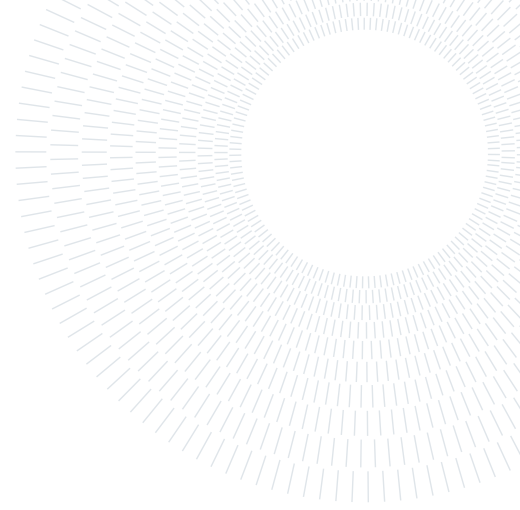




POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE



A Human-in-the-Loop Approach for Post-hoc Explainability of CNN-based Image Classification

TESI DI LAUREA MAGISTRALE IN
COMPUTER SCIENCE AND ENGINEERING - INGEGNERIA INFORMATICA

Antonio De Santis, Matteo Bianchi, 10639721, 10582370

Advisor:
Prof. Marco Brambilla

Co-advisors:
Dr. Andrea Tocchetti

Academic year:
2021-2022

Abstract: Transparency and explainability in image classification are essential factors for establishing trust in machine learning models and detecting biases and errors. State-of-the-art explainability methods generate heatmaps that highlight the regions of the image where a specific class is identified, without providing a complete explanation of how the model arrived at its decision. Striving to cover such a need, we propose a post-hoc technique for generating comprehensive local explanations that provide an overview of the feature extraction process of the model. These explanations consist of a layer-wise visualization of the features extracted by the model from the input image and we refer to them as Abstract Network Visualizations (ANV). Such features are represented by heatmaps generated from clustering and merging similar Feature Maps to which we associate a weight using Grad-CAM, a local explainability technique. These heatmaps are also described by a set of labels collected by means of a gamified crowdsourcing activity, which further improves the interpretability of our local explanations. Finally, we show that labels can also enable the production of global explanations by aggregating similarly labeled maps across multiple images.

Key-words: explainability, image classification, machine learning, gamification, crowdsourcing.

1. Introduction

1.1. Context and Problem Statement

In recent years, Deep Neural Networks have transformed the field of Artificial Intelligence by revolutionizing the way machines learn. Convolutional Neural Networks have emerged as the state-of-the-art for image classification tasks [21], thanks to their exceptional ability to recognize patterns and features in images. However, as AI models have become more powerful, their decision-making process has become increasingly complex and less transparent. As a result, the use of the term *black-box* has become prevalent to describe such models, since only their input and output are known, while what happens inside is too intricate for humans to comprehend. This issue leads to opacity in AI decisions, which is a serious concern [52] that can lead users to lose trust in such systems [24]. Furthermore, blindly trusting AI can have disastrous repercussions, including the loss of human life in fields such as healthcare or autonomous driving in which image classification systems are becoming increasingly employed [9, 50]. In response to these concerns, the EU's General Data Protection Regulation

(GDPR) introduced a right to obtain an explanation for automated decisions¹. Furthermore, the European Commission later proposed the Artificial Intelligence Act (AIA) which introduced transparency requirements for high-risk AI systems².

The problem of transparency is not only a matter of trust. Debugging black-box models is a challenging task without insights into how the model generated its predictions. Without understanding the reasoning behind errors and biased predictions, it can be challenging to address and fix these errors. This increasing need for transparent AI has been recognized by the scientific community, which led to the rise of a new research area called Explainable Artificial Intelligence (XAI). XAI has made significant progress in developing techniques for producing explanations of AI decisions, although some critical limitations still need to be addressed. For image classification, these techniques mainly focus on producing visual explanations in the form of heatmaps that highlight the pixels of the image that contributed the most to the output. While these heatmaps allow users to know whether the AI is looking at the "right thing", a complete explanation should also provide insights into how the AI produced the correct classification. Furthermore, current XAI techniques tend to focus on providing *local explanations* (i.e., explanations of the output for a certain input image) only. While this is not an issue for tabular data, where locally important features can be generalized to produce *global explanations* (i.e., explanations of the overall behaviour of the system), the same cannot be said for images. If local explanations only consist of highlighted pixels, they are very difficult to generalize because the location of these pixels only has meaning in the context of the analyzed image. To overcome these limitations, local explanations must provide a comprehensive overview of the entire decision-making process of the machine. At the same time, it should be possible to aggregate them to produce global explanations since local ones may not accurately represent the model's general behaviour when taken individually.

1.2. Proposed Solution

We propose a technique called Abstract Network Visualizations (ANV) that involves the use of human knowledge to create comprehensive local explanations for image classifications performed by Convolutional Neural Networks (CNNs), without requiring any modification or performance trade-off. These visualizations offer a detailed view of the image features and patterns the CNN identifies at each stage of its execution, along with their respective importance towards the output. We believe that having an overview of the AI decision-making process can increase transparency, thus establishing greater trust in machine classifications.

These features/patterns the machine identifies are extracted by clustering feature maps (i.e, the outputs of each neuron of the network), whose importance can be computed as a scalar weight using Grad-CAM, a state-of-the-art XAI technique for the explainability of CNNs. Moreover, our approach incorporates crowdsourcing and gamification to associate visual explanations with human concepts in the form of labels. For example, if the machine detects the presence of the sky as an important feature in identifying a plane, this information is represented as a heatmap highlighting the sky, labeled with the word "sky". Such a hybrid approach that combines visual and textual explanations significantly improves interpretability [47] while enabling the generation of global explanations by aggregating multiple local ones.

1.3. Document Structure

The rest of the document is organized as follows. Section 2 provides an introduction to AI and CNNs, along with a discussion of the current state-of-the-art explainability approaches. In Section 3, Abstract Network Visualizations are presented and thoroughly explained. Section 4 describes the necessary systems implemented, including the clustering process and the crowdsourcing platform. In Section 5, we report on the experiments conducted to validate our methodology and discuss the final results obtained. Finally, in Section 6, we draw some conclusions discussing the limitations of our approach and provide insights into how these limitations can be overcome in future research.

2. Background and Related Works

This section aims to introduce Convolutional Neural Networks, the model we are seeking to explain, and the state-of-the-art explainability of these models. We begin by providing an overview of CNNs. Following, we present a summary of the available explainability techniques for CNN-based image classification and how human knowledge has been effectively incorporated into this field.

¹<https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679>

²<https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A52021PC0206>

2.1. Artificial Intelligence and Convolutional Neural Networks

Artificial Intelligence (AI) is a rapidly growing field aiming to develop systems capable of performing tasks that normally require human intelligence, such as perception, reasoning, and decision-making. One of the most widely recognized approaches is Machine Learning (ML), a subset of AI that focuses on developing algorithms allowing systems to improve their performance through experience. These algorithms are designed to automatically learn from data and make predictions without the need for explicit guidance [56].

One of the most popular and powerful types of ML approaches is called Deep Learning (DL). DL utilizes Artificial Neural Networks (ANNs), i.e., sophisticated models designed to approximate a transfer function that maps the inputs and outputs of a certain dataset. They are made up of layers of fully interconnected nodes (as represented in Figure 2), called neurons. Each neuron receives input signals from the neurons of the previous layer in the form of numerical values. These inputs are multiplied by their associated weights and then summed together. The neuron then applies a non-linear mathematical operation called activation function (e.g., ReLU) to this sum. The output of the activation function is then passed on to the next layer of neurons in the network, where it is used as input. The process is repeated until the last layer is reached, at which point its final output is provided. This structure is particularly powerful since it can approximate almost any function [55].

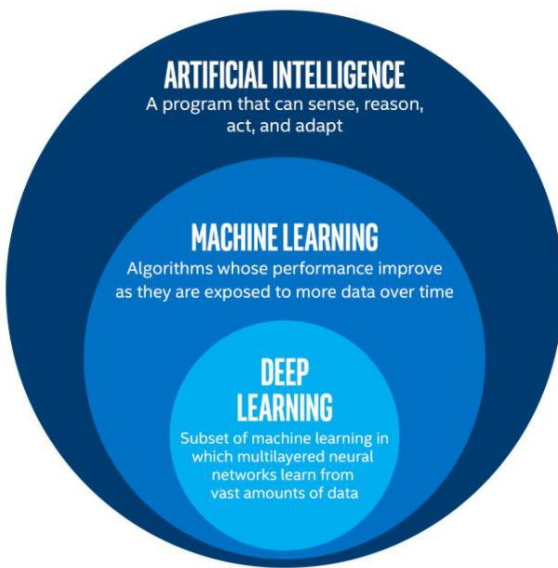


Figure 1: Venn Diagram for AI, ML & DL.

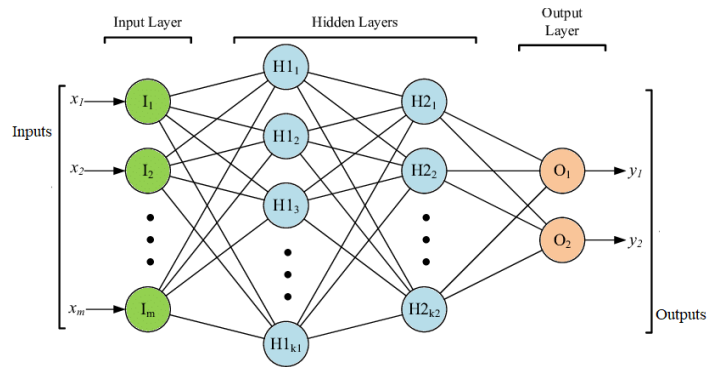


Figure 2: Example of an Artificial Neural Network with two Hidden Layers. The inputs (x) are passed to the Input Layer, processed through the Hidden Layers and a score (y) for each output class is finally computed in the Output Layer.

The main reasons ANNs and DL have become increasingly popular in recent years, besides their high flexibility, are the availability of huge amounts of data for training and the big improvements made in computational power. This combination has made it possible to build and train neural networks with many layers, called Deep Neural Networks (DNNs), which can learn very complex non-linear relationships from large amounts of data [39].

The current standard for computer vision tasks, such as image classification [22] and object detection [16], is represented by a subtype of DNNs called Convolutional Neural Network (CNN) [36]. As it is impractical to directly feed images into a basic DNN because of their high dimensionality with a multitude of non-essential information, a CNN introduces convolutional and pooling layers. The former can extract meaningful information from the input image while the latter condense this information in a lower dimensionality. Convolutional layers perform a convolution operation between the input and a set of matrices of trained weights called filters, followed by the application of a non-linear activation function. The result of these operations is a set of feature maps which represent where a certain feature is found in the image. Pooling layers are then used to reduce the size of the feature maps, condensing the extracted features. The sequential application of convolution and pooling is named *feature extraction*. When the dimensionality of the image is reduced to the level of a vector of numerical values, it is fed to a set of fully connected layers for classification. An example of a basic CNN architecture is shown in Figure 3.

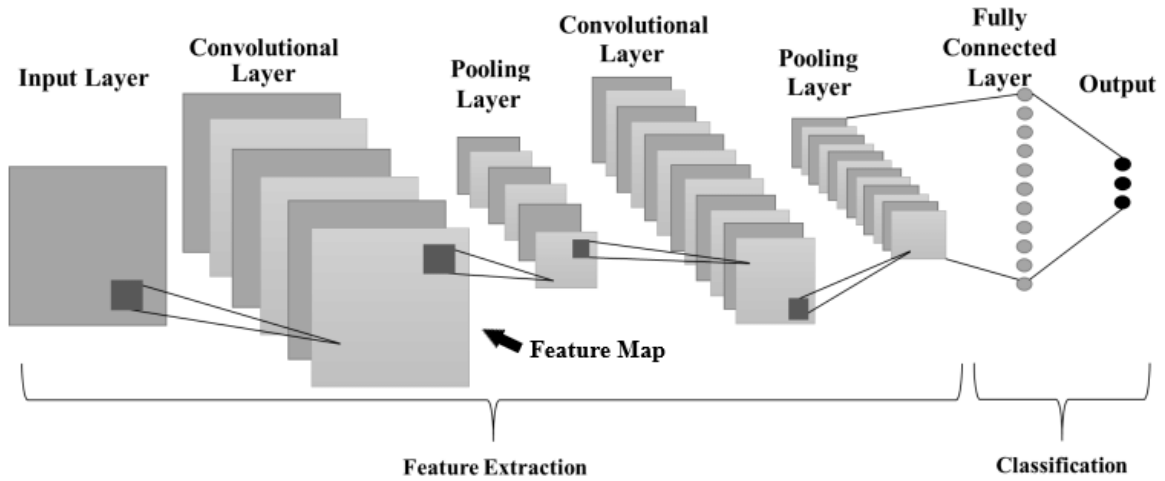


Figure 3: An example of a typical CNN consisting of multiple convolutional and pooling layers, followed by one fully connected layer.

The development of CNNs, or more in general of DNNs, improved the performance and the accuracy of ML models in a way that completely revolutionized the field of artificial intelligence. At the same time, the quantity and complexity of mathematical operations they perform have made their decision process incredibly difficult to be understood. For this reason, they are usually referred to as “black-box” models. The output of these models is known and often very accurate, although their internal logic is too complex to be interpreted by humans.

2.2. Explainability

The lack of transparency in black-box models has hindered trust and acceptance towards AI especially in sensitive domains such as medicine, defence, finance, and law where the consequences of a decision can be significant [12, 52]. Aiming to address this issue, the field of Explainable Artificial Intelligence (XAI) was created to make ML systems explainable. While there is no universal agreement on the definition of explainability, it can be described as a method of building an interface between humans and the AI system that can provide accurate explanations of the AI decisions that are also comprehensible to humans [3].

Developing an interface to understand the decision-making process of a trained black-box model is referred to as Post-hoc Explainability [57]. In contrast, Ante-hoc Explainability aims to make a model interpretable from the start of the training process by using so-called *white-box* models (e.g. Decision Trees) that are inherently transparent but not always as accurate or powerful as black-box models. The act of modifying the architecture of a black-box model to improve its transparency is also considered ante-hoc explainability since it requires to re-design and re-train the model. An example of a successful implementation of this idea is the interpretable CNN designed by *Zhang et al.* [59], which is characterized by a modification in the training algorithm to push the filters of the last convolutional layer towards the representation of a specific object part (e.g., head or torso of an animal). It is possible to gain insight about the patterns learnt by the CNN by then visualizing the outputs of these filters for different inputs. Another interesting example is InterpNET [5] which is an interpretable model based on a concatenation of a CNN with a language-generating neural network. InterpNET can provide both a classification of the image and its description which can be used as an explanation, although the drawback is the need for a big dataset of explanations and a potential problem of "explaining the explainer" (i.e., the fact that the explanation process is a black-box that in turn must be explained to be trustworthy). While ante-hoc explainability is an important aspect of model development, it has major limitations and it may not always be practical or possible for deployed black-box models. In these scenarios, post-hoc techniques become essential for understanding and interpreting the outputs of these models.

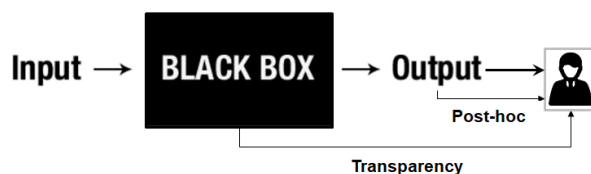


Figure 4: Two main categories of Explainable AI: transparency design (ante-hoc) and post-hoc [57].

The scientific literature has further classified post-hoc explainability into two main categories: model-agnostic and model-specific [17].

2.2.1 Model-Agnostic Explainability

Model-agnostic approaches for explainable AI refer to XAI methods that are independent of the underlying ML model and can be applied to any model regardless of its architecture or type. These methods don't access the internal workings of the model but instead focus on studying the input-output relationship. One of the most well-known contributions to the model-agnostic approach is the technique of Local Interpretable Model-Agnostic Explanations (LIME) [38]. LIME aims to explain the predictions of a black-box model by generating a set of perturbed versions of the input and analyzing how the output changes for each perturbation. This allows LIME to compute the most important input features contributing to the prediction made by the black-box model. In the context of image classification, features are represented by regions of the image (as represented in Figure 5), hence LIME provides visual explanations.

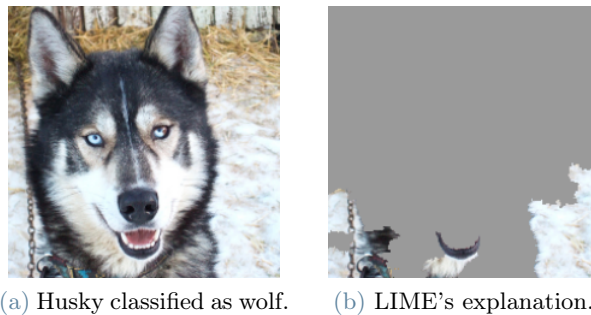


Figure 5: LIME was able to identify that the region of the image containing the snow was the most important in the classification [38].

A similar approach to LIME is CIU (Contextual Importance and Utility) [2]. It explains a model's predictions by identifying the input features deemed important for the model's outcome (Contextual Importance) while assessing the positivity or negativity of their contribution to the prediction (Contextual Utility) and providing an estimation of how favorable or unfavorable the feature value is for a given output class. Besides being in its early development, CIU has shown performance comparable to LIME [15] or even superior, especially in the medical domain [20]. Another very popular model-agnostic technique that focuses on measuring feature importance is SHapley Additive exPlanations (SHAP) [27]. It computes the contribution of each feature for a specific prediction by considering all possible coalitions of features and averaging their output predictions. This is done by using Shapley values, a method from cooperative game theory used to distribute a value among multiple players. For images, SHAP assigns importance scores to each pixel by calculating its contribution to the final prediction and then it provides a heatmap of the most important regions of the image. SHAP, LIME, and CIU are straightforward techniques to generate simple visual explanations for the predictions of any black-box model and they represent the state-of-the-art for model-agnostic explainability. However, their main limitation is that they don't extract information directly from the internal structure of a model since they only have access to inputs and outputs, and therefore may not be able to provide a robust explanation for certain predictions [1].

2.2.2 Model-Specific Explainability for CNNs

Model-specific approaches are techniques for explaining the decisions made by a specific ML model. These methods try to "open the black-box" and reverse engineer its internal structure to provide insights into how the algorithm is making its decisions. In our case, we consider CNN architectures. Zeiler *et al.* [58] were among the first to examine the internal workings of CNNs through their research on Deconvolutional Networks. They created a method for reconstructing activations of intermediate layers and projecting them back to the input pixel space, hence providing a visual representation of the information extracted (i.e., features) by the network from the input image up to a chosen layer. In a later study, Simonyan *et al.* [42] proposed a gradient-based visualization method to generate saliency maps, which they demonstrated to be a generalization of the deconvolution approach. Gradient-based visualization uses the back-propagation method to compute the gradients of the score function (i.e., the transfer function) with respect to the input image. The gradient reflects the responsiveness of the score to changes in a given pixel, i.e., higher gradients indicate that the score is more susceptible to changes for that pixel and, as a result, the pixel has a greater impact on the prediction. Hence,

gradients can be used to generate saliency maps that emphasize the important pixels in the image. SmoothGrad [43] and Guided Backpropagation [44] are two widely adopted techniques that are built upon this idea to further improve the quality of visualizations that can be generated.

