ECLAD: Extracting Concepts with Local Aggregated Descriptors

Anonymous authors
Paper under double-blind review

Abstract

Convolutional neural networks (CNNs) are increasingly being used in critical systems, where robustness and alignment are crucial. In this context, the field of explainable artificial intelligence has proposed the generation of high-level explanations of the prediction process of CNNs through concept extraction. While these methods can detect whether or not a concept is present in an image, they are unable to determine its location. What is more, a fair comparison of such approaches is difficult due to a lack of proper validation procedures. To address these issues, we propose a novel method for automatic concept extraction and localization based on representations obtained through pixel-wise aggregations of CNN activation maps. Further, we introduce a process for the validation of concept-extraction techniques based on synthetic datasets with pixel-wise annotations of their main components, reducing the need for human intervention. Extensive experimentation on both synthetic and real-world datasets demonstrates that our method outperforms state-of-the-art alternatives.

1 Introduction

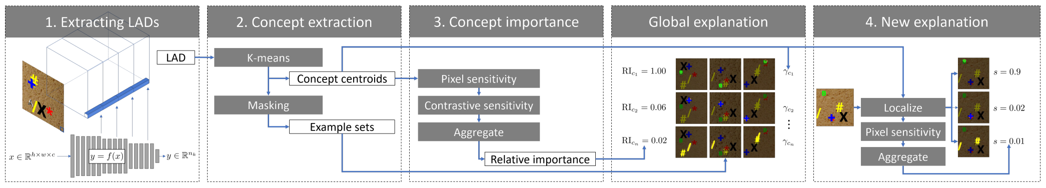
As convolutional neural networks (CNN) become increasingly used in critical real-world applications (e.g., quality control (Wang et al., 2018) or medical diagnosis (Benjamens et al., 2020)), there is an urgent need to understand their inner workings. This has led to a growing adoption of explainability methods during the lifecycle of models (Burkart & Huber, 2021; Dhanorkar et al., 2021; Wijaya et al., 2021) in an effort to increase transparency and trust, convey a sense of causality, ensure alignment, and make adjustments when necessary (Bhatt et al., 2020; Arrieta et al., 2020).

In particular, post-hoc visual explanations of CNNs have proven to be useful for detecting undesired biases or unexpected behaviors in models (Singh et al., 2020; Tjoa & Guan, 2021). In recent years, post-hoc visual explanations have been tackled by either (i) adopting feature attribution methods (Selvaraju et al., 2020; Shrikumar et al., 2017; Qi et al., 2019), or (ii) mining higher level features through concept extraction (CE) techniques (Kim et al., 2018; Ghorbani et al., 2019; Yeh et al., 2020). Explanations provided by the first approach are termed *local explanations* as they focus on analyzing single data instances, while those of the second approach are *global explanations* as they focus on obtaining features pertaining to the understanding of the model as a whole. Although these two approaches are widely used, both have significant limitations.

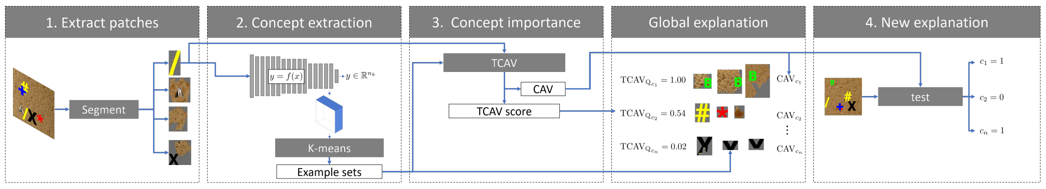
As a practical example, let us consider a CNN model for the classification of parts in a quality control process (good, scratched, and deformed edges). During the lifecycle of the CNN, explainable artificial intelligence (XAI) may be used to detect undesired behaviors and to better understand which features are present in the acquired data. This will increase trust in the model and ensure a high-level alignment with expert knowledge.

Feature attribution (local explanation) methods can be used to determine (for *single images*) whether the pixels in the scratched or deformed regions are important for image classification, yet they do not tell us which groups of pixels are contextually related (composing a scratch), or whether the model distinguishes pixels in a scratch from pixels in a deformed edge. Moreover, recent studies have shown that feature attribution methods can be noisy and misleading (Adebayo et al., 2018).

Concept extraction (global explanation) methods can analyze a model in the context of a dataset and *return different sets of images* representing concepts – in the case of the example mentioned above,



(a) ECLAD



(b) ACE

Figure 1: Proposed concept extraction technique ECLAD 1(a) in comparison to a state-of-the-art alternative ACE 1(b). ECLAD extracts a representation per pixel (LAD) before clustering and extracting concepts. ACE segments each image and uses a single representation to describe each patch before clustering. For new explanations, ECLAD provides the localization of each concept in the image, whereas ACE only tests whether each concept exists in the image.

samples of scratches or deformed edges. These sets represent the concepts learned by a model during training and are accompanied by a score denoting their importance in the model's prediction process (Kim et al., 2018; Ghorbani et al., 2019). When explaining new instances, these methods can determine whether a concept is present, but not where it is, i.e., current methods do not localize the pixels containing each concept (where is the scratch, or deformed edge). A representative method is shown in Figure 1(b). This is a severe limitation in many applications, as posterior to the CE process, the results are not being used to explain abnormal behaviors in detail, increasing the possibility of biased interpretations. For example, a problematic instance of a piece with deformed edges and a shiny patch elsewhere may be erroneously detected as scratched by a CNN. A human may then erroneously interpret the edge as the problematic region, whereas the unusual shiny region was the confounding factor. The same issue makes the objective validation of CE techniques difficult, as interpreting which cues relate to a concept as well as its relation to the ground truth requires human intervention.

Our work focuses on concept-based explanations, i.e., global explanations, and specifically on their inability to provide a straightforward concept localization. We propose *Extracting Concepts with Local Aggregated Descriptors* (ECLAD) as a method for CE that – posterior to its global execution – is able to localize concepts and quantify their importance for a single image prediction, as shown in Figure 1(a). In contrast to previous CE methods, we do not encode an image as a single flattened activation map, but rather as a set of pixel-wise aggregations of the activation maps of multiple layers. We call these pixel-wise representations *local aggregated descriptors* (LADs).

As an orthogonal contribution, we address the challenge of quantitatively validating CE techniques, as current alternatives require human intervention to associate important concepts with the ground truth. We propose an alternative process for the validation of CE methods that requires no intervention to verify the correctness of extracted concepts. We achieve this by spatially associating labelled components on images (primitives) with regions related to the extracted concepts. As other CE methods do not provide a straightforward localization of the concepts, we segment the images and test each patch in search of every extracted concept. This process can be used to validate any other CE technique and provide a more objective performance measure. In this paper, we use this process to validate the correct performance of ECLAD using new synthetic datasets.

In summary, our main contributions are: (1) We propose a concept extraction method based on local aggregated descriptors that can extract global concepts and localize them in single images. (2) We propose a process for validating concept extraction techniques by using pixel-level ground truth to relate extracted concepts with primitives of a synthetic dataset. (3) We validate our methods with multiple synthetic datasets (making them public), as well as real-world use cases.

2 Related work on concept extraction

The goal of concept-based explanation methods is to extract high-level features that relate to the decision-making process of a CNN. To achieve this, ante-hoc approaches have proposed distinctive CNN architectures, constraining the representations learned in their latent spaces (Chen et al., 2019; Koh et al., 2020; Chen et al., 2020; Utkin et al., 2021; Goyal et al., 2019; Tran et al., 2021). In contrast, post-hoc approaches extract patches from images, and cluster them based on their representations inside the latent space of the CNNs (Ghorbani et al., 2019; Ge et al., 2021).

Our work lies in the category of post-hoc concept extraction (CE), where the standard approach for extracting concepts is the algorithm *Automatic Concept-based Explanations* (ACE) (Ghorbani et al., 2019), depicted in Figure 1(b). ACE uses a segmentation technique to extract patches before encoding them through the CNN. Afterwards, ACE clusters these representations and scores the importance of each cluster using the concept-testing algorithm TCAV (Kim et al., 2018). Other studies have built on the CE capabilities of ACE, by focusing on concept completeness (Yeh et al., 2020) (ConceptShap), or structural relations between concepts (Ge et al., 2021; Kori et al., 2020). In essence, these methods assess whether an image contains a concept, and to what extent that concept influences the prediction of said image, but not where the concept is located, omitting relevant spatial information. Our approach uses LADs to represent each pixel rather than employing a single flattened activation map describing a whole image. In contrast to the above-mentioned works, this approach allows for a straightforward localization of pixels considered part of a concept.

The second main contribution of this paper is a method for validating CE techniques. Such a validation process has proven challenging for most studies to date, and three principal approaches have been followed. The first consists of using image classification datasets, performing CE over a trained model, and visually (qualitatively) inspecting the results for specific classes (Ghorbani et al., 2019; Schrouff et al., 2021; Ge et al., 2021; Kori et al., 2020; Liu & Arik, 2020). The second approach builds on the first, performing a user study to either measure the meaningfulness of extracted concepts, or link them to the main unannotated attributes of each class (Ghorbani et al., 2019; Yeh et al., 2020). The third approach utilizes datasets (synthetic or natural) with labels denoting the presence of an attribute in each image (Goyal et al., 2019; Yeh et al., 2020). This approach allows for a quantitative evaluation of the correlation between the labels and the extracted concepts, yet it does not ensure that the same visual cue is responsible for both. Our proposed validation approach uses tailored synthetic datasets with pixel-level annotations for each primitive (visual cues or concepts present on each image) to objectively relate them to extracted concepts through a distance metric. This measure enables an automatic assessment of whether the important extracted concepts coincide with the primitives used to compose the images of each class. To our knowledge, this is the first CE validation technique that uses pixel-wise annotations to verify whether visual cues from an important extracted concept are related to dataset primitives without human intervention.

3 ECLAD

We present *Extracting Concepts with Local Aggregated Descriptors* (ECLAD) as an explanation method for CNNs that extracts concepts (meaningful representations that a model has learned) using a pixel-wise aggregation of activation maps. Its main premise is that the activation maps of the multiple low, mid, and high level layers can be re-scaled and composed at a pixel level to obtain a comprehensive description of how a neural network encodes a location of an image (including its surrounding context). Consequently, this encoding can be used to mine for concepts.

We introduce ECLAD in four three. First, we specify what we mean by local aggregated descriptors (LADs). Second, we describe the process of CE by clustering LADs. Third, we propose a metrics of the relative importance of each extracted concept. A detailed pseudocode of ECLAD's usage is provided in the appendix B.

3.1 Local aggregated descriptors

CNN classification models approximate the mapping of images x of dimensions $(h, w, 3)$ (for RGB images), to a vector y corresponding to the probability of the input belonging to n_{k} classes with a function $f : x \in \mathbb{R}^{h \times w \times 3} \rightarrow y \in \mathbb{R}^{n_{\mathrm{k}}}$. In CNNs, a partial evaluation until a layer l, yields an activation map $a_l = f_l(x)$, belonging to a latent space $a_l \in \mathbb{R}^{h_l \times w_l \times c_l}$, where the dimensions depend on the input as well as the type and quantity of layers evaluated.

To exploit the progressive information encoding of the CNNs, we propose the aggregation of the activation maps of a predefined set $L = \{l_1, \cdots, l_{n_l}\}$ of n_l layers. We obtain the aggregated descriptor of an image x_i, by first computing a_l for each layer $l \in L$. Then, we upscale each a_l to the spatial dimensions of x_i using bilinear interpolation (f_U). Finally, we concatenate the resulting maps alongside their third dimension (depth)

$$d_{x_i} = \left[f_U(f_{l_1}(x_i)) \cdots f_U(f_{l_{n_l}}(x_i)) \right]. \tag{1}$$

We obtain the descriptor $d_{x_i} \in \mathbb{R}^{h \times w \times c^*}$, where c^* is the sum of the number of units for all layers in L. *Local aggregated descriptors* (LADs) refer to each pixel $d_{x_i,(a,b)} \in \mathbb{R}^{1 \times 1 \times c^*}$ of the tensor d_{x_i}, where (a, b) denotes the position along the width and height of d_{x_i}. LADs contain information about how the CNN encodes a pixel and its surrounding context in different abstraction levels.

3.2 Concept extraction

In contrast to a flattened activation map, LADs contain information about different levels of abstraction, while remaining equivariant to translation. We can exploit these properties by performing concept mining using all LADs from a dataset E,

$$D = \{d_{x_i,(a,b)} \mid a \in \{1, \ldots, h\}, b \in \{1, \ldots, w\}, x_i \in E\}.$$

We use minibatch k-means (Sculley, 2010) to compensate for the computational cost of clustering LADs. We take batches of n_i images, compute their descriptors d_{x_i}, vectorize them, and join them (obtaining a subset of D) before applying one clustering step. This approach can be used to evaluate a full dataset or a limited subset of all classes, without scaling memory requirements. Subsequently, we obtain the set $\Gamma = \{\gamma_{c_1}, \cdots, \gamma_{c_{n_c}}\}$ of centroids $\gamma_{c_j} \in \mathbb{R}^{1 \times 1 \times c^*}$ defining the concepts.

To locate a concept c_j in an image x_i, we create a mask $m_{x_i}^{c_j} \in \mathbb{R}^{h \times w \times 1}$ by analyzing each $d_{x_i,(a,b)}$ and assessing whether it belongs to the cluster defined by γ_{c_j}. This allows for a direct evaluation of an image, identifying not only whether it contains a concept, but also where it is located (localization). We use the masks $m_{x_i}^{c_j} = \text{Mask}(x_i, \gamma_{c_j})$ to attenuate unrelated pixels by a factor $\lambda \in (0, 1]$ and obtain the human-understandable sets of examples $\varepsilon_{c_j} = \{(1 - \lambda) m_{x_i}^{c_j} \odot x_i + \lambda x_i \mid x_i \in E\}$.

3.3 Concept importance

In contrast to other approaches (Goyal et al., 2019; Ghorbani et al., 2019), we extract concepts on a pixel level by using LADs. Therefore, we compute the sensitivity on a pixel level, and aggregate it for each concept. We use the same approach and also aggregate gradients to obtain $g_{x_i} \in \mathbb{R}^{h \times w \times c^*}$,

$$g_{x_i} = \left[f_U(\nabla_{l_1} h_{l_1}^k(f_{l_1}(x_i))) \cdots f_U(\nabla_{l_{n_l}} h_{l_{n_l}}^k(f_{l_{n_l}}(x_i))) \right],$$

where $\nabla_l h_l^k(f_l(x))$ is the gradient of the prediction for the class k with respect to an activation map $a_l = f_l(x)$. Then, the sensitivity $s_{x_i,(a,b)}^k$ of a pixel becomes the dot product between its local aggregated gradient and its LAD, $s_{x_i,(a,b)}^k = (g_{x_i,(a,b)})^T \cdot d_{x_i,(a,b)}$.

We propose the contrastive sensitivity $\text{CS}_{c_j}^k$ as the difference between the average sensitivity of a concept c_j towards the class k for all images of a class k minus the average for the rest of the dataset:

$$\text{CS}_{c_j}^k = \frac{1}{|S_{c_j}^{k,E_k}|} \sum_{s_{c_j}^k \in S_{c_j}^{k,E_k}} s_{c_j}^k - \frac{1}{|S_{c_j}^{k,E_{k'}}|} \sum_{s_{c_j}^k \in S_{c_j}^{k,E_{k'}}} s_{c_j}^k, \tag{2}$$

where $S_{c_j}^{k,E_k}$ is the set of sensitivities (towards class k) for all pixels in all images of E_k belonging to the concept c_j. Similarly, $S_{c_j}^{k,E_{k'}}$ is the set of sensitivities (towards class k) for all pixels in all images of $E \setminus E_k$ belonging to the concept c_j. This measure quantifies how important a concept c_j is with respect to a specific class k. In addition, we propose a relative importance measure RI_c,

$$\text{RI}_{c_j} = \text{CS}_{c_j}^{k_{c_j}} / \max_{c_q,k}(|\ \text{CS}_{c_q}^k\ |)\ ;\ k_{c_j} = \operatorname*{argmax}_k(|\ \text{CS}_{c_j}^k\ |). \tag{3}$$

This metric not only represents how important a concept is for a single class, but also how important it is for samples of a class in contrast to samples of other classes. The relative importance (RI_{c_j}) is a scaled value denoting the highest contrastive sensitivity of a concept, and normalized across all concepts. Moreover, RI_{c_j} allows for the extraction of attributive and counterfactual concepts.

Our importance score is directly tied to (A) the spatial regions containing a concept, and (B) the magnitude of the sensitivity of units in the selected layers for different class images. The objective of this metric is to better represent the inner workings of a CNN. By doing so we avoid three known limitations of TCAV and concept Shapely values. First, by relying on (A) we avoid the misscoring of co-occurring concepts, a known limitation of Shapely values. Second, by relying on (B) we avoid misscoring concepts which are in a similar general direction when the latent space of a CNN is not zero centered. Third, (A) and (B) allow us to give a relative importance to co-occurring concepts by comparing the magnitude of their sensitivities. Our importance metric aims to directly reflect these dynamics of CNNs (and their internal activations), which are not captured through either TCAV scores or Shapely values.

4 Validation of concept extraction techniques

We propose a method for the quantitative validation of CE techniques with minimal human intervention. This method is not meant to replace usability studies with humans, which seek to understand explanations in human-AI systems (Mueller et al., 2019). Rather, it is a quantitative approach to score CE techniques purely based on synthetic datasets.

In an ideal case, we have an unbiased classification dataset of images and their labels. Within all high level features contained in the data, we have a subset of *important features* which are the differentiating factors between the labels, and a subset of *unimportant features*. We build upon the assumption that after training, a model learns to predict the labels by detecting a subset of the important features (possibly disregarding correlated features (Geirhos et al., 2020)). Then, a CE algorithm analyzes the model and dataset, extracting a set of concepts and scoring their importance.

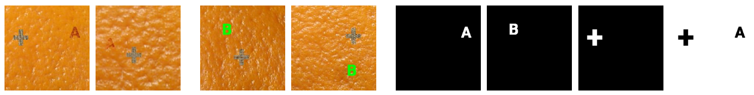
We denote as *aligned* concepts, those spatially related to the important features of the dataset (which were learned by the model). Similarly, we denote as *unaligned* concepts, those representing unimportant or unannotated features of the dataset that are irrelevant for performing the desired task. In an ideal case, where the features of a dataset were perfectly learned by a trained model, the performance of a CE method will be reflected by two metrics. **Representation correctness**: extracted *aligned* concepts should be spatially close (e.g. overlapping) with the important features of the dataset. **Importance correctness**: extracted *aligned* concepts should be scored as important (e.g. 1.0), and *unaligned* concepts as unimportant (e.g. 0.0).

With the ideal case in mind, we propose a validation procedure that aims to evaluate representation and importance correctness of a CE technique. First, we create a set of synthetic datasets, including masks for the base components of the images. Second, we train a set of models for each dataset. Third, we execute the CE method. Fourth, we compute localization masks for each extracted concept for each dataset image. Fifth, we associate each concept to the ground truth masks using a distance metric. This allows us to classify the concepts as aligned or not. Finally, we quantify the **importance correctness** and **representation correctness** of the extracted concepts.

4.1 Synthetic datasets

Our validation process requires unbiased image classification datasets that can be learned by different models and have an annotated ground truth locating each element of the images. To mitigate the risk of introducing biases, or uncontrolled features in the datasets, we focus on the low-complexity task of character classification. We created six synthetic datasets for the validation of ECLAD (e.g., dataset AB in Figure 2), described in detail in Appendix.

We generate the images of each classification task by overlapping multiple elements. Each element (e.g., red A, a gray background) is the combination of multiple features (e.g., "is red", "has the form of an A"), and is generated by a mask (denoted *primitive*) filled with a specific texture. In each dataset, we select a subset of features and their primitives to define each class, marking them as



(a) Class A. (b) Class B. (c) Primitives p_1, p_2, p_3, and p_4.

