EXplainable Neural-Symbolic Learning (*X-NeSyL*) methodology to fuse deep learning representations with expert knowledge graphs: the MonuMAI cultural heritage use case

Natalia Díaz-Rodríguez[*a,f,c], Alberto Lamas[*f,c], Jules Sanchez[*a], Gianni Franchi[a], Ivan Donadello[b], Siham Tabik[f,c], David Filliat[a], Policarpo Cruz[e], Rosana Montes[c,g], Francisco Herrera[f,c,d]

[a] *U2IS, ENSTA, Institut Polytechnique Paris and Inria Flowers, 91762, Palaiseau, France*
[b] *Free University of Bozen-Bolzano, 39100, Italy*
[c] *DaSCI Andalusian Institute in Data Science and Computational Intelligence, University of Granada, 18071 Granada, Spain*
[d] *Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, 21589, Saudi Arabia*
[e] *Department of Art History, University of Granada, 18071 Granada, Spain*
[f] *Department of Computer Science and Artificial Intelligence, University of Granada, 18071 Granada, Spain*
[g] *Department of Software Engineering, University of Granada, 18071 Granada, Spain*

Abstract

The latest Deep Learning (DL) models for detection and classification have achieved an unprecedented performance over classical machine learning algorithms. However, DL models are black-box methods hard to debug, interpret, and certify. DL alone cannot provide explanations that can be validated by a non technical audience such as end-users or domain experts. In contrast, symbolic AI systems that convert concepts into rules or symbols –such as knowledge graphs– are easier to explain. However, they present lower generalisation and scaling capabilities. A very important challenge is to fuse DL representations with expert knowledge. One way to address this challenge, as well as the performance-explainability trade-off is by leveraging the best of both streams without obviating domain expert knowledge. In this paper, we tackle such problem by considering the symbolic knowledge is expressed in form of a domain expert knowledge graph. We present the eXplainable Neural-symbolic learning (*X-NeSyL*) methodology, designed to learn both symbolic and deep representations, together with an explainability metric to assess the level of alignment of machine and human expert explanations. The ultimate objective is to fuse DL representations with expert domain knowledge during the learning process so it serves as a sound basis for explainability. In particular, X-NeSyL methodology involves the concrete use of two notions of explanation, both at inference and training time respectively: 1) *EXPLANet*: Expert-aligned eXplainable Part-based cLAssifier NETwork Architecture, a compositional convolutional neural network that makes use of symbolic representations, and 2) *SHAP-Backprop*, an explainable AI-informed training procedure that corrects and guides the DL process to align with such symbolic representations in form of knowledge graphs. We showcase X-NeSyL methodology using MonuMAI dataset for monument facade image classification, and demonstrate that with our approach, it is possible to improve explainability at the same time as performance.

Keywords: Explainable Artificial Intelligence, Deep Learning, Neural-symbolic learning, Expert Knowledge Graphs, Compositionality, Part-based Object Detection and Classification

[*] Equal contribution. Corresponding author e-mail: nataliadiaz@ugr.es.

1. Introduction

Currently, Deep Learning (DL) constitutes the state-of-the art models in many problems [54, 40, 95, 90, 46]. These models are opaque, complex and hard to debug, which makes their use unsafe in critical applications such as healthcare and high-risk scenarios. Furthermore, DL often requires a large amount of training data with over-simplified annotations that obviate an important part of centuries-long knowledge from domain experts. At the same time, DL generally uses correlation shortcuts to produce their outputs, which makes them finicky and difficult to correct. On the contrary, most classical symbolic AI approaches are interpretable but do not reach neither similar levels of performance nor scalability.

Among the potential solutions to clarify the decision process of a DL model, the topic of eXplainable AI (XAI) emerges. Given an audience, an XAI system produces details or reasons to make its functioning clear or easy to understand [4, 37]. To make black box Deep Learning methods more interpretable, a large amount of works exposed their vulnerabilities and sensitivity and came up with visual interpretation techniques, such as attribution or saliency maps [97, 79, 68]. However, the explanations provided by these methods, often in form of heatmaps, are not always enough, i.e. they are not easy to quantify, correct, nor convey to non technical audiences [45, 93, 91, 60, 39, 2, 47].

Having both specific and broad audiences of AI models contributes towards inclusiveness and accessibility, both part of the principles for responsible [4] and human-centric AI [69]. Furthermore, as advocated in [24], broadening the inclusion of different minorities and audiences can facilitate the evaluation of AI models when the objective is deploying human-centred AI systems.

A very critical challenge is thus to blend DL representations with domain expert knowledge. This leads us to draw inspiration from Neural-Symbolic (NeSy) learning [18, 8], a learning paradigm composed by both neural (or sub-symbolic) and symbolic AI components. An interesting challenge consists of bringing explainability in this fusion through the alignment of such learned and symbolic representations [6]. In order to pursue this idea further, we approach this quest by considering the expert knowledge to be in form of a KG.

Since our ultimate objective is fusing DL representations and domain expert representations, to fill this gap we propose the eXplainable Neural-symbolic (*X-NeSyL*) learning methodology, to bring explainability in the process. X-NeSyL methodology is aimed to make neural-symbolic models explainable, while providing more universal explanations for both end-users and domain experts. X-NeSyL methodology is designed to enhance both performance and explainability of DL, in particular, a convolutional neural network (CNN) classification model. X-NeSyL methodology is constituted by three main components:

1. A symbolic processing component to process symbolic representations, in our case we model explicit knowledge from domain experts with knowledge graphs.

2. A neural processing component to learn neural representations, *EXPLANet*: eXplainable Part-based cLAssifying NETwork architecture. EXPLANet is a compositional deep architecture that allows to classify an object by its detected parts.

3. An XAI-informed training procedure, able to guide the model to align its outputs with the symbolic explanation and penalize it accordingly when this is not the case. We propose SHAP-backprop to align the representations of a deep CNN with the symbolic one from a knowledge graph, thanks to a SHAP Attribution Graph (SAG) and a misattribution function.

The election of these components is designed to enhance a DL model by endowing its output with explanations at two levels:

- *Enhancement of the explanation at inference time*: We extend the classifier inference procedure to not only classify, but also detect what will serve as basis for the explanation. These components should be possible to be specified through the symbolic component, e.g., a knowledge graph that

acts as gold standard explanation from the expert. EXPLANet is proposed here to classify an object based on the detected object-parts, and thus, has the role of facilitating the mapping of neural representations to symbols.

- *Enhancement of the explanation at training time*: We penalize the original model at this second training phase, aimed towards improving the original classifier, thanks to an XAI technique called Shapley analysis [59] that assesses the contribution of each feature to a model output. *SHAP-backprop* training procedure is presented to adjust the model using a misattribution function that quantifies the error coming from the contribution of features (object-parts) attributed to the output (expressed in a SHAP Attribution Graph, *SAG*) not in agreement with the theoretical contribution expressed by the expert knowledge graph.

Together with the X-NeSyL methodology, this paper contributes an explainability metric to evaluate the interpretability of the model, SHAP GED (SHAP Graph Edit Distance), that measures the degree of alignment between the symbolic (expert) and neural (machine) representations. The objective of this metric is to gauge the alignment between the explanation from the model and the explanation from the human target audience that validates it.

We illustrate the use of X-NeSyL methodology through a guiding use case on monument architectural style classification and its dataset named MonuMAI [51]. We selected this dataset because it includes object-part-based annotations which make it suitable for assessing our proposal.

The pipeline components of the X-NeSyL methodology are summarized in Fig. 1. They are meant to complete a versatile template architecture with pluggable modular components to make possible the fusion of representations of different nature. X-NeSyL methodology can be adapted to the needs of the use case, and allows the model to train in a continual learning [55] setting.

The experiments to validate the X-NeSyL methodology make evident the well known interpretability-performance trade-off with respect to traditional training with an improvement of 3.6 % with respect to the state of the art (*MonuNet* [51]) on MonuMAI dataset. In terms of explainability, our contributed interpretability metric, SHAP GED, reports a gain of up to 0.38 –from 0.93 to 0.55–. The experimental study shows that X-NeSyL methodology makes it possible for CNNs to gain explainability and performance.

The rest of this paper is organized as follows: First we present the literature around XAI, and compositional, part-based classifiers in Section 2. We present a set of frameworks on Neural-Symbolic integration as a basis and promising body of research to attain XAI in Section 3. We describe X-NeSyL methodology in Section 4. Its core components are presented therein, Section 4.1.1 presents the symbolic component, i.e., how KGs can be used to represent symbolic expert knowledge to be leveraged by a DL model, Section 4.2 presents the neural representation component describing EXPLANet architecture, and Section 4.3 the XAI-guided training method *SHAP-Backprop*. X-NeSyL methodology is evaluated through the proposed explainability metric SHAP GED, presented and illustrated in Section 5. The complete methodology pipeline is illustrated through a driving use case on MonuMAI cultural heritage application in Section 6. Section 7 we discuss results, alternative perspectives, and open research avenues for the future. Finally, the Appendix includes additional experiments with an extra dataset, PASCAL-Part.

2. Related work: Explainable deep learning and compositional part-based classification

An increasing number of reviews and surveys is providing diverse classifications of XAI methods [4, 12, 37]. We particularly focus on attribution methods, i.e. XAI methods that relate a particular output of a DL model to their input variables. The most popular methods are model agnostic and provide visual explanations in form of heatmaps, saliency maps or class activation methods.

This section reviews three types of XAI attribution methods [68, 4], 1) local explanations, 2) saliency maps and 3) compositional part-based classification models.

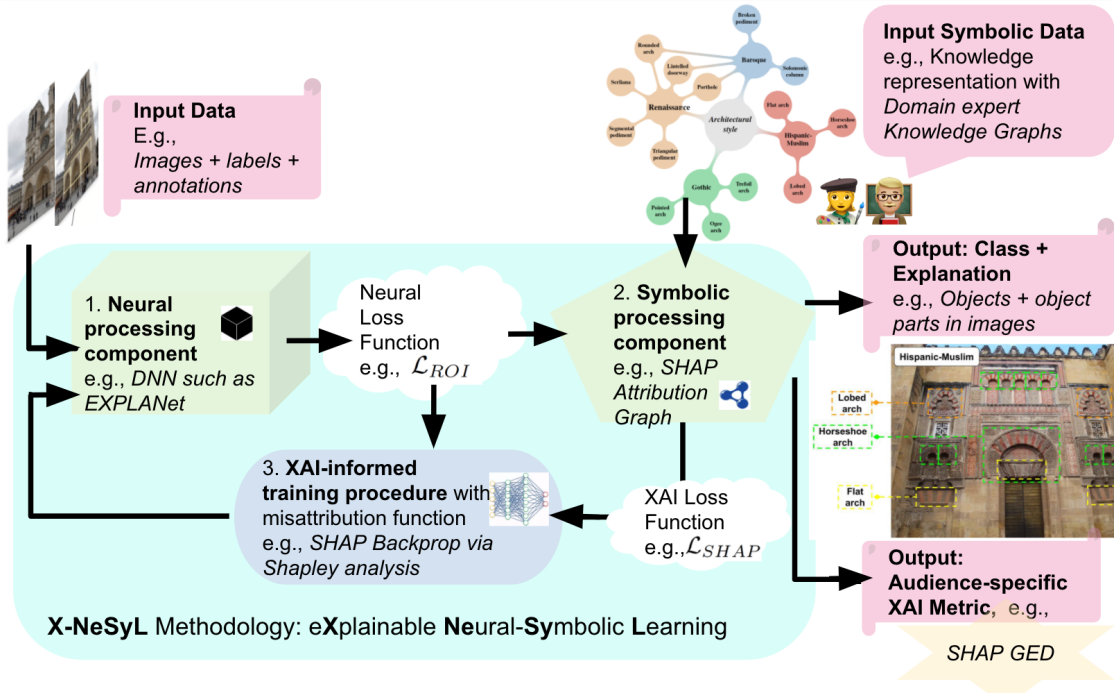


Figure 1: Proposed X-NeSyL methodology for eXplainable Neural-Symbolic learning. We illustrate the components of the methodology with examples of MonuMAI use case processed with *EXPLANet* part-based object detection and classification model, knowledge graphs, and *SHAP-Backprop* training procedure.

2.1. Local explanations

These methods are increasingly model agnostic, i.e., independent of the underlying black box model. The main intuition behind this type of methods is that they start from any specific point in the input space and explore its neighborhood to understand what has caused the prediction.

One of the simplest approaches in this context is *LIME* [72]. It explains the decision of the model by example. That is, in image classification, LIME segments the input image into super-pixels and then generates a data set of perturbed instances by turning some of the super-pixels into gray color. Each perturbed instance is then analyzed by the machine learning model to get a decision. Afterwards, a simple linear locally weighted model is trained on this data set. As result, the super-pixel with the highest weight is presented as explanation. One of the limitations of this approach is that it cannot generalize to unseen instances.

To increase generalization to unseen images, the same researchers introduced *Anchors* [73]. Instead of using a simple linear model, Anchors use reinforcement learning techniques in combination with a graph search algorithm. They express the explanation using simple $if - then$ rules applicable to other instances.

Following the same philosophy, *Minimal input deformation* [34], uses blur as perturbation technique to learn the perturbation mask that minimizes the class score. This can be seen as a counterfactual explanation strategy. The authors showed that the obtained image masks provide better explanations than the obtained by previous gradient-based saliency methods and their variants.

