KI - Künstliche Intelligenz (2022) 36:225–235
https://doi.org/10.1007/s13218-022-00771-9

TECHNICAL CONTRIBUTION

A Neural-Symbolic Approach for Explanation Generation Based on Sub-concept Detection: An Application of Metric Learning for Low-Time-Budget Labeling

Johannes Rabold[1]

Received: 10 February 2022 / Accepted: 21 June 2022 / Published online: 9 July 2022

Abstract
Deep learning methods, although effective in their assigned tasks, are mostly black-boxes with respect to their inner workings. For image classification with CNNs, there exists a variety of visual explanation methods that highlight parts of input images that were relevant for the classification result. But in many domains visual highlighting may not be expressive enough when the classification relies on complex relations within visual concepts. This paper presents an approach to enrich visual explanations with verbal local explanations, emphasizing important relational information. The proposed SymMetric algorithm combines metric learning and inductive logic programming (ILP). Labels given by a human for a small subset of important image parts are first generalized to a neighborhood of similar images using a learned distance metric. The information about labels and their spatial relations is then used to build background knowledge for ILP and ultimately to learn a first-order theory that locally explains the black-box with respect to the given image. The approach is evaluated with the Dogs vs. Cats data set demonstrating the generalization ability of metric learning and with Picasso Faces to illustrate recognition of spatial meaningful constellations of sub-concepts and creation of an expressive explanation.

Keywords Explainable artificial intelligence · Neural-symbolic integration · Metric learning

1 Introduction

Research on explainable artificial intelligence (XAI) provides methods to make decisions of black-box classifiers comprehensible and transparent [1]. What kind of explanations are helpful, depends on the recipient—such as developer, domain expert, or end user—and on the domain. In this paper explanation generation for domain experts based on image data are addressed. Relevant application domains are medical diagnostics [29] or quality control in industrial production. As it is also argued in previous work, in such highly specialized domains, established explanation methods based on visual highlighting typically are not expressive enough to communicate the relevant information [23–25]. For instance, the type of a tumor might depend on its relative position to muscle tissue, or it might be relevant whether some defect is located on a supporting part or not to reject a produced part.

In this work, I present SymMetric, a novel neural-symbolic approach [26] to generate such local relational explanations for images classified with CNNs. Specifically, explanations are based on semantic sub-concepts in an image and their spatial relations. Naming parts in an image (e.g. whiskers, ears, eyes of a cat) is normally an easy task to perform by humans. But a labeling process of per-pixel annotations of human understandable sub-concepts can become tedious when the number of images becomes large. Although there exists a variety of per-pixel labeled data sets, these collections are typically general-purpose data sets for broadly spread research and may not be specialized to the particular information need of the user. My approach aims at providing explanations, even when there is no sufficiently pre-labeled data set at hand. Additionally, human users might have only a certain budget of images they are willing to label. Therefore, the approach in this paper is utilizing a human in the loop that provides a small number of annotations for sub-concepts. Metric learning is then used to automatically label a neighborhood of images around the image whose

✉ Johannes Rabold
johannes.rabold@uni-bamberg.de

1 Cognitive Systems, Otto-Friedrich-Universität, Bamberg, Bavaria, Germany

classification result needs to be explained. The labeled sub-concepts together with extracted spatial relations are generalized with inductive logic programming (ILP) [20] to form first-order logic rules as expressive *verbal explanations*. Before the approach is explained in detail, in the next sections I will briefly position it in the greater context of XAI. Further, I will cover the theoretical background of the used methods.

2 Expressive Explainable Artificial Intelligence

It is generally known that deep learning models are omnipresent in machine learning research and are responsible for many state of the art approaches. A major downside however is their black-box nature [1]. The inner process of how the model came to a decision is mostly obscure and can hardly be audited by a user. To deal with this problem, the emerging field of Explainable Artificial Intelligence (XAI) came up with a number of approaches to break open the black-box and to make model decisions more transparent to users [1].

XAI methods can be classified into either being model-specific (can only be applied to a specific type of models) or model-agnostic (model type does not matter). Further, a distinction is made by whether the explanation is local (the model decision for a single instance is explained) or global (the complete model is explained) [2]. In this work, I describe a model-specific, local method capable of generating expressive verbal explanations of the decision of a trained black-box for a given instance.

Many local explanation approaches for image instances give an attribution score for parts of the original image [3, 19, 27, 30]. Users are then able to quickly see via a color coding on the image which parts of the image are contributing most to the final model decision. As already outlined in previous works [23–25], simply showing what regions are most important is not enough. Most of these approaches are incapable of semantically grasping what the highlighted objects mean to humans. A simple example is the mere highlighting of an eye region in a setting where the model told that a dementia patient suffers from pain [35]. Without further context, it might not become evident that the eye has to be closed in order for the model to output “pain”. In this case, the explanation should transduce the verbal information of the *value* of an attribute in the image.

Another application where verbal context is important is industrial quality control [22]. It might not only be important that a blowhole occurred in a component, but also the information if it sits on a supporting part. This could then mean that the component has to be scrapped. In this scenario, the *relation* between regions located in the image should be part of an explanation.

Introducing verbal, *symbolic* knowledge in the otherwise connectionist world of neural network systems is nothing new. Neural-symbolic integration (NSI) aims at fulfilling the most important cognitive abilities according to [33]: not only learning from experience but also reasoning about what has been learned [11]. NSI yielded several approaches to connect sub-symbolic learning with symbolic reasoning, effectively integrating interpretable background knowledge into an otherwise obscure black-box [4, 9].

In this work, I put my focus on extracting knowledge from already trained models. This is in contrast to several NSI approaches where the symbolic knowledge is a fixed part of the end-to-end differentiable pipeline and learned in the training phase (see citations above). An advantage of implementing a post-hoc approach is the possibility to better involve the user in the explanation process. The training of the model can run as quickly as possible beforehand and the user can then later decide how much time she/he will spend on getting an explanation. Also, the user can decide afterwards what kind of domain knowledge to use (see also end of this chapter).

