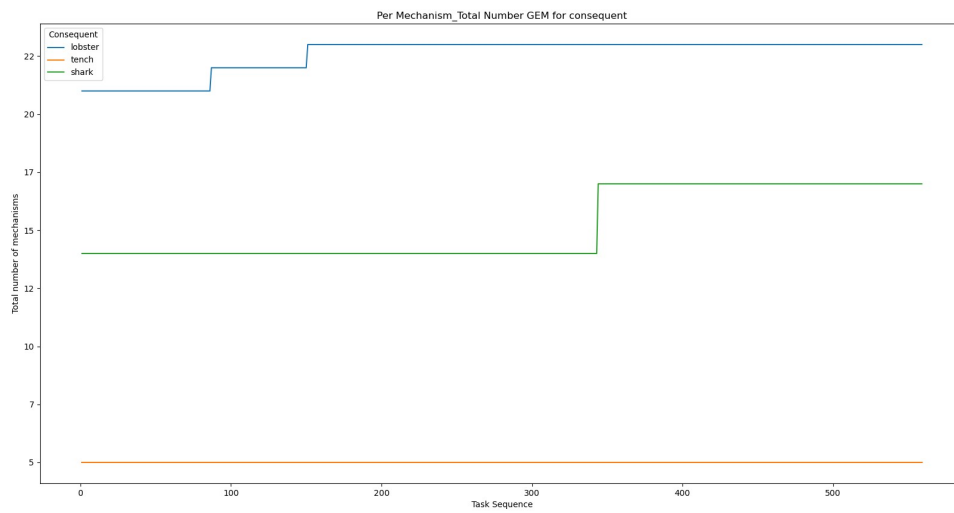


Figure 7.7: Increase in GEMs count across the session for the fish data set

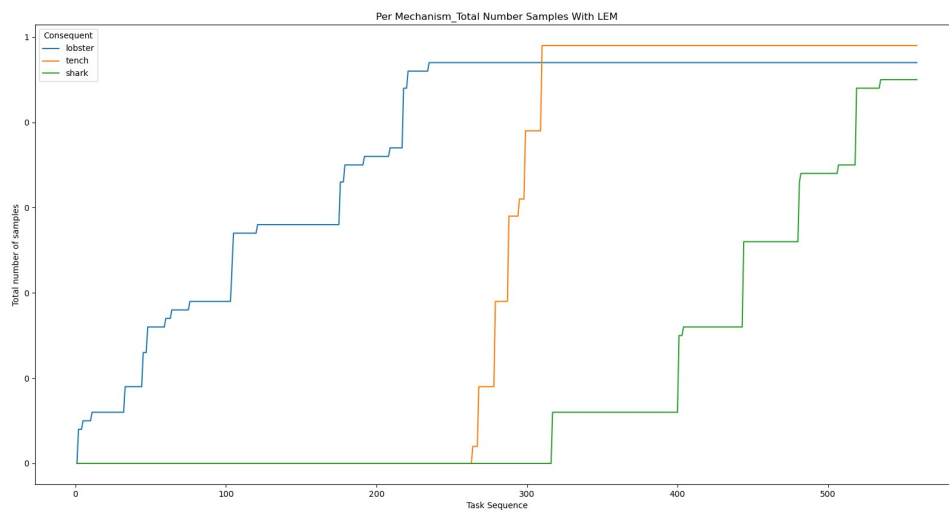


Figure 7.8: Increase in LEMs count across the session for the fish data set

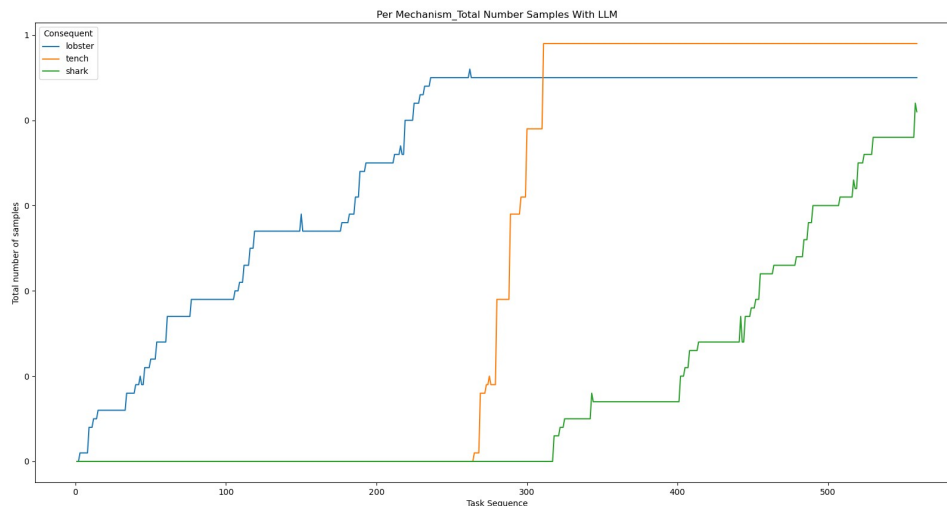


Figure 7.9: Increase in LLMs count across the session for the fish data set

is what we also observe when plotting the number of LEMs per typicality over time. In a similar fashion, we expect mechanisms of higher typicality to be more often LLMs than mechanisms of much lower typicality since they would be very uncommon edge cases. This is also what we observe when plotting the evolutions of LLMs over time per typicality.

Satisfaction of LEMs by LLMs

Next to observing the individual numbers of GEM, LEM, and LLM, we can reason over the connection between these numbers in terms of whether GEMs and LEMs are satisfied by LLMs. Over time, we discover the expectations for more samples, hence GEMs become LEMs at higher frequency. Simultaneously, we identify the model mechanisms (LLMs) over time. Note that a GEM for one sample can only be identified as an LLM if it has already been identified as an LLM in the current configuration. Hence, the ratio of times one GEM is expected and satisfied over expected can only be above 1. And we expect this ratio to decrease once a new LEM is identified, and increase once the LEM is identified as LLM. This is exactly what we observe when plotting the ratio of the number of times one mechanism is an LLM versus the number of times it is an LEM. Besides, we can again delve into deeper analysis based on typicalities. And as explained earlier, we can expect mechanisms of reasonable typicality to be satisfied more often than others, and hence to present a ratio closer to 1 than the others that should be closer to 0. This is also what we observe. These are again signs that the computations in our system are correct.

In the lenient case, one mechanism is considered to have a counter-example solely when it is the highest-typicality LEM for a sample and it is not satisfied (if another LEM is satisfied for this sample, then we do not conclude anything about the other LEMs). For implementation purposes, anytime a new LEM is identified, it is marked as not being satisfied and hence constituting a counter-example, until the satisfaction has actually been checked for this sample and a LLM has potentially been identified. Hence, anytime new LEMs are found, new counter-examples should “appear” in the system, and anytime satisfaction is actually checked, the number of counter-examples might decrease. Overall, over time, the number of counter-examples should increase, since we biased our dataset and model, so it is sound to assume that often the model follows behaviors (mechanisms) that are not expected at all. When plotting the absolute number of counter examples over time (i.e., the number of samples that are counter-examples) or the number of mechanisms for which a counter-example exist, this is again exactly the behavior we identify as shown in Figure 7.10.