SHAP (SHapley Additive exPlanation) [59] is a framework for interpreting predictions based on coalitional game theory and theoretically optimal Shapley Values. It explains the prediction of an instance x by computing the contribution of each feature to the prediction. The feature values of x act as players in a coalition. The computed Shapley values tell us how to fairly distribute the prediction among the features. A player can be an individual feature value for tabular data, or a set of feature values, i.e., a vector of values. In general, SHAP is not applied directly to images but can be applied to a vector of intermediate features, i.e., the output of a high-level layer of a neural network.

DeepLIFT [81] main specificity is that it computes importance scores based on differences with a reference (in case of images, a black image). There is a connection between DeepLIFT and Shapley values. The Shapley values measure the average marginal effect of including an input over all possible orderings in which inputs can be included. If we define "including" an input as setting it to its actual value instead of its reference value, DeepLIFT can be thought of as a fast approximation of the Shapley values. It can as well be seen as an extension of SHAP for images.

2.2. Explanations using saliency maps

Saliency maps based post-hoc explanation methods have been a very powerful tool in explaining deep CNNs, as they propose an easily interpretable map. A saliency map is actually a heatmap that is usually superimposed over the input image to emphasize the pixels of the image that were more important for the prediction.

DeconvNet & Guided Backpropagation [82] [97] are the first approaches for saliency maps based methods. DeconvNets [98] consist of running the model backwards to map the activations from intermediate convolution layers back to the input pixel space. Guided backpropagation is a variant of the standard deconvolution approach that is meant to work on every type of CNNs, even if no pooling layers are present. The main difference lies on the way ReLU functions are handled in the different cases.

LRP (layer-wise relevance propagation): [5, 10] introduce a novel way to consider the operation done inside a neural architecture with the concept of *relevance*. Given their definition of relevance and adding certain properties, *relevance* intuitively corresponds to the local contribution to the prediction function f(x). The idea of LRP is to compute feature relevance thanks to a backward pass and thus, it yields a pixel-wise heat map.

PatternNet & PatternAttribution [48] take the previous work a step further by applying a proper statistical framework to the intuition behind. More precisely, they build on slightly more recent work

called DTD (deep Taylor decomposition) as introduced in [67]. The key idea of DTD is to decompose the activation of a neuron in terms of contributions from its inputs. This is achieved using a first-order Taylor expansion around a root point x_0. The difficulty in the application of DTD is the choice of the root point x_0, for which many options are available. *PatternAttribution* is a DTD extension that learns from data how to set the root point. This way the function extracts the signal from the data, and it maps *attribution* back to the input space, which is the same idea of *relevance*. *PatternNet* yields a layer-wise back-projection of the estimated signal to the input space.

CAM (Class Activation Mapping) [100] has as goal to leverage the effect of Global Average Pooling layers to a localization of deep representations for CNNs. A class activation map for a particular category indicates the discriminative image regions used by the CNN to identify that category. The process for the basic CAM approach is to put a Global Average Pooling layer on top of the convolution network and then perform classification. The layer right before the Global Average Pooling is then visualized.

Grad-CAM & Grad-CAM++ [79, 14] emerged from the need to go faster than CAM and avoid a training procedure to happen. The idea of class activation mapping is kept, but to build weights on the features maps (convolution layers) it is using a backpropagated gradient from the score given to a specific class. The advantage is that no architectural changes or re-training is needed, contrary to the architectural constrains of the CAM approach. With a simple change, Grad-CAM can similarly provide counterfactual activations for a specific class. Grad-CAM++ expands Grad-CAM with an improved way to process the weights of the feature maps. Grad-CAM is widely used for interpreting CNNs.

Score-CAM [92] is a novel approach in between CAM-like approaches and local explanations. The main idea behind this new approach is that it does not need to backpropagate any signal inside the architecture and as such, only a forward pass is needed.

Integrated gradient [85] proposes a new way to look at the issue. The underlying idea relies on the complexity of ensuring that a visualisation is correct besides being visually appealing and making sense. Here it introduces two axioms that attribution methods should follow, called *sensitivity* and *implementation invariance*.

Sensitivity: An attribution method satisfies Sensitivity if for every input and baseline that differ in one feature but have different predictions, the differing feature is given a non-zero attribution.

Implementation invariance: Two networks are functionally equivalent if their outputs are equal for all inputs, despite having very different implementations. Attribution methods should satisfy Implementation Invariance, i.e., attributions should always be identical for two functionally equivalent networks.

Some easily applicable sanity checks can be done to verify that a method is dependant on the parameters and the training set [2]. For instance, *integrated gradient* stems from path methods, and verify both precedent axioms. We consider the straight line path (in R^n) from the base line between x_0 and x in the input, and compute the gradients at all points along the path. Integrated gradients are obtained by accumulating these gradients. Specifically, integrated gradients are defined as the path integral of the gradients along the straight line path from the baseline x_0 to the input x.

All these methods seem visually appealing, but most of them rely on heuristics about what we want to look at, more or less well defined. Some works have been proposed to check the validity of such methods as tools to understand the underlying process inside neural architectures.

It has also been highlighted by the research community that saliency methods must be used with caution, and not blindly trusted, given their sensibility to data and training procedures [91, 47].

2.3. Compositional Part-based Classification Models

Compositionality [3] in computer vision refers to the capacity to represent complex concepts (from objects, to procedures, to beliefs) by combining simpler parts [33, 3]. Despite CNNs being not inherently compositional, compositionality is a desirable property for them to be learned [83]. For instance, hand-written symbols can be learned from only a few examples using a compositional representation of the

strokes [50]. The compositionality of neural networks has also been regarded as key to integrate symbolism and connectionism [44, 62].

Part-based object recognition is an example of semantic compositionality and a classic paradigm, where the idea is to gather local level information to make a global classification. In [21], the authors propose a pipeline that first groups pixels into superpixels, then does segmentation at the superpixel-level, transforming this segmentation into a feature vector and finally classifying the global image thanks to this feature vector. Similar work is proposed by [42], where they extend it to 3D data. Here the idea is to classify part of the image into a predefined class, and then use those intermediate predictions to provide a classification of the whole image. The authors of [7] also define mid level features that capture local structure such as vertical or horizontal edges, Haar filters and so on. However they are closer to dictionary learning than to the work we propose in this paper.

One of the most well known object parts detection model is [32]. It provides object detection based on mixtures of multiscale deformable part models, based on data mining of hard negative examples with partially labelled data to train a latent SVM. Evaluation is done in PASCAL object detection challenge (PASCAL VOC benchmark).

Finally, more recently, semi-supervised processes were developed such as [35]. They are proposing a two step neural architecture for fine-grained image classification aided by local detections. The idea is that positive proposal regions are highlighting varied complementary information, and that all this information should be used. In order to do that, first an unsupervised detection model is made by alternatively applying a CRF and Mask-RCNN (given an initial approximation with CAM). Then having a detection model and thus the positive region proposal they are fed to a Bi-Directional LSTM that will produce a meaningful feature vector accumulating information across all regions and then be able to classify the image. It can be seen as unsupervised part-based classification.

3. Neural-Symbolic (NeSy) Integration models

One of the most promising approaches to merge deep representations with symbolic knowledge representation that allows explainability to deep neural networks (such as CNNs) is the Neural-Symbolic (NeSy) integration. NeSy integration aims at joining standard symbolic reasoning with neural networks in order to achieve the best of both fields and soften their limitations. A complete survey of this method is provided in [18, 19]. Indeed, symbolic reasoning is able to work in presence of few data as it constrains entities through relations. However, it has limited robustness in data errors e and requires formalized background knowledge. On the other hand, neural networks are fast and able to infer knowledge. However, they require a lot of data and have limited reasoning properties. Their integration overcomes these limitations and, as stated by [9, 88], improves the explainability of the learned models. In the following, we present the main NeSy frameworks.

Many NeSy frameworks treat logical rules as constraints to be embedded in a vector space. In most of the cases these constraints are encoded into the regularization term of the loss function in order to maximize their own satisfiability. Logic Tensor Networks [80] and Semantic Based Regularization [25] perform the embedding of First-Order Fuzzy logic constraints. The idea is to jointly maximize both training data and constraints. Both methods are able to learn in presence of constraints and perform logical reasoning over data. In Semantic Based Regularization the representations of logical predicates are learnt by kernel machines, whereas Logic Tensor Networks learn the predicates with tensor networks. Other differences regard the dealing of the existential quantifier: skolemization for Logic Tensor Networks, or conjunction of all possible groundings for Semantic Based Regularization. Logic Tensor Networks have been applied to semantic image interpretation by [28] and to zero-shot learning by [27]. Semantic Based Regularization has been applied, for example, to the prediction of protein interactions by [75] and to image classification [25]. Both Logic Tensor Networks and Semantic Based Regularization show how background knowledge is able to i) improve the results and ii) counterbalance the effect of noisy or

scarce training data. Minervini at al proposed a regularization method for the loss function that leverages adversarial examples [65]: the method firstly generates samples that maximize the unsatisfaction of the constraints, then the neural network is optimized to increase their satisfaction. Krieken et al. [89] proposed another regularization technique applied to Semi-Supervised Learning where the regularization term is calculated from the unlabelled data. In the work of [94], propositional knowledge is injected into a neural network by maximizing the probability of the knowledge to be true.

Other works use different techniques but keep the idea of defining logical operators in terms of differentiable functions (e.g., [87, 20]). Relational Neural Machines [63] is a framework developed that integrates neural networks with a First-Order Logic reasoner. In the first stage, a neural network computes the initial predictions for the atomic formulas, whereas, in a second stage, a graphical model represents a probability distribution over the set of atomic formulas. Another strategy is to directly inject background knowledge into the neural network structure as done in [17]. Here, the knowledge is injected in the model by adding new layers to the neural network that encode the fuzzy-logic operator in a differentiable way. Then, the background knowledge is enforced both at inference and training time. In addition, weights are assigned to rules as learnable parameters. This allows for dealing with situations where the given knowledge contains errors or it is softly satisfied by the data without a priori knowledge about the degree of satisfaction.

The combination of logic programming with neural networks is another exploited NeSy technique. Neural Theorem Prover [74] is an extension of the logic programming language Prolog where the crisp atom unification is soften by using a similarity function of the atoms projected in an embedding space. Neural Theorem Prover defines a differentiable version of the backward chaining method (used by Prolog) with the result of learning a latent predicate representation through an optimisation of their distributed representations. *DeepProbLog* [61] integrates probabilistic logic programming (*ProbLog* [70]) with (deep) neural networks. In this manner, the explicit expressiveness of logical reasoning is combined with the abilities of deep nets.

Finally, NeSy systems have also shown to be effective for learning logical constraints from KGs [29], for inferring causal graphs from time series [96], for learning to explain logic inductive learning [58] and for generating symbolic explanations of DL models [86, 78, 30].

4. EXplainable Neural-Symbolic (X-NeSyL) learning methodology

One challenge of the latest DL models today is producing not only accurate but also reliable outputs, i.e., outputs whose explanations agree with the ground truth, and even better, agree with a human expert on the subject. X-NeSyL methodology is aimed at filling this gap, and getting model outputs and experts explanations to coincide. In order to tackle the concrete problem of fusing DL representations with domain expert knowledge in form of knowledge graphs, in this section we present the three main ingredients that compose the X-NeSyL methodology: 1) the symbolic knowledge representation component, 2) the neural representation learning component, and 3) the alignment mechanism for both representations to align, i.e., correct the model during training or penalize it when disagreeing with the expert knowledge.

First, in Section 4.1 we present the symbolic component that serves to endow the model with interpretability –which will be in form of knowledge graphs–, then in Section 4.2 the neural representation learning component –that will serve to reach the best performance– and finally, in Section 4.3 the XAI-guided training procedure that makes both components align with SHAP-Backprop during training of the DL model.

4.1. Symbolic knowledge representation for including human experts in the loop

Symbolic AI methods are interpretable and intuitive (e.g. they use rules, language, ontologies, fuzzy logics, etc.). They are normally used for knowledge representation. Since we advocate for leveraging the best of both, symbolic and neural representation learning currents, in order to make the latter more

explainable, here we choose a simple form of representing expert knowledge, with knowledge graphs. Right after, in order to demonstrate the practical usage of X-NeSyL methodology, we present the running use case using knowledge graphs that will demonstrate the usage of this methodology thorough the paper.

4.1.1. Knowledge Graphs

Different options exist to leverage a KG as a versatile element to convey explanations [53]. We inspire ourselves by NeSy frameworks for XAI using ontologies and KGs [11, 6, 16], on explanations of image and tabular data-based models and, more broadly, on the XAI literature [37]. We focused more precisely on attribution methods that try to measure the importance of the different parts of the input toward the output. We provide a formalization of the domain expert data into a semantic OWL2-based KG that is actually leveraged by the detector and classifier DL model.

In this work we present a new training procedure to enhance interpretability of part-based classifier, given an appropriate KG. It is based on Shapley values (or SHAP) [59] which outputs feature attribution of the various part elements toward the final classification, which we compare with the KG. We use the SHAP information to weight the loss that we backpropagate [52] at training time.

Alongside standard images and annotations we have in our various datasets, we also have expert knowledge information. This information is usually encoded in knowledge graphs (KGs), such as the one in Figure 7.

A knowledge graph $\mathcal{G}$ is formalized as a subset of triples from $\mathcal{E} \times \mathcal{R} \times \mathcal{E}$, with $\mathcal{E}$ the set of entities and $\mathcal{R}$ the set of relations. A single triple (e_i, r, e_j) means that entity e_i is related to e_j through relation r. In the context of part-based classification, such graph encodes the $partOf$ relationship (that is, $\mathcal{R} = \{partOf\}$) between elements (*parts*) and the (*whole*) object they belong to.