Efforts have also been made to extract visual features from images and using them as building blocks for symbolic reasoning engines [7]. The extraction is done via finding points of interest with a purely data-driven approach. In contrast, my approach utilizes attribution methods to automatically find parts in the image that are important to a classifier.

An interesting connection between the predictive power of neural networks and the interpretability of symbolic approaches was made in [16] where a set of visual signals is directly incorporated in a logic reasoning engine. The approach needs to have the components over which is reasoned readily available (e.g. the image parts where handwritten characters are present in a Sudoku grid). My approach chooses image parts that are important to the reasoning automatically by going over the complete image instance at once.

With a huge variety in application areas where explanations are paramount, also the number of different users with different explanation needs grows. As also highlighted in [28], there can not be “one size-fits all” and the context in which an explanation is issued has to be taken into account. The paper makes the suggestion of involving the user directly in the explanation finding process, which I will directly pick up in this paper. My approach implements a human in the loop where the user can choose the building blocks of the explanation themselves.

An often overlooked challenge when it comes to quick decision making is the time factor. Users of an ML system critical for safety or health, that only have a low time-budget, might need an almost immediate explanation to base a decision on. Recent work [13] suggests to lower the complexity of an explanation in such scenarios. My approach of utilizing verbal explanations aims at reducing the complexity by narrowing the semantic gap between the natural language understanding of a human user and the processes inside the black-box. Additionally, I incorporate mechanisms that allow for expressive explanation generation by only minimal interaction with the system needed.

3 Theoretical Background

We want to create explanations that are close to the original behavior of the black-box at the locality of a given image. Therefore, visual attribution methods to quantify the importance of image parts with respect to the classification of a given machine learning model are employed:

3.1 GradCAM

As a method of quantifying the importance of instance parts in my approach, GradCAM is used [30]. With this established attribution approach for images, importance values can be assigned to the so called feature vectors on a particular layer L in a black-box neural network B. Let us define a feature vector to be one vector in the tensor output $B_L(s)$ of a convolution layer $L \in B$ when feed forwarding an image s. That is, feature vector A_{ij} is the complete array of all filter neuron responses at one location (i, j) in the tensor of activation maps $A = B_L(s)$. To get an importance value for a feature vector with respect to the classification output $B(s)$, GradCAM first finds contribution values α_k for all activation maps $A_k \in B_L(s)$. α_k is computed by taking the average over the complete matrix result of calculating the derivative of $B(s)$ with respect to A_k. These contribution values are now taken as coefficients for a weighted combination over all activation maps A_k. After feeding the resulting values in a sigmoid function, we are left with a 2D map G of importance values for the feature vectors at each location in the activation maps of $B_l(s)$ with respect to $B(s)$.

3.2 Concept Vector Analysis

Neurons in intermediate layers of CNNs act as feature detectors (aka filters). The assumption that single neurons specialized in detecting very distinct semantic sub-concepts in images was abandoned ever since the paper featuring the Net2Vec approach by [10]. The authors propose a method of quantifying the sub-concept detection power of neurons and concluded that it usually takes multiple neurons that contribute to detecting a given concept. By finding weights for a linear combination of neuron outputs, they propose an approach to find embeddings for given sub-concepts; the weights constituting an embedding vector. They train linear classifiers to detect the occurrence of a sub-concept at a particular image location given by the feature vector. This strain of research suggests that there exist linear transformations of feature vectors that put vectors emerging from similar visual sub-concepts in the image closer together in terms of a particular similarity metric. Fong and Vedaldi [10] suggest the cosine similarity (see Eq. 1) to be suited best, taking knowledge from word embeddings in text classification [18].

$$D(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{||\mathbf{x}||\ ||\mathbf{y}||} \tag{1}$$

3.3 Metric Learning

The field of research concerned with the supervised learning of custom distance metrics and their respective transformations on vectors is called *metric learning* [6, 15]. Given a distance function $D(x, y)$, e.g. the cosine distance between vectors x and y, metric learning aims to find a new distance function $D'(x, y)$ that is “better suited” for calculating similarity between vectors in a given set V. Usually the set contains labeled vectors and a supervised metric learning approach constructs D' in such a way that vectors with the same label are calculated to be closer together by D'. For this purpose, the Large Margin Nearest Neighbor algorithm (LMNN) is employed [34]. As the name suggests, LMNN aims at learning a D' that, when used as metric for a traditional k nearest neighbor classification yields high accuracy on a given labeled data set of points V. D' is formulated as a generalization of the Euclidean distance function, referred to as Mahalanobis distance (see Eq. 2; Σ denotes the covariance matrix of data V).

$$D'(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^\top \Sigma^{-1} (\mathbf{x} - \mathbf{y})} \tag{2}$$

The algorithm distinguishes between target points T and impostor points I. T's are pre-calculated as the $|T_v| = k$ nearest neighbors for all $v \in V$ that share the same label with v. I_v's are such points for all $v \in V$ that are among v's k nearest neighbors, but do not share the same label with v. LMNN is stated as an optimization problem on the covariance matrix Σ with a twofold goal: minimizing the Mahalanobis distance between instances $v \in V$ and their respective target points T_v and penalizing distances to I_v's that are closer to v than the T_v's plus one unit. This ensures that a large margin is established between the preferred target points and the unwanted impostor points. In practice, the learned function can be regarded as a transformation θ for the original data V which can then be applied to the vectors in the data yielding $V' = apply(V, \theta)$. The original function D can then be used as usual on the transformed data V'. That way, e.g. clustering algorithms can use traditional metrics like the cosine distance.

3.4 Inductive Logic Programming

In 1988, Michie [17] came up with three levels of criteria evaluating machine learning approaches: weak, strong and ultra strong machine learning. While weak ML systems merely become better in their predictive power when provided with more data, strong ML systems provide their hypotheses in human understandable symbolic form. An ultra-strong ML system is further able to provide a user with a knowledge gain beyond simply studying the training data set.