Despite their effectiveness and ease of implementation, it has been shown that relying on gradients can sometimes result in an inaccurate assessment of feature importance. According to *Sundararajan et al.* [45], this is due to the saturation of gradients, which occurs when the score function flattens in the vicinity of the input. Consequently, the gradients become either very small or equal to zero even if the inputs have significant importance due to a significant output value of the score function. A solution to the saturation problem was provided by the DeepLIFT [41] method, which calculates the contribution of each feature in relation to a baseline input (i.e., a black image). In 2017, *Sundararajan et al.* [46] introduced Integrated Gradients, combining the gradient-based approach with the baseline input concept used in DeepLIFT. This methodology was able to overcome the limitations of gradient-based methods by integrating the gradients of the prediction over a path from a baseline input to the actual input. Additionally, it addressed the issue of implementation invariance, which occurs when the method is potentially sensitive to unimportant model implementation aspects, one of the main limitations of DeepLIFT. Nevertheless, a major disadvantage of Integrated Gradients is that it is significantly more computationally intensive than other gradient-based techniques and despite being more faithful this is rarely a worthwhile trade-off [28].

A different approach for extracting feature importance is through the extraction of activation maps (i.e., feature maps) which was originally presented by *Zhou et al.* [61]. They proposed substituting the complicated-to-understand fully connected layers with a Global Average Pooling (GAP) layer to reduce the feature maps into a single scalar value that represents a particular feature. Then, it is possible (as shown in Figure 6) to do a weighted linear sum of the feature maps to obtain a Class Activation Map (CAM), since the weights can be interpreted as the feature maps’ importance towards the prediction of a certain class. The CAM can be upscaled to the size of the input image and turned into a heatmap that can be used to identify the most important regions for a particular class. The CAM can be interpreted as a weighted average of the sections of the input image that were observed by the CNN just before the final prediction is made.

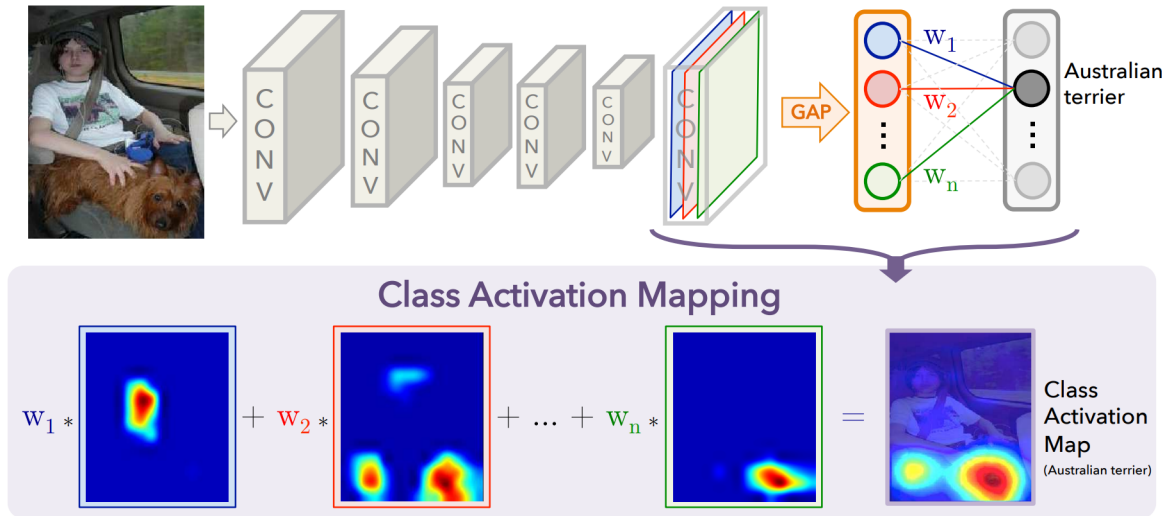


Figure 6: The fully connected layers of a CNN are substituted by a GAP and its weights towards a specific class are extracted. The CAM is then computed through a weighted sum of the feature maps and highlights the class-specific most important regions [61].

Compared to the methods mentioned so far, CAM stands out as the only method that can provide class-specific explanations. Its visualizations are unique for each class as they exclusively highlight the regions that are relevant to the chosen class. On the other hand, the main limitation of CAM is that it can only be applied to a specific kind of CNNs employing a GAP layer before prediction. For architectures using fully connected layers, CAM requires architectural changes that are significant enough to stress the concept of post-hoc explainability, such as substituting and re-training the last layer which can sometimes even hinder model accuracy [25]. *Selvaraju et al.* [40] later introduced a generalized version of CAM called Gradient-weighted Class Activation Mapping (Grad-CAM) which can be applied to any CNN architecture. The main idea behind Grad-CAM is that the weights needed to combine the feature maps can be calculated by applying a global average pooling on the gradients of the score (before the softmax) for a given class with respect to the feature maps (2.1). The final maps for a specific class are then generated by applying a ReLU to the weighted sum of the feature maps (2.2), ensuring that only features that positively impact the prediction are taken into consideration. The authors refer

to the final map as Class-discriminative Location Map since it highlights the class location.

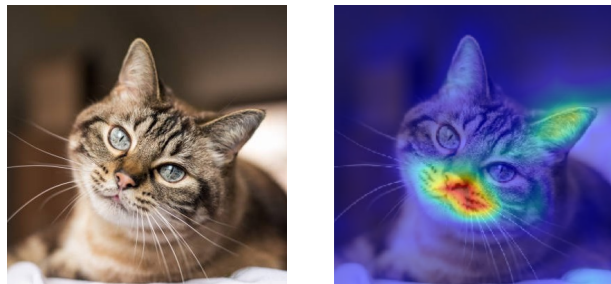
$$w_k^c = \frac{1}{Z} \underbrace{\sum_i \sum_j}_{\text{global avg pooling}} \overbrace{\frac{\partial y^c}{\partial A_{ij}^k}}^{\text{gradients}} \quad (2.1)$$

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left(\overbrace{\sum_k w_k^c A^k}^{\text{weighted sum}} \right) \quad (2.2)$$

Since Grad-CAM lacks fine-grained pixel-scale representations, it is often used in conjunction with other methods that provide greater detail in contexts where a higher degree of pixel-level detail is required. An example of such a combination is Guided Grad-CAM [40], which combines Grad-CAM with Guided Backpropagation.

Grad-CAM was a big step forward thanks to its computational efficiency, wide applicability and capability to outperform other visualization methods in a variety of contexts [11, 60]. Furthermore, another advantage of Grad-CAM is that it can be applied to any convolutional layer, although visualizations become progressively worse while moving to shallower layers as they have smaller receptive fields (i.e., elements of the feature map refer to a smaller area of the input image) and therefore focus on less semantic local features. An improved version of Grad-CAM called Grad-CAM++ was proposed by *Chattopadhyay et al.* [10]. This new version incorporates higher-order derivatives into the calculation of the class activation maps by considering second and third-order gradients and only using the positive ones. *Lerma et al.* [23] later observed that Grad-CAM++ is nearly equivalent to standard Grad-CAM with positive gradients, which they refer to as Grad-CAM⁺. The Grad-CAM⁺ algorithm differs from the original Grad-CAM in that it applies a ReLU directly to the gradients before the GAP step. According to *Lerma et al.*, the reason for this modification is to eliminate negative gradients which are associated with regions containing features from classes other than the one being analyzed. Several other variants of Grad-CAM have been developed (e.g., Eigen-CAM [34], Smooth Grad-CAM++ [35], and Score-CAM [53]). They usually perform better than Grad-CAM in specific contexts and slightly worse in others.

The presented methods generate heatmaps that highlight regions of the image with the highest influence on the prediction. However, what is lacking is an explanation of why these regions are relevant. While these regions are important as they contain patterns that the network has learnt to recognize, the interpretation of what these patterns may consist of is highly subjective. For example, if a Grad-CAM heatmap highlights the region of a cat's nose as being important for the prediction "cat", this might suggest that the network is using information about the nose to make the prediction. However, without further information about the specific features that the CNN has learnt, it is impossible to know for certain. Following on the cat's image example, it could be that the network is using information about the texture of the fur near the nose, or about the presence of whiskers, to make its prediction (see Figure 7).



(a) Correctly classified cat. (b) Grad-CAM explanation.

Figure 7: Grad-CAM highlights the region near the nose as the most significant for predicting a "cat", but it cannot be said for certain whether the network learnt to recognize the nose specifically, or possibly the fur or whiskers.

The authors of Grad-CAM proposed a possible way to address this issue by combining textual and visual explanations. Their idea was to combine Grad-CAM with a technique proposed by *Bau et al.* [6] to automatically assign names to neurons by training the model using images manually labeled with the names of the objects or features present in it. These names are then used as a textual description of what the Grad-CAM generated heatmap is highlighting to significantly improve the comprehensibility of these explanations.

Another way to identify the features a CNN has learnt was proposed by *Kim et al.* [19] in the form of a technique named Testing with Concept Activation Vectors (TCAV). This technique makes it possible to determine the importance of a specific feature or human concept for predicting a specific class. The method works by inputting

example images containing only one specific feature and observing the network’s predictions. For instance, they could input an image of stripes in a CNN that classifies zebras to determine if the network is using the concept of stripes in its prediction. This method can be used in detecting specific biases in neural networks (e.g., related to race) and can be considered complementary to Grad-CAM. Indeed, it provides a global understanding of model behaviour (i.e., global explanations) while Grad-CAM provides explanations of a specific prediction (i.e., local explanations). Both Grad-CAM and TCAV are still very effective in detecting biases and explaining AI decisions but at the same time, it has been shown that they can also introduce bias in the explanations. For Grad-CAM this happens when humans misinterpret why a region is highlighted due to biased assumptions while for TCAV the bias can be introduced by the images that are selected for representing a feature [49]. Therefore, it is crucial for future explainability frameworks that include these techniques to develop methods for mitigating any potential bias they may introduce.

2.2.3 Human Knowledge and Crowdsourcing

Despite the multitude of techniques and advancements in AI explainability, there are still limitations in ensuring that explanations are fully accurate and understandable from a human perspective. For this reason, many researchers [13, 29, 54] have turned to human-centred (i.e., human-in-the-loop) techniques that utilize human insight and reasoning to improve explanations of machine learning models. *Mishra et al.* [31] developed a crowdsourcing method for collecting high-level concepts for image classification explanations. The method involves presenting users with images and their correct labels, asking them to outline the location of the entity together with the features they used to identify it. The resulting data is then aggregated on a per-image and per-class basis to generate both global and local explanations. Later, *Tocchetti et al.* [48] proposed a two-player gamified crowdsourcing activity for collecting human concepts that can then be used for explainability purposes. In the proposed activity, one player is asked to guess the entity in a picture without seeing it by asking about its features through closed questions to the other player who provides the answers and takes note of the guessed features while also outlining them on the image. These methods can generate simple explanations and collect human-interpretable features using only input images. Despite easing the data collection process, such approaches should be combined with existing visual explanation techniques to contextualize them to an actual ML model prediction.

Among the researchers who employed human knowledge in conjunction with other explainability methods were *Lu et al.* [26]. They proposed using a game based on "Peek-a-Boom" to evaluate visual explanations generated by different XAI techniques (i.e., Grad-CAM, SmoothGrad, Guided Backpropagation, and Saliency Maps). In their implementation, only a small part of an image is initially shown, starting from the region deemed the most important by an explainability method. If the player cannot guess the image, more pixels are revealed. The number of pixels needed for the player to guess correctly determines a score for each explainability method, representing how well humans can interpret it. Grad-CAM ended up with the highest score since it was able to convey more human-understandable information in fewer pixels.



Figure 8: Example of a Peek-a-Boom game where an image of a cat is gradually revealed to the players at different exposure rates [26].

Human-in-the-loop approaches are not just effective at evaluating explanations but they also have the potential to significantly improve them [14]. *Mitsuhashi et al.* [32] proposed a framework to improve the explainability and the performance of Attention Branch Networks (ABNs) (i.e., a DNN that integrates attention mechanisms into the standard CNN) by combining human knowledge and Grad-CAM location maps. They had these maps manually edited by human experts and then used them to fine-tune the attention process. Another approach that focuses on global explainability was introduced by *Balayn et al.* [4]. They suggested augmenting the heatmaps generated by explainability methods by incorporating semantic concepts through crowdsourcing annotations. The main advantage of their approach is that the annotations are aggregated to allow the use of different statistical mining techniques to generate global explanations about the model behaviour. Overall, these methods demonstrated the value of incorporating human knowledge in the explainability of ML models, hence foreseeing a promising direction for advancing the field of XAI.

3. Methodology

3.1. Abstract Network Visualizations

State-of-the-art XAI methods provide explanations for image classification predictions, by identifying the most important region of the image that contributed to them. However, while these methods can explain where the entity of the classification was identified, they do not offer a complete understanding of the machine rationale. In order to have a comprehensive understanding of the rationale for an AI decision, we need to examine the entire AI decision-making process. In the case of CNNs for image classification, this process includes multiple stages of feature extraction from input images that happens through multiple layers. For example, considering the classification of a dog's image, one layer might identify the quadruped shape, the subsequent could extract more specific features such as ears or eyes, and then the final layer could identify the dog's head and body and consequently classify the image as "dog". Having an overview of this feature extraction process can significantly enhance the transparency of the classification. For this reason, the main objective of this work is to develop a process to generate local explanations in the form of Abstract Network Visualizations (ANV), providing human-understandable and accurate visualizations of the features extracted by the CNN at each layer. An ANV (as shown in Figure 9) is composed of layers, each consisting of a set of heatmaps, which provide a visual representation of the areas of the input image where important features were identified.

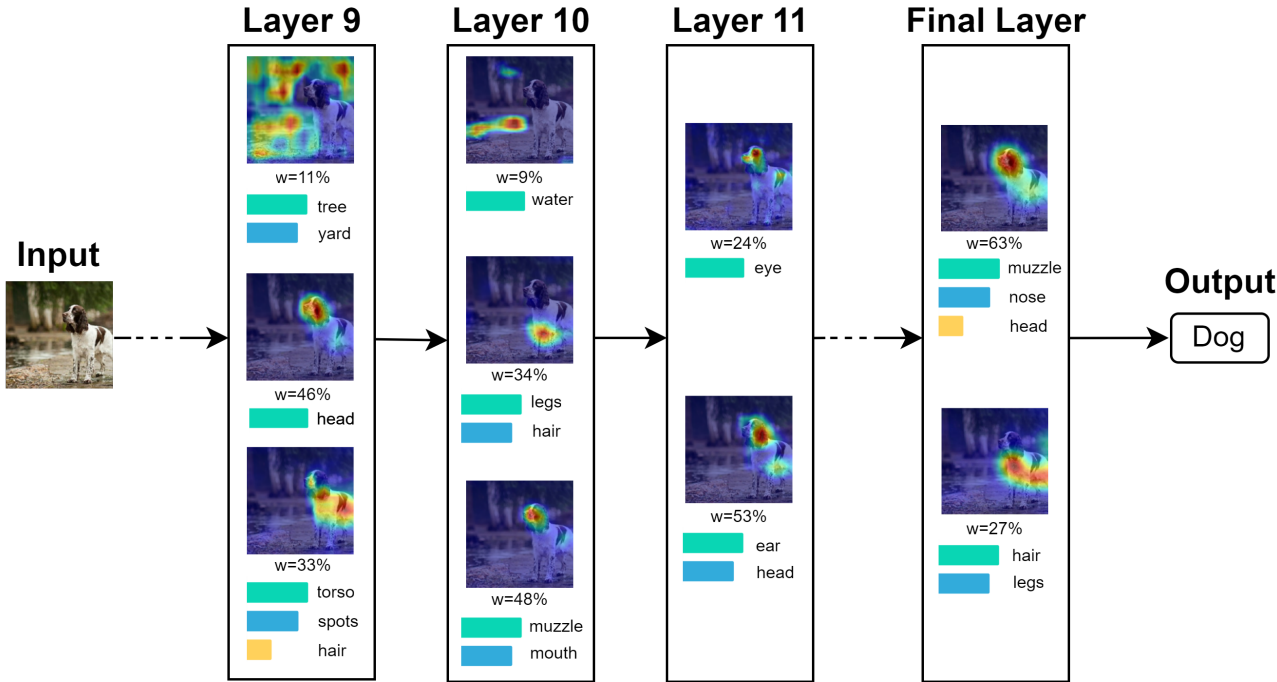


Figure 9: The ANV shows the feature extraction process layer by layer. Each layer contains a variable amount of heatmaps that visually represent the features extracted by the network in that layer. Each heatmap is associated with a weight and a set of labels that contains the human concepts describing these features. The weights represent the contribution of each feature towards the output. They do not sum up to 100% because features with very low weight are excluded.

These heatmaps represent groups of feature maps clustered by similarity (i.e., feature maps focusing on the same region of the image are grouped together). The number of heatmaps is not fixed and can be different for each layer. The final visualization also includes, for each heatmap, the relative importance of its corresponding group of feature maps with respect to the final predicted class. In addition, each heatmap in the ANV is linked to a set of crowdsourced labels that indicate the human concepts it represents. We use a set of labels instead of a single one because sometimes feature maps highlight a combination of multiple features. For instance, a heatmap might highlight a person, the sky, and a plane as this combination leads the network to predict the parachute class, rather than any of these three features individually. Moreover, labels obtained through crowdsourcing may not have equal relevance since one could be mentioned more frequently than another, hence they are ranked and represented in a plot. Additionally, labels provided directly by humans have the unique advantage of being human-understandable, enhancing the interpretability of our explanations.

The ANV can be built considering all layers of the CNN, as well as a selected subgroup of interest. Since shallow layers usually focus on detecting basic shape information (e.g., edges, outlines, corners, etc.), it might be a more efficient approach to focus on deeper layers which should contain more semantic concepts as their receptive fields are bigger (see Section 2.2.2). After deconstructing the feature extraction process into its constituent parts, it is possible to analyze each part individually to identify any potential biases. For example, it might be discovered that the classification of a golf ball is strongly influenced by the importance given to the "grass" feature. However, this approach also allows for the investigation of potential correlations between features extracted by different layers leading to a more comprehensive understanding of how they relate to each other. A significant advantage of using crowdsourced labels to describe the extracted features is the ability to aggregate multiple local explanations, thereby extending the explanation from a local to a global perspective. By analyzing the features extracted by a CNN to recognize a particular class across multiple images, we can develop a global explanation of how the network generally identifies that class. Our explainability method is post-hoc, meaning it doesn't require any modification or re-training of the model. Additionally, ANVs can potentially explain both correct and incorrect classifications, although, for the purposes of this work, we will focus solely on the former. In the next section, we will provide a detailed overview of the process and the essential steps required for generating the presented visualizations.

3.2. Method Overview

Given a CNN trained for image classification and an input image, the following steps must be followed to build the ANV, as outlined in Figure 10:

1. *Feature Maps Analysis*: Feature maps and their relative weights are extracted and clustered. Clusters are merged to generate representative heatmaps for each cluster that we refer to as cluster maps.
2. *Human Knowledge Collection*: In this step, labels are collected through crowdsourcing to enhance the interpretability of previously generated Cluster Maps.
3. *Label Analysis*: The collected labels are processed using data analysis techniques to make them structured and free of errors. Cluster maps with similar labels are also merged.