Therefore, since we have $|\mathcal{R}| = 1$, and to relate to various attribution functions presented in Section 2.1 and 2.2, we can see the KG as the theoretical attribution graph.

The attribution graph encodes whether an element contributes positively or negatively towards a prediction. This way the KG can be rewritten as $KG = (KG_1, ..., KG_n)$ (one entry for each macro label) with $KG_i = (KG_{i,1}, KG_{i,2}, ..., KG_{i,m})$ (one entry for each element that is *part-of* the object), $KG_{i,k} = \{-1, 1\}$.

If a link between an element and a macro (object-level) label exists in the theoretical KG, then it means such element is typical of that label and should count positively toward this prediction, thus, its entry in the matrix representing the KG is equal to t. If no such link exists, then it means it is not typical of the macro label and should contribute negatively, thus its entry in the matrix KG is equal to $-t$.In our case we choose values of the KG edges to be binary, and since we set $|t| = 1$, $KG_{i,k} = \{-1, 1\}$. Seeing a KG as a feature attribution graph is not the only way to model a KG; we can also encode KGs as ontologies. It is worth mentioning that ontologies can be seen as a set of triples with the format (subject, predicate, object) or (subject, property, value) where edges can have varying semantic meaning following constraints from Description Logics [1].

Modeling the graph as an adjacency matrix is not appropriate since architectural style nodes and architectural elements nodes are playing two very different roles. Instead, we model the graph as a directed graph, with edges from the architectural element toward the architectural styles.

4.1.2. A driving use case on cultural heritage: MonuMAI architectural style facade image classification

The latest deep learning models have focused on (whole) object classification. We choose part-based datasets as a straight forward way to leverage extra label information to produce explanations that are compositional and very close to human reasoning, i.e., explaining a concept or object based on its parts.

In this work, we interested ourselves in the MonuMAI (Monument with Mathematics and Artificial Intelligence) [51] citizen science application and corresponding dataset collected through the application, because it complies with the required compositional labels in a object detection task, based on object parts. At the same time, facade classification by pointing relevant architectonic elements is an interesting use

case application of XAI. We use this example thorough the article as a guiding application use case that perfectly serves to demonstrate the usage of our part-based model and pipeline for explainability.

The MonuMAI project has been developed at the University of Granada (Spain) and has involved citizens in creating and increasing the size of the training dataset through a smartphone app[1].

The MonuMAI dataset

MonuMAI dataset allows to classify architectural style classification from facade images; it includes $1,092$ high quality photographs, where the monument facade is centered and fills most of the image. Most images were taken by smartphone cameras thanks to the MonuMAI app. The rest of images were selected from the Internet. The dataset was annotated by art experts for two tasks, image classification and object detection as shown Figure 6. All images belong to facades of historical buildings that are labelled as one out of four different styles (detailed in Table 2 and Table 1): Renaissance, Gothic, Baroque and Hispanic-Muslim. Besides this label given to an image, every image is labeled with key architectural elements belonging to one of fourteen categories with a total of $4,583$ annotated elements (detailed in Table 1). Each element is supposed to be typical of one or two styles, and should almost not appear inside facade of the other styles. Examples for each style and each element are in Fig. 2 and 3, while the MonuMAI dataset labels used are shown in Figs. 4 and 5.

Architectural element	**Count**	**Element rate (%)**	**Architectural style**
Horseshoe arch	452	9.86	Hispanic-muslim
Lobed arch	131	2.86	Hispanic-muslim
Flat arch	105	2.29	Hispanic-muslim
Pointed arch	322	7.03	Gothic
Ogee arch	154	3.36	Gothic
Trefoil arch	47	1.03	Gothic
Triangular pediment	586	12.79	Renaissance
Segmental pediment	184	4.01	Renaissance
Serliana	62	1.35	Renaissance
Porthole	356	7.77	Renaissance/Baroque
Lintelled doorway	898	19.59	Renaissance/Baroque
Rounded arch	706	15.40	Renaissance/Baroque
Broken pediment	388	8.47	Baroque
Solomonic column	192	4.19	Baroque

Table 1: Characteristics of the architectural styles dataset, where count is the number of occurrences of an element in the dataset, and element rate is the ratio between the number of occurrences of an element and the total number of all elements.

Architectural style	**#Images**	**Ratio** (%)
Hispanic-Muslim	253	23.17
Gothic	238	21.79
Renaissance	247	22.62
Baroque	354	32.42

Table 2: Characteristics of MonuMAI architectural style classification dataset (#images represents the number of images).

Apart from MonuMAI dataset, and in order to draw more general conclusions on our work, we used a dataset with similar hierarchy to MonuMAI. Additional results for PASCAL-Part [15] dataset are in the

[1]Mobile App available in the project website: monumai.ugr.es



Figure 2: Extract from MonuMAI dataset. From left to right and top to bottom: hispanic-muslim, gothic, renaissance, baroque (architectural style classes).



Figure 3: 14 architectonic elements (*object parts to be detected*) from MonuMAI dataset used in our experiments as *object-part* explanation to classify a *whole* facade architectural style. From left to right and top to bottom: horseshoe arch, pointed arch, ogee arch, lobed arch, flat arch, serliana, solomonic column, triangular pediment, segmental pediment, broken pediment, rounded arch, trefoil arch, porthole, lintelled doorway. The dataset shows that there are architectural elements very distinctive. Even with a very low number of instances in the dataset, object detectors are able to recognize them, e.g., lobed arch, or solomonic column.

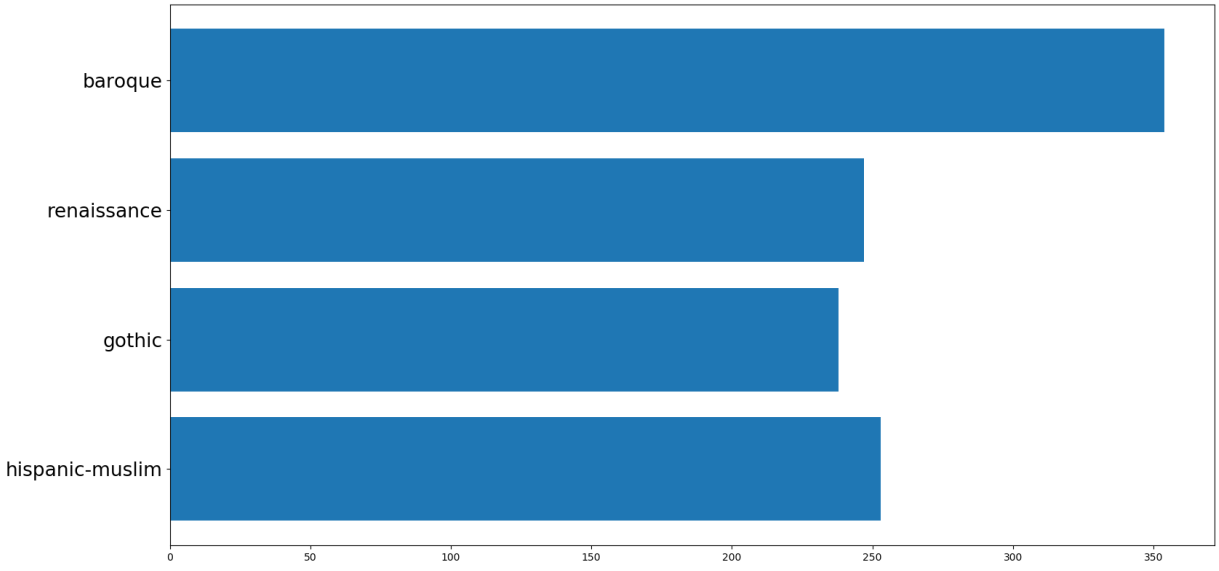


Figure 4: Distribution of style classes in MonuMAI dataset.

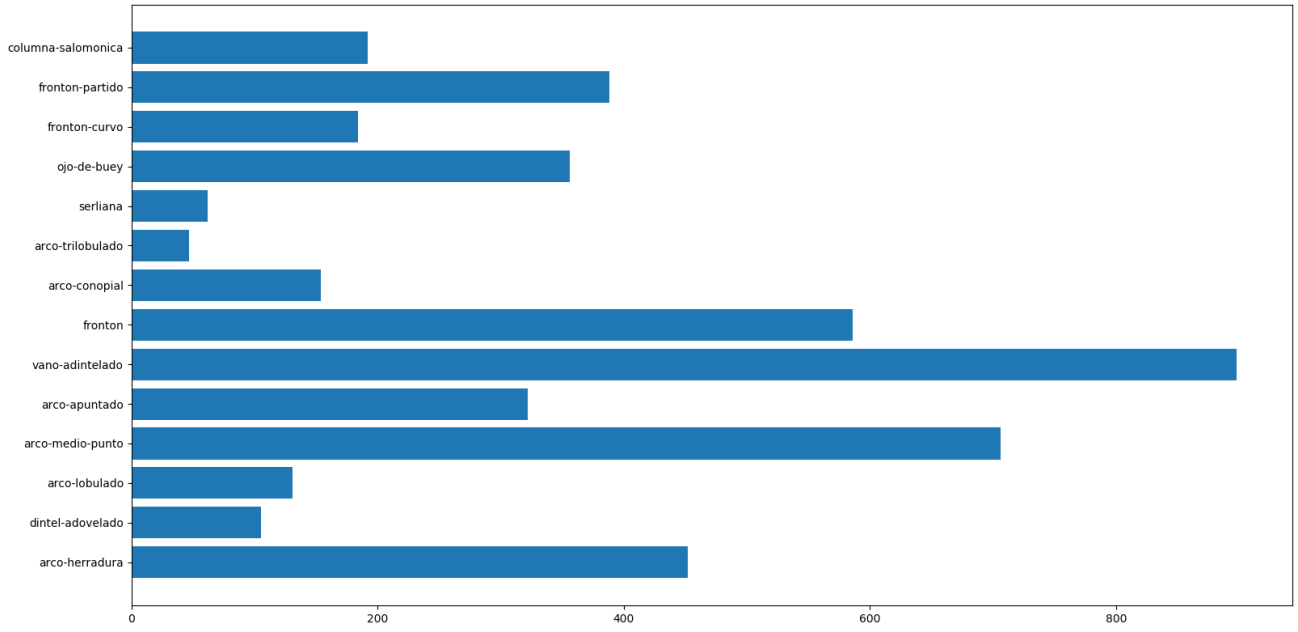


Figure 5: Distribution of object-parts in MonuMAI dataset.

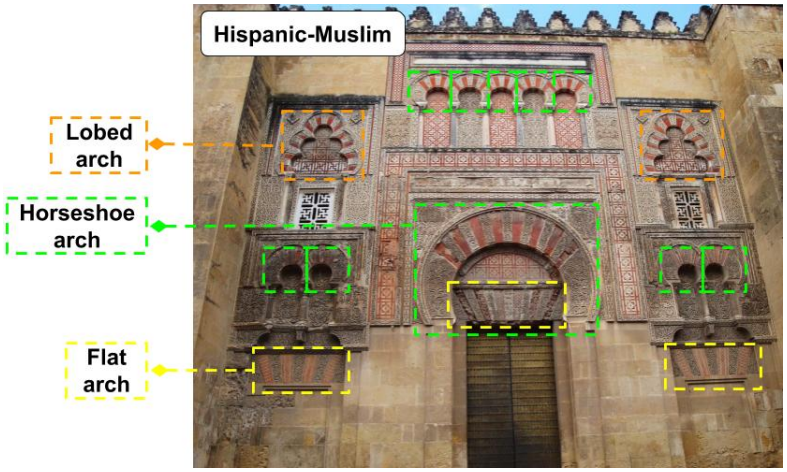


Figure 6: Illustration of the two annotation levels of architectural style and elements on MonuMAI dataset. This image is labeled as Hispanic-Muslim and includes different annotated elements, e.g., two lobed arches (Source: [51]).

Appendix.

MonuMAI's Knowledge Graph

The original design of MonuMAI dataset and MonuNet baseline architecture [51] use the KG exclusively as a design tool to visualize the architectural style of a monument facade based on the identified parts, but it is not explicitly used in the model. In contrast, we change that to go further, in order to guarantee a reproducible and explainable decision process that aligns with the expert knowledge. We will see in Section 4.3.3 how KGs can be used in a detection + classification architecture, during training, since EXPLANet is designed to incorporate the knowledge in the KG. Besides the trust gain, we aim at easing the understanding of flaws and limitations of the model, along with failure cases. This way, requesting new data to experts would be backed up by proper explanations and it would be effortless to target new and relevant data collection.

The KG corresponding to MonuMAI dataset has only fourteen object classes and four architectural styles. Each architectural element is linked to at least one style. Each link between the two sets symbolizes that an element is typical and expected in the style it is linked to.

- *Renaissance*: rounded arch, triangular pediment, segmental pediment, porthole, lintelled doorway, serliana.
- *Baroque*: rounded arch, lintelled doorway, porthole, broken pediment, solomonic column.
- *Hispanic-muslim*: flat arch, horseshoe arch, lobed arch.
- *Gothic*: trefoil arch, ogree arch, pointed arch.

MonuMAI's KG is depicted in Figure 7, where the root is the Architectural Style class (which inherits from the *Thing* top-most class in OWL). Note there is one more dimension in the KG, the *leaf* level of the original MonuMAI graph in [51] that represents some characteristics of the architectural elements, but it is not used in the current work.