Figure 2: Synthetic dataset AB. 2(a) and 2(b) show examples of classes *A* and *B*, respectively. 2(c) Shows example masks $m_{x_i}^{p_o}$ for the primitives *A*, *plus*, *background* and *B* respectively.

important, and creating the labels. Unimportant features are balanced between classes. Thus, each datapoint is composed of an image x_i, their label y_i, and a mask $m_{x_i}^{p_o}$ for each primitive p_o.

4.2 Model training

The main problem faced in the validation of CE techniques is that there are no ground truth regarding what a models learns. thus, we assume that models perfectly learns simple datasets. To mitigate possible biases, we train models with multiple architectures and random seeds for each dataset.

4.3 Concept extraction

We perform CE over every trained model and its dataset. As a result of this step, every extracted concept must have at least a related importance score and vector representation. In the case of ACE-based methods, this vector representation is the *concept activation vector* (CAV) (Kim et al., 2018). Similarly, the execution of ConceptShap yields a vector representation akin to CAVs. In our case, ECLAD provides a centroid γ_c associated with every concept. The representation vectors will be used for concept localization, and the importance scores will be used for correctness quantification. As long as these two are provided (importance score and vector representation), a CE technique can be validated using the proposed process.

4.4 Concept localization

We use the results of the previous step to localize all concepts present in the synthetic datasets. For ACE and ConceptShap, we perform concept localization by segmenting each image and testing whether each patch contains each concept. For ECLAD, we use the descriptor d_{x_i} of every image, and compute a mask $m_{x_i}^{c_j}$ for every concept. As a result of this step, every tested CE approach generates a binary mask $m_{x_i}^{c_j}$ for each concept c_j of each model, for every image x_i in a dataset E.

4.5 Concept association

The process of associating concepts and the important features of a dataset has previously been performed through human inspection (Ghorbani et al., 2019; Ge et al., 2021). This association allows the comparison of extracted concepts and the dataset's intended features. It allows for a subjective judgement of the correctness of the CE methods. To perform this association automatically, we introduce the distance DST_{p_o,c_j} measuring how close a concept is to the features of a dataset.

To measure the *spatial association* of a concept and a feature, we consider partial overlapping as well as spatial closeness. We compute this distance through the comparison of the concept masks $m_{x_i}^{c_j}$ and the primitives $m_{x_i}^{p_o}$ of the features. In cases where a concept detects the surrounding of a primitive (when activation maps become off-centered through a CNN), existing metrics (e.g. Jaccard index, adjusted rand score) perform poorly, this limitations are discussed in F.6. We propose an expressive metric, computed by adding the euclidean distance between each pixel on a mask to the nearest element of another mask. This metric results in a zero value if the mask of the primitive and the concept are overlapping, and increases as the masks separate. We compute a one-way distance between the mask $m_{x_i}^{p_o}$ of a primitive p_o and the mask $m_{x_i}^{c_j}$ of a concept c_j as $\text{dst}_{p_o,c_j}(x_i) = \text{sum}\left(m_{x_i}^{p_o} \odot \text{EDT}(\overline{m}_{x_i}^{c_j})\right)$, where $\text{EDT}()$ refers to the euclidean distance transform; $\overline{m}_{x_i}^{c_j}$ is the negated mask $m_{x_i}^{c_j}$; and $\odot$ denotes the element-wise multiplication of matrices. We estimate the association distance between c_j and p_o by computing the average two-way distance for E,

$$\mathrm{DST}_{p_o,c_j} = \frac{1}{|E|} \sum_{x_i \in E} \mathrm{dst}_{p_o,c_j}(x_i) + \mathrm{dst}_{c_j,p_o}(x_i). \tag{4}$$

Using this distance, we can associate each concept to its closest primitive, $p_{c_j} = \underset{p_o}{\mathrm{argmin}}(\mathrm{DST}_{p_o,c_j})$. Finally, we can use this association to classify each concept as *aligned* if p_{c_j} is an important primitive and $\mathrm{DST}_{p_{c_j},c_j}$ is below a defined threshold t_{DST}, or otherwise as *unaligned*. The usage of t_{DST} reduces the number of aligned concepts lacking semantic meaning.

4.6 Correctness quantification

We quantify the correctness of the CE method using two proxy metrics based on the desired behavior for representation and importance correctness. We compute the **representation correctness**, as the negative average association distance of all aligned concepts extracted from all models:

$$\mathrm{RC}_{\mathrm{CE}} = \frac{1}{|C_a|} \sum_{c_j \in C_a} -\mathrm{DST}_{p_{c_j},c_j}\,, \tag{5}$$

where C_a denotes the set of all aligned concepts extracted from all models in the validation process. In an ideal case, the value of $\mathrm{RC}_{\mathrm{CE}}$ would be zero, meaning that there is a subset of extracted concepts which correctly represents the important features of the datasets learned by the model.

To quantify the **importance correctness**, we compute the average absolute importance of all aligned concepts C_a minus the average absolute importance for the unaligned concepts C_u. We then normalize by the maximum importance of all concepts:

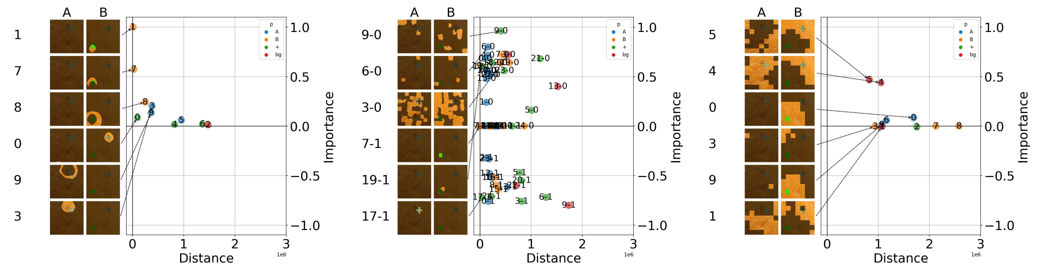
$$\mathrm{IC}_{\mathrm{CE}} = \frac{1}{\max\limits_{c_q \in C_a \cup C_u} \left(|\mathrm{I}_{c_q}|\right)} \left(\frac{1}{|C_a|} \sum_{c_j \in C_a} |\mathrm{I}_{c_j}| - \frac{1}{|C_u|} \sum_{c_j \in C_u} |\mathrm{I}_{c_j}| \right), \tag{6}$$

where $|\mathrm{I}_{c_j}|$ refers to the absolute value of the importance of c_j. In the case of ECLAD, we use the relative importance score $\mathrm{I}_{c_j} = \mathrm{RI}_{c_j}$. For ConceptShap, we use the Shapley values associated with each concept. Finally, for ACE, we scale the sensitivity score, $\mathrm{I}_{c_j} = 2 \times \mathrm{TCAV}_Q - 1$, so that unimportant concepts have a value of 0, and important concepts have a value of 1 or -1.

5 Results

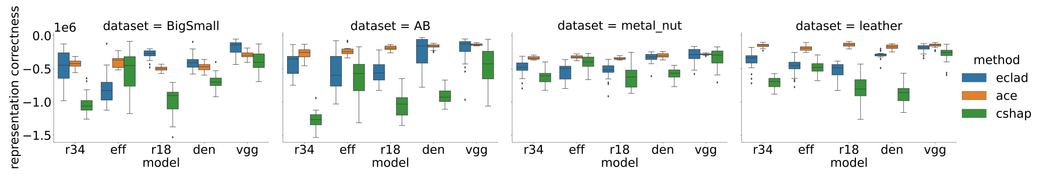
In this section, we present experimental results for our method ECLAD, in comparison with ACE and ConceptShap, through five CNN architectures (ResNet-18, ResNet-34, DenseNet-121, EfficientNet-B0, and VGG16 (He et al., 2015; Huang et al., 2016; Tan & Le, 2019; Simonyan & Zisserman, 2015)), each of which is trained using 20 random seeds for six synthetic datasets, as well as two industrial datasets (subsets of the MVTec-AD dataset (Bergmann et al., 2021)). We compare the performance of the three algorithms using the representation and importance correctness metrics introduced in Section 4. We first focus on the results for a single ResNet-34 model (Figure 3) trained for the AB synthetic dataset. We then provide a comparison of $\mathrm{RC}_{\mathrm{CE}}$ and $\mathrm{IC}_{\mathrm{CE}}$ (for all models and random seeds) for four datasets, as shown in Figure 4. In the following, we report the key findings using representative results, while further details on the setup and results are provided in the appendix.

Eclad achieves a high representation correctness while maintaining their relation to the local representations within the latent space of the models. In Figure 3, we can see how the aligned concepts from ECLAD (c_1 and c_3) have a smaller association distance as well as higher importance scores in comparison to those from ACE (c_{7-1}) and ConceptShap (c_4 and c_0). ACE concepts are extracted using the segmentation algorithm SLIC, which allows a more precise segmentation of the image features (as seen in Figure 4(a)). Yet, said process disregards local activations, and in turn affects the scoring of aligned concepts. This effect is seen for concept c_{7-1} in Figure 3(b), where a perfectly segmented character **B** is scored as non important. Similarly, the optimization process of learning concepts in ConceptShap, can lead to issues when extracting correlated concepts (e.g., inversion c_5, or redundancy c_0-c_9 in Figure 3(c)).

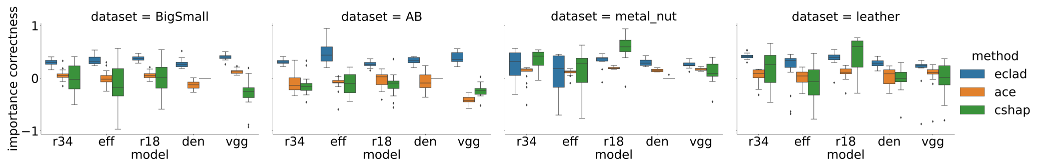


(a) Concepts ECLAD (b) Concepts ACE (c) Concepts ConceptShap

Figure 3: Concepts extracted from a Resnet34 trained on the AB dataset. The extracted concepts from each CE method are plotted in relation to their importance (y-axis), and the distance (x-axis) towards their closest primitive (hue). In an ideal case, important concepts will be closely related to the important primitives (e.g., c_1 in 3(a)), while concepts unrelated to important primitives will be scored as unimportant (e.g., c_0 3(a)).



(a) Comparison of representation correctness.



(b) Comparison of importance correctness.

Figure 4: Comparison of representation correctness 4(a) and importance correctness 4(b) for five models trained on four representative datasets (two synthetic and two real-world use cases). An ideal CE method will have a representation correctness (negative distance between aligned concepts and important primitives) close to zero, and an importance correctness (relative difference between the importance of aligned and unaligned concepts) close to one. In both 4(a) and 4(b), higher is better.

ECLADs importance scoring allows for a better differentiation between aligned and unaligned concepts. This phenomenon can be seen in Figure 3, where the most important concepts extracted by ECLAD (c_1 and c_7) are closely associated with the features that should be important for the model. In contrast, TCAV scores (ACE), assume that each concept is encoded in a distinctive direction in the latent space of a network. This assumption can entangle spurious patterns with a concept (e.g., random patches and the character **A** in c_{3-0} in Figure 3(b)). In the case of ConceptShap, the Shapely value of inversely correlated concepts (e.g., characters **A** and **B**), can be truncated, independent of the extent to which a regions become activated or actually contribute to a prediction (e.g., c_4 and c_9 in Figure. 3(c)). Globally, this is reflected in ECLAD having a larger importance correctness score for most datasets and model combinations, as seen in Figure 4(b).

Two counterintuitive results appear in Figure 4. First, in most cases ACE outperformed ECLAD and ConceptShap representation correctness, yet underperformed w.r.t. ECLAD in importance correctness. This result is caused by the crisp segmentation provided by SLIC. In contrast the diffusion of information across the network leads to coarse concepts in the case of ECLAD and ConceptShap. Second, the industrial datasets (metal nut and leather) lacked annotations for other non-important cues (e.g., center of a metal nut, standard edge of the nut). This complicates a clean assessment of which concepts are associated to important features and which are not. This demonstrates the importance of having confounding factors annotated when testing CE techniques.

In real-world use cases, ECLAD provides insightful explanations for understanding CNNs. To test the performance of ECLAD in real-world use cases, we trained a DenseNet-121 on datasets for concrete crack (Özgenel & Sorguç, 2018), metal casting defects (Dabhi, 2020), and diabetic retinopathy classification (Society, 2019). The extracted concepts with importance scores higher than 0.8 are shown in Figure 5. In the three domains, the most important extracted concepts can be associated with the main cues expected from each dataset (cracks in Fig. 5(a), pinholes in Fig. 5(b), and exudates-aneurysms in Fig. 5(c) and 5(d)).

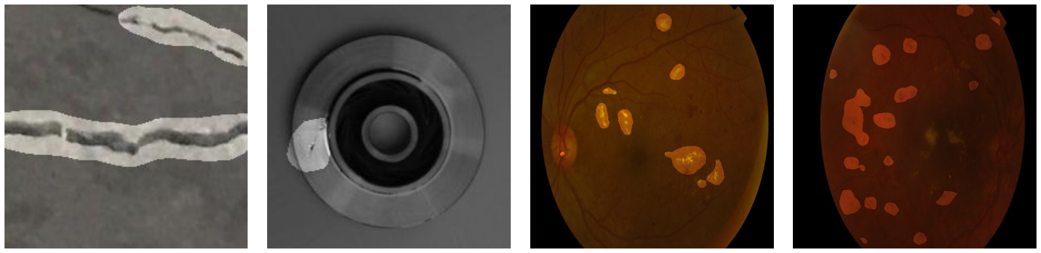

(a) Concrete c_1 (cracks) (b) Casting c_1 (pinholes) (c) APTOS c_1 (exudates) (d) APTOS c_2 (aneurysm)

Figure 5: Most important concepts ($\mathrm{RI}_{c_j} > 0.8$) extracted with ECLAD from a DenseNet-121 trained for the classification of concrete cracks 5(a), metal casting defects 5(b), and diabetic retinopathy 5(c), 5(d). These examples show that ECLAD is able to extract and localize important/meaningful concepts related to the most important visual cues of each task, being scratches 5(a), pinholes 5(b), exudates 5(c), and micro-aneurysm 5(d) respectively.

An ablation study was performed to explore the impact of key components of ECLAD, obtaining key insights. First, using a single layer for concept extraction highly depends on the depth of the chosen layer. In comparison, by combining layers from multiple depths, ECLAD allows the extraction of mid and high level concepts without the complexity of fine-tuning the selected layer. Second, more than three layers help compensate halo effects on extracted concepts (representations dilate through the network), as well as mid level concepts which are not present in higher layers. Third, higher number of concepts will cause a progressive slicing of important concepts (without affecting their RI_{c_j}) Finally, using coarse interpolation methods (f_U) will impact the boundaries of the extracted concepts, but not the concepts themselves. The chosen parameters for the presented analysis are a balance between performance and computational cost. The complete details on the ablation study are presented on the appendix F.5.

6 Conclusions

We propose ECLAD as a concept extraction (CE) technique, based on local aggregate descriptors (LADs). Our algorithm focuses on how CNNs represent pixels internally, allowing a more reliable CE and importance scoring. In addition, it provides the novel ability to localize, in new instances, which regions of an image contain the visual cues related to each concept.

As an orthogonal contribution, we propose an automatic validation process for CE techniques, which minimizes the need for human intervention. Our validation process is based on two novel metrics, measuring the importance and visual cues of concepts w.r.t. the ground truth of synthetic datasets. We provide six new synthetic datasets that can be used for testing CE methods. The proposed datasets and validation method proved effective in comparing ECLAD, ACE, and ConceptShap. As validation procedures that forego (subjective) human judgement are largely missing in the area, we hope our contribution becomes helpful, provides a quantitative approach for evaluating CE.

While ECLAD performed reliably in the studied cases, the results also raise relevant questions for future research. First, during the initial CE, ECLAD can be more computationally expensive than CAV based methods, as the base clustering is performed over the representations of pixels and not images or patches. Second, the localization of concepts in new images strongly depends on the CNN architecture being studied, as not all CNNs represent local information with the same fidelity. Finally, for more complex tasks with a higher number of features, the number of extracted concepts will have to be adjusted accordingly to avoid relevant features being clustered together.

REFERENCES

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 9525–9536, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/294a8ed24b1ad22ec2e7efea049b8737-Abstract.html.

Alejandro Barredo Arrieta, Natalia Díaz Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (XAI): concepts, taxonomies, opportunities and challenges toward responsible AI. *Inf. Fusion*, 58:82–115, 2020. doi: 10.1016/j.inffus.2019.12.012. URL https://doi.org/10.1016/j.inffus.2019.12.012.

Stan Benjamens, Pranavsingh Dhunnoo, and Bertalan Meskó. The state of artificial intelligence-based fda-approved medical devices and algorithms: an online database. *npj Digital Medicine*, 3 (1):118, Sep 2020. ISSN 2398-6352. doi: 10.1038/s41746-020-00324-0.

Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *Int. J. Comput. Vis.*, 129(4):1038–1059, 2021. doi: 10.1007/s11263-020-01400-4. URL https://doi.org/10.1007/s11263-020-01400-4.

Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José M. F. Moura, and Peter Eckersley. Explainable machine learning in deployment. In Mireille Hildebrandt, Carlos Castillo, L. Elisa Celis, Salvatore Ruggieri, Linnet Taylor, and Gabriela Zanfir-Fortuna (eds.), *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*, pp. 648–657. ACM, 2020. doi: 10.1145/3351095.3375624. URL https://doi.org/10.1145/3351095.3375624.

Nadia Burkart and Marco F. Huber. A survey on the explainability of supervised machine learning. *J. Artif. Intell. Res.*, 70:245–317, 2021. doi: 10.1613/jair.1.12228. URL https://doi.org/10.1613/jair.1.12228.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan Su. This looks like that: Deep learning for interpretable image recognition. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8928–8939, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/adf7ee2dcf142b0e11888e72b43fcb75-Abstract.html.

Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nat. Mach. Intell.*, 2(12):772–782, 2020. doi: 10.1038/s42256-020-00265-z. URL https://doi.org/10.1038/s42256-020-00265-z.

R Dabhi. Casting product image data for quality inspection. *Kaggle.com*, 2020.

Shipi Dhanorkar, Christine T. Wolf, Kun Qian, Anbang Xu, Lucian Popa, and Yunyao Li. Who needs to know what, when?: Broadening the explainable AI (XAI) design space by looking at explanations across the AI lifecycle. In Wendy Ju, Lora Oehlberg, Sean Follmer, Sarah E. Fox, and Stacey Kuznetsov (eds.), *DIS '21: Designing Interactive Systems Conference 2021, Virtual Event, USA, 28 June, July 2, 2021*, pp. 1591–1602. ACM, 2021. doi: 10.1145/3461778.3462131. URL https://doi.org/10.1145/3461778.3462131.

Yunhao Ge, Yao Xiao, Zhi Xu, Meng Zheng, Srikrishna Karanam, Terrence Chen, Laurent Itti, and Ziyan Wu. A peek into the reasoning of neural networks: Interpreting with structural visual concepts. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pp. 2195–2204. Computer Vision Foundation / IEEE, 2021. URL https://openaccess.thecvf.com/content/CVPR2021/html/Ge_A_Peek_

Into_the_Reasoning_of_Neural_Networks_Interpreting_With_CVPR_2021_paper.html.

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard S. Zemel, Wieland Brendel, Matthias Bethge, and Felix A. Wichmann. Shortcut learning in deep neural networks. *Nat. Mach. Intell.*, 2(11):665–673, 2020. doi: 10.1038/s42256-020-00257-z. URL https://doi.org/10.1038/s42256-020-00257-z.

Amirata Ghorbani, James Wexler, James Y. Zou, and Been Kim. Towards automatic concept-based explanations. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 9273–9282, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/77d2afcb31f6493e350fca61764efb9a-Abstract.html.