Inductive logic programming (ILP) is a field of machine learning approaches, that aim at inducing logic programs from symbolic examples and background knowledge (BK) [20]. The symbolic nature of their induced theses puts ILP systems into the *strong* category of Michies scale. The conducted experiments of [21] further show that hypotheses induced by ILP can help humans to better understand the underlying concept of a given ML task, lifting ILP to the ultra-strong level of Michies scale. Thus, I claim that ILP is perfectly suited for generating explanations for the decision of black-box classifiers.

Unlike with neural networks (numerical) or decision trees (numerical/categorical), the input of ILP consists of a collection of first-order logic literals. The examples are e.g. given as facts (e.g. `okay(e1)`, `not okay(e2)`) where the predicate indicates membership to either the positive or negative class. The BK literals act as means to further describing the examples symbolically (e.g. `contains(e1, p42)`, `left_of(p42, p43)` stating that example `e1` contains part `p42` etc). ILP methods are generally designed to induce hypotheses that classify as many positive examples as possible to be positive by avoiding to classify as many negative examples to be positive. The hypotheses are constructed by using predicates from the BK to form a set of first-order logic clauses.

The ILP framework Aleph [32] is a general purpose library, that allows flexible induction of first-order hypotheses from examples and BK. Aleph induces a hypothesis comprised of disjunctively connected first-order horn clauses, given positive (E^+) and negative (E^-) examples. The hypothesis is induced by using the BK literals to build preconditions (rule bodies) to the rules in the hypothesis. A mode-guided specific-to-general refinement search builds rules that entail as many positive examples as possible by not entailing the negative ones. The general steps of the algorithm are as follows:

1. Select a positive example $e \in E^+$. If E^+ is empty, halt.
2. Construct the most-specific clause C that entails e and that is in the language-constraints imposed by the modes.
3. Generalize the clause by removing literals from it. The new subset of literals is found by maximizing a user-defined score function (usually defined as a best coverage fit for the examples over the clause).
4. Add the clause with the best score to the hypothesis. All examples covered by this clause are removed from the set of examples.
5. Repeat from step 1.

An example rule that could be induced from examples and BK can look as follows:

$$\texttt{okay(A)} : -\texttt{contains(A,B)}, \texttt{contains(A,C)}, \texttt{left\_of(B,C)}.$$

Note that the syntax of the logic programming language Prolog is used where variables are upper case and constants are lower case letters. Rules start with the rule head followed by `:-` (a left facing arrow) and the conjunctively connected preconditions (The conjunction is expressed by a comma). The rule ends with a period. The above example can be interpreted as follows: instance `A` is okay if it contains parts `B` and `C` and `B` is left of `C` in the image instance.

4 SymMetric: Finding Verbal Explanations for CNN Classification Results

In the following, the SymMetric approach of obtaining an expressive local explanation for a black-box classification result of a given image is outlined. Assume a black-box B given as a CNN that was pre-trained on a binary classification task. B's architecture and parameters need to be able to be examined (model-specific approach). Assume further an image instance P, that was assigned a positive classification result $B(P) = \oplus$ and available images from a test set T that was never seen before during the training of B. Figure 1 shows the general workflow of SymMetric as process graph.

4.1 Most Important Feature Vectors for Symbolic Explanations

To find an explanation that explains B in the vicinity of the given instance P, first a neighborhood S of user-defined size N of positive and negative instances s_i that are close to P according to a similarity metric D is obtained. It is ensured that S contains $N/2$ positive and negative images. Since full access to the structure and parameters of B is assumed, we can define D as a similarity metric between the output vectors $B_{FE}(s_i)$ of the flattening layer, that is, the output of the last layer of the feature extractor of B when feed-forwarding image instances s_i. For the experiments, the cosine similarity metric (see Eq. 1) has turned out to work best. I chose this model-specific approach over image-specific similarity

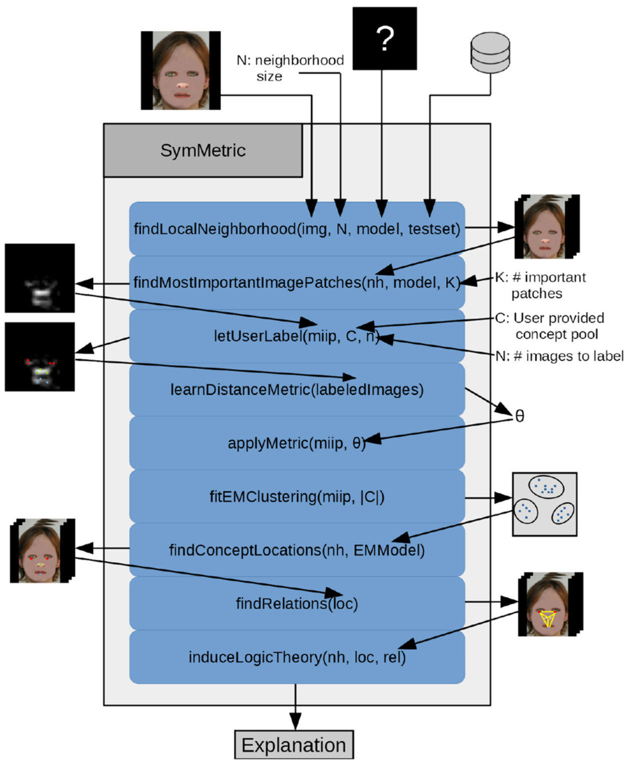
N: neighborhood size
?
SymMetric
findLocalNeighborhood(img, N, model, testset)
findMostImportantImagePatches(nh, model, K)
K: # important patches
letUserLabel(miip, C, n)
C: User provided concept pool
N: # images to label
learnDistanceMetric(labeledImages)
θ
applyMetric(miip, θ)
fitEMClustering(miip, |C|)
findConceptLocations(nh, EMModel)
findRelations(loc)
induceLogicTheory(nh, loc, rel)
Explanation

Fig. 1 The process graph for SymMetric yielding an expressive verbal local explanation for the classification of an image

measures to stay close to the network we want to explain and to not rely on data specific features.