We also explored the possibility of rewriting the looser structure captured in the KG as an ontology, using the OWL2 format. We did not limit ourselves to copying the hierarchy of the original KG, but rather added some categories to keep the ontology flexible to allow further expansions in the future. Three main classes are modelled in this ontology: A *Facade* represents an input image as a concept. A facade is linked to one and only one[2] *ArchitecturalStyle* through the relation *exhibitsArchStyle*, for which four styles can we used (others could be added by defining new classes). A facade can be linked to any number of *ArchitecturalElement* identified on it through the relation (i.e. OWL *object property*) *hasArchElement*.

ArchitecturalElement represents the class of architectural elements identified before, and is divided in subcategories based on the type of elements such as "Arch" or Window". This subcategorization, which does not exist in the original KG, was designed with the possibility of adding constraints between subcategories, such as an "arch" is probably higher in space than a "column", or an arch's lowest point is higher than a column's lowest point. Such geometrical or spatial constraints were not explored further, as it required extra expertise modelling from architecture experts, but could be easily added in future work.

Finally, the concept *ArchitecturalElement* is linked to an *ArchitecturalStyle* object through the object property *isTypicalOf*.

This multi-type edge ontology formulation allows us to see the problem of style classification as a problem of KG edge detection or link prediction among two nodes, in our case, between a facade instance and a style instance. This approach was unsuccessful (discussed in Section 6.4).

[2] In this study, as in MonuMAI, we represent the predominant one. Future work could consider the blend of more than one present style.

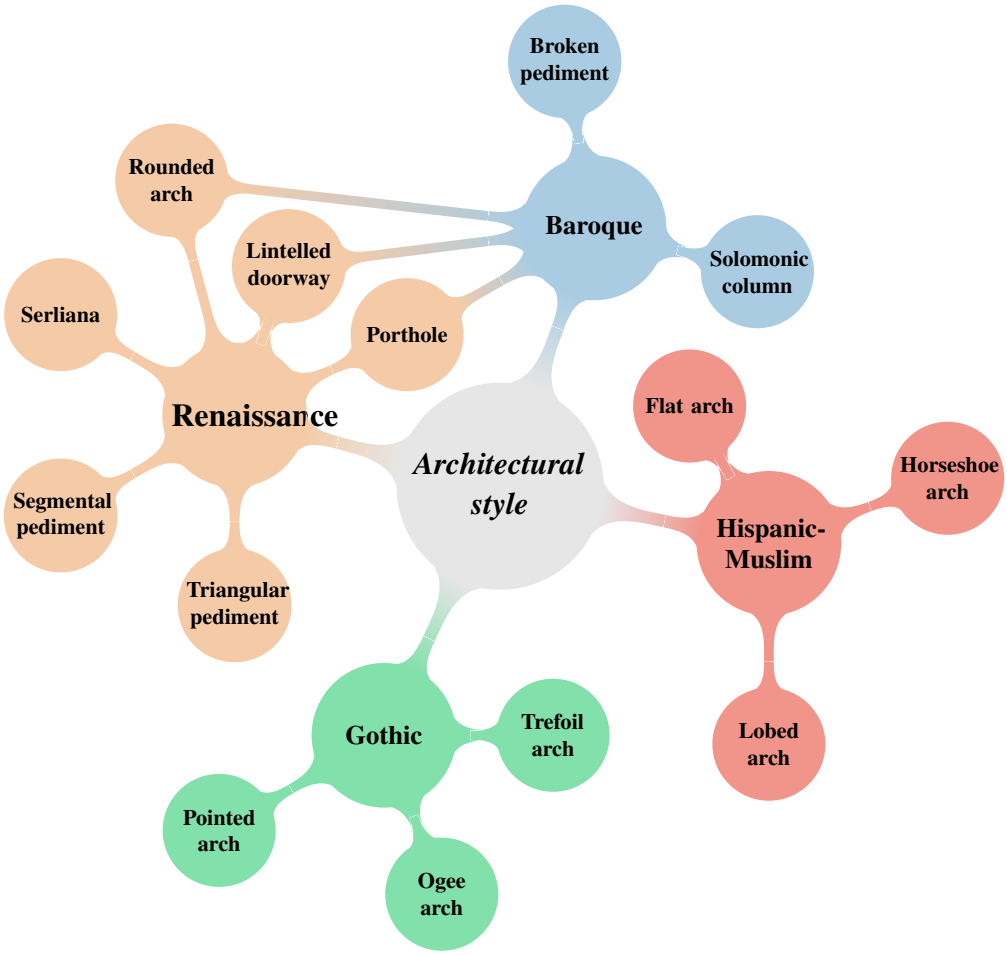


Figure 7: Simplified MonuMAI knowledge graph constructed based on art historians expert knowledge [51].

The KG formulation presented in Section 4.1.1 can be seen as a semantic restriction of the ontology we propose, where we kept only the triples including *isTypicalOf* relation and expanded the KG with a virtual relation *isNotTypicalOf*, to link together all elements with all the styles. This way the KG is a directed graph with edges going from the architectural element toward the architectural style. Because we restrict ourselves to only one relational object property and its inverse, the edges bear either *positive* or *negative* information, which motivates our modeling choice of having value ± 1 for formulated $K_{i,j,k}$ edges.

4.2. EXPLANet: Expert-aligned eXplainable Part-based cLAssifier NETwork Architecture

Previous section detailed the symbolic representation mechanism within the X-NeSyL methodology. While KGs serve the purpose of interpretable knowledge, in this section we present the neural representation learning component, mainly responsible for high performance in today's AI systems.

Our ultimate goal in this work is making DL models more trustworthy when it comes to the level of their explanations, and their agreement with domain experts. We will thus follow a human-in-the-loop [41] approach.

Typically, to identify the class of a given object, e.g., an aeroplane, a human first identifies the key parts of that object, e.g., left wing, right wing, tail; then, based on the combination of these elements and the importance of each single element, he/she concludes the final object class.

We focus on compositional part-based classification because it provides a common framework to assess part- and whole object based explanations. To achieve this we want to enforce the model to align with a priori expert knowledge. In particular, we built a new model called *EXPLANet*: Expert-aligned eXplainable Part-based cLAssifier NETwork Architecture, whose design is inspired by the way humans identify the class of an object.

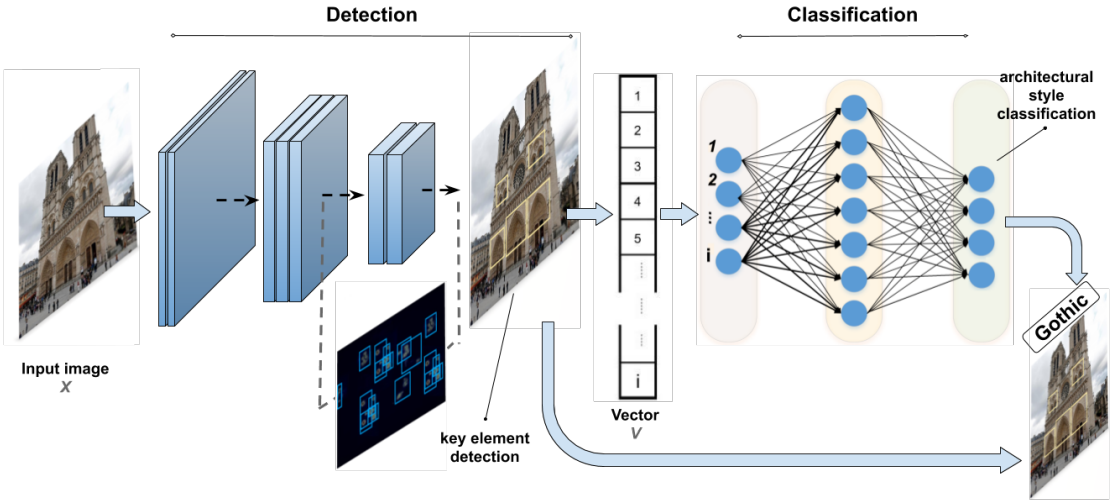


Figure 8: Proposed EXPLANet architecture processing examples from MonuMAI dataset.

EXPLANet, is a two-stage classification model as depicted in Figure 8. The first stage detects the object-parts present in the input image and outputs an embedding vector that encodes the importance, quantity and combinations of the detected object-parts. This information is used by the second stage to predict the class of the whole object present in the input image. More precisely:

1. The first stage is a detection module, which can be an object detector such as Faster R-CNN [71] or RetinaNet [56]. Let us consider that there are n object-part classes. This module is trained to detect the key object-part classes existent in the input image, and outputs M predicted regions. Each one is

represented by its bounding box coordinates and a vector of size n representing the probability of the n object-part classes. Let us denote $p_m \in \mathbb{R}^n$ (with $m \in [1, M]$) the probability vector of detecting object-part m. First we process all p_m by setting non maximal probabilities to zero, and denoting this new score p'_m, being p' also a vector of size n. Let us denote vector v the final descriptor of the image. We build v by accumulating the probabilities of p'_m such that:

$$v = \sum_{m=1}^{M} p'_m \qquad , \quad v \in \mathbb{R}^n \tag{1}$$

where vector v aggregates the confidence of each predicted object-part. M is the amount of bounding box regions predicted. In the case of MonuMAI dataset, to classify 4 object classes (i.e., architectural styles), we have $n = 14$ (object-part) labels, but this can be extended to an arbitrary n and M.

Large values in v mean that the input image contains a large number of object-part i elements with a high confidence prediction, whereas a low value means that predictions had low confidence. Intermediate values are harder to interpret, as they could be a small amount of high confidence predictions or a large amount of low confidence predictions, but the idea is that there is probably some objects of these kinds in the image. This *object-parts vector* can be seen as tabular data where each object part can be considered a feature (to be explained later by an XAI method). We will see in next section how a SHAP analysis can study the contribution of each actual object part present in the image to the actual final object classification.

Note that this probability aggregating scheme is for Faster R-CNN. For each object detected, Faster R-CNN keeps only one prediction (the most likely), whereas RetinaNet keeps all predictions (the whole probability vector). We found this to be more stable for training the RetinaNet framework.

2. The second stage of EXPLANet is a classification network, which is actually a two-layer multi-layer perceptron (MLP), that uses the embedding information (i.e., takes the previous detector output as input) to perform the final classification. This stage outputs the final object class based on the importance of the present key object parts detected in the input image.

The goal of such design is to facilitate the reproduction of the thought process of an expert, which is to first localize and identify key elements (e.g., in the case of architectural style classification of a facade, various types of arches or columns) and then use this information to deduce the final class (e.g., its overall style). However, EXPLANet architecture alone does not control for expert knowledge alignment. Next section introduces the next step of the pipeline, an XAI based training procedure and loss function to actually verify that this happens, and when this is not the case, correct the learning.

4.3. SHAP-Backprop: An XAI-informed training procedure and XAI loss based on the SHAP attribution graph (SAG)

After having presented the symbolic and neural knowledge processing components of X-NeSyL, we proceed to detail the XAI-informed training procedure to make the most of the best of both worlds, interpretable representations, and deep representations.

More concretely, this section presents how to use a model agnostic XAI technique to make a DL (CNN-based) model more explainable by aligning the test-set feature attribution with the expert theoretical attribution. Both knowledge bases will be encoded in KGs.

4.3.1. SHAP values for explainable AI feature contribution analysis

SHAP is a local explanation method [59, 66] that for every singular prediction it assigns to each feature an importance value regarding the prediction. It tells if a feature contributed to the current prediction

and gives information about how strongly it contributed. These are the Shapley values of a conditional expectation function of the original model. In our case we computed them with Kernel SHAP [59].

Similarly to LIME [72], Kernel SHAP is a model agnostic algorithm to compute SHAP values. In LIME, the loss function, weighting kernel and regularization term are chosen heuristically, while in SHAP they are chosen in a way that they satisfy the SHAP properties. See details in [59].

The idea of computing SHAP is to check whether object parts have the expected importance on the object class prediction (e.g. whether the presence of a horseshoe arch contributes to Hispanic-Muslim class). SHAP computation happens always in a per class basis, as the computation is regarding binary classification (belonging to class C vs not).

In our part-based pipeline, we apply SHAP at the tabular data level, i.e., after the aggregation function. As such, SHAP's only input is the feature vector that contains the information about parts detected before. Throughout this section, when we refer to feature value, we refer to this feature vector, and a feature value means one entry of this vector. As such, each feature value encodes the information about one element (either an architectural element for MonuMAI or an object part for PASCAL-Part) from our knowledge model. The final class prediction performed afterwards is done by the classification module of our part-based model, given such a feature vector.

In Figs. 9 and 10 we performed the SHAP analysis over the whole validation set. In practice it means that SHAP values were computed for each element of the validation set and plotted on the same graph. Then for each feature of the feature vector, in our case for each architectural element, we plot all SHAP values for this specific element found in the dataset, and we color them based on the feature value. They are plotted line-wise and each dot represents the feature value of a specific datapoint, i.e., image. High feature values (regarding the range they can take) are colored pink and low feature values in blue. Here, if an element is detected several times or with high detection confidence, it will be pink (blue for less detection confidence or less frequency). Then, horizontally, are shown the SHAP values, where high (absolute) values have high impact on the prediction[3].

If we compare the SHAP plots with the KG, here we do not observe any large amount of *outliers* or datapoints not coinciding with the domain expert KG acting as ground truth (in Fig. 12 right). We now need to be able to use this information automatically. Pink and blue (high and low) values of datapoint features can appear both in right and left sides of the plots, meaning its value can contribute towards predicting the considered class or not, respectively. However, in our case, only pink datapoints being on the positive (right) side of SHAP plot represent the *correct* behaviour if such element is also present in the KG. In that case, their feature value loss will not be penalized during training, as they match the expert KG (considered as GT). The rest of datapoints' SHAP values (blue in right side, pink and blue in left side) will be used by SHAP-Backprop to correct the predicted object class.