Yash Goyal, Uri Shalit, and Been Kim. Explaining classifiers with causal concept effect (cace). *CoRR*, abs/1907.07165, 2019. URL http://arxiv.org/abs/1907.07165.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In Jennifer G. Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2673–2682. PMLR, 2018. URL http://proceedings.mlr.press/v80/kim18d.html.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5338–5348. PMLR, 2020. URL http://proceedings.mlr.press/v119/koh20a.html.

Avinash Kori, Parth Natekar, Ganapathy Krishnamurthi, and Balaji Srinivasan. Abstracting deep neural networks into concept graphs for concept level interpretability. *CoRR*, abs/2008.06457, 2020. URL https://arxiv.org/abs/2008.06457.

Yu-Han Liu and Sercan Ö. Arik. Explaining deep neural networks using unsupervised clustering. *CoRR*, abs/2007.07477, 2020. URL https://arxiv.org/abs/2007.07477.

P Mallikarjuna, Alireza Tavakoli Targhi, Mario Fritz, Eric Hayman, Barbara Caputo, and Jan-Olof Eklundh. The kth-tips2 database. *Computational Vision and Active Perception Laboratory, Stockholm, Sweden*, 11, 2006.

Shane T. Mueller, Robert R. Hoffman, William J. Clancey, Abigail Emrey, and Gary Klein. Explanation in human-ai systems: A literature meta-review, synopsis of key ideas and publications, and bibliography for explainable AI. *CoRR*, abs/1902.01876, 2019. URL http://arxiv.org/abs/1902.01876.

Ç F Özgenel and A Gönenç Sorguç. Performance comparison of pretrained convolutional neural networks on crack detection in buildings. In *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*, volume 35, pp. 1–8. IAARC Publications, 2018.

Zhongang Qi, S. Khorram, and Fuxin Li. Visualizing deep networks by optimizing with integrated gradients. In *CVPR Workshops*, 2019.

Jessica Schrouff, Sebastien Baur, Shaobo Hou, Diana Mincu, Eric Loreaux, Ralph Blanes, James Wexler, Alan Karthikesalingam, and Been Kim. Best of both worlds: local and global explanations with human-understandable concepts. *CoRR*, abs/2106.08641, 2021. URL `https://arxiv.org/abs/2106.08641`.

D. Sculley. Web-scale k-means clustering. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti (eds.), *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pp. 1177–1178. ACM, 2010. doi: 10.1145/1772690.1772862. URL `https://doi.org/10.1145/1772690.1772862`.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vis.*, 128(2):336–359, 2020. doi: 10.1007/s11263-019-01228-7. URL `https://doi.org/10.1007/s11263-019-01228-7`.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3145–3153. PMLR, 2017. URL `http://proceedings.mlr.press/v70/shrikumar17a.html`.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.1556`.

Amitojdeep Singh, Sourya Sengupta, and Vasudevan Lakshminarayanan. Explainable deep learning models in medical image analysis. *J. Imaging*, 6(6):52, 2020. doi: 10.3390/jimaging6060052. URL `https://doi.org/10.3390/jimaging6060052`.

Asia Pacific Tele-Ophthalmology Society. Aptos 2019 blindness detection. *Kaggle.com*, 2019.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114. PMLR, 2019. URL `http://proceedings.mlr.press/v97/tan19a.html`.

Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (XAI): toward medical XAI. *IEEE Trans. Neural Networks Learn. Syst.*, 32(11):4793–4813, 2021. doi: 10.1109/TNNLS.2020.3027314. URL `https://doi.org/10.1109/TNNLS.2020.3027314`.

Thien Q. Tran, Kazuto Fukuchi, Youhei Akimoto, and Jun Sakuma. Unsupervised causal binary concepts discovery with VAE for black-box model explanation. *CoRR*, abs/2109.04518, 2021. URL `https://arxiv.org/abs/2109.04518`.

Lev V. Utkin, Pavel D. Drobintsev, Maxim Kovalev, and Andrei V. Konstantinov. Combining an autoencoder and a variational autoencoder for explaining the machine learning model predictions. In *28th Conference of Open Innovations Association, FRUCT 2021, Moscow, Russia, January 27-29, 2021*, pp. 489–494. IEEE, 2021. doi: 10.23919/FRUCT50888.2021.9347612. URL `https://doi.org/10.23919/FRUCT50888.2021.9347612`.

Tian Wang, Yang Chen, Meina Qiao, and Hichem Snoussi. A fast and robust convolutional neural network-based defect detection model in product quality control. *The International Journal of Advanced Manufacturing Technology*, 94(9):3465–3471, February 2018.

Maleakhi A. Wijaya, Dmitry Kazhdan, Botty Dimanov, and Mateja Jamnik. Failing conceptually: Concept-based explanations of dataset shift. *CoRR*, abs/2104.08952, 2021. URL `https://arxiv.org/abs/2104.08952`.

Chih-Kuan Yeh, Been Kim, Sercan Ömer Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. In

Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/ecb287ff763c169694f682af52c1f309-Abstract.html`.

A APPENDIX

The appendices of this work contain extended information pertaining to three principal topics. First, we provide a pseudocode of the proposed method. Second, we discuss in detail the experimental setup required to perform the experiments C. Third, we describe the new synthetic datasets and provide extended results for each one D. Foruth, we describe the real-world datasets used for both quantitative validation and testing in other domains E. Fifth, we discuss the computational cost of ECLAD, ACE, and ConceptShap. Sixth, We describe the ablation study performed over key components of ECLAD. Finally, we compare the proposed association distance with other existing alternatives.

B PSEUDOCODE

ECLAD is designed to be first executed over a complete dataset to generate a global explanation. The resulting set of centroids Γ can then be used to localize each concept for new input images. The global execution of ECLAD is described in Algorithm 1.

Algorithm 1 ECLAD (global)

```
Require: model $f$, dataset $E$, number of output classes $n_{\mathrm{k}}$, layers $L$, number of concepts $n_{\mathrm{c}}$, mini
    batch size $n_{\mathrm{i}}$, mask attenuation $\lambda$
 1: $D \leftarrow \mathrm{GetLADs}(f, E, L)$
 2: $\Gamma \leftarrow \mathrm{MiniBatchKmeans}(D, n_{\mathrm{i}}, n_{\mathrm{c}})$
 3: for $\gamma_{c_j} \in \Gamma$ do
 4:     $\varepsilon_{c_j} \leftarrow \{(1-\lambda)(\mathrm{Mask}(x_i, \gamma_{c_j}) \odot x_i) + \lambda x_i \mid x_i \in E\}$
 5:     for $k \in \{1, 2, \ldots, n_{\mathrm{k}}\}$ do
 6:         create $S_{c_j}^{k,E_k}, S_{c_j}^{k,E_{k'}}$
 7:         compute $\mathrm{CS}_{c_j}^{k}$
 8:     end for
 9:     compute $\mathrm{RI}_{c_j}$
10: end for
11: return $\{(\gamma_{c_j}, \varepsilon_{c_j}, \mathrm{RI}_{c_j}) \mid \gamma_{c_j} \in \Gamma\}$
```

As a result of executing ECLAD, we obtain for each concept c_j: a centroid γ_{c_j} which serves as an anchor; an example set ε_{c_j} of human-understandable visualizations; and a relative importance score RI_{c_j}, which describes how important each concept is for the overall predictions of the model.

We perform the localization of each concept c_j in a new image by extracting the mask $m_{x_i}^{c_j}$, for each concept defined by $\gamma_{c_j} \in \Gamma$. The resulting masks serve as an explanation of where the different concepts are located. In addition, the average sensitivities of all pixels in an image belonging to a concept can be used as local measures of importance.

C EXPERIMENTAL SETUP

We performed all experiments in servers with Intel® Xeon® Gold 6330 CPU and a NVIDIA A100 GPU. We implemented ECLAD, ACE, and ConceptShap using Pytorch 1.11, and the different model architectures using the *PyTorch Image Models* (TIMM) library. As part of the supplementary material, we make available the code of the experiments, as well as the created datasets under an MIT license. Both items available on the link: https://drive.google.com/drive/folders/16CjAvk8H1VAD2-rNiy0HV3OmzDlrwXo5?usp=sharing

Analyzed model architectures. During the experiments, we trained and analyzed five different CNN architectures (ResNet-18 (He et al., 2015), ResNet-34 (He et al., 2015), DenseNet-121 (Huang et al., 2016), EfficientNet-B0 (Tan & Le, 2019), and VGG16 (Simonyan & Zisserman, 2015)). For each model, we selected four layers for executing ECLAD, and the last one (l_4) was used for ConceptShap. For ACE, we used the output of the average pooling before the fully connected layers of each model, as advised in TCAV (Kim et al., 2018). The list of layers $L = l_1, l_2, l_3, l_4$, and l_{tcav} for each model are provided in Table 1.

Table 1: Synthetic datasets.

MODEL	LAYERS	LAYER
RESNET-18	l_1	LAYER 1, BLOCK 1, RELU 2
RESNET-18	l_2	LAYER 2, BLOCK 1, RELU 2
RESNET-18	l_3	LAYER 3, BLOCK 1, RELU 2
RESNET-18	l_4	LAYER 4, BLOCK 1, RELU 2
RESNET-18	l_{tcav}	AVERAGE POOLING BEFORE FC LAYERS
RESNET-34	l_1	LAYER 1, BLOCK 2, RELU 2
RESNET-34	l_2	LAYER 2, BLOCK 3, RELU 2
RESNET-34	l_3	LAYER 3, BLOCK 5, RELU 2
RESNET-34	l_4	LAYER 4, BLOCK 2, RELU 2
RESNET-34	l_{tcav}	AVERAGE POOLING BEFORE FC LAYERS
DENSENET-121	l_1	TRANSITION LAYER 1, CONV
DENSENET-121	l_2	TRANSITION LAYER 2, CONV
DENSENET-121	l_3	TRANSITION LAYER 3, CONV
DENSENET-121	l_4	DENSE BLOCK 4, DENSE LAYER 16, CONV 2
DENSENET-121	l_{tcav}	AVERAGE POOLING BEFORE FC LAYERS
EFFICIENTNET-B0	l_1	BLOCK 3, INVERTED RESIDUAL 2, CONV_PWL
EFFICIENTNET-B0	l_2	BLOCK 4, INVERTED RESIDUAL 2, CONV_PWL
EFFICIENTNET-B0	l_3	BLOCK 5, INVERTED RESIDUAL 3, CONV_PWL
EFFICIENTNET-B0	l_4	BLOCK 6, INVERTED RESIDUAL 0, CONV_PWL
EFFICIENTNET-B0	l_{tcav}	AVERAGE POOLING BEFORE FC LAYERS
VGG16	l_1	MAXPOOLING AFTER CONV 2-2
VGG16	l_2	MAXPOOLING AFTER CONV 3-3
VGG16	l_3	MAXPOOLING AFTER CONV 4-3
VGG16	l_4	MAXPOOLING AFTER CONV 5-3
VGG16	l_{tcav}	AVERAGE POOLING BEFORE FC LAYERS

Model training. We performed the model training using an SGD optimizer with a learning rate of 0.1 and momentum of 0.9. We used a *reduce lr on plateau* scheduler with a factor of 0.1 based on the *negative log likelihood loss* of the models. The data was split into 0.85 for training and 0.15 for testing, with mini-batches of 24 images sampled and balanced between the classes. In addition, we used random color jitter, and affine transforms for data augmentation.

ECLAD. We perform all ECLAD analysis with the same sets of parameters. Each execution was performed using a maximum of 200 images from each class, extracting 10 concepts, and using 2 images per clustering minibatch (100352 LADs). The layers used for each model are shown in Table 1.

ACE. We perform all ACE analysis with the same parameters used by Ghorbani et al. (2019). We performed a SLIC segmentation over 20 images of each class, with sigma of 1.0 and compactness of 20.0 for 15, 50, and 80 segments. Subsequently, we resized and padded each patch before evaluating it in a model to extract the activation map of the selected layer. We then extracted 25 clusters using k-means over the flattened activation maps. Finally, we used each group of clustered patches to perform 50 repetitions of TCAV and obtain the TCAV_Q score of each concept.

ConceptShap. We perform all ConceptShap analysis with the same sets of parameters. 10 concepts were extracted at each run with $\beta = 1.0 \times 10^{-7}$, $\lambda_1 = 1 \times 10^{-7}$, $\lambda_2 = 2 \times 10^{-7}$. These values were obtained empirically after exploring values in orders of magnitude from 1×10^{-1} to 1×10^{-10}. The Shapely values for the concepts were approximated with Monte-Carlo sampling with $100 \times n_c$ samples. As a cutting threshold for localizing each concept, we used the mean values of the projection of the activation map over the concept vectors, which worked well in comparison to other fixed thresholds.

D SYNTHETIC DATASETS

The six synthetic datasets created for the validation of ECLAD are summarized in the Table 2. All synthetic datasets were created using alphabetical characters and filling them with either solid colors or textures from the KTH-TIPS dataset (Mallikarjuna et al., 2006). Each dataset is composed of 200 RGB images of 224×224 pixels per class.

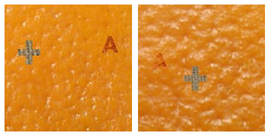
Table 2: Synthetic datasets

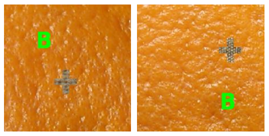
Name	Class 0	Class 1	Primitives
AB	A	B	A, B, +, background
ABplus	A	B	A, B, *, /, #, X, background
Big-Small	Big \emph{B}	Small \emph{B}	Big \emph{B}, Small \emph{B}, +, background
CO	C	O	C, O, +, background
colorGB	B	G	representative character (green or blue), intrusive green character, +, background
isA	isA	notA	A, other characters (B-H), background

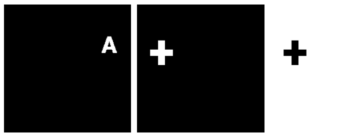
Each subsection contains a description of how the synthetic dataset was created, including example images of each class and the primitives. Similarly, for each dataset, we provide a sample result for each analysis (ECLAD, ACE, ConceptShap) and model trained in said dataset.

D.1 AB DATASET

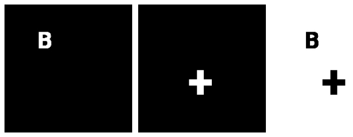
The *AB* dataset corresponds to a simple classification between images containing a character *A* or a character *B*. The character *A*, denoted as the primitive p_1, is filled with a cork texture and only appears in class A. The character *B*, denoted as the primitive p_2, is filled green, and only appears in class B. The primitive p_3 contains a $+$ filled with a cotton texture, which is an irrelevant feature appearing in all the images. Finally, p_4 refers to the background, filled with an orange peel texture. Examples of the class images and the primitives are presented in 6.


(a) Examples class A


(b) Examples class B



(c) Primitives p_1, p_3, and p_4 form class A.

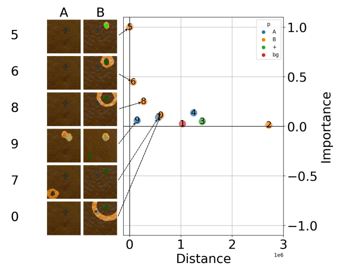


(d) Primitives p_2, p_3, and p_4 from class B.

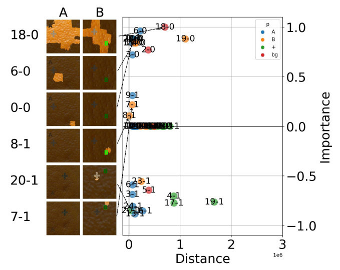
Figure 6: Dataset AB, composed of class A (6(a)) and B (6(b)). Primitives p_1, p_3, and p_4 from class A are shown in Figure 6(c). Primitives p_2, p_3, and p_4 appearing in class B are shown in Figure 6(b).

This simple dataset can be used as a sanity check for CE methods. The characters A and B are different in both form and texture, which facilitates classification. Similarly, regardless of the principle for the classification (form or color), the primitives will still be the same for both base concepts.

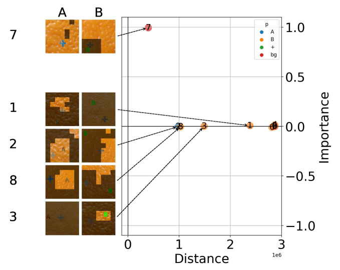
D.1.1 SAMPLE RESULTS FOR EACH MODEL TRAINED WITH THE DATASET AB



(a) Concepts ECLAD

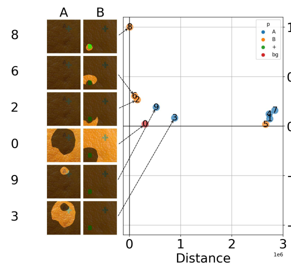


(b) Concepts ACE

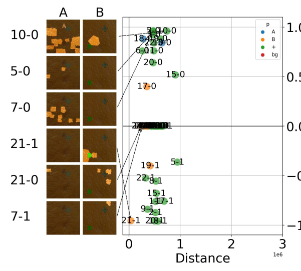


(c) Concepts ConceptShap

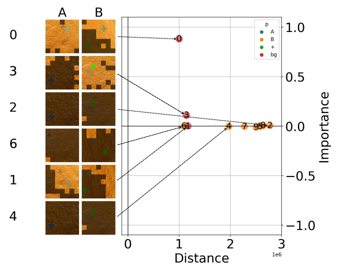
Figure 7: Concepts extracted from a resnet18 trained in the AB dataset.



(a) Concepts ECLAD

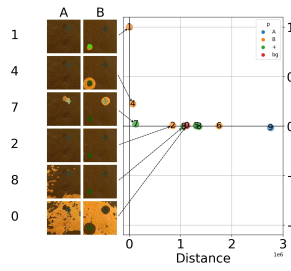


(b) Concepts ACE

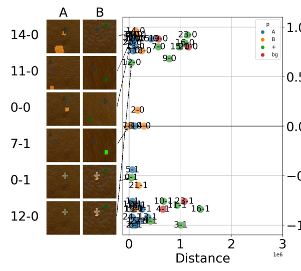


(c) Concepts ConceptShap

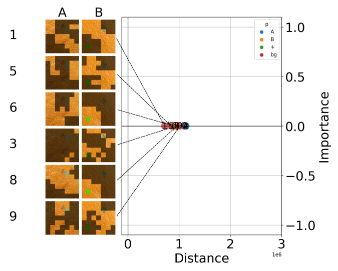
Figure 8: Concepts extracted from a resnet34 trained in the AB dataset.



(a) Concepts ECLAD

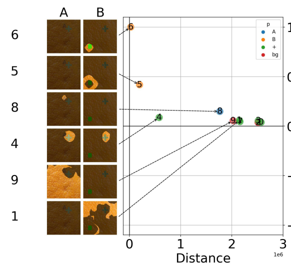


(b) Concepts ACE

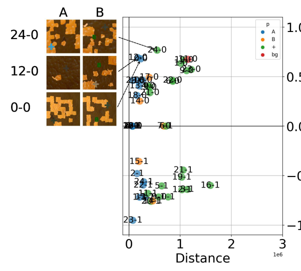


(c) Concepts ConceptShap

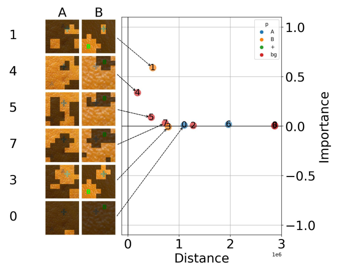
Figure 9: Concepts extracted from a densenet121 trained in the AB dataset.