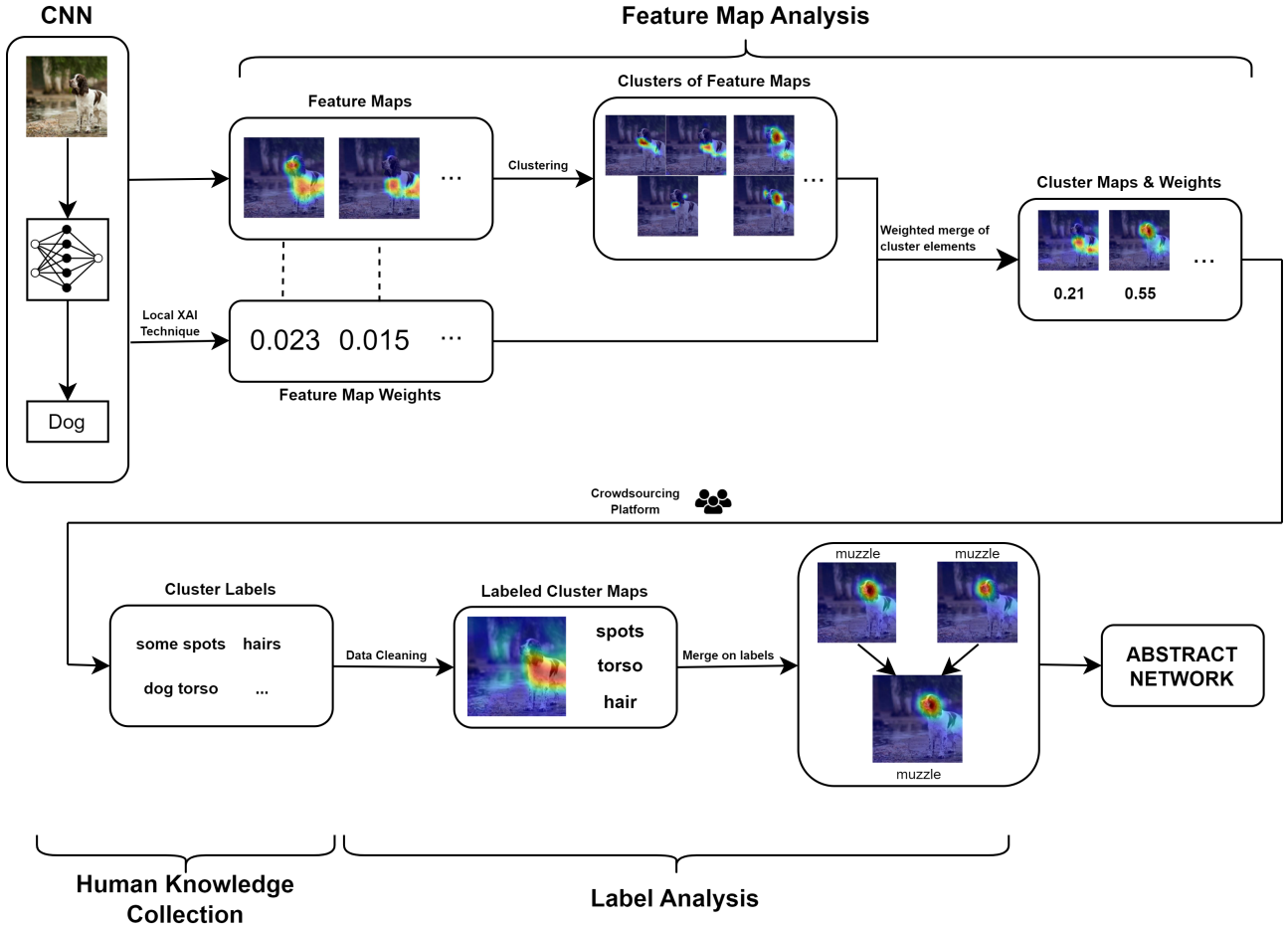


Figure 10: A pipeline depicting the process for building the ANVs. The process involves three main steps. In the first step, feature maps and their weights are extracted from the CNN. These feature maps are clustered to generate representative heatmaps (cluster maps) for each cluster. In the second step, human knowledge is collected to assign labels for the clusters. In the final step, the labels collected from the previous step are cleaned up and cluster maps with the same labels are merged.

3.3. Feature Maps Analysis

The first step of our process is to extract as much meaningful information from executing the CNN on an input image in terms of feature maps and their importance. These feature maps are then clustered and merged to generate a set of cluster maps for each layer.

3.3.1 Feature Maps and Weights

Feature maps are extracted after applying the activation function for each convolutional layer. In case the activation function is not a ReLU, a ReLU is applied to the feature maps to consider positive activations only. Some of these feature maps may become empty and thus are removed. Next, feature maps are associated with their corresponding class-specific weights towards the predicted class, which will be computed using a local explainability method. For this purpose, we used the Grad-CAM algorithm (2.1) as it is a straightforward approach that works with any CNN architecture. We have opted to use the version of Grad-CAM that utilizes Guided ReLU, a modified version that only considers positive gradients to regions of positive activations (i.e., the positive elements of the feature maps). Practically, this means that non-positive gradients and gradients corresponding to non-positive activations are both set to zero before the GAP. This modification has been shown to perform better by *Selvaraju et al.* [40], especially when the objective is not to discriminate between two classes in the same image, which is not our case since in ANVs we consider only the weights towards the predicted class.

We performed unit normalization to enhance the interpretability of these weights. This technique guarantees that the total weight for each layer sums up to one, allowing for their importance to be visualized as percentages

per layer. Feature maps whose weights are equal to or less than zero are removed and a positive weight threshold can optionally be selected to exclude feature maps with low importance, further optimizing the clustering process. The threshold of importance is relative to the number of feature maps a layer produces, hence it is a heuristic that varies based on the layer and the network of interest. The idea behind this thresholding is to reduce the number of feature maps while retaining the vast majority (e.g., 70-90%) of the total weight. At this stage, the number of feature maps per layer can often be in the order of hundreds or more, which can be an overwhelming amount of information for humans to handle. However, they can be clustered and merged together into cluster maps since multiple filters of the same layer typically produce very similar feature maps. In the next section, we'll explore how this can be accomplished.

3.3.2 Pre-processing

Some pre-processing steps are necessary to improve clustering performance. Feature maps are normalized using min-max normalization (i.e., scaling their values to a range between 0 and 1, based on the minimum and maximum values of the feature map). This normalization makes the feature maps comparable in terms of which image regions they are focusing on the most. The subsequent pre-processing step consists in performing a dimensionality reduction algorithm since feature maps are highly susceptible to the curse of dimensionality. For this purpose, we used a combination of two techniques: Principal Component Analysis (PCA) [30] and t-distributed Stochastic Neighbor Embedding (t-SNE) [51]. The reason behind using t-SNE is its ability to visualize data in a low-dimensional space while preserving its local structure, which makes it powerful for identifying clusters in complex datasets such as images. We use PCA to reduce the number of dimensions to a reasonable amount (e.g., 30-50) before applying t-SNE since it requires a high computational effort. The parameters of these algorithms cannot be standardized and must be selected specifically for each model and layer because the number and size of feature maps can vary across different models and layers. While a possible heuristic is outlined in Section 4.2, it is important to optimize these parameters based on the results obtained in the given context.

3.3.3 Clustering

After performing dimensionality reduction, we apply a clustering algorithm for each layer of the network. In our method, we use Agglomerative Clustering, a widely employed type of Hierarchical Clustering. This technique is more suitable for our problem than density-based and centroid-based approaches as the former removes noisy points that could be important in our analysis while the latter performs better under specific assumptions (e.g., spherical distribution of variables) that are unlikely to always be true for our data. Furthermore, hierarchical clustering has the advantage that it produces many clustering results for an incremental number of clusters allowing for freedom in its choice.

The optimal number of clusters can be different for each layer and depends on the number of features a convolution layer extracts for a specific input image, which can't be known beforehand. Therefore, we used the silhouette score metric (i.e., a measure of cohesion and separation of clusters) to select the optimal number of clusters for a given input image and layer. Generally, the evaluation of this metric involves visual inspection. However, we rely only on the average silhouette score as visual inspection is impractical due to the large number of clustering executions required per image. However, we cannot consider every possible number of clusters since the ultimate objective is to produce explanations that need to be understood by humans. Therefore, the number of clusters per layer should not be overwhelmingly high. Having too many clusters can also be problematic during the labeling phase, as it would require a substantial number of participants. For these reasons, a reasonable range (e.g., 3-8) must be chosen based on model size, availability of resources for crowdsourcing, and human comprehensibility. Regarding the choice of parameters for the clustering algorithm, it follows the same principle discussed at the end of Section 3.3.2 for pre-processing parameters.

3.3.4 Merging Feature Maps

Once the clustering process is complete, each cluster is merged using a weighted average approach. This produces cluster maps representing an entire cluster of feature maps. Each cluster map is assigned a weight value that indicates its significance towards the predicted class. This value is computed by summing the weights of all feature maps belonging to that cluster. More formally, these weights are derived for a given cluster starting from the Grad-CAM formulation (2.2), which can be applied to every layer. Considering n clusters, by grouping the terms $w_j A^j$ that correspond to feature maps belonging to a specific cluster C_i , we can factor out the sum of their weights to obtain a new term w_{C_i} which represents the cluster weight, multiplied by a term A_{C_i} that is the cluster map obtained through the weighted average. The Equation (3.1) shows that when using this method

to compute the clusters' weight, the Grad-CAM location map remains unchanged, hence cluster maps can be seen as condensed feature maps.

$$L_{Grad-CAM} = \sum_k w_k A^k = \sum_{i=1}^n \sum_{j \in C_i} w_j A^j = \sum_{i=1}^n \left(\underbrace{\left(\sum_{j \in C_i} w_j \right)}_{w_{C_i}} \cdot \overbrace{\frac{\sum_{j \in C_i} w_j A^j}{\sum_{j \in C_i} w_j}}^{A_{C_i}} \right) = \sum_{i=1}^n w_{C_i} A_{C_i} \quad (3.1)$$

After obtaining a set of cluster maps for every layer and their corresponding weight, we can optionally choose a threshold to exclude cluster maps with low overall weight, although this decision depends on the resources available for the data collection process. At this point, we are ready to enhance the interpretability of the cluster maps through the employment of human knowledge which will be the topic of the following section.

3.4. Human Knowledge Collection

The goal of this step is to collect labels, representing the human concepts highlighted in each cluster map. In order to ensure these concepts are human-understandable, they are obtained through crowdsourcing, i.e., the practice of obtaining knowledge from a heterogeneous group of people often referred to as "crowd". This enables us to associate visual explanations with meaningful concepts that are familiar and interpretable to humans. Furthermore, collecting labels from multiple people with a variegated background can significantly reduce the interpretation bias addressed at the end of Section 2.2.2.

3.4.1 Masking Cluster Maps

Before designing the crowdsourcing activity, we need to address what precisely participants should see during the labeling of cluster maps. The general approach to obtain an interpretable visualization of a feature map is to generate an overlay of the input image and the feature map after normalization, up-scaling, blurring, and color mapping. We use the same technique for cluster maps, obtaining an overlay such as the one shown in Figure 11a. However, asking participants to label these cluster map overlays has a problem. If we let humans know the subject of the image before labeling, they will most likely lose focus on the highlighted areas. Hence, the labels might not accurately describe the highlighting features. For example, a cluster map highlighting a portion of grass on a soccer field may be labeled as "soccer field" instead of "grass", if the entire image is displayed. Although the label "soccer field" is acceptable, "grass" is more focused and describes only the highlighted area. Hence, we need to hide the non-highlighted portions to prevent such behaviours. This can be achieved by computing a mask (i.e., a binary image) that defines which pixels to show. A masked image is then obtained by overlaying the mask on top of the input image, as shown in Figure 11b.

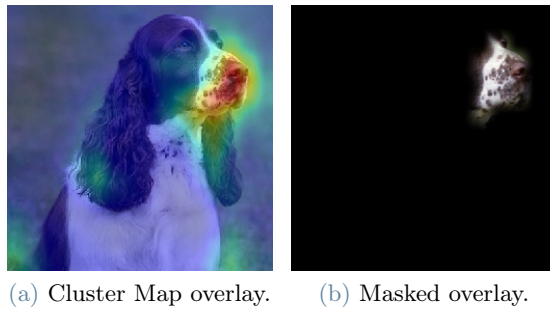


Figure 11: An example of an overlay of a cluster map that focuses on a dog's muzzle, along with its corresponding masked image that reveals only the highlighted area.

By choosing an estimate³ of the percentage of pixels to show, we can filter only the most important pixels of a cluster map by computing the percentile relative to the former estimate and considering only the values of the map greater than or equal to the percentile value. Such a percentage parameter can be adjusted to create

³Note that repeated values may be present, particularly when up-scaling cluster maps significantly smaller than the original image. Consequently, the number of values that are greater than or equal to the percentile might be slightly higher than the chosen percentage. Additionally, blurring can be applied to enhance the visualization quality, which can modify the actual number of pixels displayed.

different masks showing gradually larger portions, which will be useful in the subsequent step. The purpose of the parametrization is to produce a finite number of masks, with a percentile that increases non-linearly, since the information density decreases while moving away from the most highlighted pixels. An example can be seen in Figure 12.

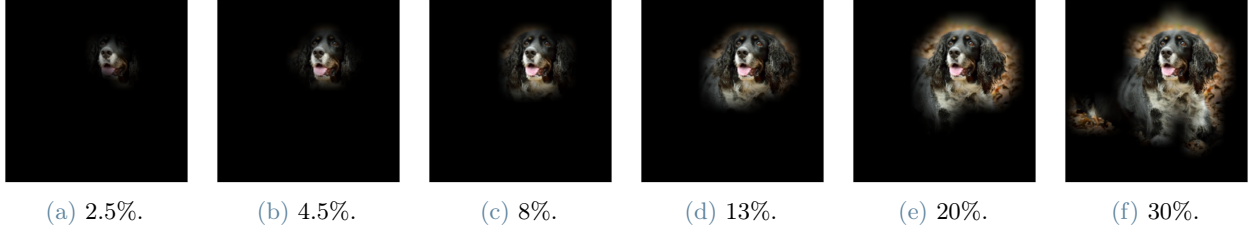


Figure 12: Six masks generated by gradually increasing the pixel percentage parameter.

3.4.2 Gamification

For collecting the labels, we employ a gamified activity since it offers several benefits. In particular, it makes it easier to attract a wide range of participants, including those who may not find a survey interesting. Additionally, according to *Morschheuser et al.* [33] gamification can significantly increase engagement, leading participants to put more effort and thought into their responses, ultimately resulting in higher-quality labels. However, such a choice is not given solely by these benefits. The main reason behind gamification is that we want to make participants behave similarly to the neural network by having them observe and analyze features to guess the correct class. The actual activity consists in playing an online game we designed called *Deep Reveal* (whose process is reported in Figure 13) in which users are presented with the masked image of a cluster map and are required to guess its class and explain their decision. Naturally, the same individual won't play with cluster maps generated from the same input image multiple times.

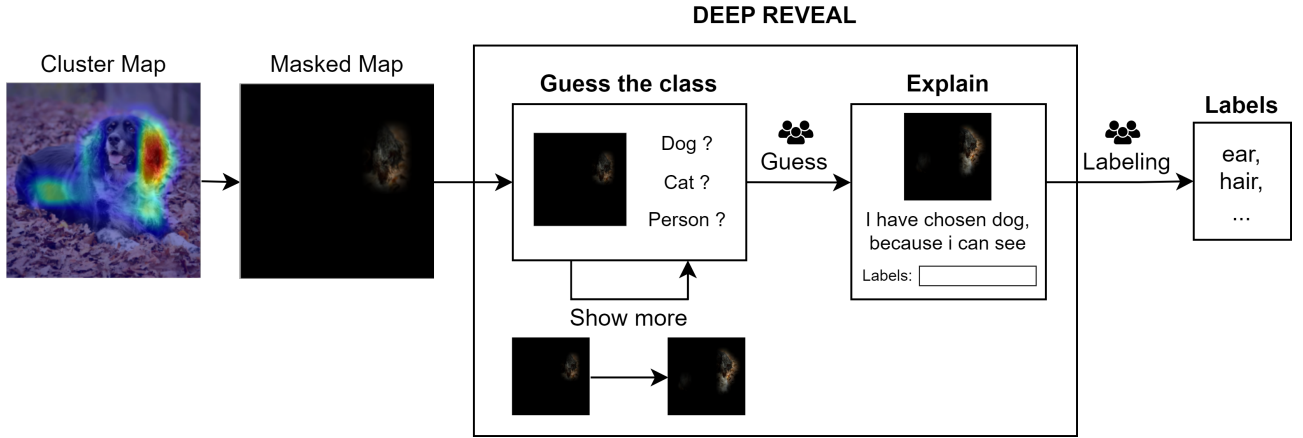


Figure 13: A pipeline describing the label collection process through *Deep Reveal*. The masked version of the cluster map is shown to the user who can try to guess right away or show more of the image. After guessing, users are asked to provide the labels to explain their decision.

Users can choose a class from a set of options, which are a subset of the model's classes chosen at random including the correct one. The number of options should not exceed a reasonable amount (e.g., five to ten) since displaying every possible class is not feasible, given that models might have hundreds of classes. Additionally, using deterministic approaches to select the options, such as considering the prediction confidences, can introduce bias and lead to recognizable patterns for users. For instance, when the model identifies a dog, the class "cat" may have a higher confidence score than other classes. As a result, the options "dog" and "cat" may frequently appear together when the correct class is "dog". Such a pattern may lead the user to choose the option "dog" independently of what he sees.

Similarly to the Peek-a-Boom game described in Section 2.2.3, users of *Deep Reveal* can gradually increase the displayed area, allowing them to get more clues. Such a mechanism is implemented by generating multiple masks as shown in Section 3.4.1. However, a user can increase the displayed area up to five times (i.e., generating five additional masks), after which the game provides them with the option to resign. Analyzing lost games and

resignations is also important since it helps in determining whether the cluster map is focusing on the subject or not.

Once the user selects an option, the game prompts them to specify which features they recognized that helped them guess. These inputs are then used as labels for the cluster maps. The reason we ask participants to guess the image before labeling is to focus their attention on the features that are truly discriminative. Additionally, the label generation should happen openly to avoid introducing any biases, meaning players should not be limited to a finite number of characteristics to choose from. This is because users may infer something that they would not have otherwise seen if provided with a range of options. We also include a scoring system (e.g., the more users reveal of the image the fewer points they earn) and a leaderboard to further increase engagement and competition. Additionally, some masked maps are purposefully very easy, allowing the exclusion of data collected from untrustworthy users. Further details about *Deep Reveal* design and implementation are presented in Section 4.4.

3.5. Label Analysis

After collecting sufficient labels for each cluster map, we proceed with the label analysis step. Since we let users insert labels openly to avoid any bias, the data could contain errors and impurities (e.g., long phrases, synonyms, stop-words, misspelt words, etc.). For this reason, it is required to perform data cleaning on the collected labels. More specifically, labels consisting of multiple words are split into different labels, and stop-words are discarded. Then, we manually map labels referring to the same feature to a single word (e.g., "column", "pillar" and "pilaster" all become "pillar") to handle synonyms and misspellings. Alternatively, it is possible to apply Natural Language Processing (NLP) techniques to automate the process. However, due to the lack of context and the fact that humans use words with different meanings to refer to the same visual entity (e.g., logo and brand, grass and field), these techniques may not be precise enough for our problem. However, further research could address the applicability of such techniques in a context similar to ours.

The next step is to evaluate each label by assigning them a score that allows us to emphasize the most relevant ones within each cluster. This score takes into account the label frequency as well as the percentage of the image revealed to the humans who assigned that label. The method for computing the score is an implementation-dependent heuristic in which we give more importance to the frequency. Our formulation (Equation 3.2) shows how to compute the $score(C, l)$ for label l in cluster C . The frequency of l in C is multiplied by a term that accounts for the average number of "show more" used by users when inserting l . The value of this term ranges from 1 when l is observed using only the smallest masks (i.e., no hints used) to 0.5 when l is observed using only the largest mask (i.e., all hints used). For intermediate values, the term decreases linearly, taking into account the average number of hints used while submitting l . Here, $n_masks - 1$ denotes the number of hints available when playing. The score of labels coming only from users who guessed wrongly counts only as one-fourth of a normal score as they have a higher probability of being imprecise.

$$score(C, l) = frequency(C, l) \times \left(1 - \frac{avg_show_more(C, l)}{2 \times (n_masks - 1)} \right) \quad (3.2)$$

Assigning a score to each label allows us to identify the labels that best describe their respective cluster maps. However, it is possible that certain cluster maps may represent the same feature and, therefore, be labeled in a similar manner. This can happen due to imperfections in the clustering process, or because the same feature is present in different regions of the image. For example, in an image of a parachute, there could be multiple cluster maps looking at different regions of the sky, all labeled as "sky". Similarly, a fish with multiple fins could have multiple cluster maps labeled as "fin", looking at different regions of the image. Since cluster maps are meant to represent the different features extracted at each layer, the final step before constructing the ANV is to merge clusters with the same most relevant labels. More specifically, we merged the clusters that shared at least one among the labels with the maximum score, which could be more than one in case of a score draw. Furthermore, if two clusters had one best label in common, and one of them shared one of its best labels with a third cluster, the three are merged (see Algorithm 1).