An example of computation of SHAP values on a single feature vector is in Table 3. On the right there is the feature vector, and on the left the SHAP values for each object class for each object part. We highlighted in green positive values and in red negative values.

4.3.2. SAG: SHAP Attribution Graph to compute an XAI loss and explainability metric

By measuring how interpretable our model is, in the form of a KG, we want to be able to tell if the decision process of our model is similar to how an expert mentally organizes its knowledge. As highlighted in the previous section, thanks to SHAP we can see how each feature value impacts the predicted macro label and thus, how each part of an object class impacts the predicted label. Based on this, we can create a *SHAP attribution graph (SAG)*. In this graph, the nodes are the object (macro) labels, and the parts are linked to a macro label if according to the SHAP algorithm, it played a contribution role toward predicting this label.

[3] See tutorial https://christophm.github.io/interpretable-ml-book/shap.html and SHAP source code in https://github.com/slundberg/shap.

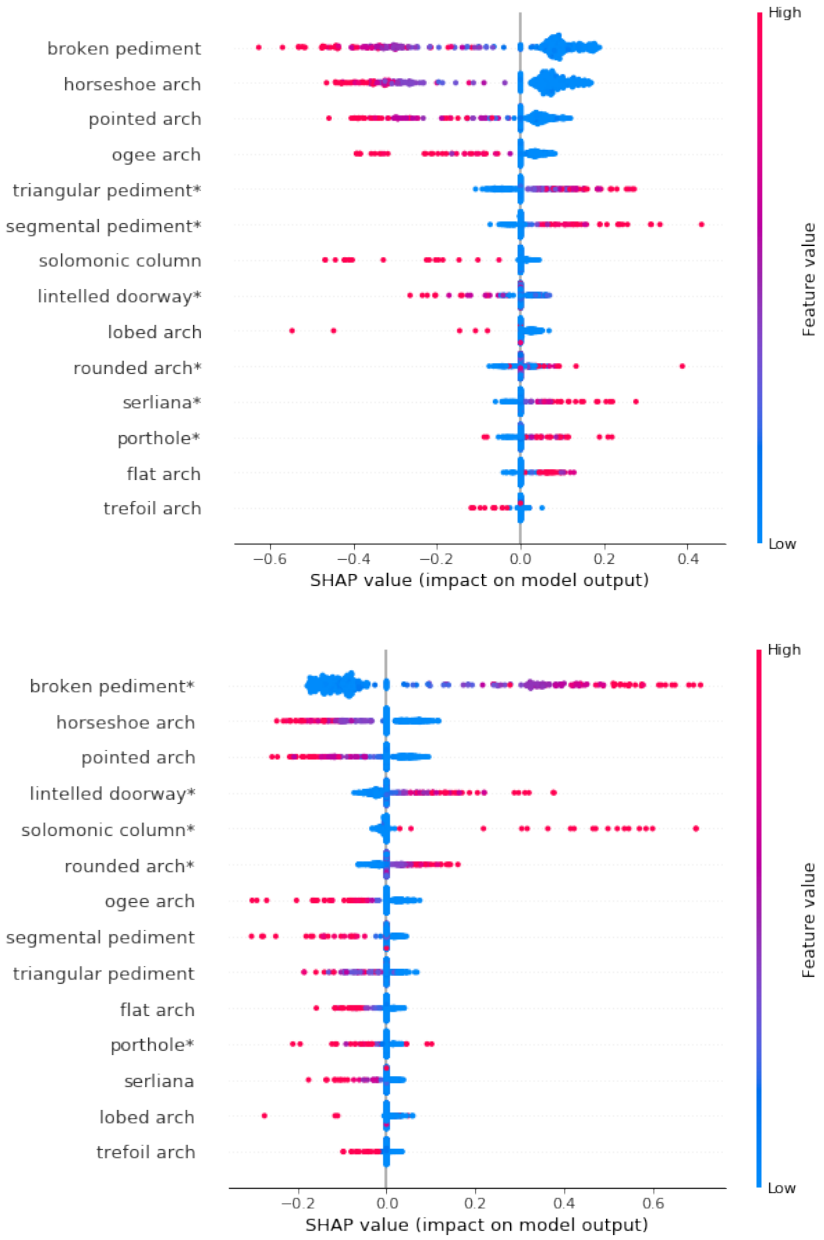


Figure 9: SHAP analysis (global explanation as SHAP summary plot) for Renaissance (up) and Baroque (down) architectural styles. Each feature (horizontal row) represents a global summary of the distribution of feature importance over the validation set. Features with asterisk in their name represent the expert ground truth (nodes with a positive value t in the expert KG). Example: In this case, there are features not representing the GT for the class Renaissance, e.g., *flat arch* (*dintel adovelado*) is contributing to the prediction, while the KG says it should not. Other than this, generally, datapoints reflecting the GT for an accurate model such as this one, acknowledge properly attributed object part contributions to the classification in the right side of the plot, in pink.

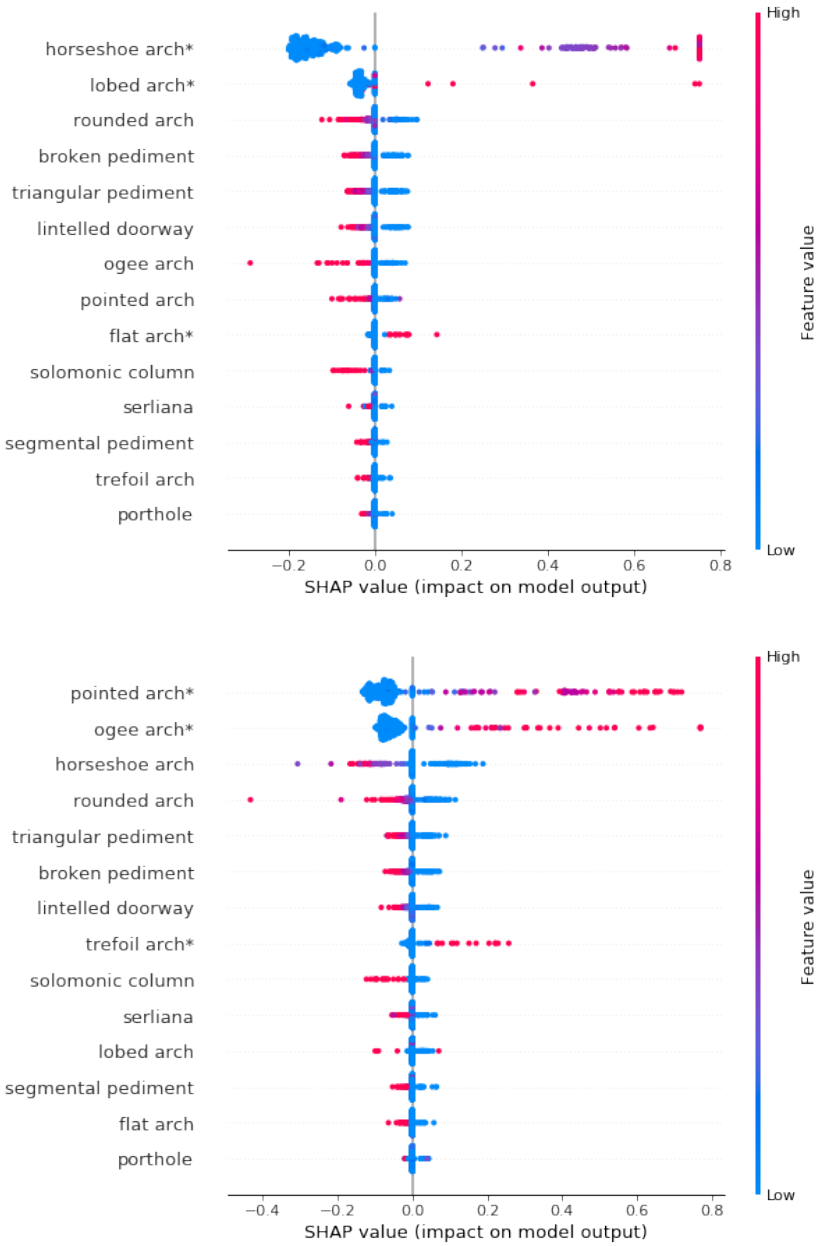


Figure 10: SHAP analysis (global explanation as SHAP summary plot) for Muslim (up) and Gothic (down) architectural styles. Each feature (horizontal row) represents a global summary of the distribution of feature importance over the validation set. Features with asterisk in their name represent the expert ground truth (nodes with a positive value t in the expert KG). Generally, datapoints reflecting the GT for an accurate model such as this one, acknowledge properly attributed object part contributions to the classification in the right side of the plot, in pink.

Building the SAG is a two step process. First we extract the feature vector representing the attributes detected (float values). Thanks to the detection model we get the predicted label from it. Feature vectors are the output of the aggregation function that are fed to the classification module.

Using as hyperparameter a threshold s[4] on each feature value, we identify which architectural element we have truly detected in the image. Then, using the SHAP values computed for this feature vector, we create a SAG per image in the test set, and thus we link together feature values and predicted label probabilities inside the SAG. This way we use SHAP to analyse the output for all classes, not only the predicted one:

- Having a *positive SHAP* value means the detected feature contributes to predicting this label, given a trained classifier and an image. We thus add to the SAG such edge representing a *present feature contribution*.
- Having a *negative SHAP* value and a feature value below the threshold s means that this element is considered typical of this label and its absence is detrimental to the prediction. As such, we can link the object label and the part label in the SAG, as a *lacking feature contribution*.

Lacking feature contributions or *present* feature contributions in the form of edges in the expert KG are obtained through a projection of the SHAP attribution graph, obtained from a SHAP analysis of the trained model, on the theoretical expert KG.

An example of SAG for the architectural style classification problem is in Fig. 12.

M, R, G, B means *Hispanic-Muslim, Renaissance, Gothic* and *Baroque*, respectively. The pseudo code to generate the SAG can be found in Algorithm 1.

Algorithm 1 Computes the SHAP attribution graph (SAG) for a given inference sample.

Require: feature_vector, shap_values, Classes, Parts, part detected threshold s
  $SAG \leftarrow \{\}$
  for class in Classes **do**
    $local_shap \leftarrow shap_values[class]$
    for part in Parts **do**
      $feature_val \leftarrow feature_vector[part]$
      $shap_val \leftarrow local_shap[part]$
      if feature_val $>$ s **then**
        if shap_val $>$ 0 **then**
          ADD (part, object) edge to SAG
        end if
      else
        if shap_val $<$ 0 **then**
          ADD (part, object) edge to SAG
        end if
      end if
    end for
  end for
  return SAG

In practice this allow us to have an empirical attribution graph, the SAG (built at inference time), and a theoretical attribution graph, the KG (representing prior knowledge). We can then compare both of them.

[4]Default thresholds used in our case for detection were s = 0.05 for both Faster R-CNN and RetinaNet, as they showed to work best for numerical stability.

4.3.3. SHAP-Backprop to penalize misalignment with an expert KG

In order to improve performance and interpretability of the model, we hypothesize that incorporating the SHAP analysis during the training process can be useful to fuse explainable information and improve interpretability.

The underlying idea is that SHAP helps us understand on a local level what features are contributing toward what class. In our case, SHAP links elements to a label by telling us if it contributed toward or against this label, and how strongly. Besides this analysis, we have the KG that is embedding basically the same information. We can see the KG as a set of expected attributions, i.e., if an element is linked to the object label in the KG, it should contribute to it, otherwise it should contribute against.

Given these two facts, we can compare real attribution via SHAP analysis, that gives us the empirical attribution graph, with the theoretical attribution found in the KG. If there is a discrepancy, then we want to penalize this feature, either in the classification process or in the detection process.

Misattribution, which happens when a feature attribution is unexpected or absent, can stem from various origins. One would be a recurrent misdetection inside the dataset. As such, penalizing misattribution at detection time could help us correct those. Penalizing the classification process could be considered as well, but has not been done in this work.

A schema of this approach is presented in Fig. 11. A new misattribution loss requires an intertwined training of the classification model with the detection model to compute the SHAP analysis; however the extra required training time in practice is not a big issue. Indeed, in the initial training protocol, one would fully train the detection and then the classification. Here, we have to train the classification at each detection epoch. We expect the explainability metric, i.e., the SHAP GED between the KG and the SAG to increase thanks to this SHAP signal backpropagation.

4.3.4. L_{SHAP}: A new loss function based on SHAP values and a misattribution function β

Let N be the number of training examples and let $I = (I_1, I_2, ..., I_N)$ be the training image examples.

Let D be the detector function such as:

$$D(I_i) = (BB_i, Conf_i, Class_i), \tag{2}$$

where $BB_i = (BB_1, BB_2, ..., BB_{N_i})$ are the bounding boxes detected by D, $Conf_i$ the confidence associated to each predicted box, and $Class_i$ the predicted class of each box. The associated ground truth label is used for standard backpropagation, but we will not need it for the weighting process.

Faster R-CNN [71] object detector uses a two terms loss:

$$\mathcal{L}_{detector} = \mathcal{L}_{RPN} + \mathcal{L}_{ROI}, \tag{3}$$

where $\mathcal{L}_{ROI}$ is the loss corresponding to predicting the region of interest and is dependant on the class predicted for each BB. It is computed at the output class level, whereas $\mathcal{L}_{RPN}$ is the loss function from the region proposal network, and it is computed at the anchor level[5]. We use a weighted $\mathcal{L}_{ROI}$, since the SHAP information is computed at the output class level and not the anchor level. We can write the loss as the sum of the losses for each image, and within an image, for each BB:

$$\mathcal{L}_{ROI} = \Sigma_{i=1}^{N} \Sigma_{k=1}^{N_i} \mathcal{L}(I_i, (BB_k, Conf_k, Class_k)), \tag{4}$$

where i is the index of the considered image, k the index of a BB predicted within that image, and $\mathcal{L}$ is a weighted sum of cross entropy and smooth L_1 loss.