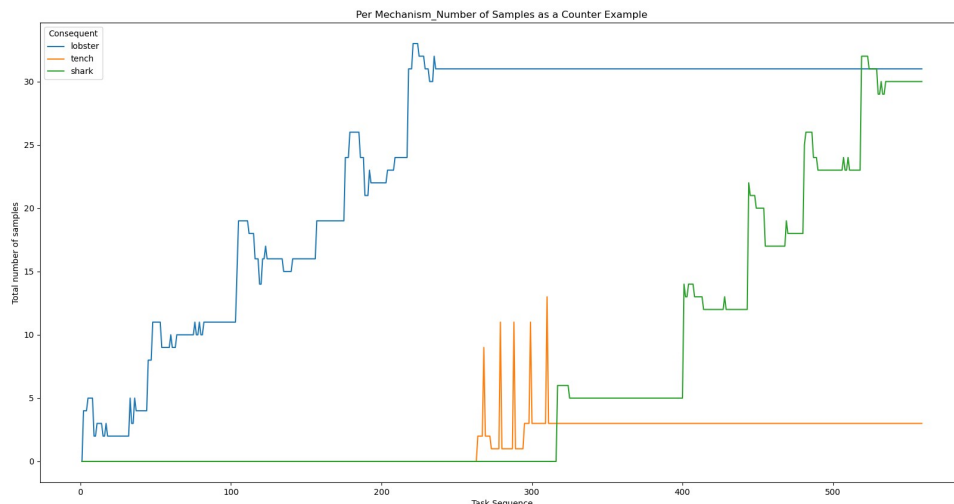


Figure 7.10: Samples as Counter example for the fish dataset

7.2. Impact of batch size on Informativeness

As previously noted, the impact of different configurations on the workflows tends to yield slight variations. In this section, our attention shifts to exploring the influence of batch sizes. Specifically, we delve into the outcomes of two distinct configurations: one with 5 samples per batch and another with 15 samples per batch. It's noteworthy that the results obtained from these varying batch sizes demonstrate a remarkable similarity when considering the same workflow. This similarity across different sample numbers within a batch leads us to a noteworthy conclusion: the adjustments in these configurations do not appear to significantly impact the comprehensiveness or informativeness of the tool's output.

7.3. Cost-efficiency in high worker-accuracy contexts

In cases where Expert workers were employed, our observations revealed a remarkable accuracy, thereby validating the superiority of expert contributions. Given the consistent precision, we found that utilizing just one expert worker suffices, aligning with our initial expectations.

7.4. Cost-efficiency in Scenarios with Lower Worker Accuracy

Conversely, we delved into scenarios involving Non-Expert workers possessing accuracy levels of 0.7 and 0.8, revealing a more intricate cost-efficiency landscape. These scenarios encompassed a diverse range of configurations, with minimum worker counts set at 3, 5, 7, and 9, in combination with thresholds of 0.6, 0.7, 0.8, and 0.9. It is important to highlight that these thresholds were exclusively applied to the "Map GEM to LEM" operation, although their implications can be extrapolated to other operations in a similar fashion. Plots for 0.7 and 0.8 were similar in their outcomes therefore we ignore one of them.

In Fig. 7.11, the outcomes are presented through a box plot that portrays key statistical measures: minimum, 1st quartile, median, 3rd quartile, and maximum worker counts. The values on the Y-axis depict the worker count, while the X-axis represents task IDs, within which lists of operations are nested. The depicted diagram showcases an aggregation of worker counts across all operations within each task.

Upon close observation, a clear pattern emerges from the plot. It becomes evident that as the threshold values increase, there is a corresponding escalation in the required worker count. This consistent trend persists across various threshold values. As anticipated, the act of elevating the threshold value for the "Map GEM to LEM" operation leads to an augmented need for human computation to achieve the target accuracy level, especially in contexts involving non-expert workers. Our empirical findings strongly validate this anticipated relationship.