Algorithm 1 Cluster merge on maximum scoring labels

```
for all  $l \in \text{layers}$  do
  repeat
    for all  $c_i, c_j \in l.\text{clusters}, i \neq j$  do
      if  $c_i.\text{best\_labels} \cap c_j.\text{best\_labels} \neq \emptyset$  then
        create new cluster having  $\text{best\_labels} = c_i.\text{best\_labels} \cup c_j.\text{best\_labels}$ 
        delete  $c_i$  and  $c_j$  from  $l.\text{clusters}$ 
        add new cluster to  $l.\text{clusters}$ 
      break
    end if
  end for
until  $l.\text{clusters}$  changes
end for
```

The cluster maps are merged through a weighted average and the final weight will be the sum of the weights of the merged maps, for the reasons discussed in Section 3.3.4. The labels of the merged clusters are also combined through a weighted average of their score. Finally, the ANV of an image can be generated by organizing its cluster maps, together with their weights and labels, into one column for each layer.

4. Implementation

In this section, we provide an overview of the systems we implemented to validate our methodology. Specifically, we will describe the CNN model, its training and the libraries and parameters utilized for feature map extraction, pre-processing, and clustering. Additionally, we provide details about the design and implementation of *Deep Reveal*. In terms of the data analysis steps, there is not much to discuss other than the usage of the Pandas library to manipulate the data extracted from the database in CSV files.

4.1. CNN Setup

Having a reference CNN model is essential before implementing the necessary systems for generating ANVs. We implemented the model using Keras, a high-level deep learning API that runs on top of TensorFlow which is a well-known software library for creating and training ML models in Python. Keras provides many pre-built models, from which we imported the VGG-16 model pre-trained with ImageNet (i.e., a large-scale image database). VGG-16 is a standard CNN architecture that consists of 13 convolutional layers, 5 max-pooling layers and 3 dense (i.e., fully connected) layers. TensorFlow enables the modification of pre-built models by changing their parameters, such as the input shape and output classes, to adapt them to different classification problems. In our case, we kept the convolutional and pooling layers of VGG-16, initialized with their ImageNet weights, while changing the input shape, dense layers, and output classes, as shown in Figure 14.

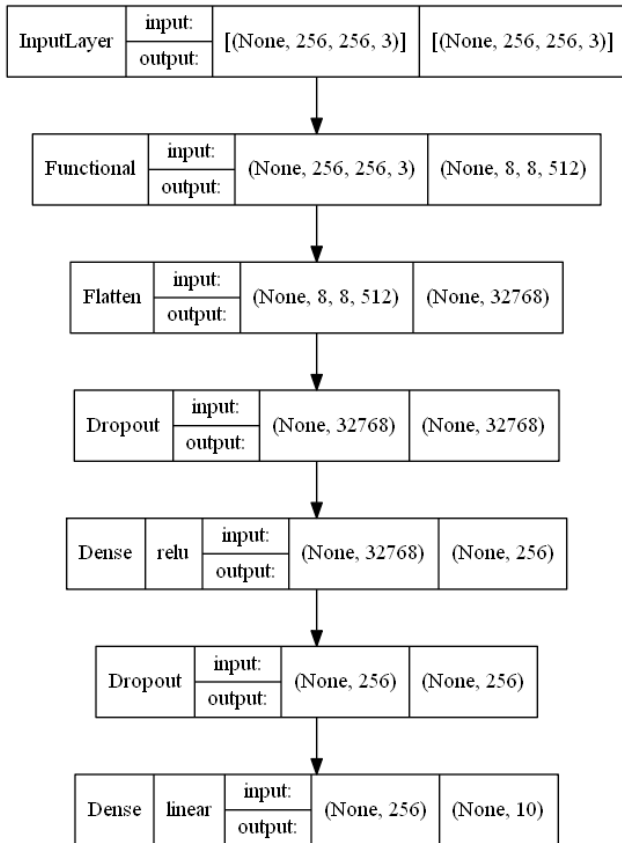


Figure 14: Visual representation of the considered CNN. The functional layer is a subnet consisting of the convolutional and pooling layers of VGG-16 for images of size 256x256.

As our dataset we used Imagenette⁴, a small subset of ImageNet consisting of ten classes and 13394 images. Its classes include Cassette Player, Chainsaw, Church, English Springer, French Horn, Garbage Truck, Gas Pump, Golf Ball, Parachute, and Tench. We applied various techniques such as image augmentation, early stopping, dropout, transfer learning, and fine-tuning to prevent excessive overfitting during training. Initially, we performed two training iterations (both initialized at 200 epochs and early stopping with patience set to 10), starting with transfer learning, allowing the network to train only its fully connected layers with a learning rate of 1e-3. Then we reduced the learning rate to 1e-5 and enabled training for the last three convolutional layers to fine-tune the model. The final accuracy for both training and validation sets were 96.65% and 97.45%, respectively. A detailed view of the training history is provided in Figure 15. Moreover, we did not conduct extensive testing as the network was not intended to be used as a classifier in real-world scenarios.

4.2. Feature Maps Extraction and Clustering

In order to extract the feature maps, we built a new Keras model for each convolutional layer of our trained model. These new models have the same input as the original model and the corresponding convolutional layer output (i.e., the feature maps) as output. The subsequent step is to calculate the weights for each feature map. Hence, we utilized TensorFlow’s automatic differentiation technique (i.e., *GradientTape*) to calculate the gradients required for implementing the Grad-CAM formula (2.1). After obtaining the weights, they are normalized using unit normalization while feature maps are normalized using max-min normalization. Furthermore, as stated in Section 3.3, we defined a threshold for considering only feature maps whose weights are above it. The value of such threshold varies according to the number of feature maps composing a layer, hence it depends on the model architecture. In our case, considering VGG-16 and its convolutional layers, we defined the thresholds shown in Table 1.

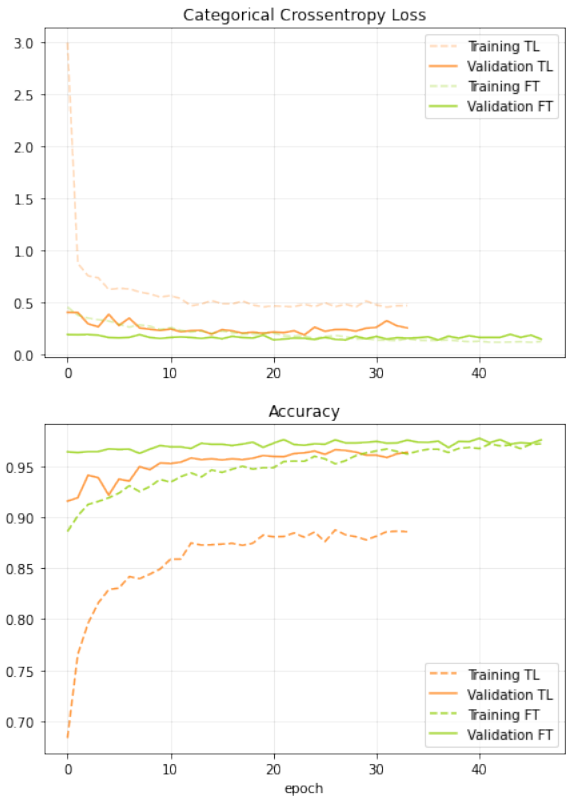


Figure 15: Plots representing the history of loss and accuracy obtained both during the transfer learning (TL) and fine-tuning (FT) phases in relation to the epochs.

⁴<https://github.com/fastai/imagenette>

Number of feature maps	64	128	256	512
Threshold	1.4	0.7	0.35	0.175

Table 1: A table that associates the number of feature maps in a layer and the weight threshold under which feature maps were excluded.

Regarding the clustering phase, we utilized the Scikit-learn library to implement the entire procedure. To begin, we selected feature maps layer by layer and applied dimensionality reduction, as stated in Section 3.3.2. For each layer, we transformed the normalized feature maps into one-dimensional vectors and then conducted PCA with a number of components equal to the minimum of either 50 or the default value defined by the library (i.e., the minimum between the number of feature maps and the number of pixels). Finally, we applied TSNE using its default parameters (i.e., 2 components with a perplexity of 30 and 1000 iterations). After completing this pre-processing step, as described in Section 3.3.3, we utilized Agglomerative Clustering with euclidean distances and Ward linkage, linking the two clusters that minimize the increase in the sum of squares distances between clusters. We tested the clustering algorithm with a number of clusters ranging from 3 to 8 and selected the solution that maximized the average silhouette score. We tested the parameters of the applied algorithms using images from the network dataset and obtained better results with the selected parameters. Next, we computed the cluster weight and cluster maps using the methods discussed in Section 3.3.4. Additionally, as a post-processing step, we applied a threshold to exclude clusters with low weights. This was chosen due to limited resources for the labeling phase, as we aimed to have more labels for the most important clusters rather than fewer labels for all the clusters. The threshold (Equation 4.1) is computed as the maximum value between one-third of the weight of the most significant cluster and half of the average weight of all clusters in that layer. This formulation is an heuristic designed specifically for our network and resources, hence similar approaches may also be applied.

$$cluster_threshold(layer) = \max(\max(cluster_weights(layer))/3, \text{mean}(cluster_weights(layer))/2) \quad (4.1)$$

4.3. Masking Cluster Maps

The masking procedure has been implemented with the Sci-kit and Numpy libraries, according to Algorithm 2. Firstly, the cluster map is up-scaled to be of the same shape of the image, then we compute the p -th percentile of the heatmap, where p is a percentage of choice. Successively the mask is computed as a matrix having entries equal to one only where the values of the heatmap are greater than or equal to the percentile computed previously. Before applying the mask to the image (i.e., showing only the pixels where the corresponding value of the mask is equal to one), we apply Gaussian blurring to the former to have a smoother visualization. For this task, we used the Scikit-image implementation of the Gaussian filter with parameters depending on the ratio between the shape of the image and the shape of the image, as shown in Table 2. Finally, we overlay the blurred mask over the image to get the masked cluster map. As specified in Section 3.4.2, the gamification process requires six masked images, each revealing an increasingly larger portion, hence it is required to select six values for the parameter p of the algorithm. We tested a wide variety of images with different values, resulting in the six values displayed in Table 3.

Input image shape	512x512				
Heatmap shape	256x256	128x128	64x64	32x32	16x16
σ parameter	(3,3)	(6,6)	(12,12)	(16,16)	(24,24)
$truncate$ parameter	1.5				

Table 2: A table representing the values chosen for the Gaussian filter implementation in function of the shape of the input image and the cluster heatmap.

Hints used	0	1	2	3	4	5
p parameter	2%	4.5%	8%	13%	20%	30%

Table 3: A list of the values chosen for the parameter of the masking algorithm when used for *Deep Reveal*.

Algorithm 2 Apply a mask to an image given a feature map and a percentile

```
function MASKIMAGE(image, heatmap, p)
  heatmap_up  $\leftarrow$  UPSCALEHEATMAP(heatmap, image.shape)            $\triangleright$  enlarges the heatmap
  percentile  $\leftarrow$  COMPUTEPERCENTILE(heatmap_up, p)              $\triangleright$  p-th percentile of the heatmap
  for pixel in heatmap_up do                                        $\triangleright$  map computation
    pixel  $\leftarrow$  1 if pixel  $\geq$  percentile else 0
  end for
  size_ratio  $\leftarrow$  image.shape/heatmap.shape
  blur_sigma  $\leftarrow$  COMPUTEBLURSIGMA(size_ratio)                  $\triangleright$  the higher ratio, the higher sigma
  mask  $\leftarrow$  APPLYGAUSSIANSFILTER(heatmap_up, blur_sigma, truncate = 1.5)
  masked_image  $\leftarrow$  APPLYMASK(image, mask)                    $\triangleright$  overlay the mask on the image
  return masked_image
end function
```

4.4. Deep Reveal

As discussed in Section 3.4, the objective of *Deep Reveal* is to enable users to participate in a gamified image-guessing activity and provide descriptive labels for the part of the images they see. In this section, we discuss the requirements of *Deep Reveal* and provide further details on its implementation.

4.4.1 Requirements and Design

Login. A login system is implemented to ensure a user does not encounter the same image more than once. To register in the system, a user only needs to provide a username, an email address (required for password reset), and a password. However, if a user does not want to register, the option to play as a guest is also available. For registered users, the system ensures that they don't encounter the same image twice. The same applies for guest users, but only within a session.

Home Page and Leaderboard. After logging in, the user is directed to the Home Page where he can access his game statistics such as his score, number of games played, win rate, and win streak. From there, the user can logout, read the game instructions, start a new game, or view the leaderboard. The leaderboard shows the top 50 users with the highest scores, along with their game statistics.

Gameplay. During the game, the players start with 300 points and are shown a partially obscured image (i.e., the masked heatmap) where only a small portion is visible. By using the "show more" button, they can increase the visible region up to five times, with each increase incurring a cost of 50 points. The players are presented with six options to choose from (i.e., five randomly chosen classes plus the right one). A win streak bonus awarded for consecutive victories is also displayed on the game screen. Once the players select an option, they are prompted to enter the characteristics that drove their choice, for which they are awarded 25 bonus points. Once they confirm their selection and the associated characteristics, the content of the image, the correct class, and the points earned are displayed. Then, the players can choose to play another game or return to the home page. Furthermore, they have the option to return to the Home Page or close the web page at any time and resume the game later.

Other requirements. Each player's third match always presents the same image which is a straightforward and easily recognizable picture of a dog (shown in Figure 17). Whether a player guesses correctly or not helps to determine which participants are more trustworthy in providing accurate labels. Additionally, the application balances the number of games per cluster map to ensure that no clusters are left with an excessive amount of labels while others are left with very few.

The interaction flow diagram illustrated in Figure 16 presents a visual representation of *Deep Reveal* application design.

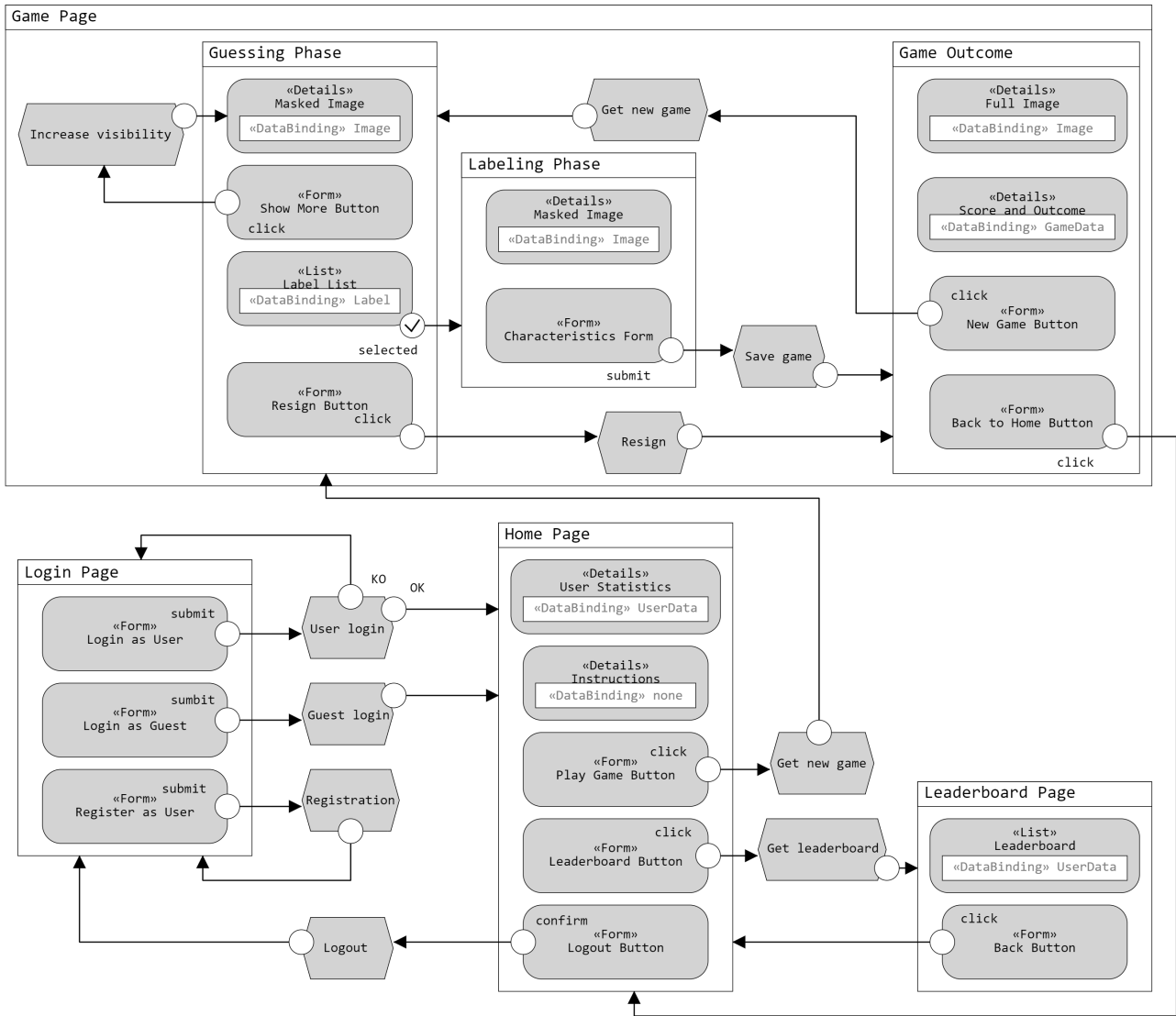


Figure 16: An IFML diagram that illustrates the interaction flow of the *Deep Reveal* application between its four main pages: Login, Home, Leaderboard, and Game. The Game Page is slightly different depending on the phase of the game. The game begins with the guessing phase, followed by the labeling phase, and finally, the outcome phase which displays the game results in terms of points earned and whether the player won or lost.

4.4.2 Implementation details

Deep Reveal was implemented as a web application to facilitate the label collection process as it is accessible from any device. The application’s backend was implemented in Python to ease the integration with previous steps. Specifically, we utilized Django REST framework to generate the application server APIs, and a PostgreSQL database to store the collected data. Python was chosen also to reuse the masked images generation algorithm (2) at run-time, which significantly reduces the required database size compared to generating all masked images beforehand.

On the client side, *Deep Reveal* was implemented using Flutter Web, a highly portable platform that also allows deployment as a mobile app or desktop app. Flutter Web is also ideal for creating game applications that require simple animations. Finally, the web application was deployed using Nginx, an open-source software for web serving, with uWSGI as a web server gateway interface. Figure 17 displays a screenshot of the Game Page during the labeling phase.