[5] Anchors are a set of predefined bounding boxes of a certain height and width. These boxes are defined to capture the scale and aspect ratio of specific objects. Height and width of anchors are a hyperparameter chosen when initializing the network.

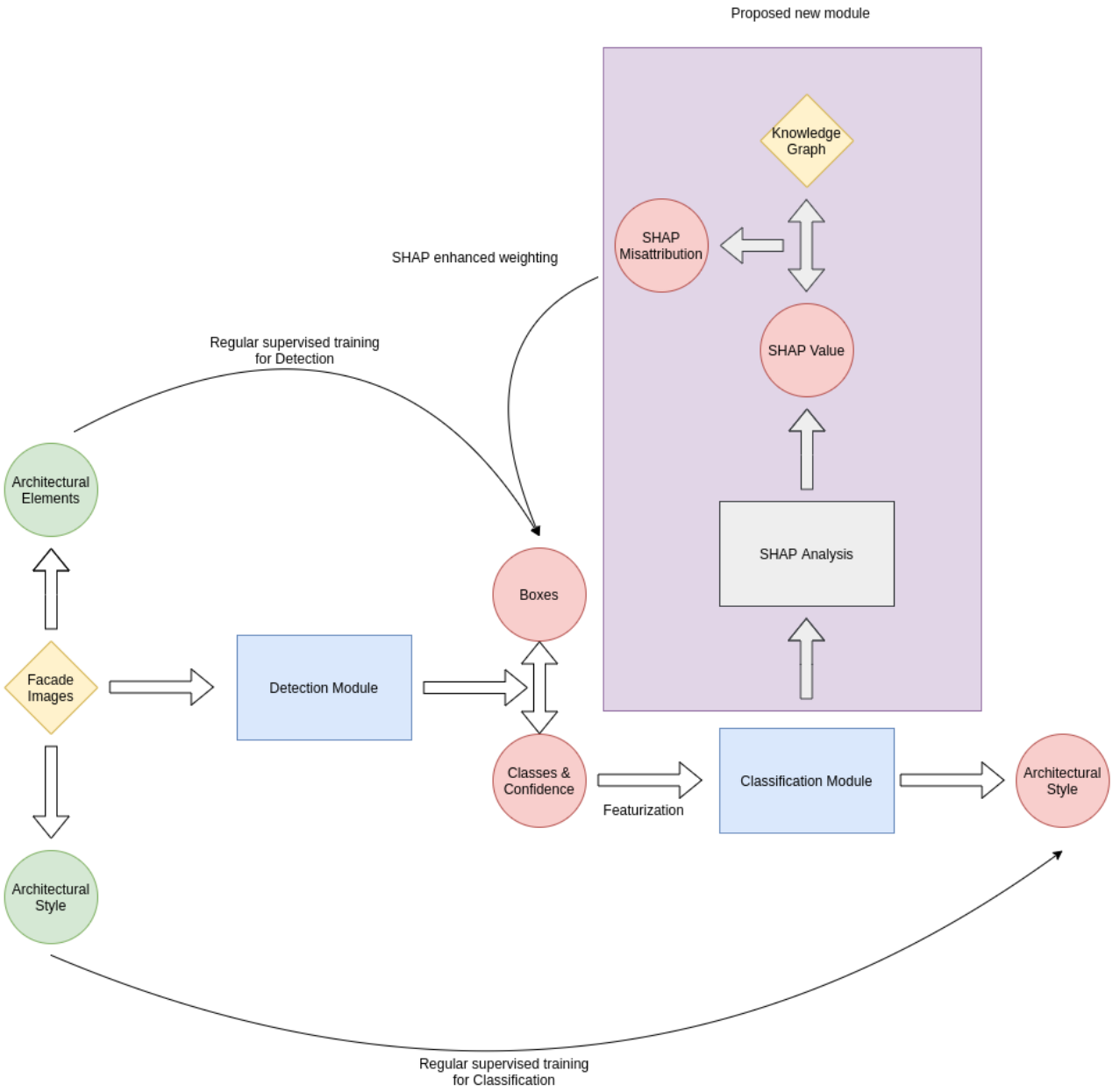


Figure 11: Diagram of proposed X-NeSyL methodology focused on the new SHAP-Backprop training procedure (in purple). In yellow the input data, in green the ground truth elements, in red predicted values (including classes and bounding boxes), and in blue trainable modules. Red blocks are the output of the various algorithms and gray blocks are untrainable elements. The purple block is the proposed X-NeSyL procedure. Thin arrows link together outputs of trainable modules along with reference elements used for backpropagation. Initially two thin arrows exist, one between the object (macro) label and the output of the classification model, and one between elements (object parts) and the output of the detection module. We add a third arrow (i.e., feedback signal) between the SHAP misattribution and the output of the object part detector as the new SHAP-penalized loss is leveraging this misattribution to penalize misalignment with the expert.

We now introduce the SHAP values, which are used as a constraining mechanism of our classifier model to be aligned with expert knowledge provided in the KG. SHAP values are computed after training the classification model.

Let $S = (S_1, S_2, ..., S_N)$ with $Si = (S_{i,1}, ..., S_{i,m})$, with m the number of different object (macro) labels, and let $S_{i,k} = (S_{i,k,1}, S_{i,k,2}, ..., S_{i,k,l})$ be the SHAP values for each training example i, where k is the macro object label, and l is the detected part. Each SHAP value is thus of size l, with l being the number of parts in the model. Furthermore, due to the nature of the output of the classification model, which are probabilities, and the way SHAP values are computed, they are bounded to be a real number in $[-1, 1]$.

The KG was already modeled as an attribution graph and corresponding matrix (in order to compute the embedding out of the KG) in Section 4.3.2, and we will be using the same notation.

Introducing the misattribution function β

To introduce SHAP-Backprop into our training, we first need to be able to compare the SHAP values with a ground truth, which here is represented by the expert KG. We thus introduce the misattribution function to assess the level of alignment of the feature attribution SHAP values with the expert KG.

The goal of the misattribution function is to quantitatively compare the SHAP values computed for the training examples with the KG. For that we assume the SHAP values are computed for all feature vectors. A misattribution value is then computed for each feature value of each feature vector. Before considering the definition of misattribution function, we can distinguish two cases when comparing these two elements, depending on the feature values observed:

A) The feature value considered is higher than a given hyperparameter v, i.e., the positive case. v symbolizes the value above which we consider a part is detected in our sample image. In our case $v = 0$.

B) The feature value is lower or equal to v: in this case we assume there is no detected part, i.e., the negative case.

Case A: In the first case, given the KG, for a SHAP attribution to be coherent, it should have the same sign as the KG. If it is the case, the misattribution is 0, i.e., there is no correction to be made and backpropagate. Otherwise, if it has opposite sign, the misattribution will depend on the SHAP value. In particular, it will be proportional to the absolute value of the SHAP attribution. We thus propose the following misattribution function:

$$\beta(S, KG, i, k, j) = (-KG_{k,j} \times S_{i,k,j})^{+}, \tag{5}$$

where i is the index of the considered image, k is the index of a given object (macro) label and j is the index of a given (object) part. This way $KG_{k,j}$ correspond to the edge value between the macro label k and the part j, where $(.)^{+}$ is the positive part of a real number. $KG_{k,j} = \pm 1$, and thus $\beta \in [0, 1]$ due to the bounding of the SHAP values. The detector output feature values are bounded due to the nature of the classification output which is in [0,1]. However, SHAP values are naturally bounded in [-1,1] [59].

Case B: Since we choose v to be 0 and v has only real values, if $v = 0$, we therefore should not backpropagate any error through the loss function, since no BB is detected for the object part.

Given the prior information in the KG, the posterior information (SHAP values post-training), and a way to compare them (attribution function $beta$), we suggest two new versions of a weighted loss, $\mathcal{L}_{SHAP}$, that will substitute the former ROI loss.

- Bounding Box-level weighting of the $\mathcal{L}_{SHAP}$ loss

This first weighted loss is at the bounding box (BB) level, meaning each BB will be weighted individually based on its label and the associated SHAP value. We propose the following loss:

$$\mathcal{L}_{SHAP} = \Sigma_{i=1}^{N} \Sigma_{k=1}^{N_i} \alpha_{BBox}(S, KG, i, C_i, Class_k) \mathcal{L}(I_i, (BB_k, Conf_k, Class_k)), \tag{6}$$

where N_i is the number of BBs predicted in image I_i, and $C = (C_1, ..., C_N)$ the ground truth (GT) labels for instance images $I = (I_1, I_2, ..., I_N)$. We propose two possible loss weighting options, depending on h,

a balancing hyperparameter (equal to 1 in our experiments), that can be linear:

$$\alpha_{BBox}(S, KG, i, C_i, Class_k) = h * \beta(S, KG, i, C_i, Class_k) + 1 \tag{7}$$

or exponential:

$$\alpha_{BBox}(S, KG, i, C_i, Class_k) = e^{h*\beta(S,KG,i,C_i,Class_k)}, \tag{8}$$

with i the index of the considered image, C_i its associated class, $Class_k$ the considered part class, KG the KG and S the SHAP values. Either way, if α is equal to 1, then the misattribution is 0 in order to maintain the value of the original loss function. Thus, $\alpha \in [1, \infty)$: the larger the misattribution, the larger the penalization.

- Instance-level weighting of the $\mathcal{L}_{SHAP}$ loss

This second weighted loss is at the instance-level, meaning we are weighting all the BBs for a given dataset instance (i.e., image) with the same value:

$$\mathcal{L}_{SHAP} = \Sigma_{i=1}^{N} \alpha_{instance}(S, KG, i, C_i) \Sigma_{k=1}^{N_i} \mathcal{L}(I_i, (BB_k, Conf_k, Class_k)), \tag{9}$$

where

$$\alpha_{instance}(S, KG, i, k) = max_{j \in [1,l]}(\alpha_{BBox}(S, KG, i, k, j)) \tag{10}$$

i.e., the instance-level weighting of the loss function considers the max BBox misattribution function value. Just as the BB level weighting, the aggregation of terms in the misattribution function can either be linear or exponential.

5. X-NeSyL methodology Evaluation: SHAP GED metric to report model explainability for end-user and domain expert audiences

Detection and classification modules of EXPLANet use mAP and Accuracy, respectively, as standard evaluation metric. In order to evaluate explainability of the model in terms of alignment with the KG, we propose the use of the SHAP Graph Edit Distance (SHAP GED) at test time. This metric has a well defined target audience: the end-user (in our case, of a citizen science application) and domain experts (art historians), i.e., users with non-technical background necessarily.

Even if the SAG above can be computed for any set of theoretical and empirical feature attribution sets, we are interested in using the GT KG in order to compute a explainability score on a test set.

The simplest way to compare two graphs is applying the GED [76]. Using straight up the GED between a KG and the SAG does not work very well, since the number of object parts (architectural elements in our case) detected vary too much from an image to another. What we do is to compare the SAG to the projection of the KG given the nodes present in the SAG. More precisely, given a SAG, we compute a new graph from the KG, where we take the subgraph of the KG that only contains the nodes in the SAG. As, such they will have the same nodes, but with the potential addition of new edges.

An example of such projection can be seen in Figure 12 (right). This way, the projection serves to only compute the relevant information given a specific image.

Once SHAP-Backprop procedure penalizes the missalignment of object parts with those of the KG (detailed in next section), we will use the SAG to compute the SHAP GED between the SAG and its projection in the KG. This procedure basically translates into counting the number of "wrong" edges in the SAG given the reference KG, i.e., the object parts that should not be present in this data point, given the predicted object label.

After detailing all necessary components to run the full pipeline of X-NeSyL methodology, together with an evaluation metric that facilitates the full process evaluation, we are in place to set up an experimental study to validate each component. It is worth noting that each component can be adapted to each use case. Next section experiments will demonstrate, with a real life dataset, how X-NeSyL methodology can facilitate the learning of explainable features, by fusing the information from deep and symbolic representations.

	Hisp.-Muslim	Gothic	Renaissance	Baroque	Feature vector
Horseshoe arch	-0.16	0	0.08	0.03	0
Lobed arch	0	0	0	0	0
Flat arch	0	0	0	0	0
Pointed arch	0	-0.15	0.07	0.04	0
Ogee arch	0	-0.08	0.01	0	0
Trefoil arch	0	0	0.04	0	0.2
Triangular pediment	0	0	0	0.06	0
Segmental pediment	0	0	0	0	0
Serliana	0	0	0	0	0
Rounded arch	0	0	0	0.03	1.35
Lintelled doorway	0	0	0	0	0
Porthole	0	0	0	0	0
Broken pediment	0	0	0.14	-0.16	0
Solomonic column	0	0	0.04	0	0

Table 3: Feature vector of a sample image and its SHAP analysis used for the construction of the SAG in Fig. 12, according to Algorithm 1.