This work assumes, that in order to find explanations that are close to the local behavior of B with respect to instance P, we need to incorporate parts of the instance that are important for B to reach the classification result of P. I utilize GradCAM to find the most important locations for all images $s \in S$. As input for GradCAM, I use the activation output $A = B_L(s)$ of the black-box when feed-forwarding s. For each instance, we are only interested in the K most important feature vectors. Each feature vector in A has a corresponding pixel patch at a specific location in s. Assume for example a layer $L \in B$ that contains half as many feature vectors in every spatial direction as the input image. The feature vector in the upper left corner would then typically correspond to the 2×2 pixel patch in the upper left corner of the input image etc. Figure 2 makes this correspondence between feature vectors and input image pixels clearer. The entirety of the K most important feature vectors and their corresponding pixel patches for all $s \in S$ is stored in the set V^*.

4.2 Low-Time-Budget User Labeling

The approach in this paper actively involves the user that has a particular explanation need. She or he labels image patches by taking labels of a pool of sub-concepts also given by the user. That way, we are not bound to an already per-pixel labeled data set like it is needed for Net2Vec.

In many application domains, the user might not have much time to put in extra effort for receiving an explanation. The experiments will show that only a small set of labeled image patches is sufficient to generalize the user annotation to unlabeled image patches sharing similar sub-concept content.

The approach to obtain the initial labels for the feature vectors is as follows:

1. Let a user input categories C into the system
2. The user is presented with the image patches taken from V^* corresponding to the most important feature vectors

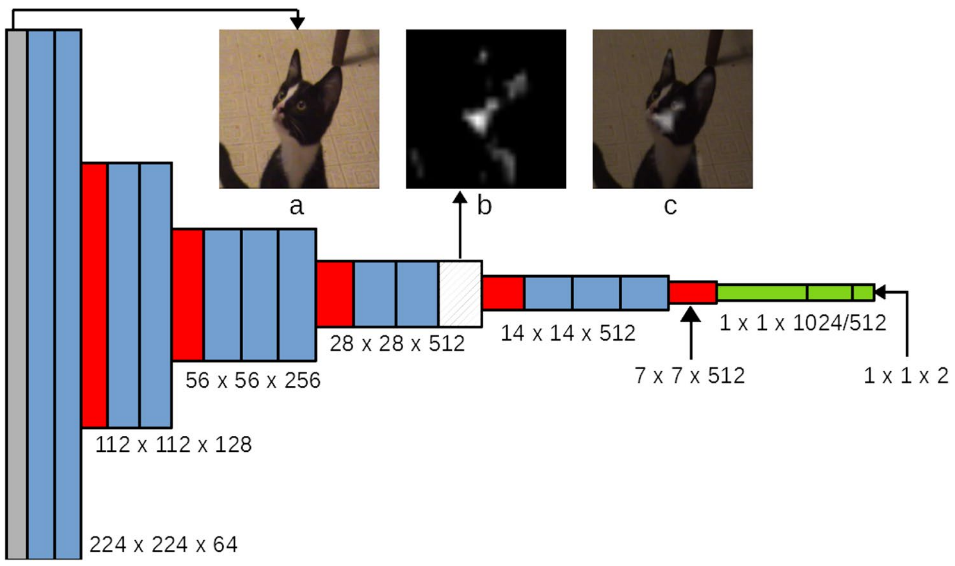

Fig. 2 The model architecture used in the experiments. Figure and architecture adapted from [31]. The last convolution layer in the fourth block (shaded) is used for extracting feature vectors. The most important feature vectors in this layer according to GradCAM when inputting image a are highlighted in heatmap b. This heatmap has a dimension of 28×28. c shows the overlap between a and b where b was up-sampled to have the same size as a. Therefore, one feature vector location corresponds to an 8×8 pixel patch in a

in a small given amount $n < |S|$ of positive and negative image instances from neighborhood S (see Sect. 4.1).
3. She/he has to assign a label to them taking categories $c \in C$ forming the set `Lab`

Taking this human in the loop approach is favorable over automatic approaches, since it gives the users the opportunity to select the semantic sub-concepts they find most informative for the target audience of the explanation. I nevertheless want to emphasize, that it is also possible to use per-pixel ground truth masks for sub-concepts if available. Bau et al. [5] for example have introduced the BRODEN data set of per-pixel segmented images showing where on the images certain colors, textures, object parts etc. can be found. The human labeling procedure from above could then be exchanged with using such a data set to form vicinity S. Of course, one has to make sure that the used data set is comparable with the original data source, meaning that similar semantic concepts are labeled.

SymMetric now performs supervised metric learning with LMNN on the labeled vectors `Lab` to find a transformation θ that brings feature vectors of similar semantics closer together according to the cosine similarity. Further, the mean vector $\overline{\mathrm{Lab}_c}$ is calculated for all labels $c \in C$ in the labeled vectors. θ is now applied to all most important feature vectors in V^* of all image instances in neighborhood S. That way, we achieve a space where also the unlabeled feature vectors are closer to their respective semantically similar vectors in terms of the cosine similarity.