(a) Concepts ECLAD

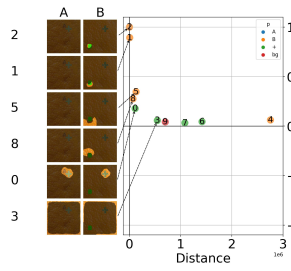


(b) Concepts ACE

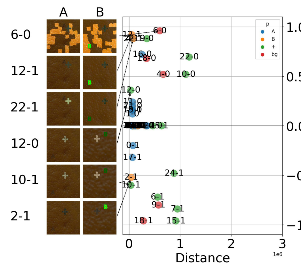


(c) Concepts ConceptShap

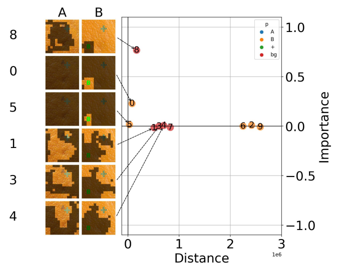
Figure 10: Concepts extracted from a efficientnet-b0 trained in the AB dataset.



(a) Concepts ECLAD



(b) Concepts ACE

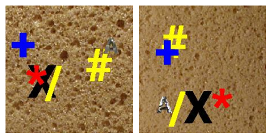


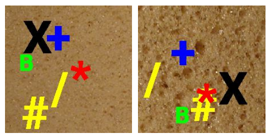
(c) Concepts ConceptShap

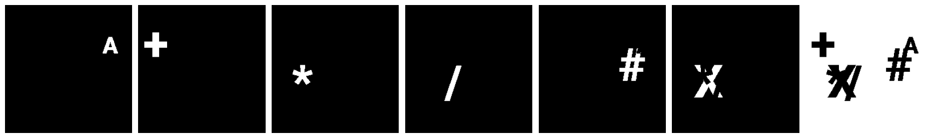
Figure 11: Concepts extracted from a vgg16 trained in the AB dataset.

D.1.2 ABPLUS DATASET

The *ABplus* dataset also consists in the classification of images containing a character *A* or a character *B*, yet, it contains a higher number of intrusive elements in comparison with the dataset *AB*. The primitive for the *A* character is p_1, filled with an aluminum foil texture, appearing only in images from the class A. The primitive for the character *B* is p_2, filled in green, appearing only in images from the class B. The rest of the primitives are balanced between the two classes, and only serve as irrelevant information. Primitives p_3, p_4, p_5, p_6, and p_7 refers to the symbols *, /, #, and **X**, respectively, all filled in solid colors. Finally, p_8 refers to the background, filled with a sponge texture.


(a) Examples class A


(b) Examples class B


(c) Primitives p_1, p_3, p_4, p_5, p_6, p_7, and p_8 from class A.

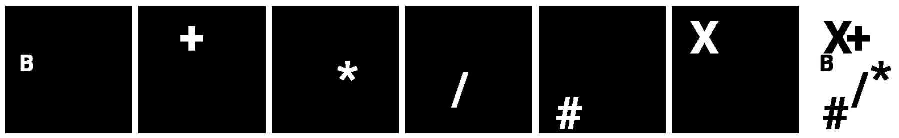
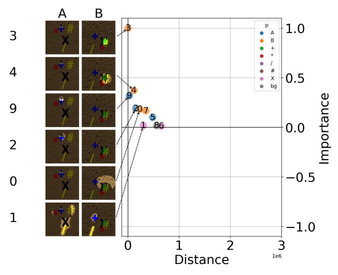

(d) Primitives p_2, p_3, p_4, p_5, p_6, p_7, and p_8 from class B.

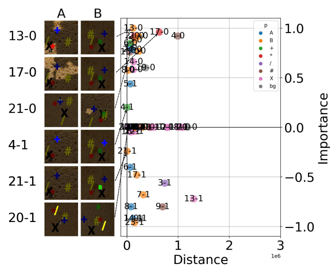
Figure 12: Examples for both classes of the dataset ABplus are shown in Figures 12(a) (class A) and Figure 12(b) (class B). The primitives of class A are shown in Figure 12(c) and the primitives of class B are shown in Figure (12(b)).

This dataset contains multiple irrelevant primitives with high contrast in random positions. This allows to test CE methods to observe and quantify their performance in cases with more feature variety. Thus, although a model may learn representations for some primitives, the irrelevant ones should be scored with low importance.

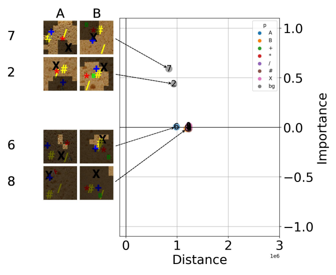
D.1.3 SAMPLE RESULTS FOR EACH MODEL TRAINED WITH THE DATASET ABPLUS



(a) Concepts ECLAD

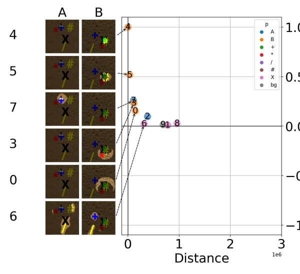


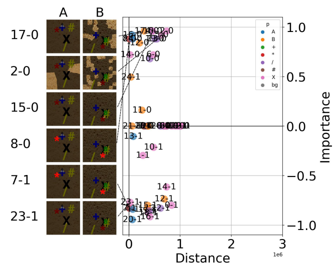
(b) Concepts ACE



(c) Concepts ConceptShap

Figure 13: Concepts extracted from a resnet18 trained in the ABplus dataset.


(a) Concepts ECLAD


(b) Concepts ACE

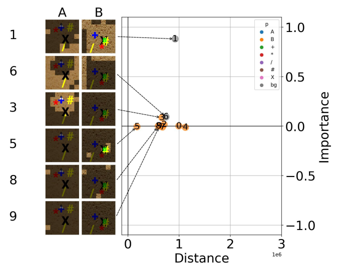
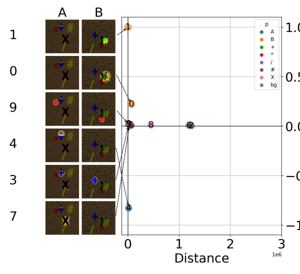
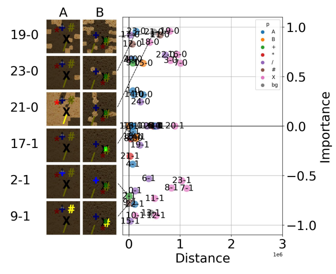

(c) Concepts ConceptShap

Figure 14: Concepts extracted from a resnet34 trained in the ABplus dataset.


(a) Concepts ECLAD


(b) Concepts ACE

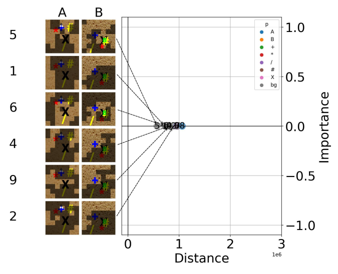
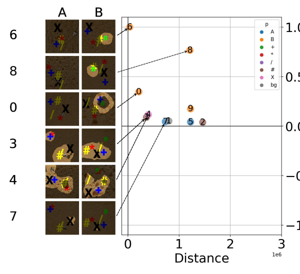
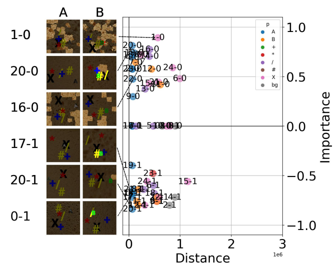

(c) Concepts ConceptShap

Figure 15: Concepts extracted from a densenet121 trained in the ABplus dataset.


(a) Concepts ECLAD


(b) Concepts ACE

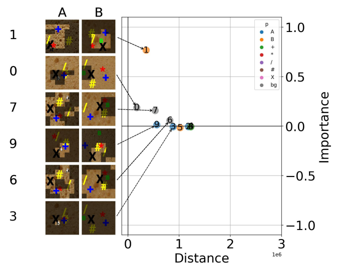
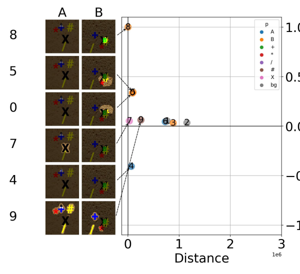
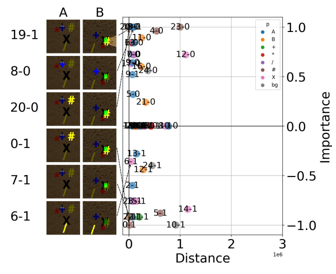

(c) Concepts ConceptShap

Figure 16: Concepts extracted from a efficientnet-b0 trained in the ABplus dataset.


(a) Concepts ECLAD


(b) Concepts ACE

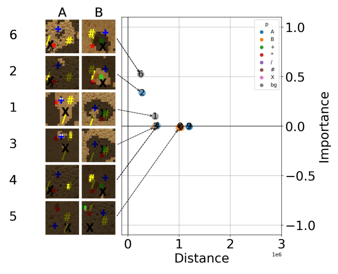
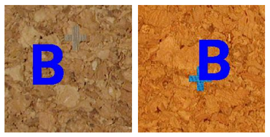
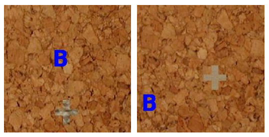

(c) Concepts ConceptShap

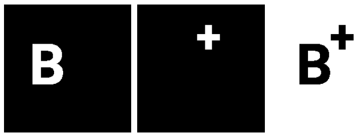
Figure 17: Concepts extracted from a vgg16 trained in the ABplus dataset.

D.1.4 Big-Small dataset

The *Big-Small* dataset, contains images of two classes. The first one, class big has 100 pixels high letters *B* filled in blue (primitive p_1). The second one, class small contains 40 pixels high letters *B* also filled in blue (primitive p_2). The only difference between the two classes is the size of the letter *B*. Primitive p_3, refers to an intrusive character $+$, filled with a cotton textile and balanced between the two classes. Finally, the background of all images is annotated as the primitive p_4, and is filled with a cork texture.


(a) Examples class big


(b) Examples class small


(c) Primitives p_1, p_3, and p_4 from class big.

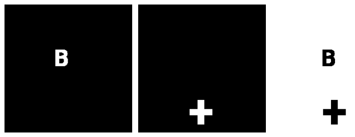
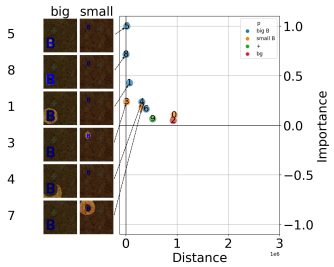
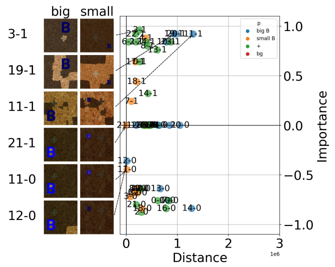

(d) Primitives p_2, p_3, and p_4 from class small.

Figure 18: Big-Small dataset. Examples of class big are shown in Figure 18(a), containing big characters *B*, and examples of the class small are shown in Figure 18(b), containing small *B*s. Class Big contains primitives p_1, p_2, and p_3 (Figure 18(c)); and class small contains primitives p_1, p_2, and p_3 shown in Figure (18(d)).

The Big-Small dataset aims to test how different CE techniques respond to instances where scale is the differentiating factor of classes. This is a common case in real-world applications such as quality control, or medical diagnosis.

D.1.5 Sample results for each model trained with the dataset BigSmall


(a) Concepts ECLAD


(b) Concepts ACE

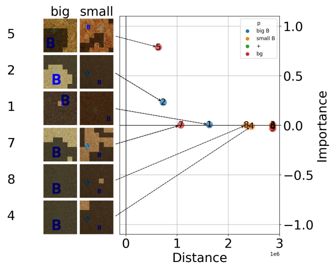
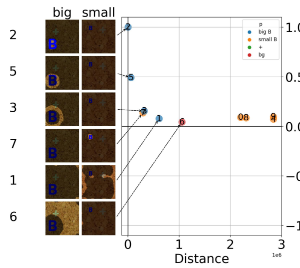
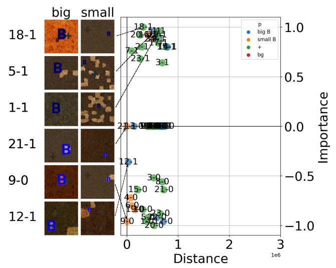

(c) Concepts ConceptShap

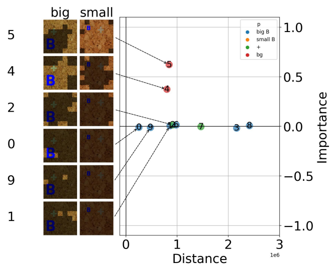
Figure 19: Concepts extracted from a resnet18 trained in the BigSmall dataset.



(a) Concepts ECLAD

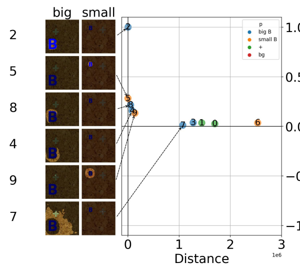


(b) Concepts ACE

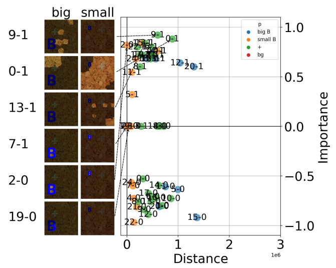


(c) Concepts ConceptShap

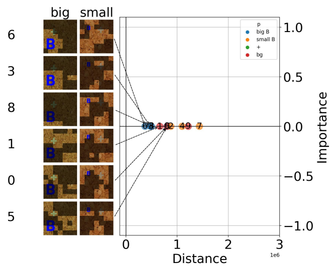
Figure 20: Concepts extracted from a resnet34 trained in the BigSmall dataset.



(a) Concepts ECLAD

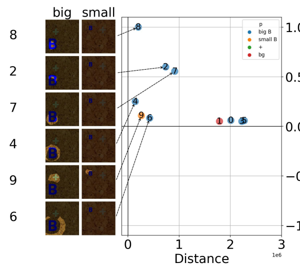


(b) Concepts ACE

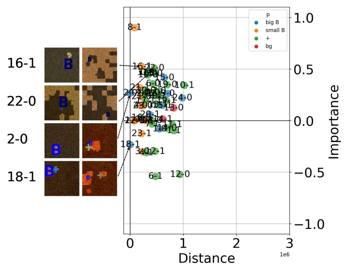


(c) Concepts ConceptShap

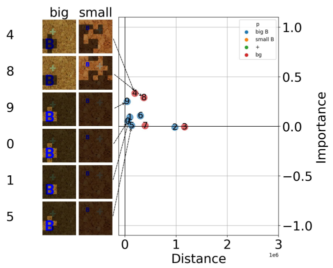
Figure 21: Concepts extracted from a densenet121 trained in the BigSmall dataset.



(a) Concepts ECLAD

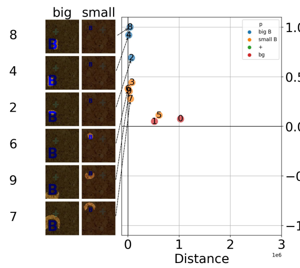


(b) Concepts ACE

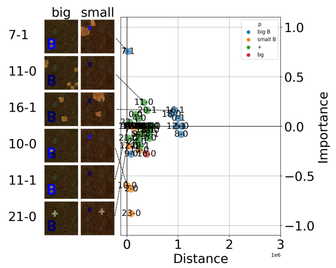


(c) Concepts ConceptShap

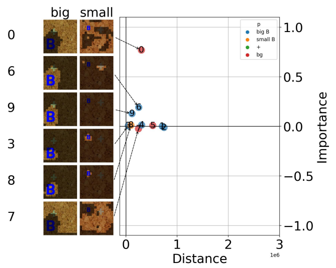
Figure 22: Concepts extracted from a efficientnet-b0 trained in the BigSmall dataset.



(a) Concepts ECLAD



(b) Concepts ACE



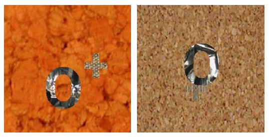
(c) Concepts ConceptShap

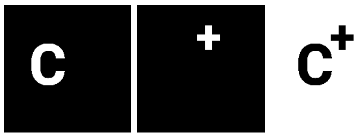
Figure 23: Concepts extracted from a vgg16 trained in the BigSmall dataset.

D.1.6 CO DATASET

The task for the dataset *CO* consists in differentiating images with a character *C* (primitive p_1) in class C, or a character *O* (primitive p_2) in class O. Both p_1 and p_2 are filled with an aluminum foil texture, and their only difference is that the *O* is closed on the right. Primitive p_3 is the character $+$ filled with a cotton texture, which appears in both classes. Similarly p_4 refers to the background, filled with a cork texture.

(a) Examples class C


(b) Examples class O


(c) Primitives p_1, p_3, and p_4 from Class C.

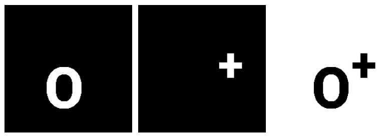
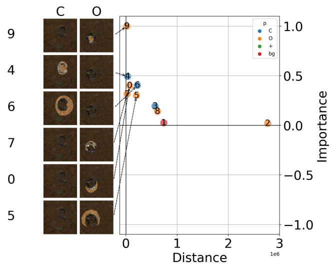

(d) Primitives p_2, p_3, and p_4 from class O.

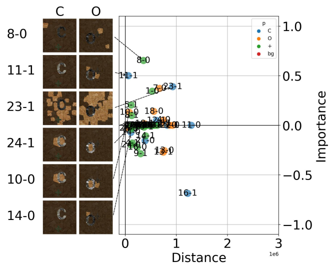
Figure 24: Examples of the dataset CO, where one class are images with the letter *C* (Figure 24(a)), and the other has images containing the letter *O* (Figure 24(b). Primitives for the class C are shown in Figure 24(c), whereas primitives of class O are shown in Figure 24(d).

This dataset was designed to test how CE algorithms perform when dealing with completeness issues. The actual difference between the two classes is the right side of the *O*, which should be the fasts way for CNNs to differentiate the images. This main feature is part of the primitives, but patch extraction techniques may have issues detecting features that are important *by omission*.

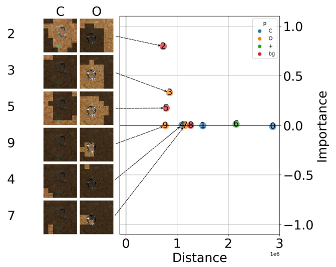
D.1.7 SAMPLE RESULTS FOR EACH MODEL TRAINED WITH THE DATASET CO



(a) Concepts ECLAD

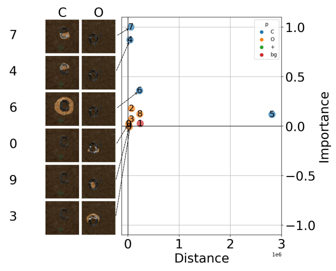


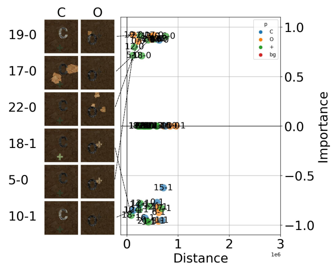
(b) Concepts ACE



(c) Concepts ConceptShap

Figure 25: Concepts extracted from a resnet18 trained in the CO dataset.


(a) Concepts ECLAD


(b) Concepts ACE

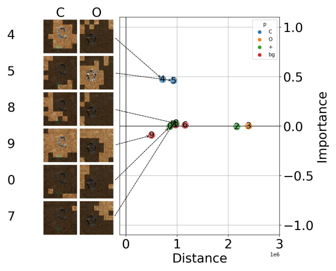
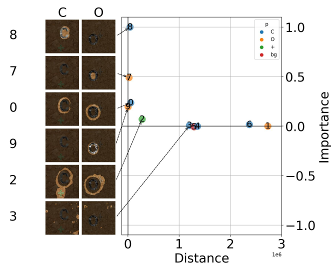
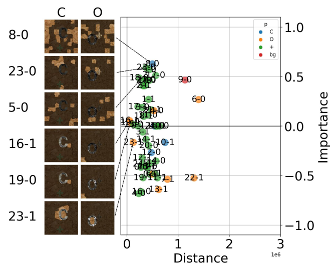

(c) Concepts ConceptShap

Figure 26: Concepts extracted from a resnet34 trained in the CO dataset.