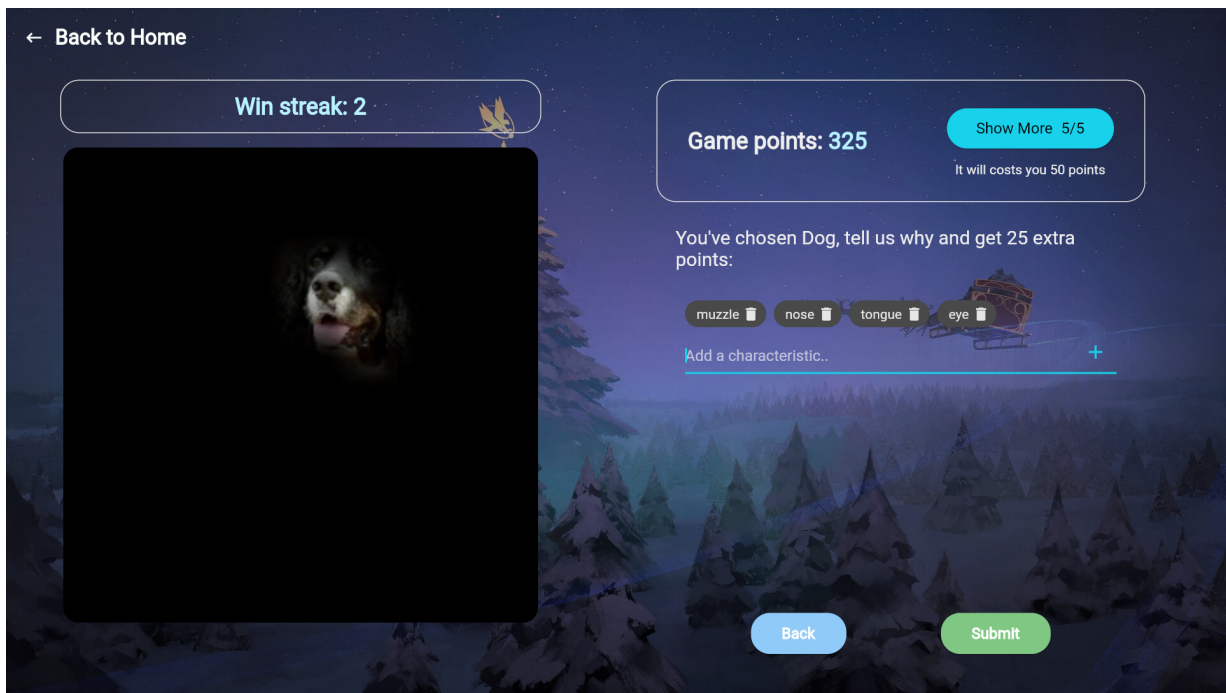


Figure 17: A screenshot showcasing the labeling phase of the game. The user is prompted to insert the characteristics he saw in the form of tags which can also consist of multiple words.

5. Experiments and Discussion

In this section, we describe the experiments we conducted to validate our methodology and discuss the final results obtained.

5.1. Experiment Setup

In this section, we report the setup of every phase of the experiment and the respective intermediate results obtained.

5.1.1 Number of images and layers. Considering the CNN presented in Section 4.1, we conducted the experiment using 5 images per class that were correctly classified by our model. Since the network classifies 10 classes, we explained a total of 50 predictions. Furthermore, knowing that we had limited resources for crowdsourcing, we decided to study only the last 9 convolutional layers of the network. The reason we focused on the deeper ones is that, as already mentioned in Section 3.1, the initial layers primarily extract basic shape information such as edges and outlines, so they focus on less semantic local features that are less associated with semantic concepts.

5.1.2 Feature Maps Analysis. Once the images were chosen, the first step was to extract the feature maps and their respective weights for each image. We then proceeded with normalization and weight thresholding. After pre-processing the feature maps using PCA and t-SNE, we performed clustering with a range of 3 to 8 clusters and selected the result with the highest silhouette score. Finally, we performed cluster thresholding using the formula described in (4.1). The final clustering results are presented in Table 4. The proportion of removed feature maps was significantly higher in deeper layers, while the total weight left over was relatively constant (refer to Figure 18). This suggests that the weights became more concentrated in deeper layers, allowing us to condensate more relevant information in fewer clusters. Finally, we merged the feature maps of each cluster, resulting in a total of 1954 cluster maps to be labeled.

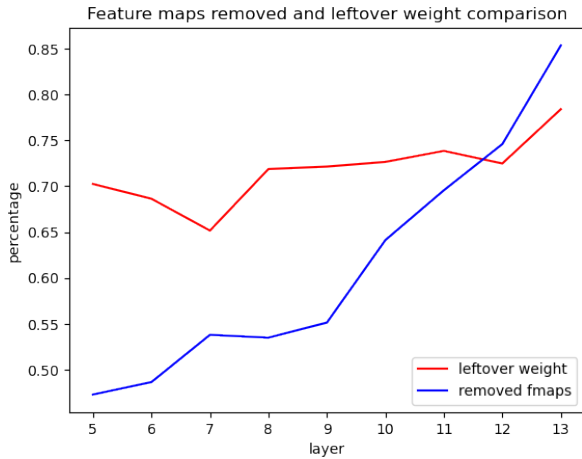


Figure 18: The graph illustrates how the percentage of removed feature maps increases as we move from shallow to deep layers. On the other hand, the total weight of the remaining feature maps remains relatively constant, showing a slight upward trend.

layer	vgg16 fmaps	remaining fmaps	leftover weight	cluster count
5	256	134.94	70.24%	4.28
6	256	131.46	68.63%	4.36
7	256	118.28	65.16%	4.78
8	512	238.14	71.87%	4.24
9	512	229.72	72.13%	4.76
10	512	183.68	72.65%	5.20
11	512	155.88	73.85%	4.58
12	512	130.10	72.48%	3.60
13	512	75.04	78.39%	3.32

Table 4: The table presents the results of the clustering phase for each layer. The figures for remaining feature maps, leftover weight, and cluster count are averages per image and are considered after the cluster thresholding. The leftover weight indicates the combined percentage of importance of the remaining feature maps.

5.1.3 *Human Knowledge Collection.* For the labeling phase, we deployed *Deep Reveal* web application and shared it with 210 participants. Furthermore, we asked them to fill out a usability and workload questionnaire (which can be found in Appendix D) at the end of the experiment to validate the design and implementation of *Deep Reveal* and find possible issues. The usability questions were based on the System Usability Scale (SUS) [7], and the workload questions were inspired by the NASA-TLX [18] method. In the questionnaire, we also added the option to give additional feedback about the app’s user experience. The results of the questionnaire are discussed in Section 5.2.3. We allowed participants to insert labels both in Italian and English to collect more labels, at the cost of having to perform a translation step during data analysis. At the end of this phase, we collected 9968 raw labels evenly distributed among the cluster maps. Only one user guessed wrongly on the test image and his data was discarded. A more comprehensive view of the results of the crowdsourcing activity is provided in Table 5.

Games	Wins	Losses	Resigns	Total raw labels	Raw labels per map	Hints per game
7948	7150	688	110	9968	5.096	1.6

Table 5: The table presents the results of the *Deep Reveal* activity. We collected approximately 5 raw labels per cluster map. Users guessed correctly in the vast majority of the games played and used an average of 1.6 hints per game.

5.1.4 *Label Analysis.* After splitting the raw labels into single words and removing stop-words, a translation from Italian to English was performed and manually validated. This is necessary to avoid losing the contextual meaning of certain labels. For example, the Italian word "Esso" is automatically translated to "it", but in the context of a gas pump, it refers to a company name. The mapping of contextual synonyms with the same word was also done manually for the same reasons. The processes of validating translations and mapping synonyms were done simultaneously providing a unique translation and mapping to every unique word. During this process, irrelevant labels (e.g., seems, like, see) were discarded. The result of these steps is a set of single-word polished labels to which we assigned a score computed using the Formula 3.2 described in Section 3.5. Finally, we merged cluster maps according to their maximum-scoring labels (Algorithm 1) and obtain the results presented in Table 6, which will be used to generate the visualizations presented in the next section.

Total labels	Unique labels	Labels per map (pre-merging)	Cluster maps (post-merging)	Weight per map (post-merging)
12082	430	6.18	1192	27.07%

Table 6: The table shows the results of the label analysis step, including the number of processed labels obtained and the cluster maps resulting from merging the labels. It’s important to note that the labels per map are non-unique and represent an average value before merging. The weight per cluster map is also an average but it is considered after merging.

5.2. Results and Discussion

We conclude by presenting and discussing the resulting ANVs and their validity as a local explanation method for our CNN. Subsequently, we explore the possibility of further aggregating the labeled visualizations to obtain global explanations for the model. The last subsection is a short discussion about the *Deep Reveal* questionnaire results.

5.2.1 Abstract Network Visualizations: Case studies

Based on the structure of the ANV outlined in Section 3.1, we organized the heatmaps for each cluster map into columns, displaying their respective weight and highest-scoring label (example in Figure 19). Moreover, each feature has a detailed visualization (example in Figure 20) which includes a plot of all labels, their respective score, the number of feature maps comprising the cluster, and additional game-related information such as the masked image, wins, losses and resigs. In this section, we present and discuss three case studies, while additional examples are available in Appendix B. All detailed visualizations for the 50 images are accessible online⁵.

The first case we discuss is the ANV for an image of a chainsaw, which is illustrated in Figure 19. It is evident that the prediction was primarily determined by the presence of a motor and a sawchain, with the former being the most important. As the layers get deeper, the focus on the motor and sawchain regions increases, which is a reasonable behaviour since these features are class-discriminative. On the other hand, non-discriminative features like "tree", "green" and "grass" acquire a certain level of relevance in more shallow layers, although their significance decreases as we go deeper, to the point where they disappear entirely. Moreover, some users recognized the motor through its orange color, a classical feature of a chainsaw motor, and it is possible that the network has learnt to recognize this pattern as well. In fact, when a fully orange image was introduced into the network, the predicted class was "Chainsaw" (although with only a 30% confidence), suggesting that analyzing human rationales can provide insights into the machine's behaviour. However, we acknowledge that these statements are conjectures that require further validation, which may be the subject of future research.

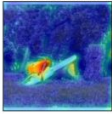
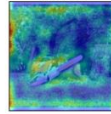
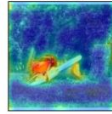
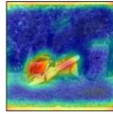
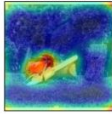
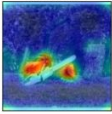
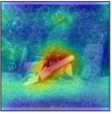
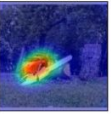
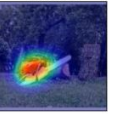
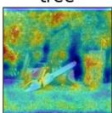
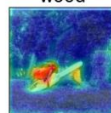
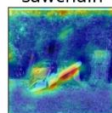
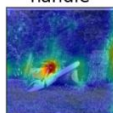
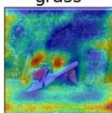
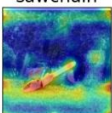
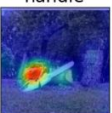




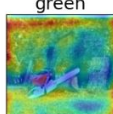
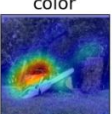
LAYER 5	LAYER 6	LAYER 7	LAYER 8	LAYER 9	LAYER 10	LAYER 11	LAYER 12	LAYER 13
32.9% sawchain 	31.71% case 	59.36% wood 	36.22% tool 	61.13% handle 	54.11% handle 	40.83% sawchain 	46.74% motor 	45.12% motor 
22.86% tree 	26.29% wood 	12.85% sawchain 	20.56% handle 	13.06% grass 	24.81% sawchain 	18.3% handle 	22.23% sawchain 	35.88% sawchain 
10.83% handle 	17.48% sawchain 		20.51% green 			14.45% color 		

Figure 19: This is an ANV for an image of a chainsaw, showcasing the labeled features extracted by the network from the last 9 convolutional layers. The weight of each feature is presented as a percentage per layer. The sum of the weights for each layer is less than 100% since some less important feature maps and clusters were removed during the process. The most important features for the prediction were the motor and the sawchain. However, other elements such as the presence of vegetation in the background, a log, and the handle also contributed to the correct classification.

Although the visualization offers a detailed explanation for the prediction, it is not flawless. For instance, in the first cluster of layer 6, it is unclear what the heatmap is highlighting, and the label "case" does not help in

⁵<https://github.com/antonio-dee/abstract-network-visualizations>

describing it. Interestingly the weight of the feature is significantly high, meaning that intermediate layers can still be quite difficult to interpret. Furthermore, the distinction between the concepts of "tool", "handle" and "motor" may seem ambiguous. However, one cluster map can represent multiple concepts, which is evident in the detailed visualization of the first cluster of Layer 10 (see Figure 20). The cluster was obtained by merging two clusters on the label "handle" and contains additional concepts such as "color", "log", "tool" and "motor". Neither can be deemed incorrect as the network is focusing on every described feature.

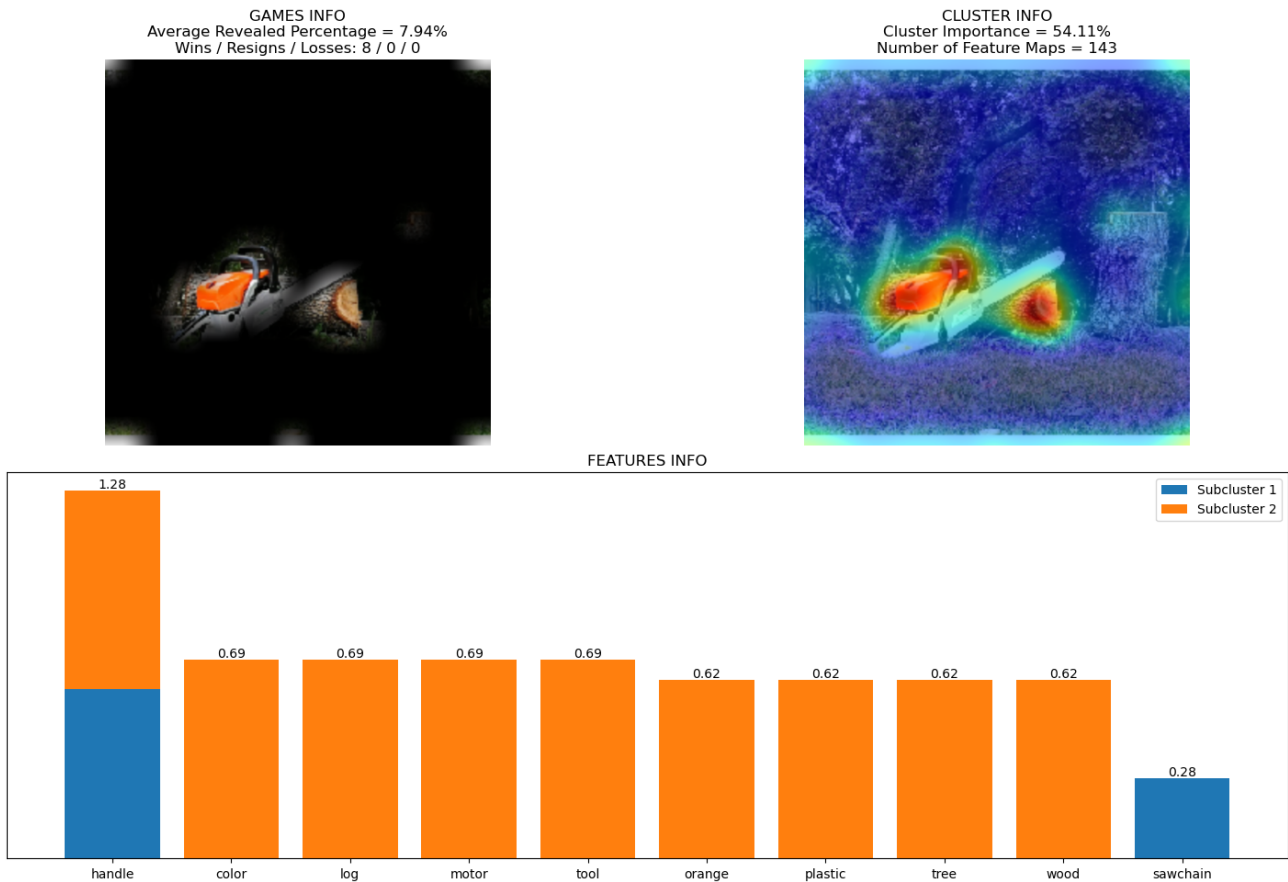


Figure 20: A detailed visualization of a cluster map relative to an image of a chainsaw. A masked image is shown on the top left, which represents the average portion of the image revealed by users when guessing the class and labeling the cluster map. The visualization also provides games information which includes the wins, resigns and losses. More information regarding the cluster is shown on the top left, including the cluster map overlay, its importance and the number of feature maps it represents. The bar plot at the bottom shows all labels describing the cluster map ordered by score. Furthermore, the plot underlines the fact that the cluster map originated from a merge between two and highlights the contributions of each one in terms of the labels' score.

The second ANV we are examining is depicted in Figure 21. The input in this case is an image of a tench, a type of fish. The primary feature used to identify the class was the presence of gills, as well as the back fin and the person visible in the background. While the person and his hand holding the fish appear to contribute to the process of recognizing the tench, unlike the background vegetation in the chainsaw example, this feature maintains a significant weight in the final layer, suggesting a slight overfitting. This is even more evident considering that the "fin" feature appears to be less significant than the person feature, despite both clusters containing the same number of feature maps. The "lake" feature also contributes to the prediction, but it disappears in the final layer. Other features that contributed to the prediction were the scales, eye, and mouth, which may have helped to identify the body and head of the fish. As a rough validation, the network was fed with an image containing only the scales, and it produced the prediction "Tench" with 90% confidence, indicating that the network is capable of associating this feature with the concept of tench.

Before moving on to the next case study, we would like to make one final observation. Sometimes, the heatmaps of the final layer may appear less precise in highlighting certain features. For example, the person in layer 12 and layer 13 of Figure 21. This is due to the gradient saturation problem which, as described in Section 2.2.2, is a significant limitation of gradient-based approaches, including Grad-CAM. Similarly, the weight of 15.39% may be a slight underestimation due to this issue. However, our method can work with any local XAI technique

that can compute a weight for each feature map, including other explainability techniques that may effectively overcome this limitation. Nonetheless, having a visualization across multiple layers could slightly mitigate this problem.

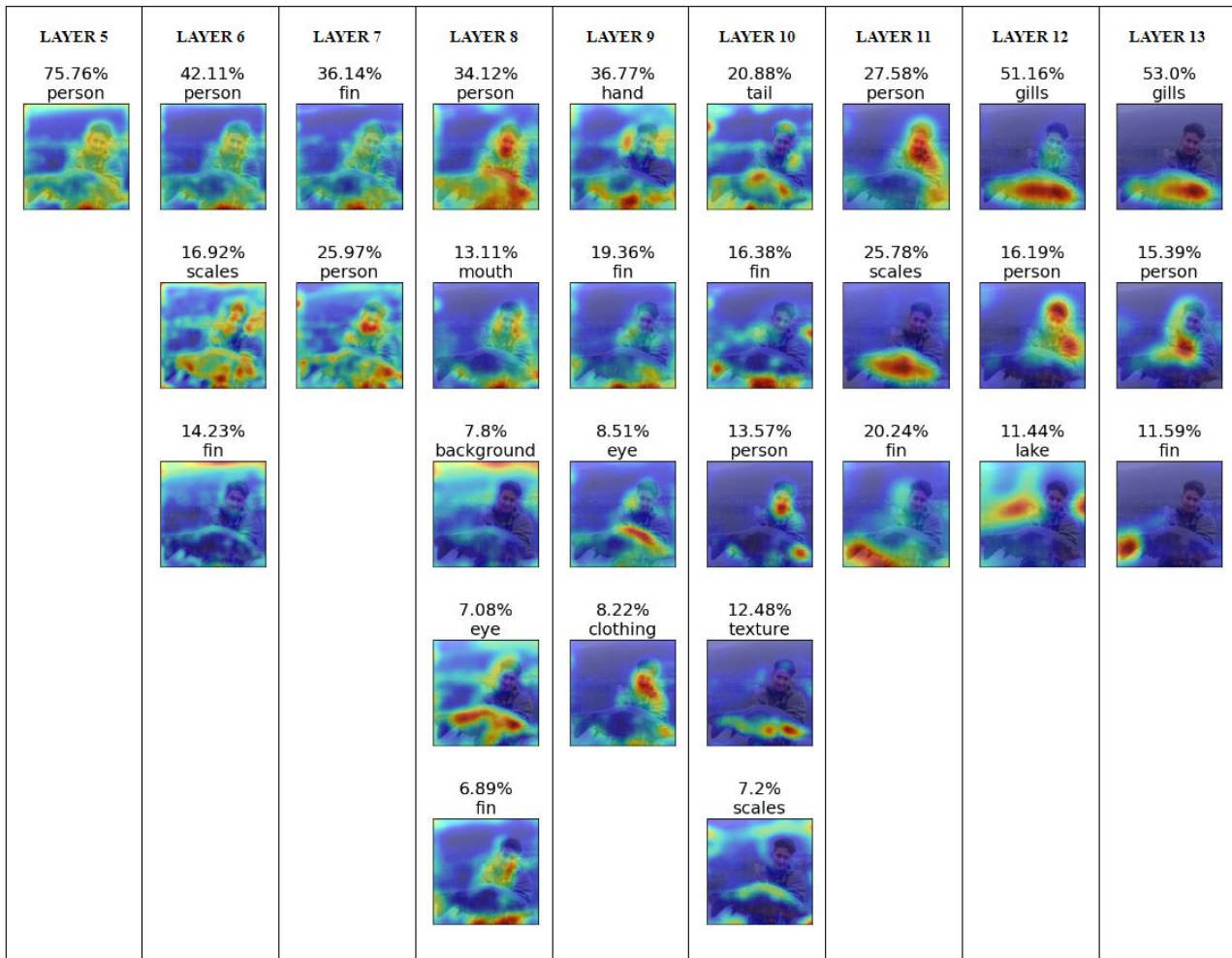


Figure 21: An ANV for an image of a tench. The most significant features for predicting its classification were the gills, fin, and the person in the background. However, the presence of a lake was also identified as a relevant feature with a non-negligible weight. Additionally, the tench classification was also influenced by the presence of its eye, mouth, and scales.