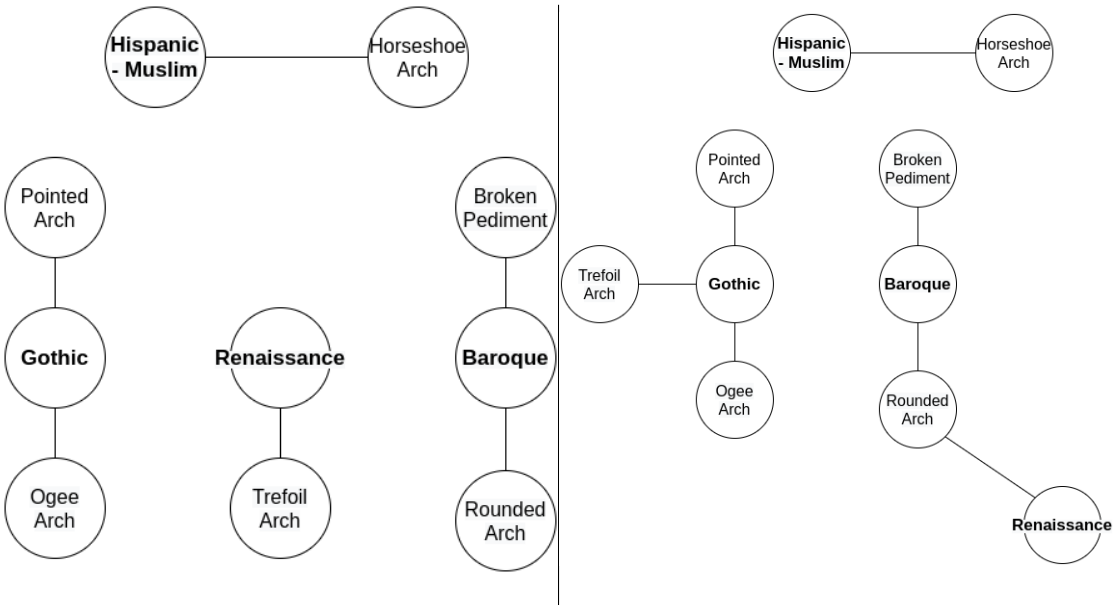


Figure 12: Example of applying SHAP analysis after training an architectural style classifier based on an architectural element detector, and computing a SHAP attribution graph (SAG, on the left). Given the nodes present in such SAG, the projection on the KG is performed (on the right) to get the non matching edges (and the graph Edit distance, GED) and compute the misattribution function β. The latter penalizes the model for assigning feature attribution that conflicts with the attribution of features an expert would assign (represented in Fig. 7 expert KG).

6. MonuMAI Case Study: Classifying monument facades architectonic styles

In order to evaluate the X-NeSyL methodology, and all inherent components including the detection and classification module within the proposed EXPLANet architecture, as well as the XAI training procedure, we perform two main studies. In the first study, we evaluate the full SHAP-Backprop training mechanism by testing the detection module followed by the classification one. In other words, we test and assess the full EXPLANet architecture. In the second study, as an ablation study to assess the influence of the detector's accuracy on the overall part-based classifier, we evaluate the detection module of EXPLANet model with two different detection models, Fast-RCNN [71] and RetinaNet [56].

In both evaluation studies, we used two datasets, MonuMAI and PASCAL-Part. For simplicity, we focus on MonuMAI dataset mainly in this section, while additional results for PASCAL-Part can be seen in the Appendix. For the remainder of the paper, we will use elements and object parts interchangeably, and macro object labels will be used to refer to the classification labels, i.e., the style of the macro object.

6.1. Experimental setup

To evaluate the classification performance we use the standard accuracy metric (equation 11).

$$accuracy = \frac{\#correct\ predictions}{\#total\ predictions} \tag{11}$$

, where $\#$ represents the number for correct and total predictions. To evaluate the detection performance, we use the standard metric mean average precision *mAP* (Eq. 12).

$$mAP = \frac{\sum_{i=1}^{K} AP_i}{K} \qquad AP_i = \frac{1}{10} \sum_{r \in [0.5,\ldots,0.95]} \int_0^1 p(r)dr \tag{12}$$

where given K categories of elements, p precision and r $recall$ define $p(r)$ as the area under the interpolated precision-recall curve for class i.

We initialized Faster R-CNN [71] and RetinaNet with the pre-trained weights on MS-COCO [57] then fine-tuned both detection architectures on the target datasets, i.e., MonuMAI or PASCAL-Part. The last two layers of Faster R-CNN were fine-tuned on the target dataset. As optimization method, we used Stochastic Gradient Descent (SGD) with learning rate of 0.0003 and a momentum of 0.1. We use *Faster R-CNN* implementation provided by native PyTorch's *torchvision* library. Concretely, Faster R-CNN is implemented using PyTorch packages *torch 1.7.0, torchvision 0.8.1*. The classifier after the detector uses Tensorflow 1.15 (*tensorflow-gpu 1.15.0*).

For the classification module, we also fine-tuned the two layer MLP with 11 intermediate neurons. We used the Adam [49] optimizer provided by Keras.

To perform an ablation study on the element or part-based detector, the original dataset is split into three categories (train, validation and test), following a 60/20/20 split. Reported results are computed on the test set.

The compositional part-based object classification with RetinaNet is trained in two phases. First the detection is trained by finetuning a RetinaNet-50 pretrained on MS COCO. We use Adam optimizer with starting learning rate (LR) of 0.00001 and a scheduler of learning rate to reduce on plateau with a patience of 4[6]. We train this way for 50 epochs. Then we freeze the whole detection weights and train only the classification. We use Adam optimizer with starting LR= 0.001 and a scheduler of LR to reduce on plateau with a patience of 4. We train this way for 25 epochs.

[6]Patience is the number of epochs taken into account for the scheduler to decide the network converged. Here, the last four.

Even if our objective was having a fully end-to-end training, the need for a quite different LR between the detection and classification modules led us to train separately for convenience, at the moment.

6.2. EXPLANet model analysis

In order to assess the advantages of EXPLANet, we consider three baselines: 1) MonuNet [51]: the architecture proposed with MonuMAI dataset, designed as a compressed architecture that is able to run in embedded devices such as smartphones[7]. 2) A simple object classifier based on vanilla ResNet-101 [38]. MonuNet is a different classification architecture to ResNet, it uses residual and inception blocks but with a more static architecture than EXPLANet that does not allow modifications or is not meant to be scalable.

3) A KG Deterministic classifier baseline, based on the same pipeline as EXPLANet, but using solely the expert KG. It first computes the detection scores and uses the same aggregation function (Eq. 1) as EXPLANet, but then the classification is done by summing for each class the contribution of all related object-parts that are typical of this object class, according to the expert KG. The resulting vector is normalized (to resize the output between 0 and 1 so it sums to 1) and an *arg max* is applied to decide the style, without any training. For an image I, the KG Deterministic classifier (DC_{KG}) is defined as follow:

$$DC_{KG}(I) = argmax_{k\in[1,m]}(confidence_k(I)) \tag{13}$$

where

$$confidence_k(I) = \sum_{j\in[1,l]} \frac{1 + KG_{k,j}}{2} v_j(I) \tag{14}$$

with v the image descriptor based on the detected object parts defined in Eq. 1, KG the knowledge graph defined in Eg. 5, and $k \in [1, m]$ the object index out of m classes and $j \in [1, l]$ the object-part index.

The results of EXPLANet classification model based on Faster R-CNN and RetinaNet detector backbones, together with these baseline classification networks are shown in Table 4 for MonuMAI dataset.

	mAP (detection)	**Accuracy** (classification)	**SHAP GED** (interpretability)
EXPLANet using Faster R-CNN backbone detector	42.5	86.8	0.93
EXPLANet using RetinaNet backbone detector	**49.5**	**90.4**	**0.86**
ResNet-101 classifier baseline	N/A	84.9	N/A
MonuNet classifier baseline	N/A	83.11	N/A
KG Deterministic classifier baseline	N/A	54.79	N/A

Table 4: Compositional vs *direct* (traditional) classification: Results of two backbone variants of EXPLANet (using object detector Faster R-CNN and RetinaNet) on MonuMAI dataset, and comparison with embedded version of the baseline model MonuNet, a vanilla classifier baseline with ResNet, and an expert KG-based deterministic (non-trained) classifier. In this baseline, the same results are obtained using majority voting based on the detection output of both object detectors, Faster R-CNN and RetinaNet (i.e., despite concrete examples having different confidence scores, the total right guesses are the same and result in the same accuracy). EXPLANet versions use the standard procedure, no SHAP-Backprop, to demonstrate that a part-based compositional classifier improves over a straight classifier (i.e., without a connected object-part detector as input).

Both versions of EXPLANet outperform the three baselines. Here results are shown with a Faster R-CNN baseline, but accuracy remains the same for RetinaNet, given that the majority voting on different attribute detection confidence scores results in the same final accuracy. In average, RetinaNet produces a detection and classification of better quality.

[7]Since MonuMaiKET detector and MonuNet classifier are not connected, MonuNet does not provide object detection.

MonuNet [51], the baseline provided by MonuMAI dataset authors, is an architecture designed for being used in mobile devices in real time. Because of its compressed design that targets embedded systems, its performance is not fully comparable with EXPLANet. However, we report it for reference as, to the best of our knowledge, it is the only previous model trained on the novel MonuMAI dataset to date.

Due to the naturally simpler nature of RetinaNet, the latter is faster to train than Faster R-CNN[8].

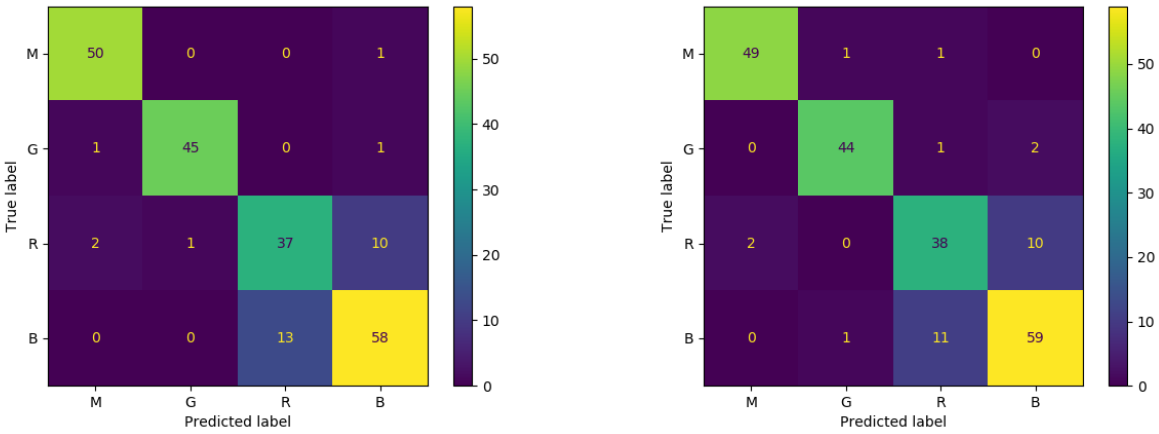


Figure 13: Confusion matrix of EXPLANet using Faster R-CNN (left) and EXPLANet using RetinaNet (right) on MonuMAI dataset for each architectural style class.

We can see the confusion matrix computed on MonuMAI for EXPLANet, using both Faster R-CNN and RetinaNet object detectors as backbones, in Fig. 13. Although a better detector (better mAP) could be intuitive to encourage a better GED, it is not expected, because the mAP evaluates the spatial location and the presence or not of a descriptor (object part), while the GED evaluates just the presence. Moreover, having no correlation among mAP and GED is reasonable, specially because mAP evaluates only one part of the model (detection), and thus it makes more sense that accuracy correlates with SHAP GED, as our results show for MonuMAI.

The object part detector module of MonuNet baseline, i.e., MonuMAIKET detector (based on Faster R-CNN detector -using ResNet101- as backbone) reaches slightly higher performance. We assume this minor difference due to the different TensorFlow and PyTorch default implementations of Faster R-CNN's inherent ResNet module versions in MonuNet and EXPLANet, respectively).

Interpretability-wise, EXPLANet with RetinaNet approach outperforms EXPLANet with Faster R-CNN. This probably stems from the object parts aggregation functions that are slightly different. In the Faster R-CNN version of EXPLANet, only the probability for the highest scoring label is kept, whereas in the case of using RetinaNet as part detector, the latter aggregates over all the scores for each example, with the sum function. This way RetinaNet is probably more robust to low score features as it always contemplates the several labels detected for each example.

At the moment minimal work is done to process the aggregating vector (i.e., vector accumulating the probabilities of each bounding box), but that could be explored in the future.

[8]We use the RetinaNet 100% PyTorch 1.4 (torch 1.7.0, torchvision 0.8.1) implementation from https://github.com/yhenon/pytorch-retinanet. Ease of use stems from the fact that if we wanted to modify the aggregation function, whether its analytical form or at the end of the detector, at which we should attach the classifier, it would be much simpler.

We observe that the more accurate the detector is, the more accurate the classification model is, and the more interpretable (lower SHAP GED) it becomes. However, future work should further investigate more complex tasks and datasets (as appendix will show on PASCAL-Part experiments), since not having annotations with sufficient quality can be detrimental to interpretability, and reinforce the interpretability-explainability trade-off.

The takeaway message from this experiment is that overall, both part-based models outperform the regular classification for MonuMAI, which means that detecting prominent and relevant object parts helps the classifier (a MLP) focusing on the important elements to draw a decision. In other words, detecting object parts helps the model putting attention on the parts of the image that matter, and making the prediction more interpretable. Furthermore, since ResNet is highly non interpretable on its own, tabular data is much easier to process for XAI methods.

6.3. SHAP-Backprop training analysis

Once assessed the EXPLANet architecture as a whole, and once performed an ablation study with respect to dependency on the object detector, we asses the different ways of weighting the training procedure penalization.