4.3 Symbolic Explanation Generation with ILP

The transformed vectors V^* are now clustered by expectation maximization (EM) clustering, taking as number of Gaussian components the number of categories $|C|$ and yielding mean vectors $\overline{\mathrm{Clus}_c}$ for $c \in C$. The cluster defining category $c^* \in C$ for a particular cluster c^0 is found by Eq. (3) (D is the cosine similarity as given by Eq. 1).

$$c^* = argmax_c[D(\overline{\mathrm{Clus}_{c^0}}, \overline{\mathrm{Lab}_c})] \tag{3}$$

Following the terminology of [10], I will call the mean vectors $\overline{\mathrm{Clus}_c}$ *sub-concept vectors*. We then are able to localize sub-concepts on all images of neighborhood S (also the unlabeled ones) automatically by going over each sub-concept $c \in C$ and finding the image location $\langle x^*, y^* \rangle$ of the transformed feature vector which is closest to the respective sub-concept vector of c according to the cosine similarity. Note that we now are not restricting ourselves to the previously found most important feature vectors V^* but we take all feature vectors V at all locations of the images into account. Given one image s, a concept c, a function A over positions xy giving the feature vector at that position, the learned transformation θ and cosine similarity D, the position of the sub-concept c is thus found with Eq. (4).

$$\langle x^*, y^* \rangle = argmax_{\langle x,y \rangle}[D(\theta(A_{xy}(s)), \overline{\mathrm{Clus}_c})] \tag{4}$$

The found location will mark the single point location of c in a given image and is the first step in finding symbolic representations of the images in our neighborhood.

In the following, I will describe how inductive logic programming (ILP) is used to generalize over the symbolic representations of the images to create verbal explanations:

In order to obtain symbolic background knowledge (BK) for Aleph to get a local explanation for the original image instance P, we first get all sub-concepts present in the images s in neighborhood S as explained above. We construct the BK with containment literals, e.g. `contains(s1, p1).` `concept(p1, eye)` to state that image `s1` contains an image patch `p1` that is of concept `eye`. To enrich the background knowledge for the images, we additionally find spatial relations between the sub-concepts. We only use the four relations `left_of`, `right_of`, `top_of` and `bottom_of` for now. Literals such as `left_of(p1, p2)` or `top_of(p3, p2)` are then added to the BK. We find these in a straight forward fashion by comparing the coordinates of the found sub-concepts. E.g. for the `top_of` relation we scan if there is a sub-concept present in the 45° section facing upwards in the image. Finally, Aleph is used to generate explanations by inducing first-order logic rules.

4.4 Algorithm

Algorithm 1 shows the precise steps that SymMetric takes. As an input it requires the original image instance P as well as a trained CNN black-box B whose output $B(P)$ we want to explain. The algorithm further requires test instances T, a convolution layer $L \in B$, the neighborhood size N, a smaller number of images $n < N$ that a user can label and the number of most important images patches K. As an output, the algorithm gives a first-order logic hypothesis T consisting of logic rules.

Algorithm 1 SymMetric: Generate local verbal explanations for the classification result of convolutional neural networks

```
Require: Original positive image instance $P$, Test set of images $T$, CNN black-box $B$, Convolution layer $L \in B$, neighborhood
    size $N$, number of images to label $n < N$, number of most important image patches $K$
 1: $S \leftarrow N$ nearest positive and negative image instances $t \in T$ to $P$ with respect to the cosine similarity metric applied on
    the output of the feature extractor $B_{FE}(t)$
 2: $V \leftarrow \{\}$
 3: for all $s \in S$ do
 4:   $A \leftarrow B_L(s)$ {Get activation output on layer $L$}
 5:   $G = (g_{ij})_{i=1,\dots,\text{HEIGHT}(A);j=1,\dots,\text{WIDTH}(A)} \leftarrow \text{GradCAM}(s, B_L(s), B(s))$ {Find the importance map with Grad-
      CAM}
 6:   patches $\leftarrow$ Upsample$(A, s)$ {Get original image patches by up-sampling the locations of $A$ to the size of $s$}
 7:   $v_s \leftarrow \{\}$
 8:   for $i = 1, \dots, \text{HEIGHT}(A); j = 1, \dots, \text{WIDTH}(A)$ do
 9:     $v_s \leftarrow v_s \cup \{\langle A_{ij}, \text{patches}_{ij}, G_{ij}\rangle\}$
10:   end for
11:   $V \leftarrow V \cup \{\{v_s\}\}$ {Add the vectors, image patches and importance information for each image $s$}
12: end for
13: $V^* \leftarrow$ MostImportant$(V, K)$ {For each image we build $V^*$ to only contain the $K$ most important locations}
14: $V_{anno} \leftarrow$ Take$(V^*, n)$ {Take $n$ images from $V^*$}
15: $C \leftarrow$ AskUserCategories
16: Lab $\leftarrow$ LetUserLabel$(V_{anno}, C)$
17: $\theta \leftarrow$ LearnTransformation(Lab) {Supervised Metric Learning}
18: $V^* \leftarrow$ apply$(V^*, \theta)$
19: Clus $\leftarrow$ EM-Clustering$(V^*, |C|)$ {Clustering with $|C|$ components}
20: $E^+ \leftarrow \{\}$
21: $E^- \leftarrow \{\}$
22: for all $v \in V$, $s \in S$ do
23:   $v \leftarrow apply(v, \theta)$ {Apply $\theta$ to all vectors in $v$}
24:   $v' \leftarrow$ predict$(v, \text{Clus})$ {Predict the concepts for all vectors in $v$}
25:   Parts $\leftarrow \{\}$
26:   for all $c \in C$ do
27:     $p_c \leftarrow$ FindHighestCosineSim$(v', c)$ {Find the location of the vector with the highest cosine similarity to the sub-
        concept vector of $c$}
28:     Parts $\leftarrow$ Parts $\cup \{\langle c, p_c\rangle\}$
29:   end for
30:   Rels $\leftarrow$ CalculateRelations(Parts)
31:   if $B(s) = \oplus$ then
32:     $E^+ \leftarrow E^+ \cup \{\langle\text{Parts, Rels}\rangle\}$ {Symbolic representation of images together with background knowledge}
33:   else
34:     $E^- \leftarrow E^- \cup \{\langle\text{Parts, Rels}\rangle\}$
35:   end if
36: end for
37: $T \leftarrow$ Aleph$(E^+, E^-)$ {Let Aleph induce a set of first-order logic rules}
38: return $T$
```