The final case study we present concerns an image of a French horn (see Figure 22). We specifically selected this example because our method performed poorly compared to other images. In this case, there were an excessive number of merging operations, mainly on the label "brass". On one hand, the feature "brass" is likely important for recognizing a French horn, but on the other hand, it was not the sole focus of every cluster map. Similar reasoning can be made for the horn and pipes. From the ANV, we may conclude that the model solely focuses on the horn, pipes, and brass to recognize a French horn, which may be an oversimplification. Furthermore, the cluster maps do not clearly show where these features were identified. This outcome can be explained by the fact that the features of the class of interest can be difficult to describe and require specific knowledge in the domain of musical instruments. If we combine this observation with the fact that a French horn is generally made of brass pipes, the result is that the vast majority of participants focused mainly on these features to recognize the class. Although a few participants included more detailed labels such as "valves" or "pistons", these did not appear frequently enough to show in the main visualization.

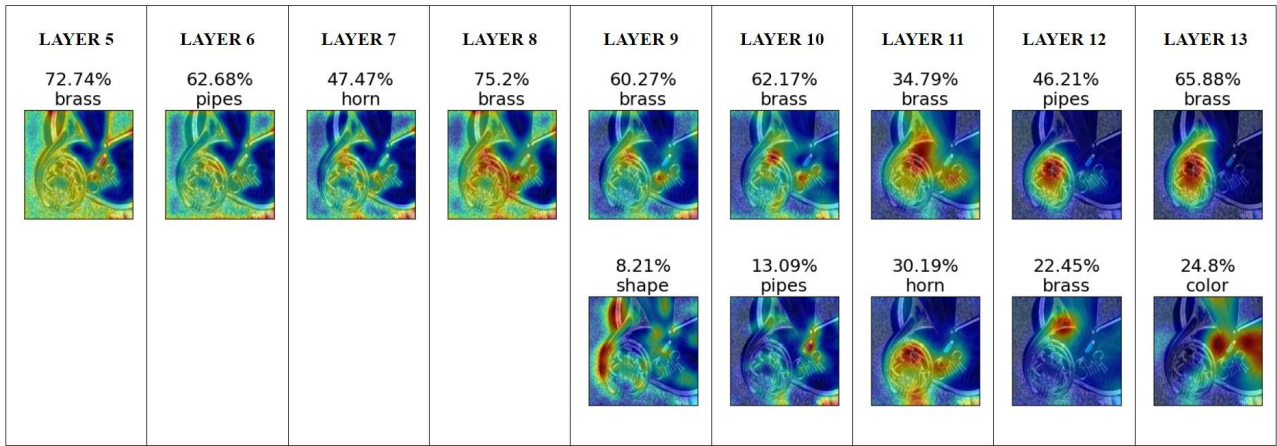


Figure 22: An ANV for an image of a French horn. The three most relevant features for the prediction were the brass, the pipes and the horn. This particular visualization is presented because it is one of the worst we obtained, which can be attributed to the fact that participants lacked domain-specific knowledge and focused on the same features in each cluster map.

From the presented case studies, we observed that ANVs are capable of providing a comprehensive overview of the features that contribute to correctly predicting a class, along with their corresponding importance and insight about the identification process of such features. These visualizations also offer valuable insights into less obvious features, such as the orange color of a chainsaw’s motor or the scales of a tench. However we need to incorporate more sophisticated validation approaches for such features since the fact that humans are able to classify an image using a certain feature suggests that the network may do the same, but it doesn’t necessarily imply that it does. Moreover, we have noticed that the lack of domain knowledge may oversimplify the visualization in some cases, which may lead to preferring domain expert users. While this could be beneficial because of the higher level of detail in the labels that they could provide, it’s important to note that a combination of expert and non-expert users could offer different levels of detail and a broader view. Furthermore, relying solely on experts could result in highly detailed visualizations, which may overestimate the network’s degree of specificity since our CNN works with very diverse classes.

5.2.2 Exploring Visualizations for Global Explanations

As mentioned in earlier sections, labeled cluster maps have the advantage that they allow for representing image features in a tabular format. This makes it easier to aggregate local explanations and create global ones. Starting from the tabular representation, various data mining and statistical analysis techniques can be applied to extract insights and knowledge regarding the overall behaviour of the model. While our work primarily focuses on using ANVs as a local explanation technique, we would like to present a possible approach for aggregating features of multiple images to generate class-specific global explanations. However, it is important to note that our experiment was limited by the relatively small sample size of only five images per class. As a result, the explanations we provide in this section should be considered a preliminary approach to the problem rather than a definitive result. Due to the limited number of images, there is a possibility of bias in our findings, since these images may not be enough to represent an entire class.

We considered only the labels with the maximum score for each cluster since the score of a label represents a proxy for its trustworthiness and ability to describe a cluster map. Furthermore, we construct our global explanations with a hierarchical visualization, grouping layers three by three, thereby extending the concept of the ANV to a global perspective. Features are ordered by their global score, computed as the sum of the scores of a label when it appears as the maximum-scoring label in a cluster. The global score serves as a measure of label quality across multiple images, therefore acting as a generalization of the original scores. For each feature, the visualization also provides a set of cluster map examples in which the label score was the highest. Additionally, each label is assigned a weight computed as the average of the weights of the cluster maps where it appeared as the maximum scoring one.

Figure 23 depicts the global explanation for the class "Chainsaw". In this visualization, we display six labels per group of layers, as the global score decreases significantly beyond this point, which could potentially impact the label reliability⁶. The main objective of the visualization is to provide an overview of the features the network

⁶For instance, a global score of 4.2 indicates that approximately 5 users added that label across the three layers and all images of that class.

generally extracts for the class "Chainsaw" and their average importance. For instance, we can observe that the network primarily focused on the handle and the sawchain in every layer and the presence of wood and gloves significantly contributed to the prediction.

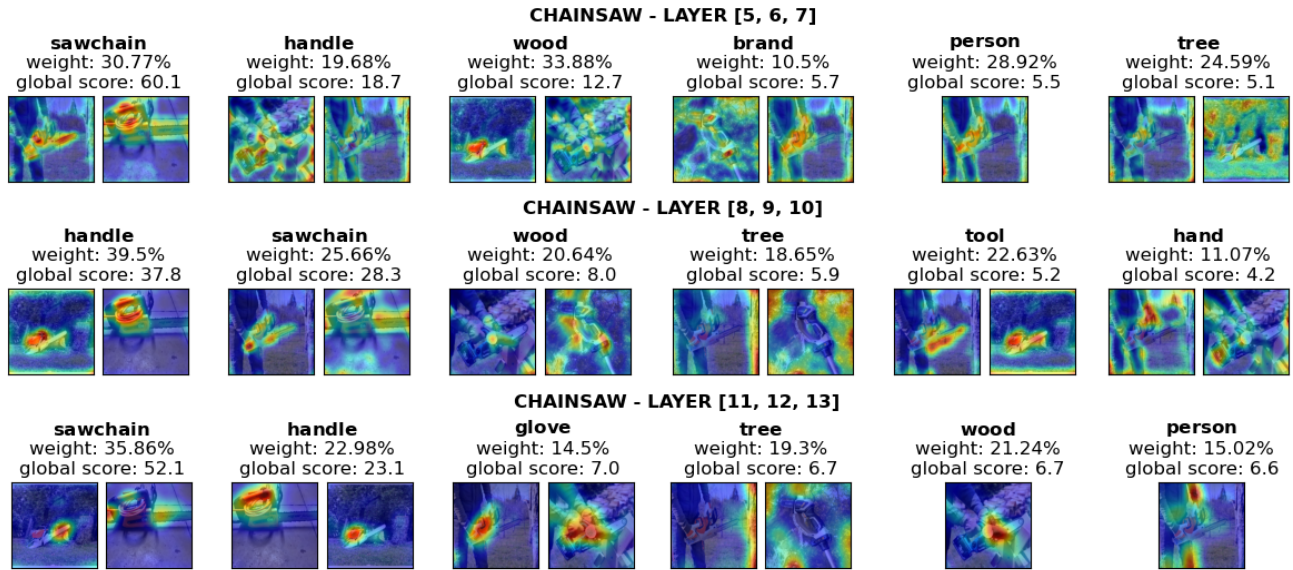


Figure 23: An example of a global explanation for the class "Chainsaw". For each row, we can observe the main features extracted from each group of layers with a weight that signifies the feature's importance in the prediction and a global score that is a measure of the label's trustworthiness.

We present another example of a global explanation for the class "Tench", as shown in Figure 24. As with the local explanations, the presence of the person and his hand is highly important. The person's weight and the hand's weight are consistently high in every group of layers, particularly in shallower layers. This highlights the importance of having a sufficient number of images for global explanations. Without images containing the person, we would not be able to identify this behaviour. As a result, other significant features may be absent from global explanations because they were not included in the five images we selected. Nonetheless, in the last layers, the fish's gills and eyes are still the predominant features in terms of average weight.

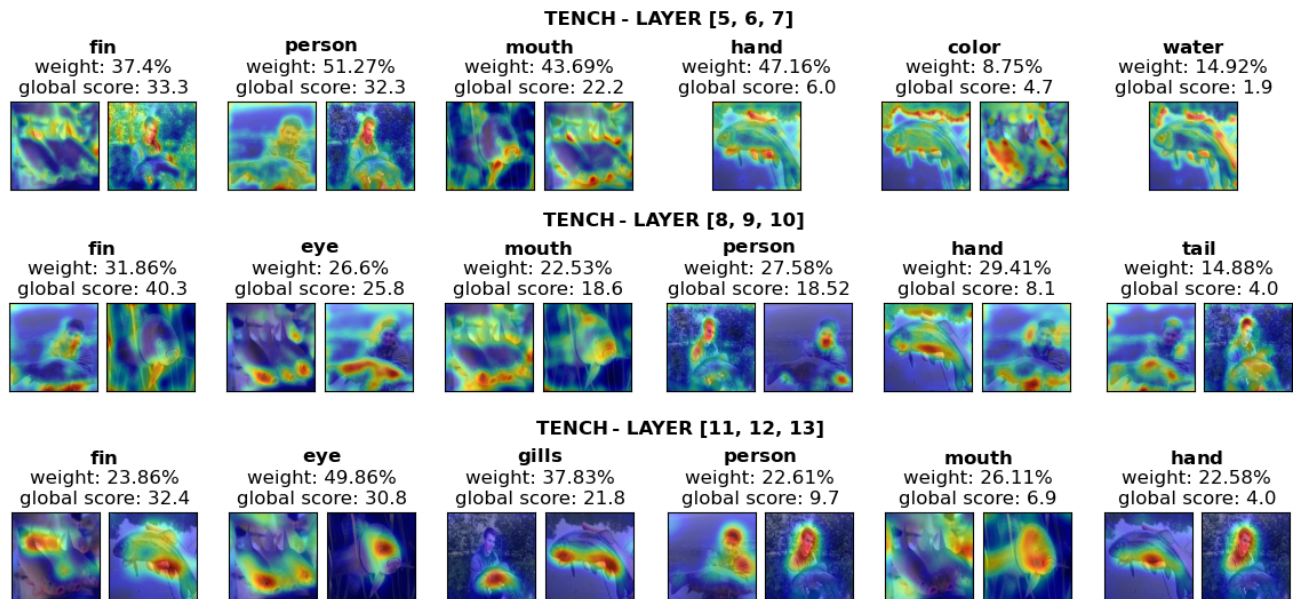


Figure 24: An example of a global explanation for the class "Tench". We chose this example to illustrate that certain features, such as the person and his hand, can be highly relevant for recognizing a class, even if we do not expect them to be. This highlights the importance of having a sufficient number of images to generate proper global explanations, as the limited samples we used may not contain all the features that the network deems relevant for predicting a certain class.

The global explanations presented in this section were generated by grouping layers into groups of three, but the size of these groups can be of any choice. We chose this grouping to avoid combining cluster maps generated from feature maps of different sizes and to prevent the overwhelming amount of information that would be shown without any grouping. By adjusting this parameter, it is also possible to generate simple and straightforward explanations by combining all layers together, as shown in Figure 25. However, it is important to note that such explanations may appear clearer but they provide a lower level of detail. Furthermore, weights are excluded from the visualization as they may lose meaning if averaged out between shallow and deep layers. Other global explanations are available in Appendix C, while results for every class are available online⁷.

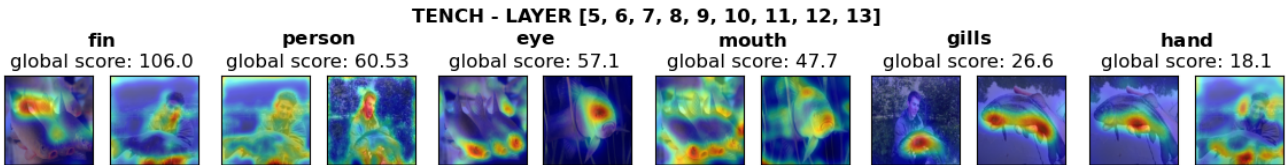


Figure 25: This figure provides a simple and straightforward global explanation for the class "Tench", generated by grouping the features extracted at all layers. It summarizes the most relevant features that the network has used to recognize the tench in our five images. The advantage of using this kind of visualization is its ease of understanding. However, it lacks significant details, such as the importance of each feature and the process by which these were identified through the various layers of the network.

5.2.3 A discussion of Deep Reveal questionnaire results

In this section, we present the results of the *Deep Reveal* participation questionnaire, which can be found in Appendix D. We also discuss how these results can suggest possible improvements in the proposed gamification activity. The questionnaire was completed by 152 participants, and the final scores for usability and workload are displayed in Table 7, along with a user experience score which was overall positive with a rating of 3.95 out of 5. The final SUS score was approximately 80.9, corresponding to a percentile ranking of 90, indicating that the system is considered between good and excellent [8]. However, the NASA-TLX score⁸ was 38.1. This score is considered somewhat high [37], suggesting that *Deep Reveal* has more room for improvement in terms of workload aspects rather than usability. One interpretation could be that guessing and labeling require effort and are not easy tasks.

Participants identified the biggest weak point of *Deep Reveal* to be the images to guess, as shown in Figure 26. Additionally, a significant amount of feedback indicated that some games were disproportionately more difficult than others, causing frustration and confusion. This can be attributed to the fact that not all cluster maps highlighted relevant information to discriminate between classes. A takeaway for future implementations is to design games with a more balanced difficulty, which could be achieved by revealing slightly more of the image if the weight of the corresponding cluster map is lower than the average. Another approach could be introducing different difficulty levels based on the cluster map weight. This idea derives from the fact that users used slightly fewer hints when playing with more important cluster maps, as shown in Figure 27.

The second most significant weakness was the process of adding characteristics. Users noted that it was sometimes challenging to assign labels to features they were unfamiliar with or that they don't know the name of. This emphasizes the importance, for real-world applications, of including expert users that have adequate knowledge about the classes to label. Furthermore, according to users, other improvements can be made in the gameplay, which was sometimes repetitive, and future implementations with better graphics and a more elaborated score system could enhance the user experience.

⁷<https://github.com/antonio-dee/abstract-network-visualizations>

⁸For practical reasons, we opted for the simplified version of the NASA-TLX that excludes the weighing step. Therefore, the resulting score should be considered more of an estimate than an accurate evaluation.

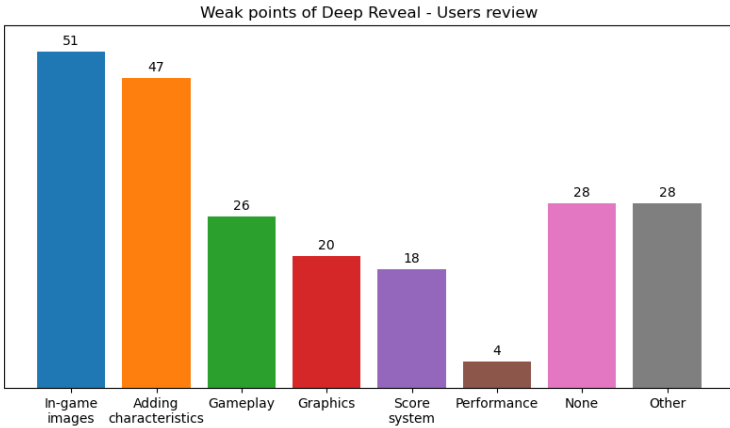


Figure 26: A visualization of the weak points of *Deep Reveal*, according to the participants of the questionnaire. The main ones were the in-game images, the process of adding characteristics and the gameplay.

Scores		
Rating	SUS	NASA-TLX
3.95/5	80.9/100	38.1/100

Table 7: The table displays the scores obtained for the System Usability Scale (SUS), NASA-TLX, and a Rating of the application. Higher scores for SUS and Rating respectively indicate better usability and overall user experience, while a higher score for NASA-TLX indicates a higher workload, therefore the lower the better.

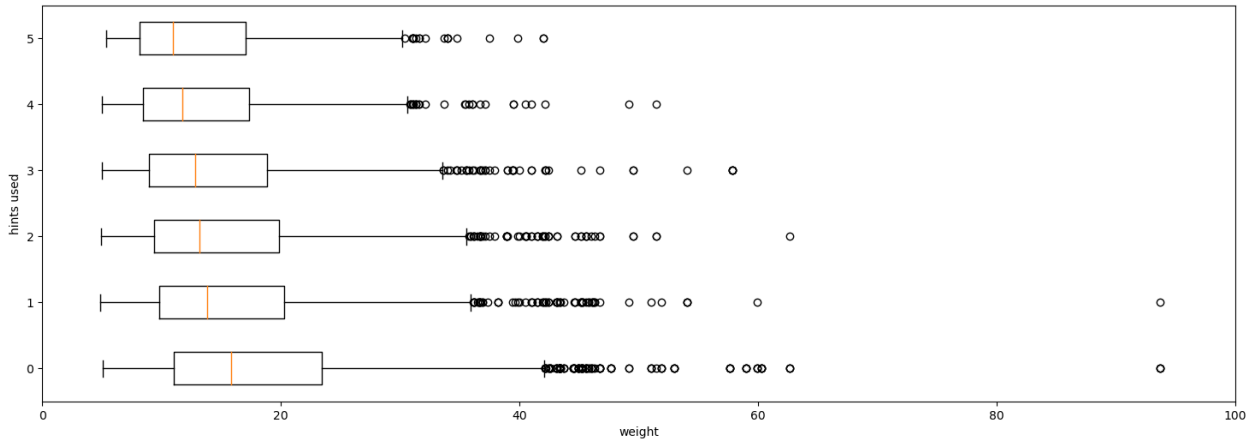


Figure 27: A box-plot depicting the relation between the number of hints used and the weight of the cluster maps. We can observe a slight trend, where additional hints correspond to a lower weight. Since the number of hints can be seen as a proxy of the difficulty, we can infer that higher weights correspond to masked cluster maps that may be easier to guess.