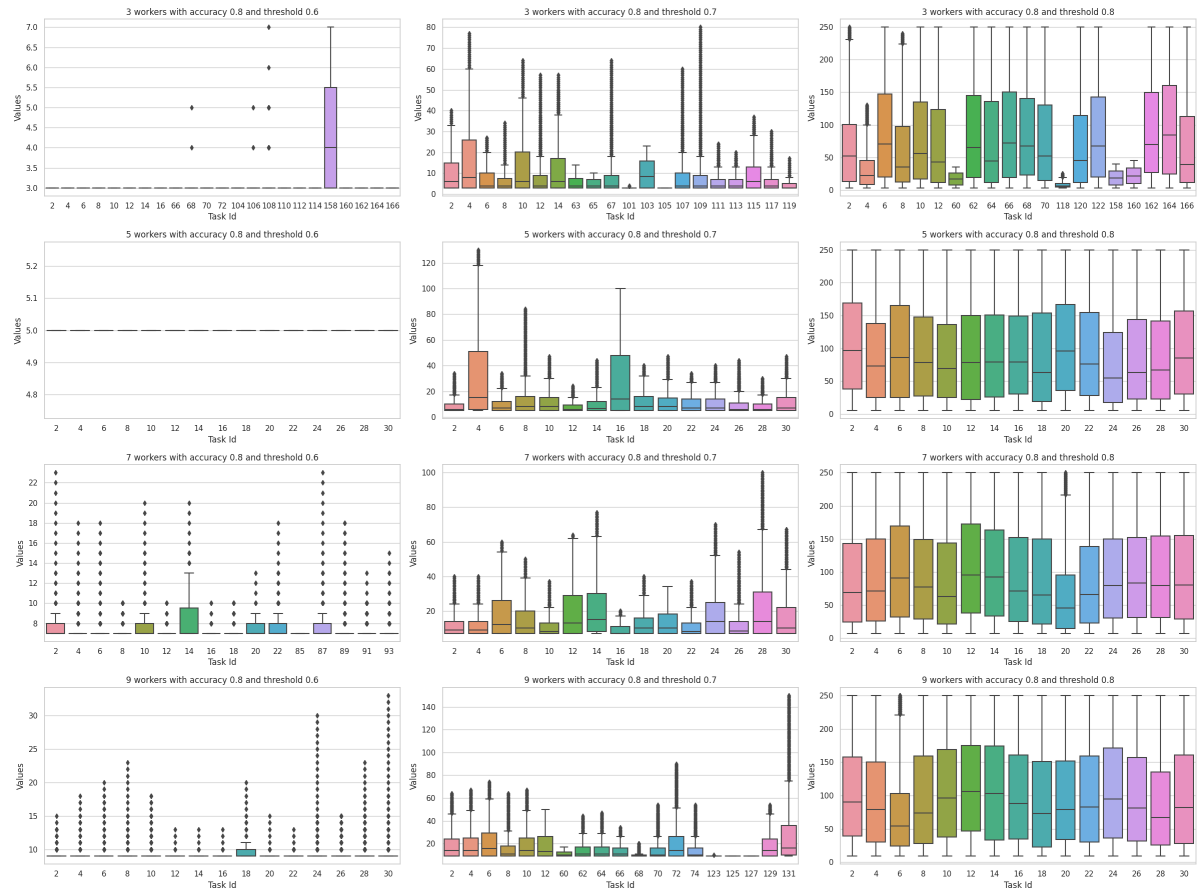
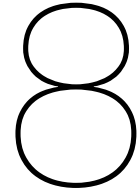


Figure 7.11: Box plot depicting the outcomes obtained from the simulator using workers with an accuracy of 0.8.

Interestingly, a comprehensive experiment employing a 0.9 threshold resulted in a consistent pattern where all operations reached the maximum worker count of 250. Given the high worker accuracy of 0.8 achieved in this specific experiment, it is reasonable to deduce that scenarios with lower worker accuracies would exhibit similar saturation, rendering further experimentation redundant. Therefore we ignored other experiments on the 0.9 threshold

Upon examination of the X-axis in each plot, a significant observation comes to light. The task IDs exhibit considerable variation, occasionally reaching a maximum of 30 and, in other instances, exceeding 100. These task IDs serve as indicators that allow us to discern differences between two consecutive IDs. If the difference exceeds 2, it indicates the execution of additional operations, while a consistent difference of 2 points toward an agreement on deviating from accuracy. Although this variability appears random, it's important to note that real-world applications rarely exhibit such patterns, as a majority of answers would typically be accurate.

This variability in task IDs can be attributed to the inherent randomness embedded within the system. On certain occasions, this randomness leads to situations where user responses deviate from accuracy. This outcome is a direct result of the foundational simplicity embedded within the structure of our simulator.



Discussion & Conclusion

8.1. Discussion of the Results: Positives, Limitations, & Future Research Opportunities

In this study, we aimed to assess and validate the correctness and informativeness of our system through a series of experiments. Our primary focus was to verify whether the system's outputs align with our expectations and whether it successfully uncovers the underlying behaviors of machine learning models. We conducted experiments using various datasets, model biases, and worker accuracies, and we analyzed the results comprehensively.