5 Conducted Experiments and Results

In the following, I discuss the conducted experiments as well as the results. For all of the experiments in this section I used a slightly altered version of the widely used VGG-16 architecture as black-box B [31]. In particular, I changed the two fully-connected layers from both 4096 to 1024 and 512 neurons and the last sigmoid layer to output 2 class estimator values. The changes in the layers are due to the significant reduction of classes from 1000 to 2. Figure 2 shows the altered architecture. For fine-tuning this network, I only keep the parameters of the first three blocks of convolution layers fixed. I took for L the last convolution layer in the 4th convolution block of B as layer to extract our feature vectors since I assume sub-concepts to emerge in higher layers rather than lower layers that typically consist of detectors for low-level features (see the introductory chapter in [12]). The selection procedure of the best layer can become a tedious process which is not in the scope of this paper but is an important future work. Fong and Vedaldi [10] and Bau et al. [5], the work they built upon, provide a good starting point for finding suited selection heuristics. The training, testing and experiments were run on our in-house GPU workstation with the following specifications: 24 CPUs of type

Table 1 Accuracy and F1 score on the training and test data sets

	Accuracy	F1 score
Train data	0.9713	0.9706
Test data	1.0	1.0

The respective train and test scores closely match and indicate that the degree of over-fitting is negligible

Table 2 Average of the respective image maximums of the cosine similarity to the sub-concept vector on previously unlabeled images

	Eye	Ear	Whiskers
Without metric learning	0.31 (0.10)	0.33 (0.11)	0.25 (0.06)
With metric learning	**0.35** (0.12)	**0.34** (0.12)	**0.29** (0.07)

Bold values indicate that there was an improvement when using metric learning when compared with the case without metric learning

The standard deviation is given in parentheses

AMD™ Ryzen™ Threadripper™ 2920X 12-Core Processor. One NVIDIA® GeForce® RTX 2080 Ti with 11019 MiB.

As a first evaluation of how the approach generalizes parsimonious sub-concept labeling from just a few examples to a larger image data set, I report results on the Kaggle Cats and Dogs data set published by Microsoft® [8]. The network was fine-tuned on the task of discriminating between images of cats and dogs using a balanced data set of 16,797 cat and dog images. After one epoch I stopped training and received an accuracy of 100 percent on 7200 test images (70:30 split). By checking the metrics on the training and test sets (see Table 1), I made sure that over-fitting is no issue.

I randomly sampled $N = 20$ images predicted to contain a cat, constituting S. The images came from a separate experimentation set of 1000 cat and dog images that were not part of the training or test set. GradCAM yielded the $K = 10$ most important image patches and corresponding feature vectors from L for each image in S. For $n = 4$ images I annotated the K most important image patches by hand. For this data set I took as categories $C = \{\text{eye}, \text{ear}, \text{whiskers}\}$. Whenever none of these categories was present in the patch, I did not consider this region in the following. After labeling, I related the respective feature vectors to that patches and performed metric learning on the vectors of these few images. I took LMNN as the supervised metric learning algorithm which yielded a transformation θ for our labeled feature vectors. Applying this transformation to all important feature vectors in S leaves us with vectors in a vector space we can cluster according to the cosine distance metric. Using EM clustering and $|C| = 3$ as the number of clusters yields sub-concept vectors representing our categories. I then transformed all feature vectors at all locations of the images in S and predicted the category cluster for them. This lets us generate heatmaps for the images with 3 heatmaps per image for the 3 categories. The heatmaps in Fig. 3 show the cosine similarity of the feature vectors to the respective sub-concept vector of 6 randomly selected images not labeled by hand.

Table 2 shows the maximum of the cosine similarity to the respective sub-concept vector averaged over $N - n = 16$ neighborhood images that were not labeled. The values are given for an experiment run with and without transforming the feature vectors using metric learning. The similarities

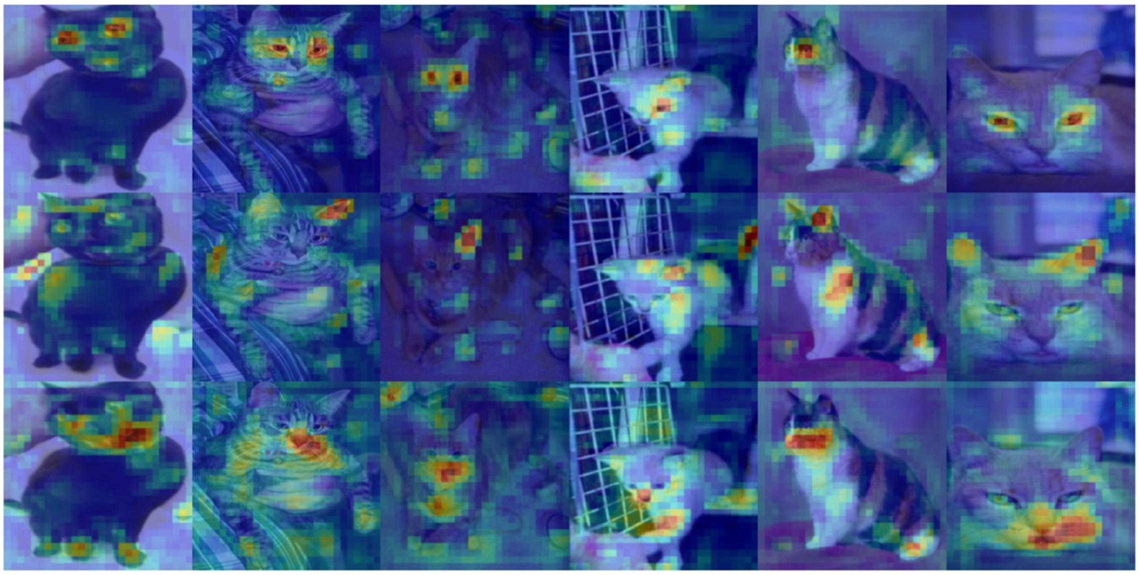

Fig. 3 Exemplary images with regions of highest cosine similarity to the respective sub-concept vector. Rows from top to bottom are the sub-concepts EYE, EAR, WHISKERS