(a) Concepts ECLAD


(b) Concepts ACE

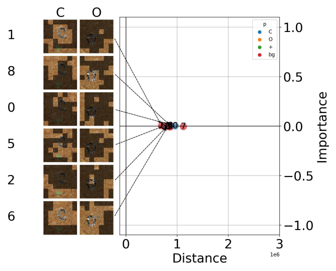
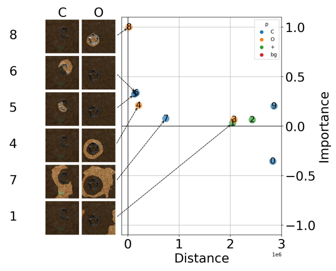
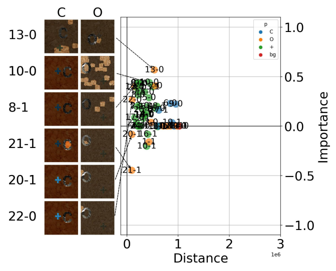

(c) Concepts ConceptShap

Figure 27: Concepts extracted from a densenet121 trained in the CO dataset.


(a) Concepts ECLAD


(b) Concepts ACE

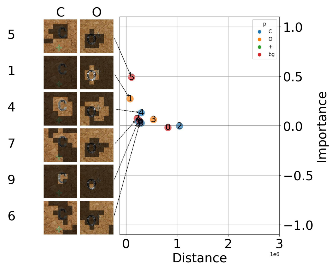
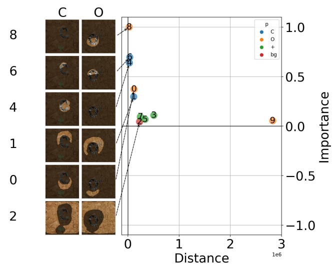
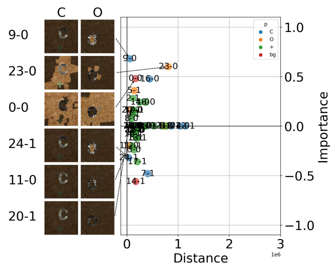

(c) Concepts ConceptShap

Figure 28: Concepts extracted from a efficientnet-b0 trained in the CO dataset.


(a) Concepts ECLAD


(b) Concepts ACE

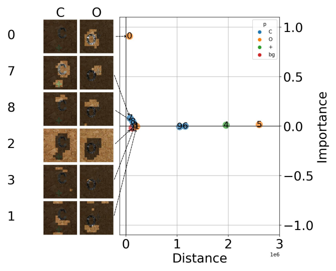
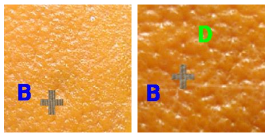

(c) Concepts ConceptShap

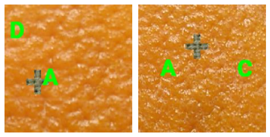
Figure 29: Concepts extracted from a vgg16 trained in the CO dataset.

D.1.8 COLORGB DATASET

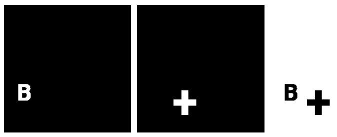
The dataset *colorGB* consists in detecting if a letter in the image is of color blue, or if all the letters are of color green. In this regard, the dataset contains four primitives. The first primitive p_1 can be either the character *A* or *B*, which is randomly selected and always appears in the images. The color of p_1 determines if the class is B blue, or G green. The second primitive p_2 is a random green character between *C*, *D* or blank (not appearing), yet, it is balanced between both classes. Finally, All images contain primitives p_3 and p_4 denoting a symbol $+$ and the background, which are filed with cotton and orange peel texture, respectively.



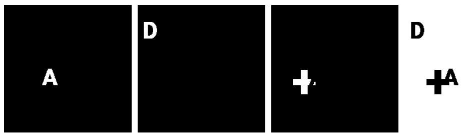
(a) Examples class B



(b) Examples class G



(c) Primitives p_1, p_3, and p_4 of class B.

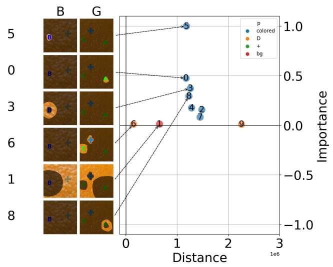


(d) Primitives p_1, p_2, p_3, and p_4 of class G.

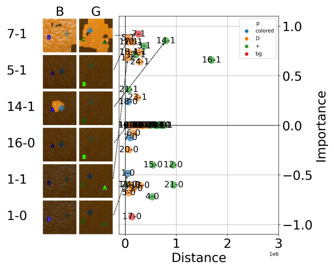
Figure 30: Dataset *colorGB*, where classes B (Figure 30(a)) and G (Figure 30(b)) are defined by the appearance of blue characters. Figure 30(c) shows the primitives of class B, where p_1 is filled blue. Figure 30(d) shows the primitives of class G, where p_1 always appears in green.

This dataset is forcing a clear color different rather than a form difference between the classes. In theory, a model should converge towards blue and green characters (*A*,*B*), possibly forcing a shortcut towards the blue color.

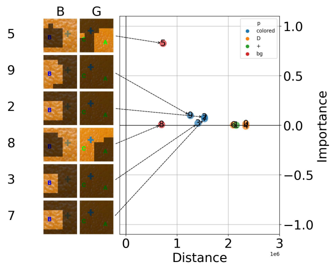
D.1.9 SAMPLE RESULTS FOR EACH MODEL TRAINED WITH THE DATASET COLORGB



(a) Concepts ECLAD

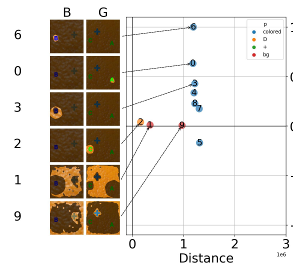


(b) Concepts ACE

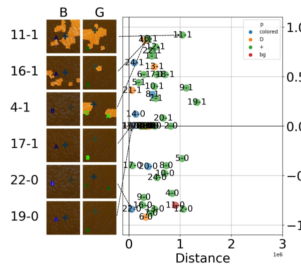


(c) Concepts ConceptShap

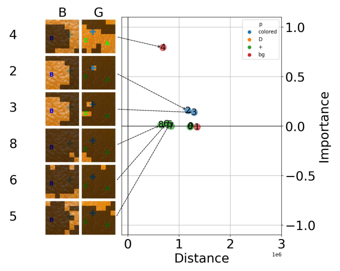
Figure 31: Concepts extracted from a resnet18 trained in the colorGB dataset.



(a) Concepts ECLAD

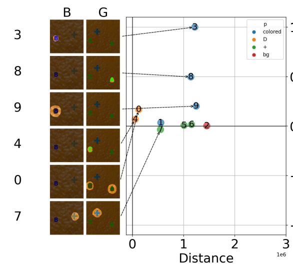


(b) Concepts ACE

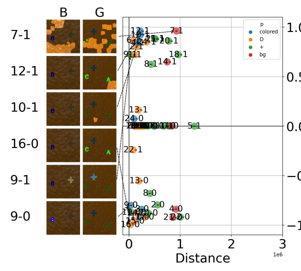


(c) Concepts ConceptShap

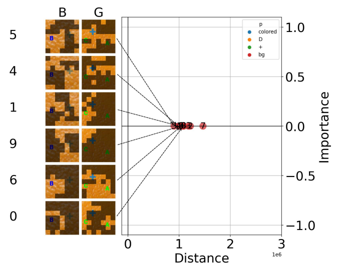
Figure 32: Concepts extracted from a resnet34 trained in the colorGB dataset.



(a) Concepts ECLAD

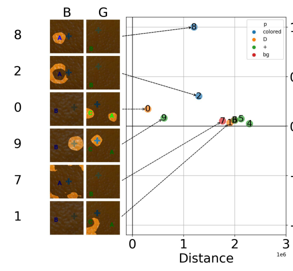


(b) Concepts ACE

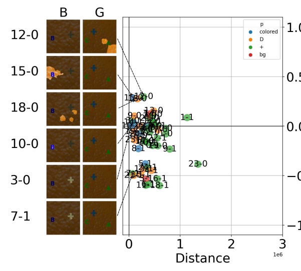


(c) Concepts ConceptShap

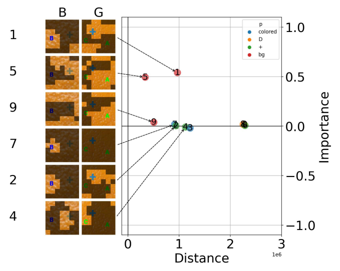
Figure 33: Concepts extracted from a densenet121 trained in the colorGB dataset.



(a) Concepts ECLAD

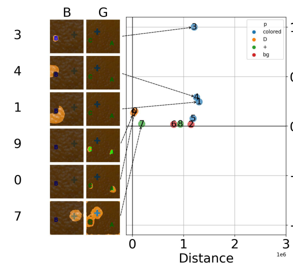


(b) Concepts ACE

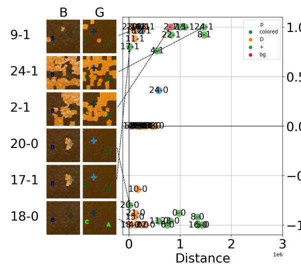


(c) Concepts ConceptShap

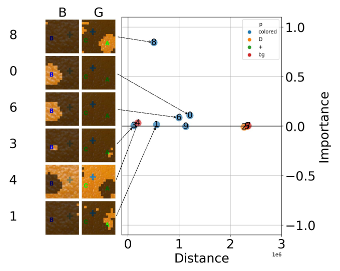
Figure 34: Concepts extracted from a efficientnet-b0 trained in the colorGB dataset.



(a) Concepts ECLAD



(b) Concepts ACE

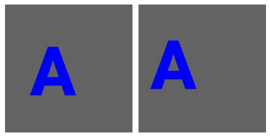


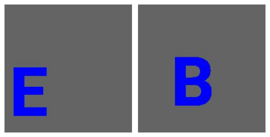
(c) Concepts ConceptShap

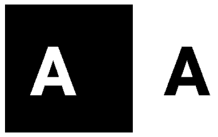
Figure 35: Concepts extracted from a vgg16 trained in the colorGB dataset.

D.1.10 ISA DATASET

The *isA* dataset consists in the classification of whether the main primitive of an image is an *A*. Its main primitive is the character *A* (p_1) in blue, which appear in all the images of the class isA. The second primitive p_2, consists in one letter from B to H, also in blue, happening only in class notA. The third primitive p_3 refers to the background filled in gray.


(a) Examples class isA


(b) Examples class notA


(c) Primitives p_1 and p_3 from class isA.

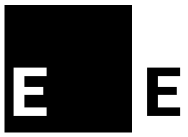
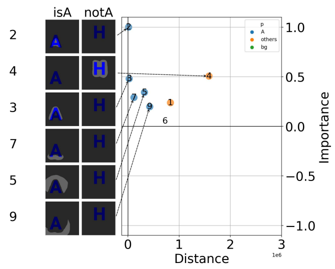

(d) Primitives p_2 and p_3 from class notA.

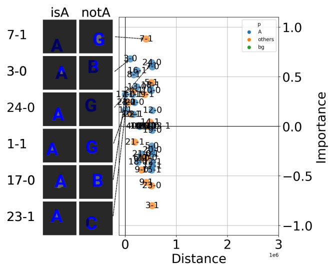
Figure 36: Dataset isA, examples of class isA and class notA are shown in Figures 24(a) and 24(b), respectively. Figure 24(c) shows the primitives p_1 and p_3 composing class isA. Figure 24(b) shows primitives p_2 and p_3 composing class notA.

The complexity unbalance of this dataset aims to test the performance of CE algorithms in cases where shortcut learning is to be expected. Thus, it is to be expected that from the extracted concepts, one or more will be related to a region of the letter *A*. This will then be measured via the spatial association of the concepts with the primitive p_1.

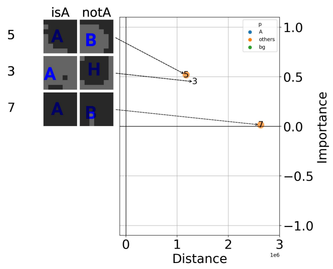
D.1.11 SAMPLE RESULTS FOR EACH MODEL TRAINED WITH THE DATASET ISA



(a) Concepts ECLAD

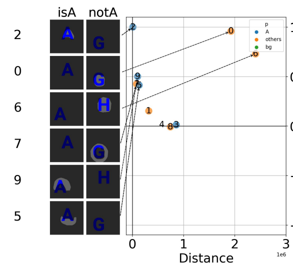


(b) Concepts ACE

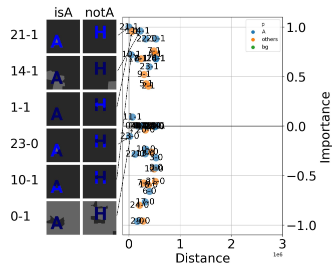


(c) Concepts ConceptShap

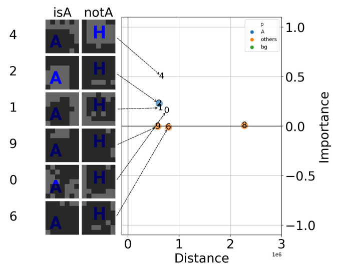
Figure 37: Concepts extracted from a resnet18 trained in the isA dataset.



(a) Concepts ECLAD

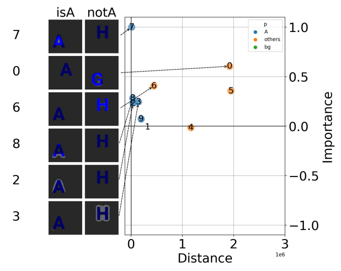


(b) Concepts ACE

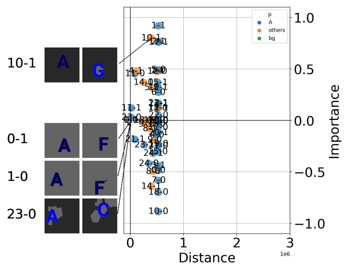


(c) Concepts ConceptShap

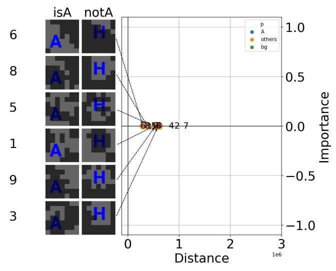
Figure 38: Concepts extracted from a resnet34 trained in the isA dataset.



(a) Concepts ECLAD

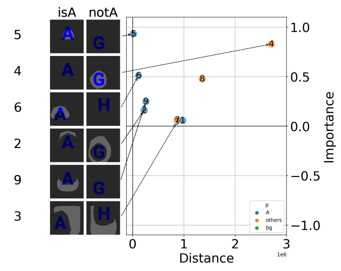


(b) Concepts ACE

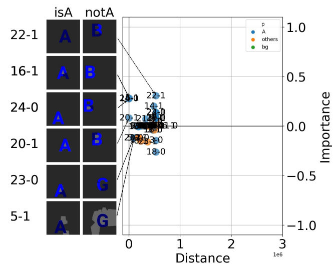


(c) Concepts ConceptShap

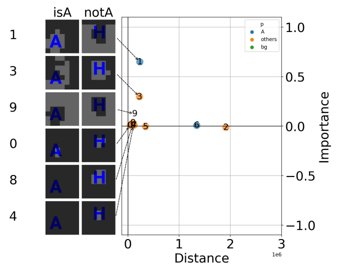
Figure 39: Concepts extracted from a densenet121 trained in the isA dataset.



(a) Concepts ECLAD

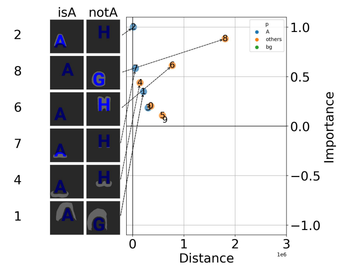


(b) Concepts ACE

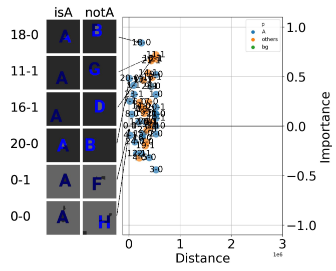


(c) Concepts ConceptShap

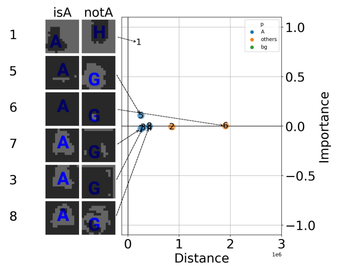
Figure 40: Concepts extracted from a efficientnet-b0 trained in the isA dataset.



(a) Concepts ECLAD



(b) Concepts ACE



(c) Concepts ConceptShap

Figure 41: Concepts extracted from a vgg16 trained in the isA dataset.

E Real-world datasets

The validation of the proposed methods was performed using two subsets (leather and metal nut) from the MVTec-AD dataset (Bergmann et al., 2021), containing pixel wise labelled data otherwise used for anomaly detection. This section contains example results for each analysis method (ECLAD, ACE, and ConceptShap) for each model trained in said datasets. In addition, ECLAD was tested in datasets for concrete crack (Özgenel & Sorguç, 2018), metal casting defects (Dabhi, 2020), and diabetic retinopathy classification (Society, 2019). These examples show the list of concepts extracted with ECLAD and their respective relative importance.

E.1 Concrete cracks dataset

The *concrete crack* dataset Özgenel & Sorguç (2018) contains images of cracked and good concrete. The main feature of the dataset is the presence of a crack (roughly though the complete image) for one of the classes. Examples of said classes are shown in Figure 42.

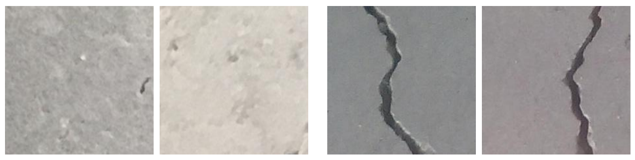

(a) Class Negative. (b) Class Positive.

Figure 42: Dataset concrete crack, composed of two classes, *Negative* without a crack (Figure 42(a)), and *Positive* with a crack (Figure 42(b)).

In an ideal scenario, the regions used by a model should coincide with the cracks appearing in the images. Thus, the extracted concepts should serve as a way to localize where the cracks are. This dataset serves as a simple domain with a real world application in civil and material engineering.

E.1.1 Sample results for concepts extracted with ECLAD for the dataset concrete cracks

Then five concepts with higher relative importance extracted through ECLAD can all be associated with the cracks in the images. These concepts can serve as a way to localize which parts in an image contain the defect which differentiates the classes of the dataset.