6. Conclusion and Future Works

We presented Abstract Network Visualizations, a novel approach for generating post-hoc local explanations within the context of CNN-based image classification. Using our approach, we generated a highly detailed visualization of the features extracted from the input image at each layer through clustering and merging of feature maps. We utilized the Grad-CAM formula to obtain a weight for each of these features and measure their contribution towards the prediction. Furthermore, we associated human-understandable concepts with these features in the form of labels by engaging humans in a gamified activity that consisted in playing an online image-guessing game named *Deep Reveal*. Finally, we showed that these labels can be aggregated to generate global explanations and provide insights into how the model recognizes a particular class. Such explanations not only increase the transparency of the model but can also be used for the diagnosis of the network, e.g. by identifying any features on which there may be overfitting.

The results of our experiments proved the potential of our explainability method, although some open questions and improvements still need to be addressed by future research. These questions include how to extend *Deep Reveal* for collecting labels relative to images predicted wrongly by the network and how to methodically validate the correctness of the labels. Regarding the former question, one approach could be to consider only the labels provided by users who made the same mistake as the machine. As for the latter question, one possible solution could be to combine our method with TCAV, using it to evaluate the model’s sensitivity to the features our method identified as the most important. Additional future work could involve exploring the possibility to associate CNN filters and the labels assigned to cluster maps once a certain level of knowledge about a specific

CNN has been obtained and labels have become saturated. This could lead to the possibility of generating an ANV of an image concurrently with the CNN execution, meaning that the crowdsourcing step is needed only once per model.

References

- [1] David Alvarez-Melis and Tommi S. Jaakkola. On the robustness of interpretability methods. *CoRR*, abs/1806.08049, 2018.
- [2] Sule Anjomshoae, Kary Främling, and Amro Najjar. Explanations of black-box model predictions by contextual importance and utility. 05 2019.
- [3] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, 2019.
- [4] Agathe Balayn, Panagiotis Soilis, Christoph Lofi, Jie Yang, and Alessandro Bozzon. What do you mean? interpreting image classification with crowdsourced concept extraction and analysis. In *Proceedings of the Web Conference 2021*, WWW '21, page 1937–1948, New York, NY, USA, 2021. Association for Computing Machinery.
- [5] Shane Barratt. Interpnet: Neural introspection for interpretable deep learning, 2017.
- [6] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. *CoRR*, abs/1704.05796, 2017.
- [7] John Brooke. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, 11 1995.
- [8] John Brooke. Sus: a retrospective. *Journal of Usability Studies*, 8:29–40, 01 2013.
- [9] Lei Cai, Jingyang Gao, and Di Zhao. A review of the application of deep learning in medical image classification and segmentation. *Annals of Translational Medicine*, 8(11), 2020.
- [10] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. *CoRR*, abs/1710.11063, 2017.
- [11] David Cian, Jan van Gemert, and Attila Lengyel. Evaluating the performance of the LIME and grad-cam explanation methods on a LEGO multi-label image classification task. *CoRR*, abs/2008.01584, 2020.
- [12] Juan Manuel Durán and Karin Rolanda Jongsma. Who is afraid of black box algorithms? on the epistemological and ethical basis of trust in medical ai. *Journal of Medical Ethics*, 47(5):329–335, 2021.
- [13] Vladimir Estivill-Castro, Eugene Gilmore, and René Hexel. Human-in-the-loop construction of decision tree classifiers with parallel coordinates. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3852–3859, 2020.
- [14] William Ferguson, Dhruv Batra, Raymond Mooney, Devi Parikh, Antonio Torralba, David Bau, David Diller, Josh Fasching, Jaden Fiotto-Kaufman, Yash Goyal, Jeff Miller, Kerry Moffitt, Alex Oca, Ramprasaath Rs, Ayush Shrivastava, Jialin Wu, and Stefan Lee. Reframing explanation as an interactive medium: The equas (explainable question answering system) project. *Applied AI Letters*, 2, 11 2021.
- [15] Kary Främling, Samanta Knapič, and Avleen Malhi. *ciu.image: An R Package for Explaining Image Classification with Contextual Importance and Utility*, pages 55–62. 07 2021.
- [16] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2013.
- [17] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Dino Pedreschi, and Fosca Giannotti. A survey of methods for explaining black box models, 2018.
- [18] Sandra G. Hart and Lowell E. Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In Peter A. Hancock and Najmedin Meshkati, editors, *Human Mental Workload*, volume 52 of *Advances in Psychology*, pages 139–183. North-Holland, 1988.

- [19] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). 2017.
- [20] Samanta Knapič, Avleen Malhi, Rohit Saluja, and Kary Främling. Explainable artificial intelligence for human decision support system in the medical domain. *Machine Learning and Knowledge Extraction*, 3(3):740–770, 2021.
- [21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012.
- [22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.
- [23] Miguel Lerma and Mirtha Lucas. Grad-cam++ is equivalent to grad-cam with positive gradients, 2022.
- [24] Zachary Chase Lipton. The mythos of model interpretability. *CoRR*, abs/1606.03490, 2016.
- [25] Tianyuan Liu, Hangbin Zheng, Jinsong Bao, Pai Zheng, Junliang Wang, Changqi Yang, and Jun Gu. An explainable laser welding defect recognition method based on multi-scale class activation mapping. *IEEE Transactions on Instrumentation and Measurement*, 71:1–12, 2022.
- [26] Xiaotian Lu, Arseny Tolmachev, Tatsuya Yamamoto, Koh Takeuchi, Seiji Okajima, Tomoyoshi Takebayashi, Koji Maruhashi, and Hisashi Kashima. Crowdsourcing evaluation of saliency-based XAI methods. *CoRR*, abs/2107.00456, 2021.
- [27] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions, 2017.
- [28] Andreas Madsen, Nicholas Meade, Vaibhav Adlakha, and Siva Reddy. Evaluating the faithfulness of importance measures in NLP by recursively masking allegedly important tokens and retraining. *CoRR*, abs/2110.08412, 2021.
- [29] Lucie Charlotte Magister, Dmitry Kazhdan, Vikash Singh, and Pietro Liò. Gcexplainer: Human-in-the-loop concept-based explanations for graph neural networks. *CoRR*, abs/2107.11889, 2021.
- [30] Andrzej Maćkiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers & Geosciences*, 19(3):303–342, 1993.
- [31] Swati Mishra and Jeffrey M. Rzeszotarski. Crowdsourcing and evaluating concept-driven explanations of machine learning models. *Proc. ACM Hum.-Comput. Interact.*, 5(CSCW1), apr 2021.
- [32] Masahiro Mitsuhashi, Hiroshi Fukui, Yusuke Sakashita, Takanori Ogata, Tsubasa Hirakawa, Takayoshi Yamashita, and Hironobu Fujiyoshi. Embedding human knowledge in deep neural network via attention map. In *VISIGRAPP*, 2019.
- [33] Benedikt Morschheuser, Juho Hamari, and Jonna Koivisto. Gamification in crowdsourcing: A review. 01 2016.
- [34] Mohammed Bany Muhammad and Mohammed Yeasin. Eigen-cam: Class activation map using principal components. *CoRR*, abs/2008.00299, 2020.
- [35] Daniel Omeiza, Skyler Speakman, Celia Cintas, and Komminist Weldemariam. Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models. *CoRR*, abs/1908.01224, 2019.
- [36] Keiron O’Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.
- [37] Atyanti Dyah Prabaswari, Chancard Basumerda, and Bagus Wahyu Utomo. The mental workload analysis of staff in study program of private educational organization. *IOP Conference Series: Materials Science and Engineering*, 528(1):012018, may 2019.
- [38] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier, 2016.
- [39] Sarat Kumar Sarvepalli. Deep learning in neural networks: The science behind an artificial brain, 10 2015.

- [40] Ramprasaath R. Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391, 2016.
- [41] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. *CoRR*, abs/1704.02685, 2017.
- [42] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2013.
- [43] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda B. Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *CoRR*, abs/1706.03825, 2017.
- [44] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806, 2014.
- [45] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Gradients of counterfactuals. *CoRR*, abs/1611.02639, 2016.
- [46] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *CoRR*, abs/1703.01365, 2017.
- [47] Maxwell Szymanski, Martijn Millecamp, and Katrien Verbert. Visual, textual or hybrid: The effect of user expertise on different explanations. In *26th International Conference on Intelligent User Interfaces, IUI '21*, page 109–119, New York, NY, USA, 2021. Association for Computing Machinery.
- [48] Andrea Tocchetti, Lorenzo Corti, Marco Brambilla, and Irene Celino. Exp-crowd: A gamified crowdsourcing framework for explainability. *Frontiers in Artificial Intelligence*, 5, 2022.
- [49] Schrasing Tong and Lalana Kagal. Investigating bias in image classification using model explanations, 2020.
- [50] Tolga Turay and Tanya Vladimirova. Toward performing image classification and object detection with convolutional neural networks in autonomous driving systems: A survey. *IEEE Access*, 10:14076–14119, 2022.
- [51] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 11 2008.
- [52] Warren von Eschenbach. Transparency and the black box problem: Why we do not trust ai. *Philosophy & Technology*, 34, 12 2021.
- [53] Haofan Wang, Mengnan Du, Fan Yang, and Zijian Zhang. Score-cam: Improved visual explanations via score-weighted class activation mapping. *CoRR*, abs/1910.01279, 2019.
- [54] Jun Wang, Changsheng Zhao, Junfu Xiang, and Kanji Uchino. Interactive topic model with enhanced interpretability. In *IUI Workshops*, 2019.
- [55] Ming-Xi Wang and Yang Qu. Approximation capabilities of neural networks on unbounded domains, 2019.
- [56] Hans-Dieter Wehle. Machine learning, deep learning, and ai: What’s the difference? 07 2017.
- [57] Feiyu Xu, Hans Uszkoreit, Yangzhou Du, Wei Fan, Dongyan Zhao, and Jun Zhu. *Explainable AI: A Brief Survey on History, Research Areas, Approaches and Challenges*, pages 563–574. 09 2019.
- [58] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.
- [59] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. *CoRR*, abs/1710.00935, 2017.
- [60] Yunyan Zhang, Daphne Hong, Daniel McClement, Olayinka Oladosu, Glen Pridham, and Garth Slaney. Grad-cam helps interpret the deep learning models trained to classify multiple sclerosis types using clinical brain magnetic resonance imaging. *Journal of Neuroscience Methods*, 353:109098, 2021.
- [61] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

A. Appendix Overview

In the appendix, we provide:

1. More examples of Abstract Network Visualizations
2. More examples of Global Explanations
3. *Deep Reveal* experiment participation questionnaire

B. More examples of Abstract Network Visualizations

Continuing from the results presented in Section 5.2.1, we also provide additional ANVs for different images and classes in Figures 28, 29, 30, and 31.

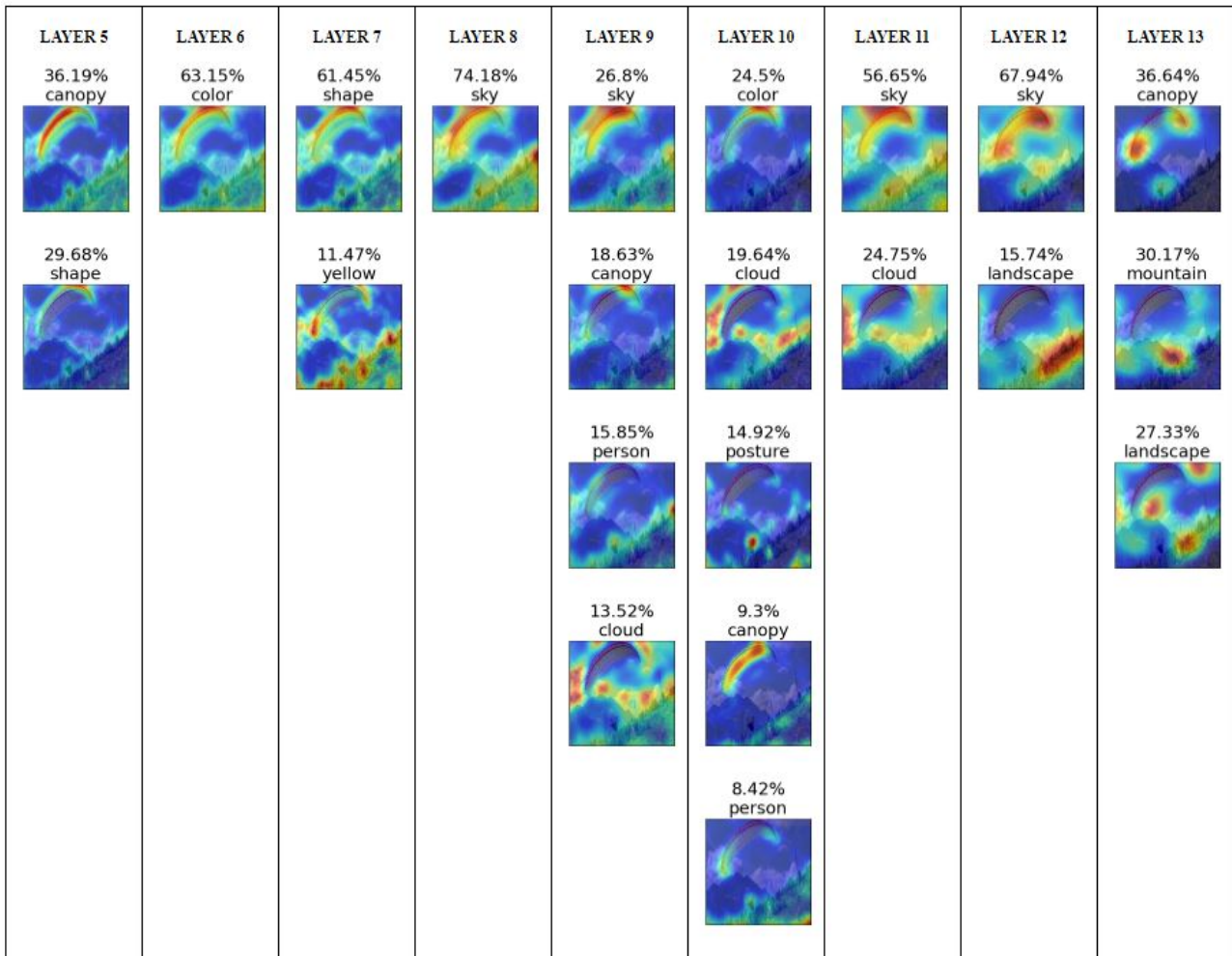


Figure 28: An ANV was generated for an image of a parachute. The network successfully recognized the shape and color of the canopy, as well as the posture of the person in the image. However, the context of the image, including the landscape, mountain, and sky, also played a significant role in predicting the classification.

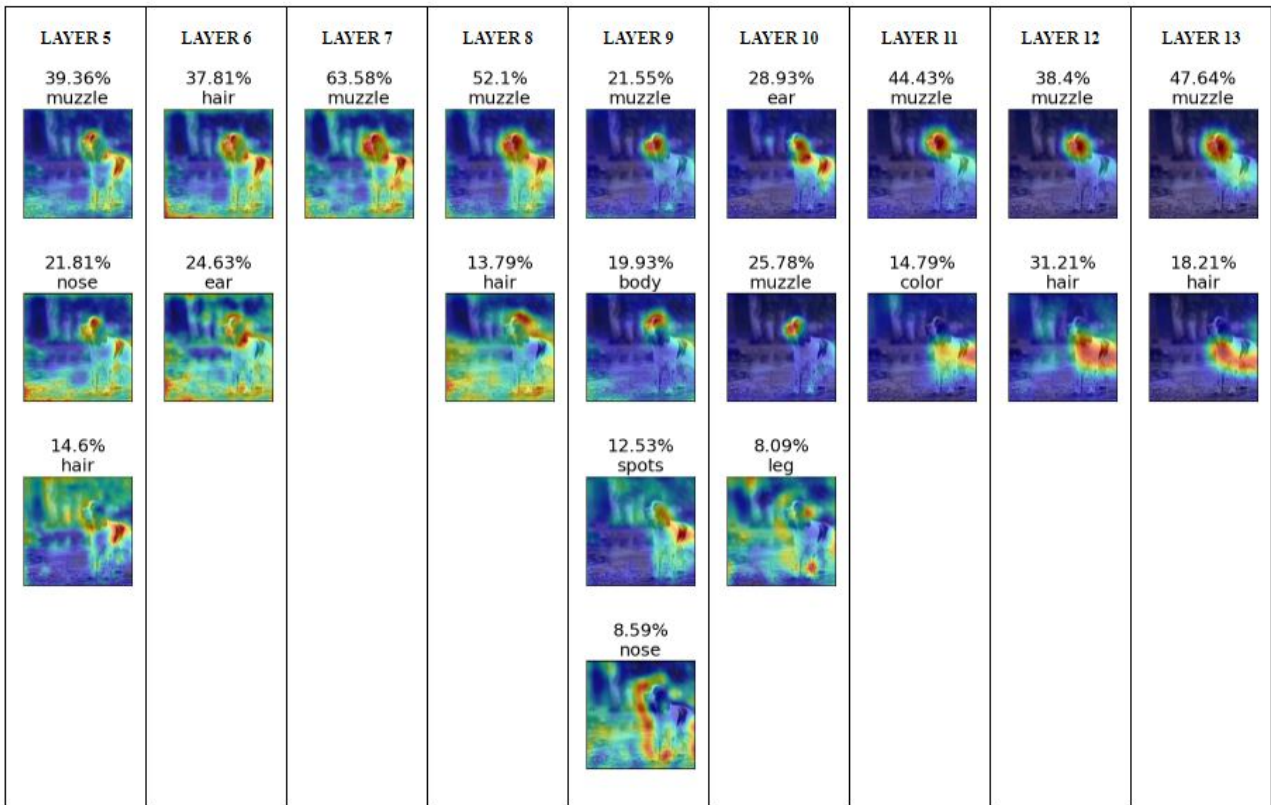


Figure 29: An ANV is presented here for an image of an English Springer. The muzzle and hair were found to be the most significant features for predicting its classification, and the network was also able to recognize its spots and ears, which are the principal characteristics of this breed. It should be noted that some labels may appear to be completely wrong, such as the "body" label in layer 9. However, this cluster map contained three labels with equal scores: "body", "head", and "face", and the former was displayed only because of the alphabetic order.

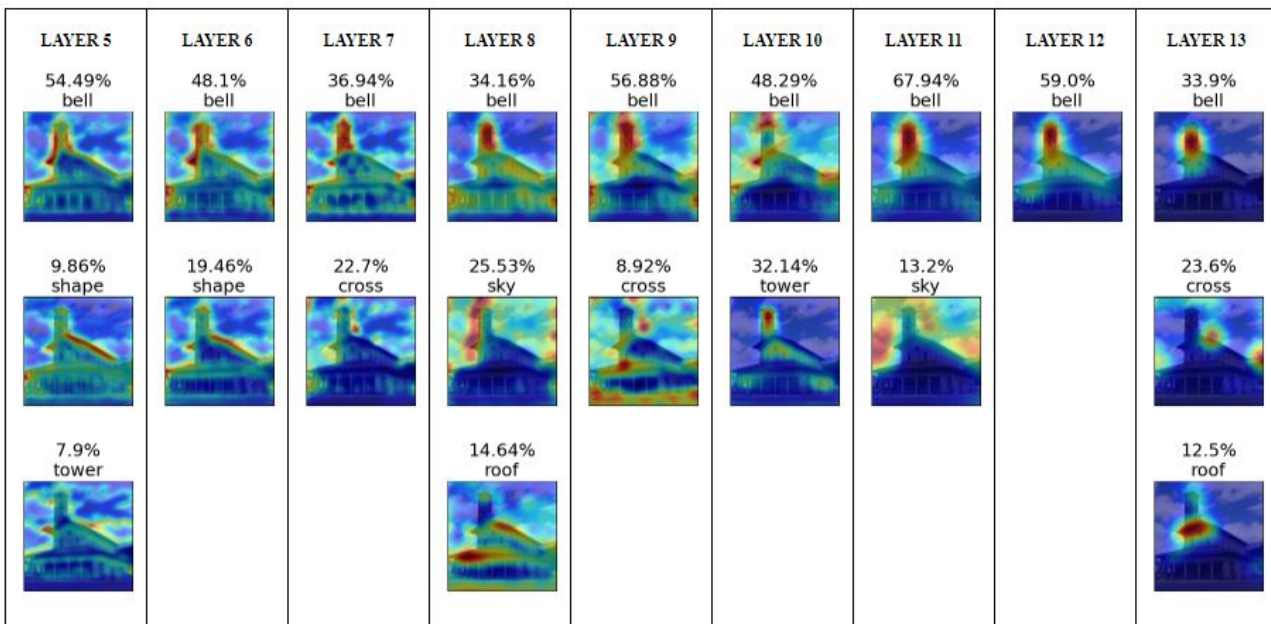


Figure 30: An ANV for an image of a church. Every layer concentrated on the bell tower, which is the most class-discriminative feature in this image together with the cross and the roof. We can also observe that shallower layers are focusing mostly on the shape of the building and the tower.