The first part of our analysis concentrated on the correctness of the system's outputs. We evaluated whether the identified Generalization Error Mechanisms (GEMs) and Local Explanation Mechanisms (LEMs) met our expectations in terms of satisfying model behaviors and identifying counter-examples. Our findings showcased that the outputs consistently aligned with our expectations. The identified GEMs often had a high likelihood of being satisfied and could effectively explain the model's behavior. Additionally, the Local Explanation Mechanisms (LEMs) correctly mapped to GEMs with high fidelity, demonstrating our system's capability to identify relevant samples that showcase model behaviors.

We further investigated how the typicality of mechanisms influenced their satisfaction rates. Our results validated our hypothesis that mechanisms of reasonable typicality were more likely to be satisfied, whereas mechanisms with extremely high or low typicality were less likely to be fulfilled. This consistency across different configurations reinforced the reliability of our system's outputs.

The analysis then shifted towards assessing the informativeness of the system's outputs. We explored how the results revealed novel insights about models that were naturally trained and had no known biases. These insights highlighted unexpected behaviors, such as the model's tendency to associate specific concepts (like the presence of people) with certain scene classes, indicating potentially incomplete learning. The system proved its ability to uncover these complex relationships within the models, enhancing its role as a tool for model interpretation.

To evaluate the efficiency of the system under different conditions, we explored scenarios with varying worker accuracies and batch sizes. Our findings indicated that the performance was consistent across different batch sizes, reaffirming its robustness. However, the impact of worker accuracy was significant, with lower worker accuracy leading to increased human computation requirements. In cases with non-expert workers, higher accuracy thresholds led to a higher demand for human input, aligning with our expectations.

8.1.1. Guidelines on the use of the system

When utilizing the system, a pivotal aspect is the developer's interaction with the current configuration settings, which are contingent upon their specific preferences and the crowd workers they intend to target. In this context, a judicious approach is recommended—one that combines flexibility with efficiency. The following discourse expounds on the optimal strategies for configuring the system to yield informative results while optimizing resource usage.

A straightforward and expedient method for developers to initiate their interactions with the system is to opt for the "all concepts" workflow. This approach not only efficiently gathers copious amounts

of information but, notably, it also demonstrates swiftness in execution. In our simulator, it consistently exhibited the shortest completion time. For developers seeking comprehensive analyses of each mechanism, our array of workflows offers a viable solution. By selecting appropriate workflows, diverse layers of insights can be extracted from the system's outputs.

While a holistic exploration of mechanisms at both class and lower levels can be alluring, a prudent suggestion is to begin with class-level mechanisms. This recommendation stems from the potential for generating an extensive number of tasks, particularly if both levels are examined concurrently. By focusing initially on class-level mechanisms, developers can circumvent unforeseen costs arising from the extensive task generation associated with delving directly into lower-level concepts. Furthermore, prior knowledge of the magnitude of sub-concepts aids in planning the execution of lower-level analyses.

When contemplating batch composition and order, developers encounter a crucial decision-making juncture. For those aiming to analyze every sample exhaustively, the choice between batch compositions and orders remains relatively unaltered. However, situations arise where developers seek a rapid overview from a compact sample set. In such instances, the "diverse composition" approach emerges as advantageous. This technique facilitates the accumulation of data from multiple classes within a limited number of batches, expeditiously providing a broad insight into the system's performance.

8.1.2. Limitations of the system

Within the context of our research, it is imperative to address certain limitations that have emerged during the development of our system. These limitations, while inherent to the current iteration, pave the way for future enhancements and refinements. In this section, we expound upon these limitations, suggesting potential avenues for improvement and innovation.

One limitation in the present system revolves around the utilization of "max/total" values in the certainty of operations. While functional, this aspect presents room for refinement. For instance, enhancing the algorithms governing these values could contribute to heightened precision and effectiveness.