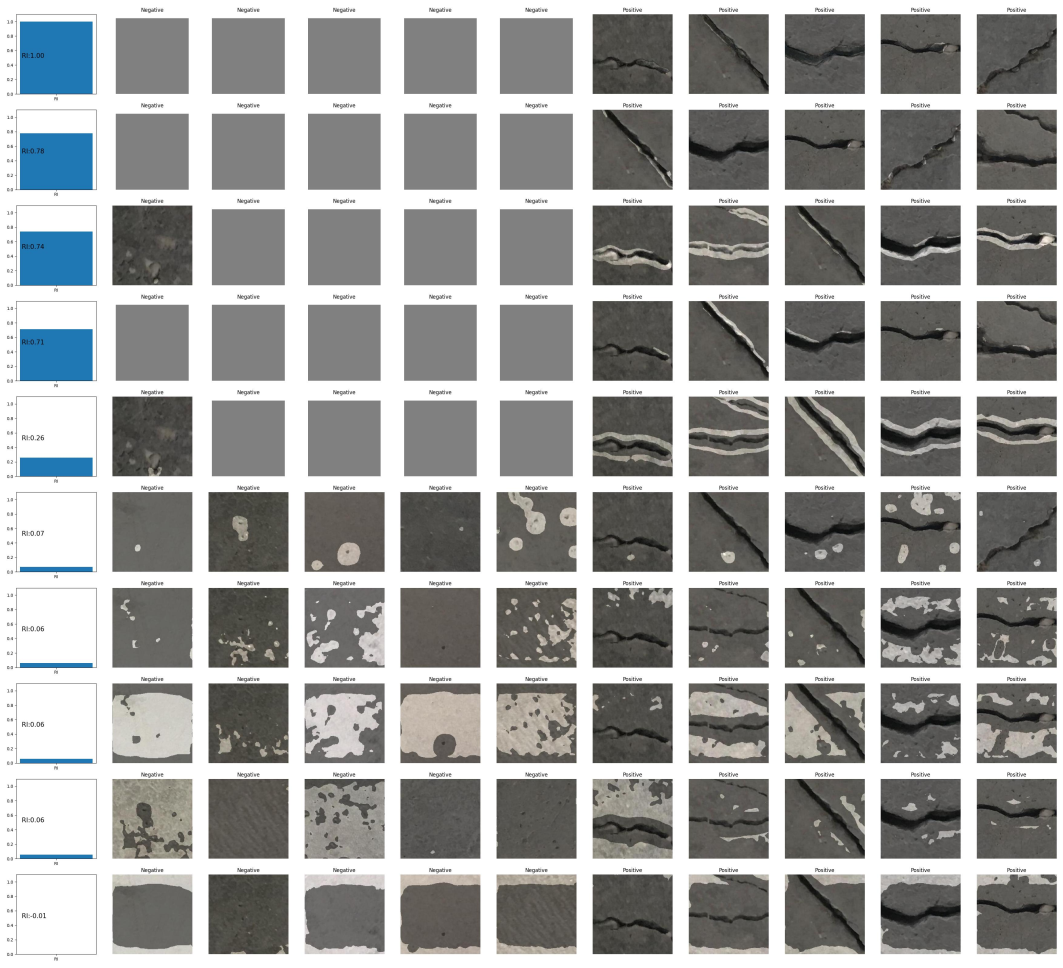
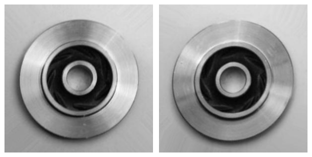


Figure 43: Concepts extracted with ECLAD from a Densenet-121 trained over the dataset concrete cracks. Each column refers to an extracted concept, ordered by their relative importance score. The first column represented their relative importance, subsequently, there are five examples of the concept detected in images of each class. A gray placeholder is used when the concept was not detected in more images of said class.

E.2 Metal casting dataset

The metal casting defects dataset Dabhi (2020) contains images of casting pieces which are classified as ok or defective. The main features of the defective class are either pinholes, or malformed edges. Examples of said classes are shown in Figure 44.


(a) Class OK.

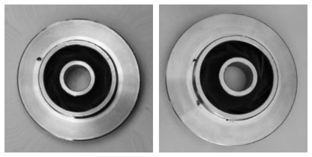

(b) Class defective.

Figure 44: Dataset metal casting, composed of two classes, *OK* with a well cast metal part (Figure 44(a)), and *defective* with a metal part with a defect on it (Figure 44(b)).

In an ideal scenario, the concepts learned by the model should be able to differentiate the defective regions of the image. Considering the low structural variability of the images, a model may also learn the presence of different structural parts (center hole, consistent edges, background).

E.2.1 Sample results for concepts extracted with ECLAD for the dataset metal casting

The concept with the higher relative importance can be associated to the pinholes or malformed edges. In contrast, other concepts with importance of less than half of the first one are associated with different structural parts of the cast piece. Finally, it is also an important insight that the least important concept is associated with the background (there is no background bias).

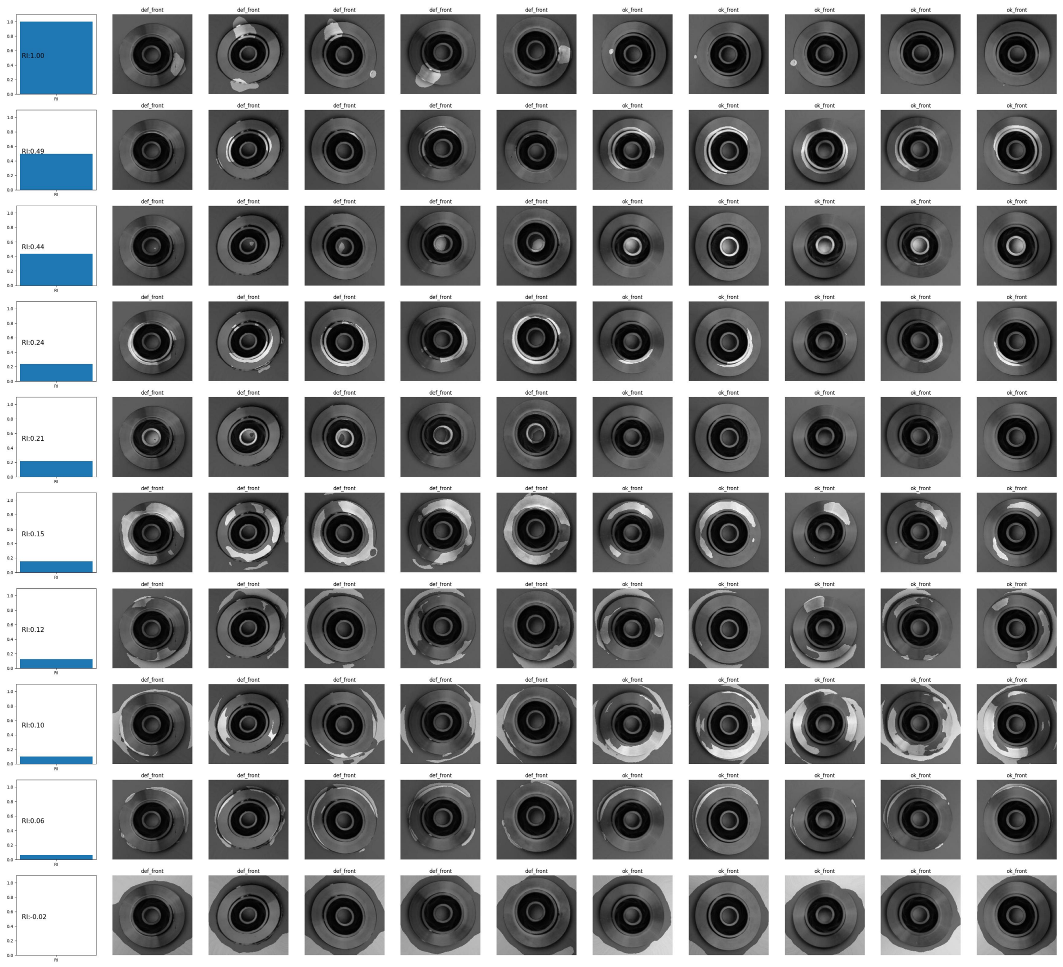
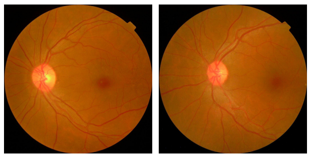
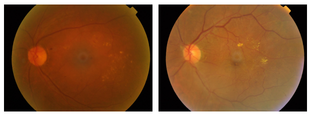


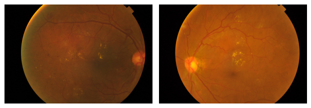
Figure 45: Concepts extracted with ECLAD from a Densenet-121 trained over the dataset metal casting. Each column refers to an extracted concept, ordered by their relative importance score. The first column represented their relative importance, subsequently, there are five examples of the concept detected in images of each class. A gray placeholder is used when the concept was not detected in more images of said class.

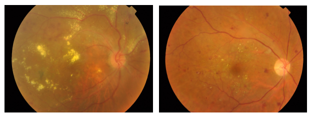
E.3 DIABETIC RETINOPATHY CLASSIFICATION DATASET

The diabetic retinopathy classification Society (2019) (APTOS), is a medical imaging dataset containing retina images taken using fundus photography. The images are classified in one of five classes, depending on the severity of their diabetic retinopathy. Examples of said classes are shown in Figure 46.


(a) Class 0 - No DR.


(b) Class 1 - Mild.


(c) Class 2 - Moderate.


(d) Class 3 - Severe.

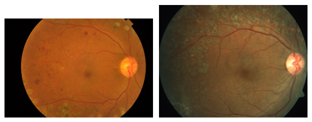

(e) Class 4 - Proliferative DR.

Figure 46: Dataset diabetic retinopathy classification, composed of five classes, *0 - No DR* with a healthy retina image (Figure 46(a)), *0 - No DR* with a healthy retina image (Figure 46(a)), *1 - Mild* with a retina image with mild retinopathy (Figure 46(b)), *2 - Moderate* with a retina image with moderate retinopathy (Figure 46(c)), *3 - Severe* with a retina image with severe retinopathy (Figure 46(d)), and *4 - Proliferative DR* with a retina image with proliferative retinopathy (Figure 46(e)).

In an ideal scenario, the extracted concepts will be aligned with the visual cues that human experts would use to diagnose diabetic retinopathy. This means, concepts with high relative importance should be associated with micro-aneurysm, exudates, issues in blood vessels, and large hemorrhages.

E.3.1 Sample results for concepts extracted with ECLAD for the dataset diabetic retinopathy classification

The two concepts with the highest relative importance are directly associated with micro-aneurysm and exudates. In addition, the third and fourth concepts used by the model are related to blood vessels of different characteristics. As a tangential insight, none of the concepts were specifically related to large hemorrhages. Paradoxically, the analyzed model does use a minimal set of visual cues important to human experts. Yet, not all cues that human experts would consider relevant are being used by the CNN in the prediction process.

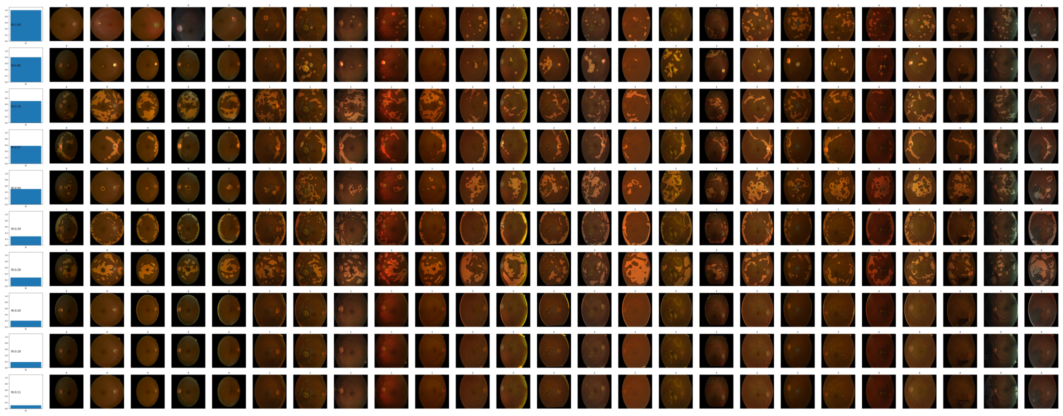


Figure 47: Concepts extracted with ECLAD from a Densenet-121 trained over the dataset diabetic retinopathy classification. Each column refers to an extracted concept, ordered by their relative importance score. The first column represented their relative importance, subsequently, there are five examples of the concept detected in images of each class. A gray placeholder is used when the concept was not detected in more images of said class.

F Computational cost and bottlenecks

Concept extraction algorithms provide global explanations in human understandable terms to improve the interpretability of neural networks. These algorithms can provide valuable insights, yet, their computational cost is significant. Most specifically, the presented methods, require the evaluation of the analyzed model a significant amount of times, as well as to execute clustering and regression techniques over large amounts of data. The computational cost of executing each one of these algorithms, not only depends on the CE method, but also on the model that is being analyzed. In this section we provide a rough approximation of the computational costs of each algorithm, in terms of their main operations. For each algorithm, we analyze the main operations on their phases of (1) identifying concepts, (2) importance scoring of concepts, and (3) usage to localize concepts, as of the authors implementation.

F.1 ECLAD

The first phase of ECLAD can be defined as the (1) Identification of concepts. This phase consists on the computation of LADs and execution of minibatch K-means. First, every batch of n_i images (from the dataset of N_e images) is evaluated on the CNN, and the activation maps from the set L of N_l layers are extracted. After extracting the activation maps, they are resized and concatenated, to obtain the descriptor d_{x_i} of each image, and latter flattening the descriptors of all images in the batch to form a vector of $h\ w\ n_i$ LADs, where each LAD has n_{LAD} dimensions (where n_{LAD} is the sum of units/neurons for all selected layers L). Finally, this vector is used to perform a step of the minibatch K-means. The time complexity of this phase is proportional to:

$$O(\text{ECLAD}_{(1)}) = \frac{N_e}{n_i}(n_i O(f_{\text{CNN}}) + n_i N_l O(f_{\text{resize}}) + O(\text{mbkmean}(h\ w\ n_i, n_{\text{LAD}}, k))) \tag{7}$$

Where, N_e denotes the size of the dataset; n_i is the number of images composing each batch; N_l is the number of layers on the set L; h and w denote the height and width of each input image; n_{LAD} is the sum of units in all layers of L; $O(f_{\text{CNN}})$ represents the complexity of evaluating an image on the selected CNN; $O(f_{\text{resize}})$ represents the complexity of resizing an activation map to size (h, w); and $O(\text{mbkmean}(h\ w\ n_i, n_{\text{LAD}}, k))$ represents the complexity of executing one step of minibatch K-means, for $h\ w\ n_i$ points of n_{LAD} dimensions, to compute k clusters.

Once the concepts have been identified, the phase (2) importance scoring of concepts starts. This phase requires the computation of $s_{x_i}^k$ for each image and each class, before aggregating the results to obtain $\text{CS}_{c_j}^k$ and RI_{c_j}. For each computation of $s_{x_i}^k$, the computation of d_{x_i} and g_{x_i} are required. Similarly, for the correct aggregation of $\text{CS}_{c_j}^k$, each LAD must be associated to the centroids extracted through minibatch K-means. The time complexity of this phase is proportional to:

$$\begin{aligned} O(\text{ECLAD}_{(2)}) =& N_k N_e (O(f_{\text{CNN}}) + N_l O(f_{\text{resize}}) + O(f_{\nabla\text{CNN}}) + 2 N_l O(f_{\text{resize}}) \\ &+ O(f_{\text{association}}) + O(f_{\text{s}})) + O(f_{\text{aggregation}}) \end{aligned} \tag{8}$$

Where N_k denotes the number of classes of the dataset; $O(f_{\nabla\text{CNN}})$ represents the complexity of computing the gradient of the CNN; $O(f_{\text{association}})$ represents the complexity of associating the LADs of an image to the centroids of the extracted concepts; $O(f_{\text{s}})$ is the complexity of computing $s_{x_i}^k$; and $O(f_{\text{aggregation}})$ represents the complexity of computing $\text{CS}_{c_j}^k$ and RI_{c_j}, based on the set of all $s_{x_i}^k$ and their associated concepts.

The rough cost of executing ECLAD is:

$$\begin{aligned}O(\text{ECLAD}) = & \\ & (1+N_k)N_eO(f_{\text{CNN}}) \\ & + N_kN_eO(f_{\nabla\text{CNN}}) \\ & + \frac{N_e}{n_i}O(\text{mbkmean}(h\ w\ n_i, n_{\text{LAD}}, k)) \\ & + (1+2N_k)NeN_lO(f_{\text{resize}}) \\ & + N_kN_eO(f_{\text{association}}) + N_kN_eO(f_{\text{s}}) + O(f_{\text{aggregation}})\end{aligned} \tag{9}$$

To evaluate new images, ECLAD's (3) usage to localize concepts, consists of three steps. First, the image is evaluated on the analyzed CNN, and the activation maps of the set of layers L are extracted. Then, these activation maps are resized and aggregated. Finally, each LAD of the resulting descriptor d_{x_i} is associated to the centroids of each concept. The time complexity of this phase is proportional to:

$$O(\text{ECLAD}_{(3)}) = O(f_{\text{CNN}}) + N_lO(f_{\text{resize}}) + O(f_{\text{aggregation}}) \tag{10}$$

From these operations, the bottlenecks are $O(f_{\text{CNN}})$, $O(\text{mbkmean}(h\ w\ n_i, n_{\text{LAD}}, k)$, and $O(f_{\text{s}})$. Where $O(f_{\text{CNN}})$ was performed on a GPU, and requires not only the memory to execute the CNN, but also to extract d_{x_i} of dimensions $h \times w \times n_{\text{LAD}}$. $O(\text{mbkmean}(h\ w\ n_i, n_{\text{LAD}}, k)$ was executed on the CPU, and could be speed up by using a GPU implementation of K-means, yet, it would also imply higher GPU requirements. Finally, $O(f_{\text{s}})$ was performed on GPU, and required the multiplication of two matrices d_{x_i} and g_{x_i}, which in case of being computed in minibatches, are of dimension $h\ w\ n_i \times n_{\text{LAD}}$ each. This operation can arise practical problems with limited GPU resources.

F.2 ACE

For a fair comparison, we will discuss the analysis of a complete dataset, and for simplification, we will assume balanced classes. The (1) identification of concepts using ACE, consists on three steps. First, each image is segmented n_s times using SLIC, to obtain a set of n_p patches. Then, each patch is resized, padded, and evaluated on the CNN, to obtain the activation map of the selected layer. Finally, for each class, the set of $\frac{N_e}{N_k}n_p$ vectors, of $h_l\ w_l\ n_{\text{layer}}$ dimensions are used to extract k clusters using K-means (where n_{layer} denotes the number of dimensions of the selected layer). The time complexity of this phase is proportional to:

$$\begin{aligned}O(\text{ACE}_{(1)}) = & N_k(\frac{N_e}{N_k}(n_sO_{\text{SLIC}} + n_pO(f_{\text{CNN}})) \\ & + O_{kmean}(\frac{N_e}{N_k}, h_l\ w_l\ n_{\text{layer}}, k))\end{aligned} \tag{11}$$

Where N_e is the size of the dataset; N_k is the number of classes of the dataset; k is the number of concepts to extract; n_s denotes the number of SLIC segmentations to perform (e.g., the ACE method segments an image to obtain [15,50,80] patches, in this case $n_s = 3$); n_p denotes the total number of patches extracted after filtering the SLIC segments (e.g., for the default parameters of ACE, $15 + 50 + 80 > n_p > 15$); h_l and w_l denote the height and width of the activation maps of the selected layer; n_{layer} denotes the number of units of the selected layer; O_{SLIC} denotes the complexity of executing the SLIC segmentation algorithm; $O(f_{\text{CNN}})$ represents the complexity of evaluating an image on the selected CNN; $O_{kmean}(\frac{N_e}{N_k}, h_l\ w_l\ n_{\text{layer}}, k))$ represent the complexity of executing the K-means algorithm for $\frac{N_e}{N_k}$ datapoints, of $h_l\ w_l\ n_{\text{layer}}$ dimensions, and k clusters.