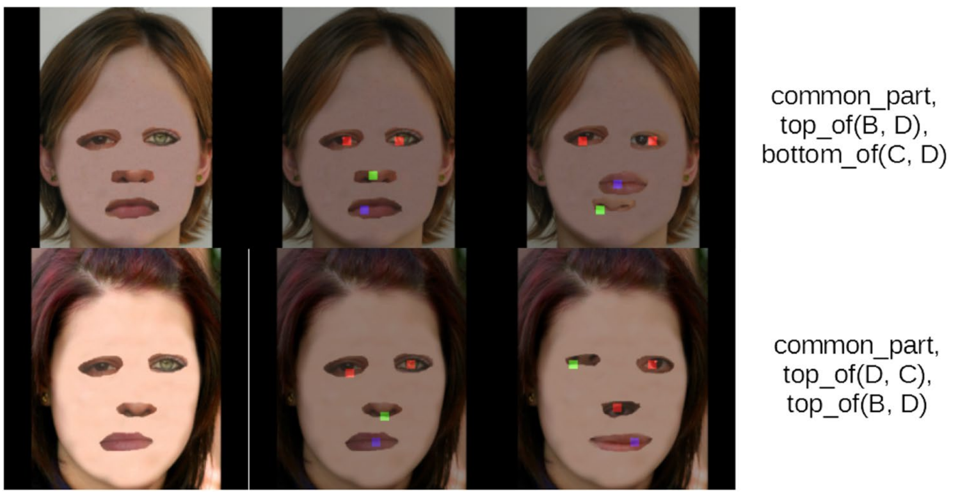

Fig. 4 First column shows the original images, second and third column the locations of the found sub-concepts for the original image and the nearest negative example respectively. Last column states the found rule. The common rule part is `face(A):- contains (A, B), contains (A, C), contains (A, D), concept (D, nose)`

calculated for the experimental setup with metric learning are always higher indicating a better fit to the labeled ground truth when compared to taking the un-transformed "raw" vectors from the probed network.

To illustrate the usability of the approach to recognize meaningful spatial relations, I subsequently present results for the expressive verbal explanation generation. As a sample data set I use the Picasso Faces data set we already created in [25]. The data set is derived from the FASSEG data set [14] of frontal face images. The underlying task of the data set was altered towards a visual relational classification task. The faces of selected individuals in FASSEG were rid of facial features like eyes, nose and mouth and were given a similar skin texture all over the face. These "canvases" of faces were then used to construct either faces where the constellations of facial features resemble "correct" faces (positive class) or where the spatial structure does not render a standard face (negative class) (see Fig. 4 for examples). The facial features were taken from a pool of features coming from the original FASSEG data set. I made sure that all the images contain exactly two eyes, one nose and one mouth. For the conducted experiment, I proceeded nearly identical to the experiment above with the following data set dependent changes: fine-tuning on a balanced data set of 9002 positive and negative images was stopped after 1 episode with an accuracy of 100 percent on 998 images. GradCAM was fed with $K = 10$ and I annotated $n = 4$ out of $N = 20$ images with categories $C = \{\text{eye}, \text{nose}, \text{mouth}\}$. LMNN was used again for the supervised metric learning approach and the learned transformation was applied to all important feature vectors in the neighborhood. Learning an EM clustering model on these vectors and predicting all transformed feature vectors in the neighborhood leaves us with a completely labeled neighborhood. Next, I transformed the images to symbolic examples with background knowledge (BK). I found the location of each sub-concept (`eye`, `nose`, `mouth`) on each image $s \in S$ by finding the location of the image patch whose feature vector is most similar to the sub-concept vector (see Sect. 4.3). For the eyes who appear two times on each image we have to deviate slightly from that approach. In order to get two points for two eyes, we do not find the closest vector in the entirety of the vectors that are part of the eye cluster. We rather find the two "super patches" that are formed by coherent patches but whose member patches have no neighbors from the respective other super patch. For each image, we now can write literals like `face(s1)` or `not face(s2)` depending on whether s_i represents a positive or negative image instance. We further write background knowledge for the occurrence of a sub-concept in an image (e.g. `contains(s1, p1)`, `concept(p1, eye)` etc.) as well as the derived spatial relations between the sub-concepts (e.g. `left_of(p1, p2)`, `bottom_of(p2, p3)`). The last step consists of inducing general rules from the examples and the BK. See Fig. 4 for the rules as well as intermediate data collected during the experiment.

The explanations are generalizations over the symbolic examples. I explicitly allowed variables to be contained in the rules. Both rules tell us that there has to be a nose in between two other sub-concepts. Substituting the variables by constants (in this case the sub-concepts) such that the resulting rule entails the original image example yields `B` to be an eye and `C` to be a mouth. This is on par with the construction of the positive examples, i.e. a normal face.

6 Discussion and Further Work

I have demonstrated that SymMetric provides users with verbal explanations that go beyond visualization of important image parts. Parsimonious specialized labeling can be generalized by using metric learning, creating a neighborhood of symbolic data around the original image instance. The metric learning helps locating sub-concepts on unlabeled data that are more similar to a given sub-concept vector with respect to the cosine similarity compared to an approach without metric learning. This can become particularly important in domains where labeled data is not available (e.g. in a specialized medical field) and practitioners want to receive expressive explanations with minimal additional effort. In the future, the claim of the helpfulness of the generated explanations needs to be further backed up with a user study aimed at practitioners that are working with black-box decision systems. Comparing performance on a particular task before and after being exposed to an explanation can demonstrate the usefulness of it. In subsequent work I also plan on investigating the effectiveness of this approach on data sets that go beyond a proof of concept.