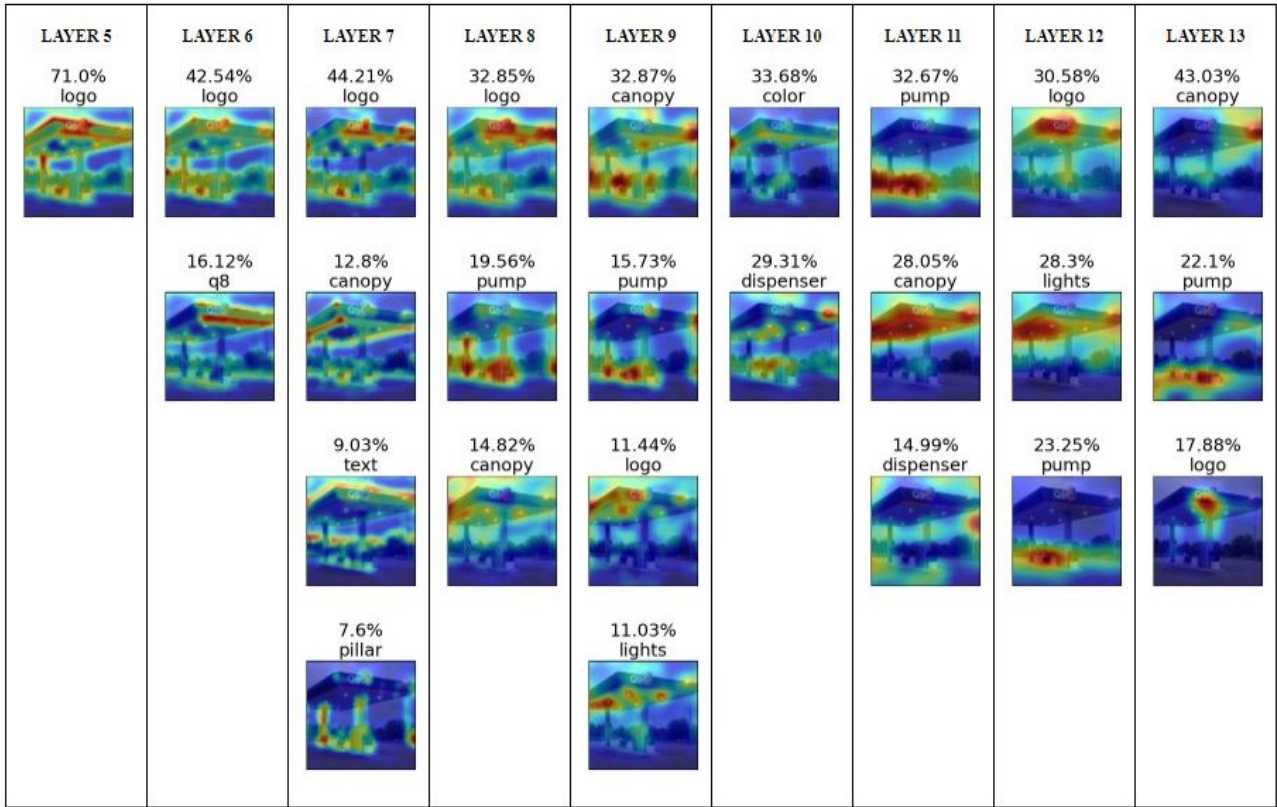


Figure 31: An ANV for an image of a gas pump. Similar to the church image, the heatmaps show that the network focused on the building's shape in the shallower layers, and more semantic features such as the canopy, the lights, and the pump itself in deeper layers. Interestingly, the visualization reveals that the network was capable of recognizing the "Q8" logo as an important feature for a gas pump. Furthermore, a weak point of this explanation is that the heatmap for the canopy in the final layer is slightly inaccurate, possibly due to the gradient saturation issue discussed in the main document (see Section 2.2.2).

C. More examples of Global Explanations

Building upon the results outlined in Section 5.2.2, we have incorporated additional global explanations for various classes illustrated in Figures 32, 33, and 34.

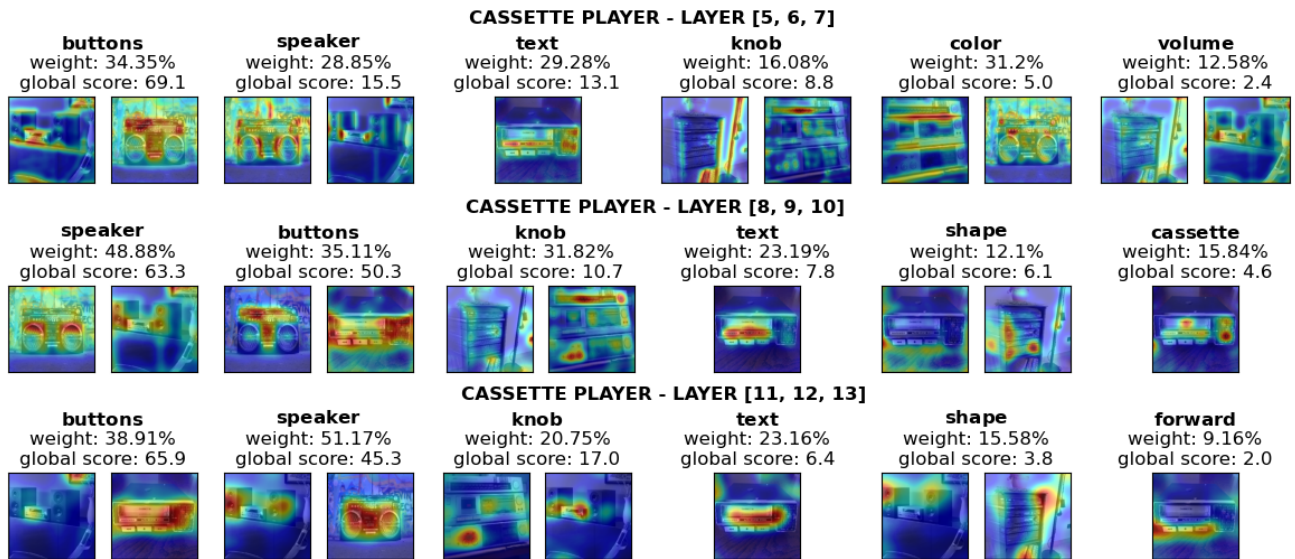


Figure 32: A global explanation for the "Cassette Player" class. The most relevant features were buttons and speakers. However, speakers are not always part of the cassette player, so assigning them high importance may not be ideal. Interestingly, the network recognized the written text "cassette" and the forward symbol.

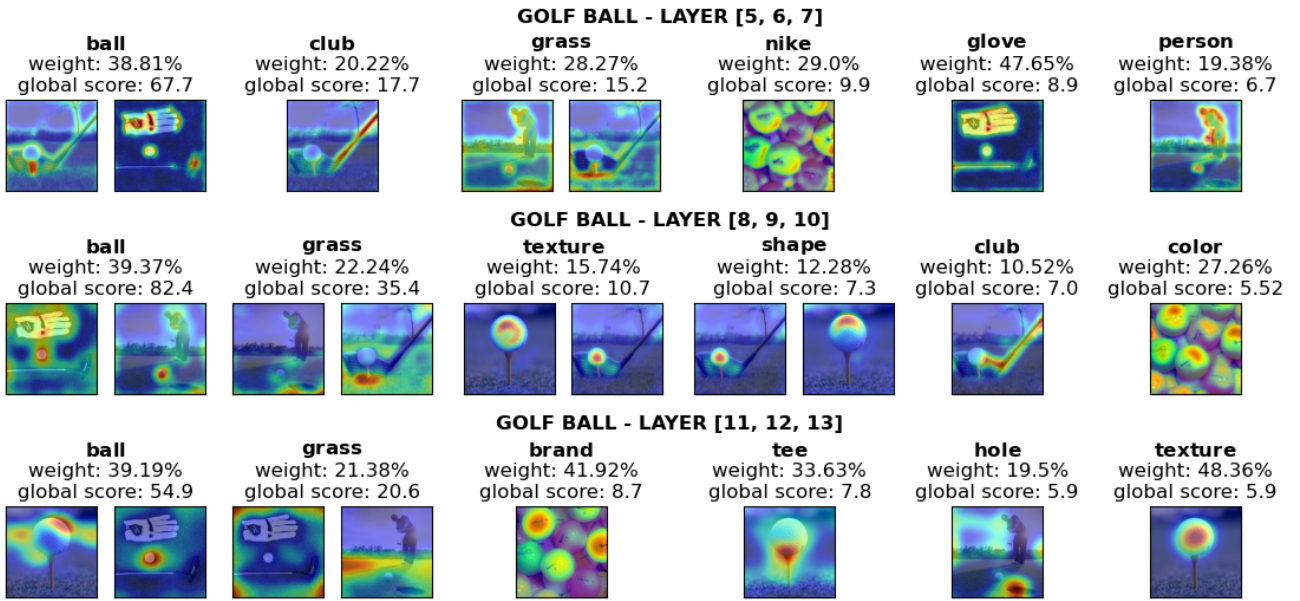


Figure 33: An example of a global explanation for the class "Golf Ball". The network examined various features such as the ball's shape and texture, while also making use of contextual clues to help in its predictions. For instance, it associated the presence of grass, tees, clubs, and holes with a golf ball context. Furthermore, some users recognized the golf balls using the Nike symbol. However, it is difficult to determine whether the network did it too, or if it solely relied on the ball's shape and texture. As stated in the main document, further validation is required for such features.

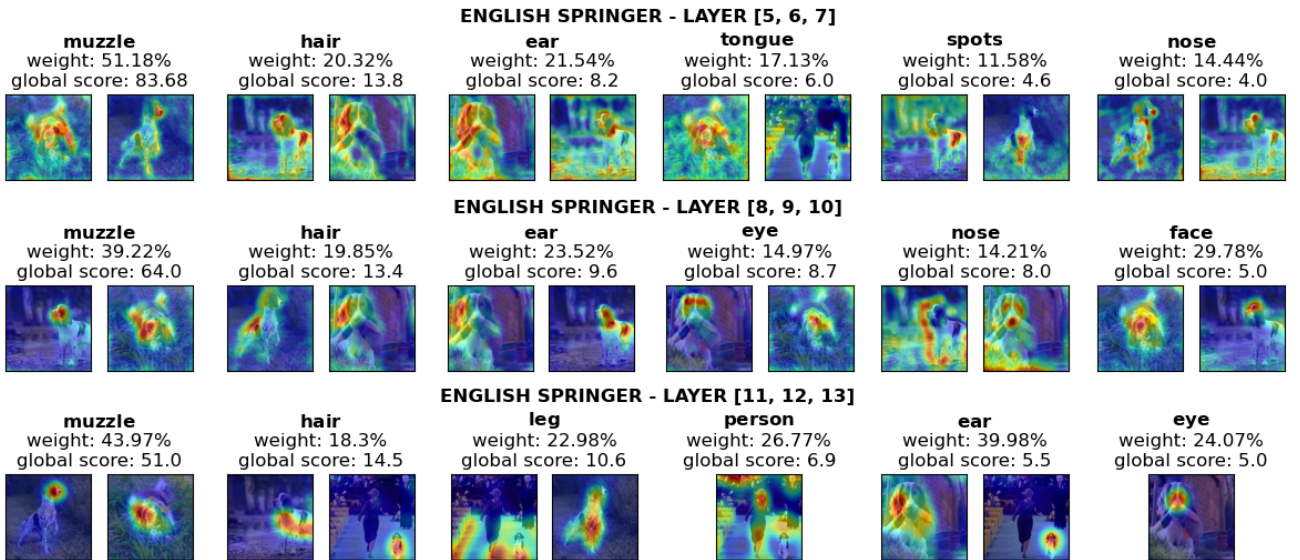


Figure 34: An example of a global explanation for the "English Springer" class. The dog's head was primarily identified by examining its muzzle, eyes, nose, and ears. Meanwhile, its body was recognized by its hair, legs, and spotting patterns. It's worth noting that the person also appears to have a higher-than-expected weight. However, this can be justified since the cluster map was also slightly focused on the dog.

D. Deep Reveal experiment participation questionnaire

D.0.1 Participant information

1. Please provide the email address that you used to register on Deep Reveal.
-

D.0.2 Usability

	Strongly Disagree				Strongly Agree
1. I think that I would like to play Deep Reveal often.	1	2	3	4	5
2. I found Deep Reveal to be unnecessarily complex.	1	2	3	4	5
3. I found Deep Reveal to be easy to use.	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use Deep Reveal.	1	2	3	4	5
5. I found the various functions of Deep Reveal were well integrated.	1	2	3	4	5
6. I thought there was too much inconsistency in Deep Reveal.	1	2	3	4	5
7. I would imagine that most people would learn to use Deep Reveal very quickly.	1	2	3	4	5
8. I found Deep Reveal very cumbersome to use.	1	2	3	4	5
9. I felt very confident using Deep Reveal.	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with Deep Reveal.	1	2	3	4	5

D.0.3 Workload

How mentally demanding was Deep Reveal?

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Very Low Very High

How strenuous was to use Deep Reveal?

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Very Low Very High

How hurried or rushed was the pace of the game?

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Very Low Very High

How successful were you in playing Deep Reveal?

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Perfect Failure

How hard did you have to work to accomplish your level of performance?

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Very Low Very High

How insecure, discouraged, irritated, stressed, and annoyed were you?

0	1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	---	----

Very Low Very High

D.0.4 Feedback

1. Which of the following would you consider to be the weak points of the game? (Multiple choices allowed)
A. Gameplay B. Graphics C. In-game images D. Score system E. Adding characteristics
F. None of the above G. Other: _____

2. How did you find the overall experience?

1	2	3	4	5
---	---	---	---	---

3. Additional feedback (Optional)

Abstract in lingua italiana

La trasparenza e la spiegabilità nei processi di classificazione di immagini sono fattori essenziali per instaurare fiducia nei modelli di apprendimento automatico e rilevare discriminazioni ed errori. Le tecniche di spiegabilità più avanzate generano mappe di calore che evidenziano le regioni dell'immagine in cui viene identificata una classe specifica, senza fornire però una spiegazione completa di come il modello sia arrivato alla sua decisione. Per rispondere a questa esigenza, proponiamo una tecnica post-hoc per la generazione di spiegazioni locali che forniscono una panoramica del processo in cui il modello estrae caratteristiche dalle immagini. Queste spiegazioni consistono in una visualizzazione delle caratteristiche estratte dal modello per ogni suo layer e sono denominate Abstract Network Visualizations (ANV). Tali caratteristiche sono rappresentate da mappe di calore generate dalla fusione di Feature Map simili raggruppate con tecniche di clustering, alle quali associamo un peso utilizzando Grad-CAM, una tecnica di spiegabilità locale. Queste mappe di calore sono, inoltre, descritte da un insieme di annotazioni raccolte tramite un'attività di crowdsourcing gamificata, che migliora ulteriormente l'interpretabilità delle nostre spiegazioni locali. Infine, dimostriamo che tali annotazioni possono anche consentire la produzione di spiegazioni globali, aggregando mappe annotate in modo simile su più immagini.

Parole chiave: spiegabilità, classificazione di immagini, apprendimento automatico, gamificazione, crowdsourcing.

Ringraziamenti

Questo spazio è dedicato a tutti coloro che in un modo o nell'altro sono stati al mio fianco nel raggiungimento di questo piccolo traguardo.

Naturalmente ringrazio il professore e mio relatore Marco Brambilla che è stato un punto di riferimento per questo lavoro ed in generale per tutto il percorso universitario inclusa la triennale.

Ringrazio Andrea Tocchetti che è stato il miglior correlatore che si potesse mai chiedere sia dal punto di vista umano che di preparazione, e non è assolutamente una esagerazione.

Ringrazio naturalmente Matteo che, oltre ad essere un amico e un ottimo collaboratore, è tra le persone che sono più grato di aver conosciuto durante questo percorso.

Ringrazio inoltre ognuna delle persone che hanno preso parte all'attività di crowdsourcing che ha reso possibile questo lavoro. È inutile dire che senza di voi questa tesi non sarebbe stata possibile e per questo il lavoro fatto ed i risultati ottenuti sono anche in parte di ognuno di voi.

Un ringraziamento speciale va poi ai miei genitori che hanno fatto sacrifici credendo in me e questo mi ha reso orgoglioso e mi ha dato senso di responsabilità. Vi voglio bene e questo traguardo lo dedico a voi.

Ringrazio mia sorella Chiara che è la persona a cui sono legato di più al mondo. Mi hai costantemente iniettato autostima e mi scuso se a volte non sono riuscito a fare altrettanto.

Ringrazio zio Mino, che è stato come un secondo padre in questi anni ed il mio principale punto di riferimento quando a 18 anni mi sono ritrovato solo e lontano da casa.

Ringrazio comunque tutta la mia famiglia, ma in special modo nonno Lucido, zia Michela e zio Sergio che purtroppo non ci sono più.

Ringrazio anche nonna Giovanna che per me è stata come una mamma e mi dispiace che questa volta non potrà essere insieme a me a festeggiare.

Ringrazio infine gli amici e compagni universitari che ho conosciuto durante questo percorso, tra cui volevo citare in particolare Riccardo che negli anni è diventato sempre più come un fratello per me.

Grazie veramente a tutti per mi avermi insegnato che le soddisfazioni più grandi si raggiungono sempre e solo insieme.

Antonio De Santis

04/05/2023

Dedico questo spazio a tutti coloro che mi hanno supportato in questo lungo percorso di crescita professionale e personale.

Innanzitutto, desidero ringraziare il mio relatore Marco Brambilla, per la possibilità di lavorare a questo progetto e per tutti i preziosi consigli durante il suo svolgimento.

Un sentito ringraziamento va al mio correlatore Andrea Tocchetti, per la disponibilità e per tutti i puntuali aiuti nella stesura di questa Tesi.

Non posso non ringraziare Antonio, il mio co-autore, con cui ho condiviso l'intero percorso universitario. Grazie per aver accettato di lavorare assieme a questo progetto, e per avermi sopportato per tutta la sua durata. Sono felice di poter condividere questo traguardo assieme a un caro amico.

Ringrazio tutti coloro che hanno partecipato e aiutato nell'esperimento di questa tesi, senza il quale tutto questo lavoro non sarebbe stato possibile.

Un Grazie a tutti i miei familiari, per aver sempre creduto in me e per avermi aiutato a credere in me stesso. In particolare, un doveroso Grazie va ai miei genitori, che nonostante i miei difetti mi sono stati vicini e mi hanno sempre sostenuto in questo insidioso percorso, nel bene e nel male. Grazie di tutto.

Infine ringrazio gli amici, universitari e non, che mi hanno aiutato a crescere e con il quale ho potuto condividere gli anni migliori della mia vita. Grazie per ogni momento.

Grazie infinite a tutti.

Matteo Bianchi

04/05/2023