After the extraction of concepts, the (2) importance scoring of concepts is performed using TCAV, for each concept of each class. The process of obtaining the CAV and TCAV score of a concept consists on four steps. First, a random set of images are sampled from the dataset to serve as a random concept. Similarly, a subset of n_b images from the concept and random concept are sampled. Second, both subsets of images are evaluated on the CNN, to obtain the flattened activation maps of the selected TCAV layer. Third, a linear classifier is trained to differentiate both subsets of vectors (we obtain a CAV from this classifier). This linear classification is performed over a dataset of $2n_b$ vectors of $h_l\ w_l\ n_{\text{layer}}$ dimensions. Fourth, we compute the TCAV score for said CAV, by evaluating every image of the associated class in the model, obtaining the directional derivative on the corresponding layer and counting how many of these directional derivatives point on the same direction as the CAV, this proportion is the TCAV score. This process is repeated n_{tcav} times for each concept and an associated concept of random images (serving as a random concept for control). The resulting sets of TCAV scores and CAVs of the concept and random concept are then compared using a t-test, to obtain a p-value stating the statistical significance of the concept. The time complexity of this phase is proportional to:

$$\begin{aligned} O(\text{ACE}_{(2)}) = & N_k K(\\ & \quad n_{\text{tcav}}(O(f_{\text{sample}}) \\ & \qquad + 2n_b O(f_{\text{CNN}}) \\ & \qquad + O(f_{\text{lin-class}}) \\ & \qquad + \frac{N_e}{N_k}(O(f_{\text{CNN}}) \\ & \qquad\quad + O(f_{\nabla\text{CNN}}) \\ & \qquad\quad + O(f_{\text{proj}})) \\ & \quad) + O(f_{\text{t-test}})) \end{aligned} \tag{12}$$

Where N_e is the size of the dataset; N_k is the number of classes of the dataset; n_{tcav} refers to the number of subsample computations to TCAV scores for each concept, to later compute the statistical significance of the concept; n_b is the size of each image subset to compute each TCAV score; $O(f_{\text{sample}})$ represents the complexity of sampling n_b images from a concept and random concept; $O(f_{\text{CNN}})$ represents the complexity of executing the CNN and extracting the selected activation map; $O(f_{\nabla\text{CNN}})$ represents the complexity of computing the gradient of the CNN w.r.t. the selected layer; $O(f_{\text{lin-class}})$ represents the complexity of fitting a linear a stochastic gradient descent classifier to $2n_b$ vectors of $h_l\ w_l\ n_{\text{layer}}$ dimensions; $O(f_{\text{proj}})$ represents the complexity of projecting the gradient of the network towards a CAV;

The rough cost of executing ACE is:

$$\begin{aligned} O(\text{ACE}) = & N_e n_s O_{\text{SLIC}} \\ & + (N_e n_p + 2N_k K n_{\text{tcav}} n_b + K n_{\text{tcav}} N_e) O(f_{\text{CNN}}) \\ & + K n_{\text{tcav}} N_e O(f_{\nabla\text{CNN}}) \\ & + N_k O_{kmean}(\frac{N_e}{N_k}, h_l\ w_l\ n_{\text{layer}}, k) \\ & + N_k K n_{\text{tcav}} O(f_{\text{lin-class}}) \\ & + N_k K n_{\text{tcav}} O(f_{\text{sample}}) \\ & + K n_{\text{tcav}} N_e O(f_{\text{proj}}) \\ & + N_k K O(f_{\text{t-test}}) \end{aligned} \tag{13}$$

To improve the efficiency of ACE, we evaluated the extracted patches, random images, and the images of every class a single time, and sampled the tensors when performing the TCAV computations. This implementation detail significantly reduced the time complexity to:

$$\begin{aligned} O(\text{ACE}) =& N_e n_s O_{\text{SLIC}} \\ &+ (N_e n_p + 2N_k K n_b + N_e) O(f_{\text{CNN}}) \\ &+ N_e O(f_{\nabla\text{CNN}}) \\ &+ N_k O_{kmean}(\frac{N_e}{N_k}, h_l\ w_l\ n_{\text{layer}}, k) \\ &+ N_k K n_{\text{tcav}} O(f_{\text{lin-class}}) \\ &+ N_k K n_{\text{tcav}} O(f_{\text{sample}}) \\ &+ K n_{\text{tcav}} N_e O(f_{\text{proj}}) \\ &+ N_k K O(f_{\text{t-test}}) \end{aligned} \tag{14}$$

To evaluate new images, ACE's (3) usage to localize concepts, consists of three steps. First, the image is segmented using SLIC n_s times. Second, each patch is evaluated on the analyzed CNN, and the activation maps of the selected layer is extracted. Then, these flattened activation maps are compared with each CAV of the class concepts. Finally, the masks of each path are aggregated to obtain the localization result. The time complexity of this phase is proportional to:

$$\begin{aligned} O(\text{ACE}_{(3)}) =& n_s O_{\text{SLIC}} \\ &+ n_p O(f_{\text{CNN}}) \\ &+ n_p O(f_{comparison}) \\ &+ O(f_{\text{aggregation}}) \end{aligned} \tag{15}$$

From these operations, $O(f_{\text{CNN}})$, O_{SLIC}, and $O_{kmean}(\frac{N_e}{N_k}, h_l\ w_l\ n_{\text{layer}}, k)$ were the bottlenecks. $O(f_{\text{CNN}})$ was performed on GPU, and also required the memory for the extraction of the selected layer. Both O_{SLIC} and $O_{kmean}(\frac{N_e}{N_k}, h_l\ w_l\ n_{\text{layer}}, k)$ were performed on CPU, which significantly increase the time requirements of ACE.

F.3 ConceptShap

The phase of (1) identification of concepts with ConceptShap is performed by including an extra set of layers at a defined point of a CNN and training said layer for n_{et} epochs. Each evaluation of the CNN is performed until a selected layer. Then, a linear projection of the resulting activation map is performed towards a lower dimensional space (of k dimensions). Afterwards, an extra pair of layers g are introduced to rescale the obtained tensor and obtain an activation map of the original size. Then, the rest of the CNN is evaluated as originally intended. In this process, the lower dimensional space is introduced as a concept space, and loss is added to it. The identification of concept is then performed by freezing the CNN weights and training the new layer (for n_{et} epochs), to optimize the performance of the model as well as the extra losses introduced on the concept space. The time complexity of this phase is proportional to:

$$O(\text{ConceptShap}_{(1)}) = N_{et} n_{et}(O(f_{\text{CNN}}) + O(f_{\nabla\text{CNN}}) + O(f_{\text{optim-step-CNN}})) \tag{16}$$

Where N_{et} refers to the training subset of the dataset N_e; n_{et} refers to the number of epochs for training the added layers; $O(f_{\text{CNN}})$ refers to the complexity of the evaluation of the CNN, including the evaluation of the new layers; $O(f_{\nabla\text{CNN}})$ and $O(f_{\text{optim-step-CNN}})$ refer to the complexity of computing the gradient of the CNN and performing an optimization step over the new layers.

After training the new projections and obtaining a concept space, each component of said space is scored, and their importance is computed based on Shapely values. Said Shapely values are obtained based on a Monte Carlo approximation. For each one of the N_{MC} samples of this approximation, the contribution is computed as the difference in completeness score between not ablating the concepts, and ablating them. The completeness score requires a complete evaluation of the validation set. In addition, when ablating the concepts, a retraining of the layers g is performed for n_{ev} epochs (to compute the completeness score). The time complexity of this phase is proportional to:

$$\begin{aligned} O(\text{ConceptShap}_{(2)}) =& N_{\text{MC}}(N_{ev}n_{ev}(O(f_{\text{CNN}}) + O(f_{\nabla\text{CNN}}) \\ &+ O(f_{\text{optim-step-CNN}})) + 2N_{ev}O(f_{\text{CNN}})) \end{aligned} \tag{17}$$

Where N_{ev} refers to the validation subset of the N_e dataset; n_{ev} refers to the number of epochs to retrain the new layers at each computation of the completeness score; N_{MC} refers to the number of samples to use when computing the Monte Carlo approximate of the shapely values of the concepts; $O(f_{\text{CNN}})$, $O(f_{\nabla\text{CNN}})$, $O(f_{\text{optim-step-CNN}})$ refer to the computational complexity of evaluating the CNN, computing its gradient and performing an optimization step over the parameters of the new layers, respectively.

The rough cost of executing ConceptSHAP is:

$$\begin{aligned} O(\text{ConceptShap}) =& \\ &(N_{et}n_{et} + N_{\text{MC}}N_{ev}n_{ev} + 2N_{\text{MC}}N_{ev})O(f_{\text{CNN}}) \\ &+ (N_{et}n_{et} + N_{\text{MC}}N_{ev}n_{ev})O(f_{\nabla\text{CNN}}) \\ &+ (N_{et}n_{et} + N_{\text{MC}}N_{ev}n_{ev})O(f_{\text{optim-step-CNN}}) \end{aligned} \tag{18}$$

To evaluate new images, ConceptShap's (3) usage to localize concepts, consists of three steps. First, the image is evaluated on the analyzed CNN, including the newly added layers for linear projection and resizing. Then, the activation map of the concept space is extracted, and based on a threshold, each dimension of said tensor is used as the mask of said concept. Finally, the extracted masks are resized to the original size of the image, and can be used to localize each concept. The time complexity of this phase is proportional to:

$$O(\text{ConceptShap}_{(3)}) = O(f_{\text{CNN}}) + O(f_{\text{aggregation}}) + O(f_{\text{threshold}}) \tag{19}$$

From these operations, the Monte Carlo approximation of the shapely values of each concept are resource intensive. Specially, since the computation of the contribution for each sample requires the retraining of the layers g. Yet, this operation is performed on GPU, which speeds it up significantly.

F.4 Comparison

The nature of the three algorithms differs significantly, and thus, they scale differently to specific parameters and operations. As an example, ECLAD requires the evaluation of a CNN $(1 + N_k)N_e$ times, and $\frac{N_e}{n_i}$ executions of minibatch K-means. ECLAD will perform efficiently (w.r.t ACE and ConceptShap) for dataset with few classes (e.g. $N_k < 20$). There is a tradeoff between speed and GPU requirements based on the minibatch size n_i, where it increases the required GPU memory (by a factor of $n_i \times n_{\text{LAD}} \times \frac{h\ w}{h_l\ w_l}$, w.r.t ACE and ConceptShap), yet, it speeds up the computations of $O(f_{\text{CNN}})$, and $O(f_{\text{s}})$. In contrast ACE requires $(1 + n_p)N_e + 2N_kKn_b$ evaluations of the CNN, N_en_s executions of SLIC, and N_k executions of K-means. This means that the number of executions of the CNN scales better to the number of classes. To increase the scalability of the method, the K-means of each class can be performed by minibatches (analog to ECLAD). Yet, the computational cost of executing N_en_s times SLIC, and N_kKn_{tcav} linear classifiers is significant, which makes it slower than ECLAD and ConceptShap for most cases. ConceptShap evaluates the analyzed CNN roughly $n_{et}N_e + (N_{\text{MC}} - 1)N_{ev}n_{et} + 2N_{\text{MC}}N_{et}$ (if we consider $N_{et} = N_{ev}$), which means it doesn't require more resources, regardless of the number of classes. In contrast, it scales poorly with the number of concepts k, as it directly influences the number of samples N_{MC} required for the convergence of the Monte Carlo approximation of the shapely values of the concepts.

As a broad summary, ECLAD provides more granular explanations, scaling well to large datasets and number of concepts. It scales linearly for the number of classes N_k, which makes it preferable when dealing with a small number of classes (e.g. $N_k < 20$). ConceptShap scales well for a large number of classes, yet it scales poorly for the number of extracted concepts. As a caution, ConceptShap and Shapely values in general have issues when dealing with correlated concepts, which can be problematic when detecting spurious correlations and their importance for a CNN. Finally, ACE scales better than ECLAD w.r.t. to the number of classes, yet, it has a significant computational cost of executing SLIC and SDG linear classification. In this regard, ACE can be parallelized and executed per class, being a better fit for large datasets with a large number of classes (e.g. N_k 1000). In our settings, ECLAD took the least time to execute, followed by ConceptShap, and finally ACE.

F.5 Ablation study

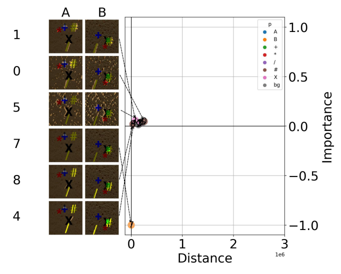
In contrast to other concept extraction methods (e.g. ACE, ConceptShap), ECLAD proposes the upscaling and aggregation of of activation maps at different levels of a neural network. On a pixel level, these local aggregated descriptors are denoted as LADs, and are used as a basis for extracting similarly encoded areas through a clustering algorithm. In this ablation study, we explore four key components of our method. First, we explore the need for aggregating information of different layers. Second, we investigate the impact of the number of aggregated layers. Third, we examine the difference of using different numbers of clusters. Finally, we assess the impact of using multiple upscaling methods.

Each study was performed using two models architectures (ResNet-18 (He et al., 2015) and DenseNet-121 (Huang et al., 2016)) trained over the *ABplus* and *leather* datasets. From each architecture we selected eight equally distributed layers, named l_1 to l_8, from which subsets were used on each run. The plots shown below result from executing ECLAD with different sets of parameters over a DenseNet-121, trained on the *ABplus* datasets, which are representative of both architectures and datasets.

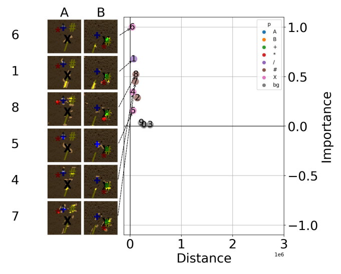
F.5.1 Aggregating activation maps

We compare the execution of ECLAD over single layers (across different depths of the network), with the standard execution using four layers equally distributed across the depth of the network. Similar to the figures shown in the result section, we provide scatter plots of association distance and importance of the extracted concepts for each run, as seen in Figure 48.

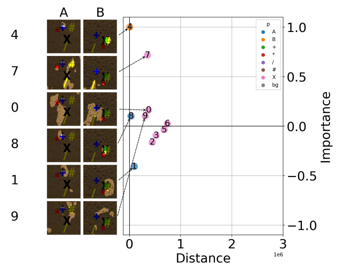
Combining layers from multiple depths allows the extraction of mid and high level concepts without the complexity of fine-tuning the selected layer. The results of performing CE over single low level layers generates concepts lacking from abstract meaning such as lateral edges found in $l1$ (c_4, c_5 in subfigure 48(a)), or multiple entangled features such as yellow, green and black edges found in $l3$ (c_6, c_1 in subfigure 48(b)). When using a single high level layer, the generated concepts disregard mid level features such as the * or + characters in the images which are not found in l_6 nor l_8. In addition, the latent representations of the features dilates, generating significant halo effects in l_6 and l_8 (c_0, c_8, c_9 in subfigure 48(e)). This makes the choice of a single layer, non trivial, as higher level layers miss existing concepts, and low level layers lack abstraction and disentanglement. In contrast, the aggregation of equally distributed layers extracted high level features characteristics of the task, such as A and B (c_2, c_4 in subfigure 48(f)), as well as other existing high level concepts differentiated by the model, such as the * or + characters (c_8, c_3 in subfigure 48(f)). By aggregating multiple layers, the resulting concepts are more defined (mitigating the halo effect of higher layers), and the selection of layers for the analysis becomes less critical.



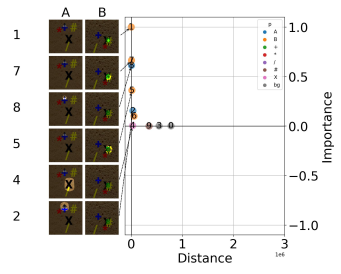
(a) $L = \{l_1\}$



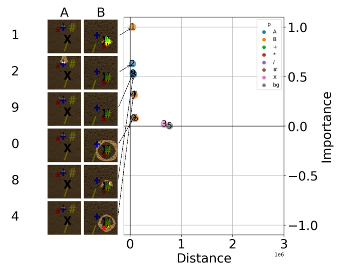
(b) $L = \{l_3\}$



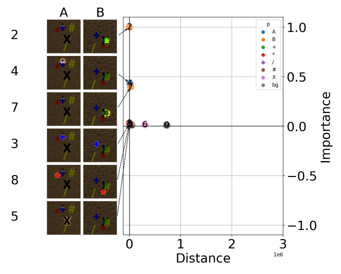
(c) $L = \{l_5\}$



(d) $L = \{l_6\}$



(e) $L = \{l_8\}$



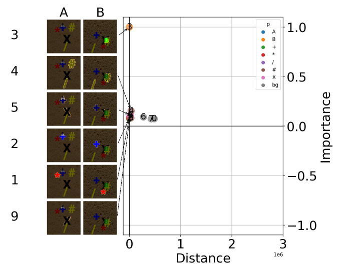
(f) $L = \{l_2, l_4, l_6, l_8\}$

Figure 48: Concepts extracted from a DenseNet-121 trained in the ABplus dataset. Subfigures 48(a) to 48(e) contain the results of single layer executions, and subfigures 48(f) contains the results of aggregating four layers. Concepts extracted from low layers lack abstract meaning and are related to texture, edges or color (e.g. c_0, c_1, c_7 in subfigure 48(a)), while concepts extracted from higher layers disregard mid level concepts (e.g. + character on the images). The results from aggregating layers (subfigure 48(f)), contain abstract concepts relevant to the classification task (e.g. c_2, c_4), without loosing specificity nor disregarding mid level concepts (e.g. c_8, c_5, c_3).

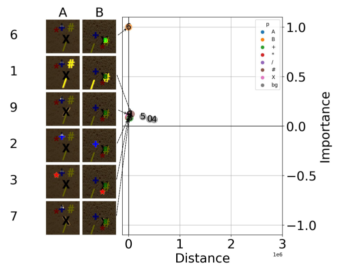
F.5.2 Number of aggregated layers

We compare the execution of ECLAD selecting different number of equally distributed layers (along the depth of the model). As discussed before, using a single layer for concept extraction can be problematic given the dilation of concepts through the CNNs, as well as the disappearance of mid level concepts through the network. In this section we compare the impact of using two or more layers for the concept extraction, the resulting scatter plots are shown in Figure 49.

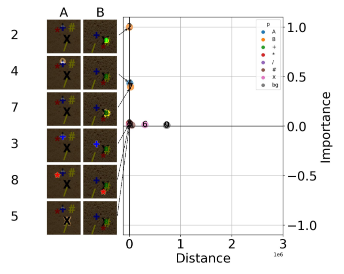
Including more layers mitigates the halo effect of important concepts and allows the inclusion of mid level concepts. The results of performing CE with two or three layers generates concepts including the most important ones, but also entangled representations of other concepts, such as edges (c_9 in subfigure 49(a), and c_7 in subfigure 49(b)), and entangled concepts such as the characters **X** and **A** (c_5 in subfigure 49(a), and c_9 in subfigure 49(b)). A possible explanation can suggest that the selected layers did not have enough information for clearly separating the different concepts (e.g. **X** and **A**), as their representations are differentiated in middle layers. ECLAD runs with more layers (e.g. 4 and 8) extract disentangled concepts as the possibility of including relevant layers increase. This can be observed in the characters **X**, ***** and **B** (e.g. c_2 and c_5 in subfigure 49(c), and c_1 and c_7 in subfigure 49(e)). Similarly, possible issues of halo effect in important concepts diminish with an increasing number of layers, which can be seen for concepts related to the character **B** with a halo in subfigure 49(c), which disappears with the subsequent inclusion of more layers. It must be mentioned that the computational cost increases with each new layer selected for the analysis, thus, our choice of four layers is a balance between computational cost and good performance.