Commenting on the potential negative societal impact of my work I believe the stated approach generally attempts to mitigate issues like the inability of auditing automated decision making. I aim at establishing trust that is justified by correctly stating the reasons for a classification decision. Highlighting spatial relations might be a first step towards a holistic auditing process, but I am aware that this is most likely still short of a complete answer.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. Adadi A, Berrada M (2018) Peeking inside the black-box: a survey on explainable artificial intelligence (xai). IEEE Access 6:52138–52160
2. Arrieta AB, Díaz-Rodríguez N, Del Ser J, Bennetot A, Tabik S, Barbado A, García S, Gil-López S, Molina D, Benjamins R et al (2020) Explainable artificial intelligence (xai): concepts, taxonomies, opportunities and challenges toward responsible ai. Inf Fusion 58:82–115
3. Bach S, Binder A, Montavon G, Klauschen F, Müller K-R, Samek W (2015) On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS One 10(7):e013040
4. Badreddine S, Garcez AD, Serafini L (2022) Logic tensor networks. Artif Intell 303:103649
5. Bau D, Zhou B, Khosla A, Oliva A, Torralba A (2017) Network dissection: quantifying interpretability of deep visual representations. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, Manhattan, pp 6541–6549
6. Bellet A, Habrard A, Sebban M (2015) Metric learning. Synth Lect Artif Intell Mach Learn 9(1):1–151
7. Dai W-Z, Muggleton S, Wen J, Tamaddoni-Nezhad A, Zhou Z-H (2017) Logical vision: one-shot meta-interpretive learning from real images. In: International conference on inductive logic programming. Springer, pp 46–62
8. Elson J, Douceur JR, Howell J, Saul J (2007) Asirra: a captcha that exploits interest-aligned manual image categorization. ACM Conf Comput Commun Secur 7:366–374
9. Evans R, Grefenstette E (2018) Learning explanatory rules from noisy data. J Artif Intell Res 61:1–64
10. Fong R, Vedaldi A (2018) Net2vec: quantifying and explaining how concepts are encoded by filters in deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. IEEE, Manhattan, pp 8730–8738
11. Garcez AD, Gori M, Lamb LC, Serafini L, Spranger M, Tran SN (2019) Neural-symbolic computing: an effective methodology for principled integration of machine learning and reasoning. arXiv preprint arXiv:1905.06088
12. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press. http://www.deeplearningbook.org
13. Guidotti R, Monreale A, Ruggieri S, Turini F, Giannotti F, Pedreschi D (2018) A survey of methods for explaining black box models. ACM Comput Surv (CSUR) 51(5):1–42
14. Khan K, Mauro M, Leonardi R (2015) Multi-class semantic segmentation of faces. In: 2015 IEEE international conference on image processing (ICIP). IEEE, pp 827–831
15. Kulis B et al (2012) Metric learning: a survey. Found Trends Mach Learn 5(4):287–364
16. Manhaeve R, Dumancic S, Kimmig A, Demeester T, De Raedt L (2018) Deepproblog: neural probabilistic logic programming. Adv Neural Inf Process Syst 31:3749–3759
17. Michie D (1988) Machine learning in the next five years. In: Proceedings of the 3rd European conference on European working session on learning. Pitman Publishing, Inc, Marshfield, pp 107–122
18. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. arXiv preprint arXiv:1310.4546
19. Montavon G, Lapuschkin S, Binder A, Samek W, Müller K-R (2017) Explaining nonlinear classification decisions with deep Taylor decomposition. Pattern Recognit 65:211–222
20. Muggleton S, De Raedt L (1994) Inductive logic programming: theory and methods. J Logic Program 19:629–679
21. Muggleton SH, Schmid U, Zeller C, Tamaddoni-Nezhad A, Besold T (2018) Ultra-strong machine learning: comprehensibility of programs learned with ilp. Mach Learn 107(7):1119–1140
22. Müller D, März M, Scheele S, Schmid U (2022) An interactive explanatory ai system for industrial quality control
23. Rabold J, Siebers M, Schmid U (2018) Explaining black-box classifiers with ilp-empowering lime with aleph to approximate nonlinear decisions with relational rules. In: International conference on inductive logic programming. Springer, pp 105–117
24. Rabold J, Deininger H, Siebers M, Schmid U (2019) Enriching visual with verbal explanations for relational concepts-combining lime with aleph. In: Joint European conference on machine learning and knowledge discovery in databases. Springer, pp 180–192

25. Rabold J, Schwalbe G, Schmid U (2020) Expressive explanations of DNNS by combining concept analysis with ilp. In: German conference on artificial intelligence (Künstliche Intelligenz). Springer, pp 148–162
26. Raedt LD, Dumancic S, Manhaeve R, Marra G (2020) From statistical relational to neuro-symbolic artificial intelligence. In: Bessiere C (ed) Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI2020, pp 4943–4950. ijcai.org. https://doi.org/10.24963/ijcai.2020/688
27. Ribeiro MT, Singh S, Guestrin C (2016) Why should i trust you? Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. Association for Computing Machinery, New York, pp 1135–1144
28. Schmid U (2021) Interactive learning with mutual explanations in relational domains. Human-like machine intelligence. pp 338
29. Schmid U, Finzel B (2020)Mutual explanations for cooperative decision making in medicine. KI-Künstliche Intelligenz. pp 1–7
30. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D (2017) Grad-cam: visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. IEEE, Piscataway, NJ, pp 618–626
31. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint, arXiv:1409.1556
32. Srinivasan A (2007) The Aleph Manual. https://www.cs.ox.ac.uk/activities/programinduction/Aleph/aleph.html. Accessed 10 Feb 2022
33. Valiant LG (2003) Three problems in computer science. J ACM (JACM) 50(1):96–99
34. Weinberger KQ, Saul LK (2009) Distance metric learning for large margin nearest neighbor classification. J Mach Learn Res 10(2):207–244
35. Weitz K, Hassan T, Schmid U, Garbas J-U (2019) Deep-learned faces of pain and emotions: elucidating the differences of facial expressions with the help of explainable ai methods. tm-Technisches Messen 86(7–8):404–412