Another limitation pertains to the restriction of annotating with a single bounding box. Recognizing the potential utility of employing multiple bounding boxes, we acknowledge the potential of expanding this functionality. A forward-looking approach involves parallelizing certain system operations. For instance, instead of relying solely on bounding boxes for annotation, integrating saliency maps could offer a more nuanced means of annotating. Analyzing pixels within these maps to detect highlighted regions could contribute to improved visibility determinations. This, in turn, holds promise for assessing full or partial visibility. By outlining this prospective development, we envision an enriched annotation process that aligns with evolving requirements.

To bolster the accuracy and reliability of the system, there is an opportunity to adopt a more granular, per-pixel annotation technique. This refined approach could be harnessed to augment the precision of saliency maps. By enabling a more automated, pixel-level method, the current reliance on worker inputs could be mitigated, leading to a more accurate and dependable system. This refinement could include the integration of a certainty threshold and the incorporation of signals within the "Collect Complete LLM" operation. Presently, the system accepts concepts proposed by over 50% of workers. However, exploring alternative strategies for concept acceptance could yield more robust outcomes.

Considering the operational infrastructure, a constraint relates to internet connectivity for workers. To address this, an implementation using web sockets could be explored. This advancement would enable seamless continuation for workers even in cases of connection interruptions. This not only enhances worker experience but also augments the system's efficiency.

Lastly, our concept's attribute scope is presently limited to color. Yet, the inclusion of additional attributes holds promise for a more comprehensive and nuanced analysis. By expanding the spectrum of attributes considered, we anticipate a richer diagnostic capability that aligns with diverse real-world scenarios.

Conceptual simplifications versus correctness of the outputs

The results obtained from our study undeniably highlight the success of our efforts. The diagnostic framework we have devised has demonstrated promising outcomes in its ability to evaluate computer vision models. The information gleaned from the outputs indicates a substantial degree of accuracy in the diagnoses. Yet, it is paramount to address the simplifications that have been strategically incorporated into our methodology. One of the notable simplifications pertains to the entities and attributes

we have considered. Our focus has primarily revolved around simpler instances of these components. This selective approach was adopted to streamline the initial stages of our framework's development. It is important to acknowledge that while this simplification introduces a certain degree of abstraction, the outputs furnished remain informative and constructive. The diagnostic information provided, although not devoid of imperfections, still holds intrinsic value for developers aiming to gain insights into their models' performance.

Despite the fact that the diagnostic outcomes generated exhibit a commendable level of informativeness, it is imperative to recognize that perfection remains an elusive goal. The conceptual simplifications we have embraced do introduce limitations. While the outcomes are certainly beneficial, there exists a scope for further refining the framework to enhance its accuracy and comprehensiveness.

Cost-efficiency

In our pursuit of cost-effectiveness, we have directed our efforts towards minimizing unnecessary expenditures while maintaining the efficacy of our system. A fundamental strategy in this regard involves the identification and elimination of avoidable patterns, thereby fostering resource optimization. To this end, we have meticulously structured the system to prevent the presence of duplicate data. This systematic approach ensures that received data triggers updates across relevant segments of the system, thus enhancing its cohesiveness and reducing redundancy.

Our current efforts represent just the initial phase of a broader journey. Anticipating future advancements, we envision the possibility of deploying alternative operation methodologies. These methodologies may harness equivalent knowledge acquisition while demanding reduced human computational input. Furthermore, we envision the development of streamlined operations that efficiently accumulate a wealth of information within a limited number of tasks. The inherent advantage of these operations lies in their capacity to swiftly gather valuable insights during the initial stages of engagement. This early efficiency benefits developers by enabling them to execute more compact sessions, enhancing the overall productivity of the system.

Scope of the system

This section delves into the defined scope of the system, marking the boundaries of our research. By elucidating what our system encompasses, we provide a clear understanding of its focus and limitations.

Our system operates exclusively within the realm of accurately labeled data. An area for future exploration lies in the domain of incorrect predictions and their corresponding saliency maps. While not considered in the current iteration of the system, we recognize the potential insights that can be gleaned from analyzing incorrect predictions. This aspect is earmarked for incorporation in the subsequent evolution of the system. By encompassing both correct and incorrect predictions, our system's diagnostic capacity could be significantly augmented, offering a more comprehensive understanding of model behavior.