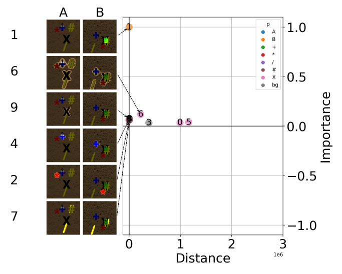
(a) $L = \{l_1, l_7\}$



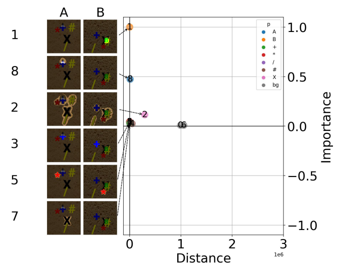
(b) $L = \{l_1, l_4, l_7\}$



(c) $L = \{l_1, l_3, l_5, l_7\}$



(d) $L = \{l_0, l_1, l_3, l_4, l_6, l_7\}$



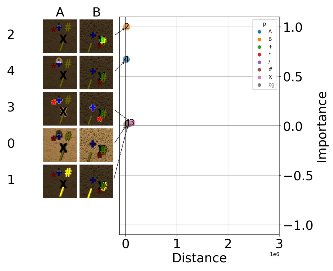
(e) $L = \{l_0, l_1, l_2, l_3, l_4, l_5, l_6, l_7\}$

Figure 49: Concepts extracted from a DenseNet-121 trained in the ABplus dataset. Subfigures 49(a) to 49(e) contain results of executing ECLAD with 2 to 8 layers equally distributed through the depth of the model. For two and three layers, the resulting concepts also include low level features such as edges (c_9 in subfigure 49(a), and c_7 in subfigure 49(b)), and entangled concepts such as the characters **X** and **A** (c_5 in subfigure 49(a), and c_9 in subfigure 49(b)). runs with four to eighth layers provide a better extraction of disentangled concepts such as the characters **X**, * and **B** (e.g. c_2 and c_5 in subfigure 49(c), and c_1 and c_7 in subfigure 49(e)). In addition, the halo effect of important concepts such as the character **B** progressively diminishes with the number of layers.

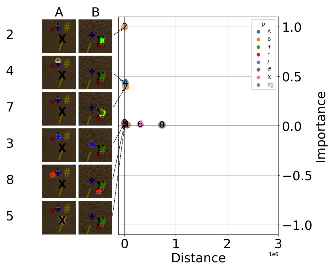
F.5.3 NUMBER OF CLUSTERS

The number of concepts to extract n_c is an important parameter for ECLAD, as it determines the number of clusters to mine using minibatch k-means over subsets of LADs. In this section we compare executions of ECLAD with different numbers of k-means clusters n_c. The results of four runs with n_c of 5, 10, 20 and 50 are shown in Figure 50.

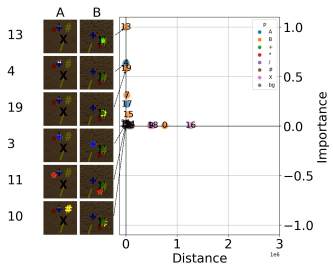
A low number of concepts will group unimportant features, and a high number of concepts will slice important features. Nonetheless, the extraction and scoring of important features is consistent. For low number of clusters, such as 5, unimportant features are grouped together in a single concept. An example can be seen in concept c_3 from subfigure 50(a), where the characters **X**, **+** and * were grouped on a single concept. These features are then split when the number of clusters is increased, as seen for $n_c = 10$ in subfigure 50(b), with concepts c_3, c_8, and c_5. For larger number of clusters such as 20 or 50, features are sliced into multiple concepts. An example can be seen in concepts c_5 and $c_3$3 of subplot 50(d) which represent the center and surrounding of character **B**, respectively. Nonetheless, the slices of important concepts are still being scored with the highest RI_{c_j}, consistently across different numbers of concepts.



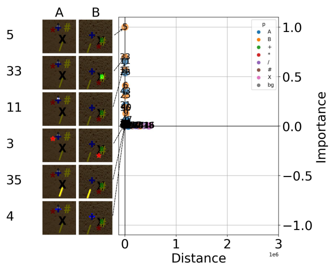
(a) $n_c = 5$



(b) $n_c = 10$



(c) $n_c = 20$

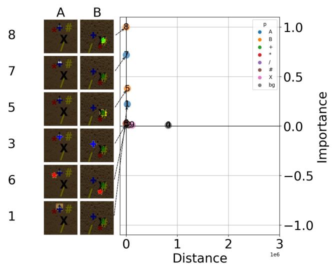


(d) $n_c = 50$

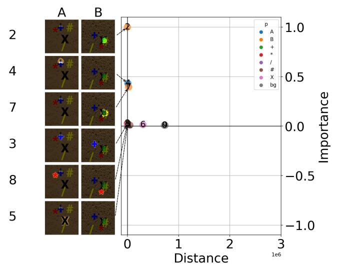
Figure 50: Concepts extracted from a DenseNet-121 trained in the ABplus dataset. Subfigures 50(a) to 50(d) contain results of executing ECLAD with 5 to 50 extracted concepts. With n_c of five, the important concepts are extracted correctly (character **B** and **A**, c_2 and c_4 in subfigure 50(a)), and unimportant concepts are presented together (c_3 in subfigure 50(a)). With an increasing number of clusters, the different features start to disentangle, e.g. characters **X**, *****, and **+**, in subfigure 50(b). Yet, for larger number of concepts such as 20 or 50, the original features such as the character **B**, start to be sliced into multiple concepts (e.g. c_5 and c_{33} in subfigure 50(d)).

F.5.4 UPSCALING METHODS

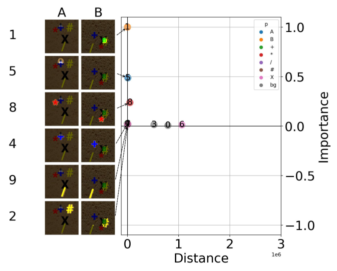
A key step of ECLAD is the upsampling of activation maps to obtain the image descriptors and LADs. The upsampling functions can have a significant impact when resizing small activation maps from high level layers. Thus we explore three alternatives in the runs below, in Figure 51 we present ECLAD results of three runs using nearest interpolation, bilinear interpolation, and bicubic interpolation.



(a) f_U = nearest interpolation.



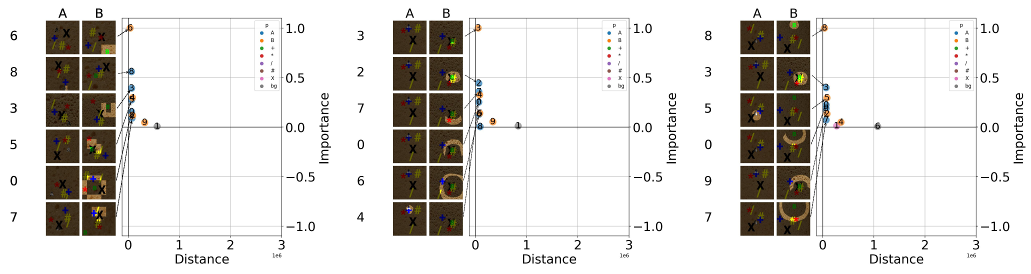
(b) f_U = bilinear interpolation.



(c) f_U = bicubic interpolation.

Figure 51: Concepts extracted from a DenseNet-121 trained in the ABplus dataset. Subfigures 51(a), 51(b), and 51(c) present the results for executing ECLAD with nearest interpolation, bilinear interpolation and bicubic interpolation respectively. The resulting concepts of the three runs contain the main features of the dataset, characters **B** and **A**, which are extracted in concepts c_8, c_7 in subfigure 51(a), c_2 and c_4 in subfigure 51(b), and c_1 and c_5 in subfigure 51(c).

Using coarse interpolation methods (f_U) will impact the boundaries of the extracted concepts, but not the concepts themselves. For the three methods, similar concepts where extracted, an example is the important character **B**, which is extracted as concepts c_8, c_2, c_1 for the nearest, bilinear, and bicubic interpolation runs respectively. A similar example is the unimportant character **+** which is extracted as concepts c_3, c_3, c_4 for the nearest, bilinear, and bicubic interpolation runs respectively. Aside from the rough boundaries of the concepts, no perceivable effect was observed on the end result of the different runs. A similar behavior was observed when analysing the results for a ResNet-18 also trained on the ABplus dataset, as shown in Figure 52



(a) f_U = nearest interpolation. (b) f_U = bilinear interpolation. (c) f_U = bicubic interpolation.

Figure 52: Concepts extracted from a Resnet-121 trained in the ABplus dataset. Subfigures 52(a), 52(b), and 52(c) present the results for executing ECLAD with nearest interpolation, bilinear interpolation and bicubic interpolation respectively. The resulting concepts of the three runs contain the main features of the dataset, characters **B** and **A**, which are extracted in concepts c_6, c_8 in subfigure 52(a), c_3 and c_2 in subfigure 52(b), and c_8 and c_5 in subfigure 52(c).

F.6 Distance metric

A key contribution of the current manuscript is the proposal of an distance metric DST_{p_o,c_j} measuring the *spatial association* between the masks of concepts and primitives. This metric takes into account overlapping and spatial closeness to mitigate the effect of associating off-centered and surrounding concept. The case of off-centered concepts can arise when the representation of a feature shifts through the filters of CNN. This phenomenon can arise in relation with the depth of a CNN unless the activation maps of multiple depths are constrained. The case of surrounding concepts can arise when a network recognizes the shape of a feature as important, and not the area of the feature itself. Thus, the the edges surrounding the shape may be recognized, and further propagated towards the exterior of an object. We seek a metric capable of relating dataset primitives and concepts even in these exceptional cases. In this section we compare the proposed DST_{p_o,c_j} association distance, with the Jaccard score used in object detection, the normalized mutual information score, and the adjusted rand score used in clustering.

As an example, the CO synthetic dataset consists in classifying images with a character **C** or a character **O** in them. Given the shape of both characters, the difference can be described as an extra right section for the character **O**, or a missing right section of the **C**. The two approaches for detecting both classes where seen in the experimentation process. For the current metric comparison, an example of overlapping related concepts is shown in Figure 53.

On the first experimental setup we compare two masks emulating a primitive and a concept, with the same general form as the offset between the characters increase. In the figure 54 we present the experimental results of comparing two masks of the character **A**, and two masks of the character **O**, at various offsets. White areas represent overlapped sections and gray areas represent non overlapping regions of both masks.

The proposed association distance DST_{p_o,c_j} **can express offsets between primitive and concept masks, with or without overlapping**. Figure 54(a), shows as the three alternative metrics decrease monotonically with a bigger offset between masks, yet, the difference can only be measured while there is a degree of overlapping. When the two masks cease to overlap, the alternative metrics do not express the offset anymore. In addition, depending on the geometry of the compared masks, the degree of overlapping may increase as the two masks shift from each other. This can be observed at 40

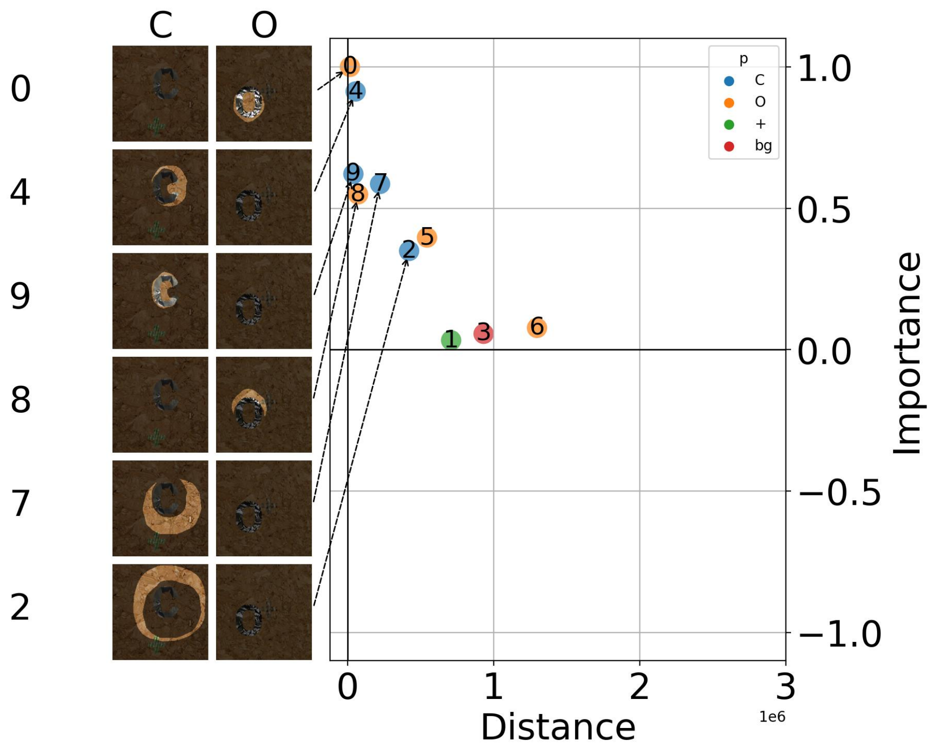
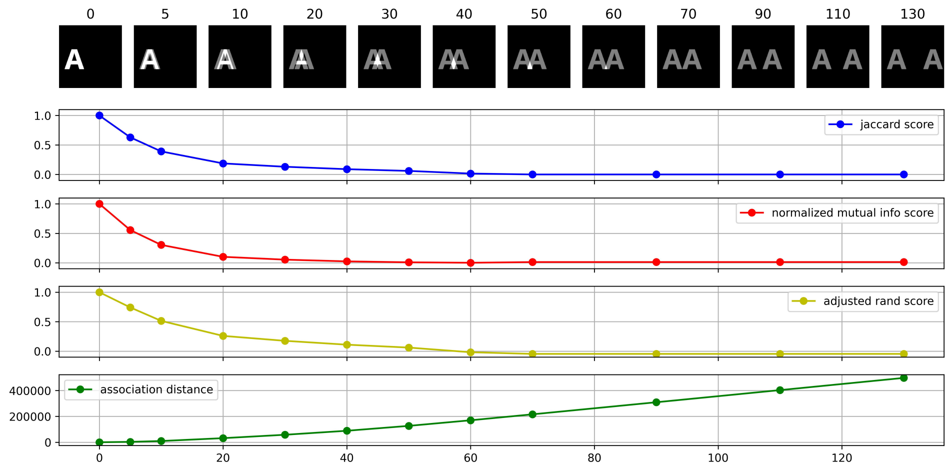


Figure 53: Concepts extracted from a ResNet-18 model traned over the CO synthetic dataset. Concept c_4 related to the missing right part of the character **C**. The concept is non-overlapping, yet, spatially related to character **C** on all images.

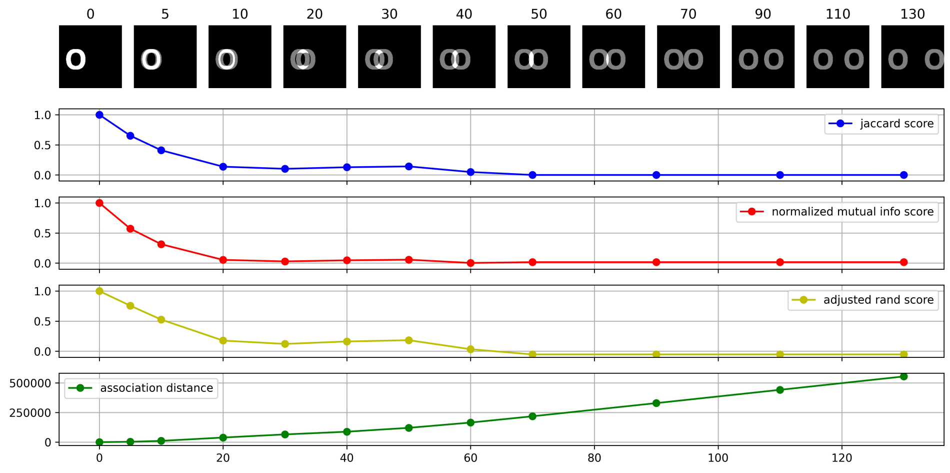
pixels offset on the subfigure 54(b), where the vertical section of the character **O** increases the overlapping. As a consequence, the Jaccard score, normalized mutual information score, and adjusted rand score, stop behaving monotonically w.r.t. the shift between the masks, which is undesired.

In other cases, a concept may represent the surroundings of a feature, or the missing counterpart of a form. In these cases, it is desired that an association metric is able to measure the degree of separation between the feature and the surrounding concept.

The proposed association distance DST_{p_o,c_j} can measure concepts surrounding primitives and express the degree of separation between both masks.. Other metrics only allow the comparison of overlapping regions, which can be problematic.

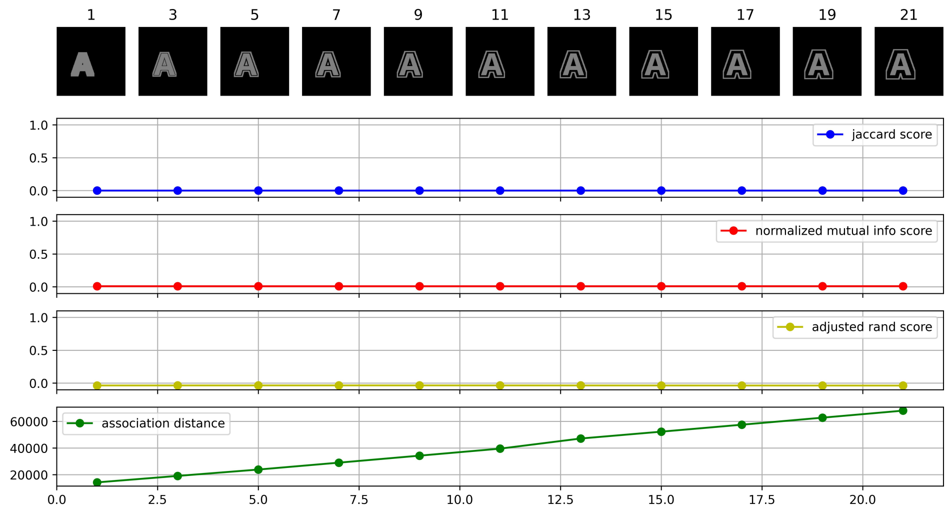


(a) Metric comparison for offsets of the character **A**.

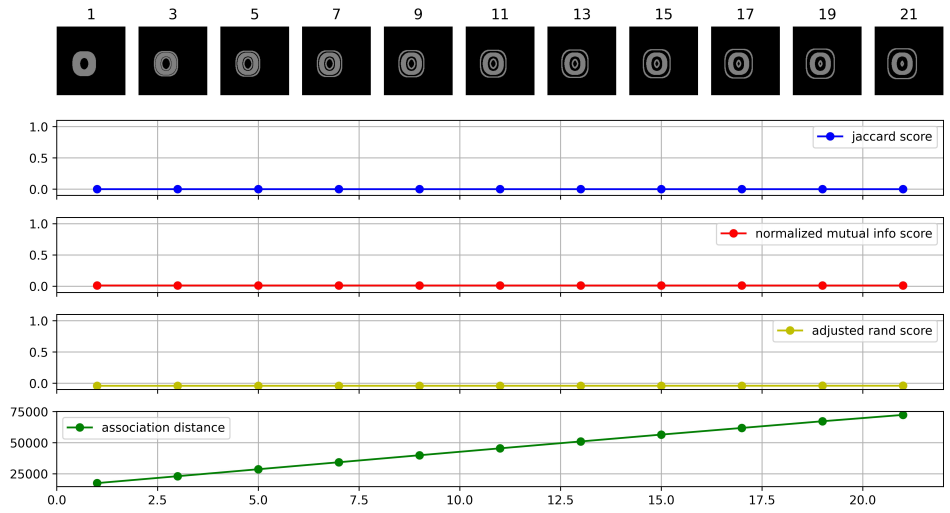


(b) Metric comparison for offsets of the character **O**.

Figure 54: Evolution of metrics at various degrees of overlapping and offset for the character A in subfigure 54(a), and for the character O in subfigure 54(b). In both cases, the jaccard score, normalized mutual information score, and adjusted rand score are proportional to the degree of overlapping, yet, do not show any difference for further offsets of the masks. In comparison, the association distance captures the differences between the masks not only the overlapping cases (d¡70), but also non overlapping shifts.



(a) Metric comparison for features surrounding the character **A**.



(b) Metric comparison for features surrounding the character **O**.

Figure 55: Evolution of metrics as a relate surrounding features distances itself from a shape. subfigure 55(a) contains the comparison of the character **A** and a concept surrounding it, initially including it's center, and subsequently only surrounding its exterior. Similarly, subfigure 55(b) contains the comparison between the character **O** and a surrounding concept. In both cases, the association distance behaves monotonically as the primitive and the concepts distance increases, yet, other metrics cannot capture this behavior.