Presently, our system's focal point is concentrated on the task of computer vision image classification. However, the horizon of applicability for this framework extends beyond this single task. An intriguing avenue for future investigation involves exploring the adaptability in diverse realms of computer vision, as well as potentially venturing into the terrain of natural language processing tasks. This expansion has the potential to unveil hitherto undiscovered insights and applications, enriching the versatility of our system.

In essence, our system's scope delineates its current reach and foreseeable evolution. As we move forward, these established boundaries will serve as guiding parameters, ensuring that our research remains well-defined yet poised for expansion. By addressing both the potential of analyzing incorrect predictions and the broader applicability across different tasks, we pave the way for the system to advance as a robust and adaptable tool in the domains of computer vision and beyond.

8.1.3. Limitations of the experimental setup

Within the context of our experimental endeavors, it is important to acknowledge certain limitations inherent to our approach. These limitations stem from a deliberate focus on crafting an initial framework designed for diagnosing computer vision models. The scope of our resources, which encompassed both time and capacity, influenced the extent of our experimental pursuits.

It is important to underline that our experiments have been conducted exclusively within a simulated environment, without involving actual crowd workers. The underlying rationale behind this approach

was to swiftly establish a foundational version of the diagnostic framework for computer vision models. The constraints we encountered in terms of available resources compelled us to prioritize simplicity in the initial phase. We emphasize that future research endeavors should focus on refining and elaborating upon the simplified framework we have introduced. An essential aspect of this refinement involves subjecting the system to empirical evaluation through experimentation with real crowd workers.

The preliminary edition of our framework has already demonstrated considerable promise in realizing the intended objectives of our study. However, it is imperative to validate our findings and draw substantiated conclusions by conducting experiments involving actual crowd workers. Furthermore, an expansion of the range of datasets employed is warranted, including datasets that hold significance in various industries such as medical domain. Despite having employed a limited number of datasets, we have managed to glean insightful results. Yet, our ambitions extend towards investigating diverse domains, such as the medical field, in order to thoroughly scrutinize the system's outputs and derive meaningful analyses.

While our present work has centered around relatively uncomplicated datasets, our aspirations encompass the examination of models that operate across more extensive classes and larger datasets. The realization of this aspiration hinges upon the availability of additional resources. Such an endeavor holds the potential to illuminate a pivotal facet of our objectives—to facilitate developers in comprehending their models more comprehensively through nuanced diagnosis.

8.2. Conclusions

In this work, we introduce a pioneering framework in the intricate realm of model diagnostics. Our contribution lies in presenting the system design and its meticulous implementation. The framework harnesses the power of a model's saliency map, combined with images from the training process, as its input. By seamlessly integrating human computation input, the system crafts operations and tasks to extract pertinent data from the input, unraveling what the model comprehends, where it falls short, and what it should ideally have grasped.

Our comprehensive analysis resoundingly affirms the harmony between the outputs and our pre-conceived expectations regarding the system's informativeness and correctness. Rigorous simulated experiments, employing annotated data from experts simulating workers, have meticulously probed the system's configurations. The culmination of these efforts yields high-quality outcomes, albeit in the backdrop of recognized simplifications. As we traverse this research trajectory, the avenues for improvement, expansion, and innovation unfold before us. The system's evolution remains ongoing, poised to reshape the landscape of model diagnostics, ushering in heightened precision, efficiency, and insights.

Gazing ahead, our research offers a guiding beacon for future endeavors. As we acknowledge and address inherent limitations, streamline operational efficiency, and precisely define the system's scope, we lay the foundation for the system's burgeoning growth. This amalgamation of insight, innovation, and adaptability epitomizes the legacy of our work. With unwavering commitment, we persist in pushing the horizons of machine learning model interpretation and comprehension, a journey that promises to redefine the boundaries of this dynamic field.

Bibliography

- Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., & Kim, B. (2018). Sanity checks for saliency maps. *Advances in neural information processing systems*, 31.
- Arous, I., Dolamic, L., Yang, J., Bhardwaj, A., Cuccu, G., & Cudré-Mauroux, P. (2021). Marta: Leveraging human rationales for explainable text classification. *AAAI*, 35(7), 5868–5876.
- Balayn, A., Rikalo, N., Lofi, C., Yang, J., & Bozzon, A. (2022). How can explainability methods be used to support bug identification in computer vision models? *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 1–16.
- Balayn, A., Rikalo, N., Yang, J., & Bozzon, A. (2023). Faulty or ready? handling failures in deep-learning computer vision models until deployment: A study of practices, challenges, and needs. *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 1–20.
- Biessmann, F., & Refiano, D. I. (2019). A psychophysics approach for quantitative comparison of interpretable computer vision models. *arXiv preprint arXiv:1912.05011*.
- Escalante, H. J., Escalera, S., Guyon, I., Baró, X., Güçlütürk, Y., Güçlü, U., van Gerven, M., & van Lier, R. (2018). *Explainable and interpretable models in computer vision and machine learning*. Springer.
- Gao, I., Ilharco, G., Lundberg, S., & Ribeiro, M. T. (2022). Adaptive testing of computer vision models. *arXiv preprint arXiv:2212.02774*.
- Ghorbani, A., Wexler, J., Zou, J. Y., & Kim, B. (2019). Towards automatic concept-based explanations. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in neural information processing systems*. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/77d2afcb31f6493e350fca61764efb9a-Paper.pdf
- Hendricks, L. A., Burns, K., Saenko, K., Darrell, T., & Rohrbach, A. (2018). Women also snowboard: Overcoming bias in captioning models. *Proceedings of the European conference on computer vision (ECCV)*, 771–787.
- Kortylewski, A., Liu, Q., Wang, A., Sun, Y., & Yuille, A. (2021). Compositional convolutional neural networks: A robust and interpretable model for object recognition under occlusion. *I. Journal of Computer Vision*, 129(3), 736–760.
- Larsson, S., & Heintz, F. (2020). Transparency in artificial intelligence. *Internet Policy Review*, 9(2).
- Leclerc, G., Salman, H., Ilyas, A., Vemprala, S., Engstrom, L., Vineet, V., Xiao, K., Zhang, P., Santurkar, S., Yang, G., et al. (2022). 3db: A framework for debugging computer vision models. *Advances in Neural Information Processing Systems*, 35, 8498–8511.
- Matsubara, M., Borromeo, R. M., Amer-Yahia, S., & Morishima, A. (2021). Task assignment strategies for crowd worker ability improvement. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW2), 1–20.
- Mundhenk, T. N., Chen, B. Y., & Friedland, G. (2019). Efficient saliency maps for explainable ai. *arXiv preprint arXiv:1911.11293*.
- Sharifi Noorian, S., Qiu, S., Gadiraju, U., Yang, J., & Bozzon, A. (2022). What should you know? a human-in-the-loop approach to unknown unknowns characterization in image recognition. *Proceedings of the ACM Web Conference 2022*, 882–892. <https://doi.org/10.1145/3485447.3512040>
- Singla, S., Nushi, B., Shah, S., Kamar, E., & Horvitz, E. (2020). Understanding failures of deep networks via robust feature extraction.
- Stacey, J., Belinkov, Y., & Rei, M. (2022). Supervising model attention with human explanations for robust natural language inference. *AAAI*, 36(10), 11349–11357.
- Wang, S., & Gong, Y. (2022). Adversarial example detection based on saliency map features. *Applied Intelligence*, 1–14.
- Wiley, V., & Lucas, T. (2018). Computer vision and image processing: A paper review. *International Journal of Artificial Intelligence Research*, 2(1), 29–36.
- Yang, J., Bozzon, A., Gadiraju, U., & Lease, M. (2023). Human-centered ai: Crowd computing. *Frontiers in Artificial Intelligence*, 6, 1161006.

Ziengs, B. (1970). A human-in-the-loop system for interpreting image recognition models. <https://repository.tudelft.nl/islandora/object/uuid:6ebbf05-e37d-45b9-b0ad-1844c9d7f298?collection=education>