Table 5 displays the computed results of X-NeSyL methodology SHAP-Backprop method on *MonuMAI* [51] dataset (see additional results for PASCAL-Part [31] in Appendix). In bold are the best results for each metric, and in italics the second best. We tested on what we call *Standard procedure*, which is the typical pipeline of training first the detection module and then the classification module in two different steps, sequentially, without SHAP-Backprop nor any other interference.

The four other cases are methods computed with SHAP-Backprop. At each epoch in the detection phase, we train a classifier and use it to compute the SHAP values. These are then used to weight the detection loss with the misattribution function presented in the previous subsection.

	mAP (detection)	**Accuracy** (classification)	**SHAP GED** (interpretability)
Standard procedure (baseline, no SHAP-Backprop)	*42.5*	86.8	0.93
Linear BBox-level weighting	42.4	86.3	*0.69*
Exponential BBox-level weighting	42.3	**88.6**	1.16
Linear instance-level weighting	41.9	87.2	**0.55**
Exponential instance-level weighting	**44.2**	**88.6**	1.09

Table 5: Impact of the new SHAP-Backprop training procedure on the mAP (detection), accuracy (classification) and SHAP GED (interpretability). Results on MonuMAI dataset with EXPLANet architecture show a positive improvement on interpretability by using SHAP-Backprop with linear instance-level weighting. SHAP GED is computed between the SAG and its projection in the expert KG. (*Standard procedure* means a sequential typical pipeline of 1) train detector, 2) train classifier with no SHAP-Backprop).

Table 5. Applying SHAP-Backprop has little effect on accuracy and mAP. Nonetheless, all but the linear BBox-level weighting increase the classifier accuracy around 1-2%. These instabilities could probably be stabilized with further domain specific fine-tuning. Furthermore, we have to take into account the stochastic nature of the training process, since the SHAP value computation is approximated using a random subset of reference examples, to be more computationally efficient.

On the other side, in terms of interpretability, we do have a more sensible improvement, (reducing SHAP GED from 0.93 to 0.55) in the case of linear instance weighting.

The takeaway of this experiment is that there are other ways to improve interpretability than a good detector. For instance, using SHAP-Backprop. In particular, its linear weighting scheme here is stronger than exponential.

The gain obtained in both dimensions is large enough to conclude that the X-NeSyL methodology helps improving interpretability in terms of SHAP GED to the expert KG.

6.4. Lessons learned

In order to create a NeSy deep learning model that is explainable, we chose the expert as target audience of the model output explanation. We then proposed a training procedure and metric to qualitatively asses domain expert based explanations. Although further explainability metrics beyond SHAP GED could be studied depending on the audience of the explanation, we showed explainability, in terms of alignment of the conceptual match with the domain expert KG is increased.

The X-NeSyL methodology with pluggable components is meant to be a generic and versatile one, i.e., a template architecture that can be adapted and customized to each use case: the symbolic component for knowledge representation, the neural component based on a compositional architecture such as EXPLANet, and the XAI-informed misattribution procedure to be applied during training. Furthermore, our experiments verified our hypotheses:

1. Overall, X-NeSyL methodology brings explainability in the fusion of deep learning representations with domain expert knowledge. X-NeSyL did have the expected effect on the MonuMAI dataset, and SHAP-Backprop improves the explainability metric (SHAP GED) on the model.

2. The intermediate learned representation of EXPLANet allows to remove noisy information. Learning to classify by detecting object parts first shows to be more interpretable can also improve precision. For instance, some monuments are defined by a single element (e.g., *broken pediment*), to which the system learns to give an importance greater than the others.

3. Not always more accurate means more interpretable. However, we showed with the MonuMAI use case that it it possible that the more accurate EXPLANet classifier model is, the more interpretable (lower SHAP GED) it becomes[9]. Even if different error weighting schemes in SHAP-Backprop display this phenomenon is not always the case, we encourage future avenues of research to further explore this hypothesis on more complex tasks.

4. Despite some weighting schemes confirming the interpretability-performance trade-off (some improve SHAP GED while worsening accuracy, and viceversa), if the main objective is improving interpretability, we recommend using SHAP-Backprop with the linear instance-level weighting scheme. This is able to improve over both interpretability and performance.

5. Linear weighting yields better results than exponential due to the distribution of the misattribution function depending on the SHAP value. Exponential weighting focuses on examples that are extremely wrong, whereas the linear one gives a similar importance to all wrong examples. As SHAP GED does not take into account the SHAP value, but only whether there is a connection in the attribution graph, negating or correcting all examples similarly is the best approach when seeking explainability. Regarding bounding box aggregation function, instance-level works better than the BBox-level because the latter might slightly overfit each box, whereas instance-level gives a more average value to each image.

As there is no consensus on how to measure explainability, especially because methods are achieving different goals, the lack of unification moved us to develop and contribute our own metric, SHAP GED, because as far as we know no other work explicitly incorporates the use of expert domain KGs in an image classification process to produce explanations for end-users and domain experts. We encourage researchers to put effort to further explore expert knowledge alignment models and develop richer metrics beyond this scope.

[9] As explained in the PASCAL-Part dataset experiments, this is due to a debility of PASCAL-Part dataset and its label names not being specifically designed for a compositional whole-part classification task (since they become noisy as they use the same object part label for semantically different concepts).

When it comes to the semantic modelling of the KG, we explored the possibility of using the ontology OWL format, but compared to standard ontologies such as ArCo [13] or the Google Knowledge Graph, our domain knowledge on architectonic styles is rather flat in terms of hierarchies of triples. We therefore did not need in this case the additional semantics modelling power of the Web Ontology Language (OWL) and limited ourselves to explain *partOf* relationships. This permitted us to simplify the explanations and edge semantics. Future work should consider more complex semantic constraints natural of OWL format.

Regarding the reproducibility or scalability of X-NeSyL methodology, manually constructing KGs for a given dataset in our case was not hard, given the scale of MonuMAI or PASCAL-Part datasets. For larger datasets where no domain expert KG is available, one debility of X-NeSyL methodology, in concretely of using KGs as symbolic knowledge representation, is the needs for the domain expert to design the KG. This may require, if experts are limited, or data is disperse and sparse, to previously recur to knowledge engineering tasks, among others, automatic knowledge base construction and datatype learning [43, 23], relation learning [64], link prediction [36, 84], concept induction [77] or entity alignment [99].

7. Conclusions and Future work

With the presented work we open up different research horizons and future avenues of work that we detail in this section.

We extensively considered what is one of the most crucial points to be addressed while developing XAI methods. Within the general needs for producing more trustworthy outputs, we tackled the challenge of fusion and alignment of deep learning representations with domain expert knowledge. To achieve this we proposed a new methodology, X-NeSyL, to fuse deep and symbolic representations thanks to an explainability feedback mechanism that facilitates the alignment of both deep and symbolic features. The part-based detection and classification, EXPLANet, and XAI-informed training procedure SHAP-Backprop leverage expert information in form of a knowledge graph. X-NeSyL could be seen as one way to attain explainable and theory-driven data science [4].

We demonstrated the full pipeline of X-NeSyL methodology on MonuMAI and PASCAL-Part datasets, and the EXPLANet model with two variants of object detectors. The fusion of learned representations of different nature through the addition of an XAI technique component facilitates the model to learn with a human expert in the loop.

X-NeSyL methodology was also validated through a contributed audience-specific explainability metric, SHAP GED, that quantifies the alignment of the X-NeSyL methodology neural model (EXPLANet) with the symbolic representation of the expert knowledge. All models, datasets, training pipeline and metric of the showcased X-NeSyL methodology are available online[10].

This approach targeted compositional object recognition based on explaining the whole through the object-parts on deep architectures. However, other non compositional semantic properties of description logics could be further modelled in order to assess, and further constrain the level of alignment of a DL model with symbolic knowledge representing the expert.

Given the diverse contributions of this work, there is a broad set of options that can follow up to improve X-NeSyL methodology. In terms of evaluation of our work, the assessment was limited by the number of available datasets that contain part-based data, which is not large, since they must include a corresponding KG as well.

The explainability metric may be refined, since the proposed vanilla version of SHAP GED might not take into account all explainable factors an expert would like to see reflected in a black box model explanation. Future work includes assessing the SHAP GED metric itself, as the most suitable graph comparison metric, and including more elaborated datasets with finer grained object-part labels.

[10]Models available online: https://github.com/JulesSanchez/X-NeSyL

The ontology alignment with the deep model predictions can be refined in many ways. For instance, instead of using a simple KG, representing the expert knowledge in a rich ontology that incorporates extra axiomatic restrictions between elements, such as spatial relations or geometric constraints, could be useful to further improve SHAP-Backprop.

One way to improve the model along these lines may be inducing spatial structure in the embedding space (e.g. with approaches such as *ConvE* [22], which uses CNNs on embeddings for link prediction). Works along these lines could be explored to make possible that a KG would be learned (instead of being given to the DL model).

An actionable future work that could be very valuable for the XAI field is providing textual explanations of the output, since even limiting the model to describe the SAG could help build trust in the model output.

To conclude, we invite researchers and domain experts to be part of the XAI debate and contribute to democratize XAI, and to collaboratively design quantitative metrics and assessment methods aimed at developers, domain experts and end-users, as target audiences of DL model explanations.

Acknowledgement(s)

This research was funded by the French ANRT (Association Nationale Recherche Technologie - ANRT) industrial Cifre PhD contract with SEGULA Technologies, the Andalusian Excellence project P18-FR-4961 and the Spanish National Project PID2020-119478GB-I00. S. Tabik was supported by the Ramon y Cajal Programme (RYC-2015-18136). N. Díaz-Rodríguez is currently supported by the Spanish Government Juan de la Cierva Incorporación contract (IJC2019-039152-I). We also acknowledge the funding for open access charge covered by Universidad de Granada / CBUA.

References

[1] *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2 edition, 2007.

[2] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 9505–9515, 2018.

[3] Jacob Andreas. Measuring compositionality in representation learning. *arXiv preprint arXiv:1902.07181*, 2019.

[4] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58:82–115, 2020.

[5] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7), 2015.

[6] Adrien Bennetot, Jean-Luc Laurent, Raja Chatila, and Natalia Díaz-Rodríguez. Towards explainable neural-symbolic visual reasoning. In *Proceedings of the Neural-symbolic learning and Reasoning Workshop, NeSy-2019 at International Joint Conference on Artificial Intelligence (IJCAI), Macau, China*, 2019.

[7] Elliot Joel Bernstein and Yali Amit. Part-based statistical models for object classification and detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 734–740. IEEE, 2005.

[8] Tarek R Besold, Artur d'Avila Garcez, Sebastian Bader, Howard Bowman, Pedro Domingos, Pascal Hitzler, Kai-Uwe Kühnberger, Luis C Lamb, Daniel Lowd, Priscila Machado Vieira Lima, et al. Neural-symbolic learning and reasoning: A survey and interpretation. *arXiv preprint arXiv:1711.03902*, 2017.

[9] Federico Bianchi, Matteo Palmonari, Pascal Hitzler, and Luciano Serafini. Complementing logical reasoning with sub-symbolic commonsense. In *RuleML+RR*, volume 11784 of *Lecture Notes in Computer Science*, pages 161–170. Springer, 2019.

[10] Alexander Binder, Sebastian Bach, Gregoire Montavon, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for deep neural network architectures. In *Information Science and Applications (ICISA) 2016*, pages 913–922. Springer, 2016.

[11] Kurt Bollacker, Natalia Díaz-Rodríguez, and Xian Li. Extending knowledge graphs with subjective influence networks for personalized fashion. In *Designing Cognitive Cities*, pages 203–233. Springer, 2019.

[12] Vanessa Buhrmester, David Münch, and Michael Arens. Analysis of explainers of black box deep neural networks for computer vision: A survey. *arXiv preprint arXiv:1911.12116*, 2019.

[13] Valentina Anita Carriero, Aldo Gangemi, Maria Letizia Mancinelli, Ludovica Marinucci, Andrea Giovanni Nuzzolese, Valentina Presutti, and Chiara Veninata. ArCo: The Italian cultural heritage knowledge graph. In *International Semantic Web Conference*, pages 36–52. Springer, 2019.

[14] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847. IEEE, 2018.

[15] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Proceedings of the IEEE Conference on Computer Vision and Rattern Recognition*, pages 1971–1978, 2014.

[16] Roberto Confalonieri, Tillman Weyde, Tarek R Besold, and Fermín Moscoso del Prado Martín. Using ontologies to enhance human understandability of global post-hoc explanations of black-box models. *Artificial Intelligence*, 296:103471, 2021.

[17] Alessandro Daniele and Luciano Serafini. Neural networks enhancement through prior logical knowledge. *CoRR*, abs/2009.06087, 2020.

[18] Artur S. d'Avila Garcez, Marco Gori, Luís C. Lamb, Luciano Serafini, Michael Spranger, and Son N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics - IfCoLog Journal of Logics and their Applications (FLAP)*, 6(4):611–632, 2019.

[19] Artur S. d'Avila Garcez, Luís C. Lamb, and Dov M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer, 2009.

[20] Artur S. d'Avila Garcez and Gerson Zaverucha. The connectionist inductive learning and logic programming system. *Applied Intelligence*, 11(1):59–77, 1999.

[21] R De Kok, T Schneider, and U Ammer. Object-based classification and applications in the alpine forest environment. *International Archives of Photogrammetry and Remote Sensing*, 32(Part 7):4–3, 1999.

[22] Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818, February 2018.

[23] Natalia Díaz-Rodríguez, Aki Härmä, Rim Helaoui, Ignacio Huitzil, Fernando Bobillo, and Umberto Straccia. Couch potato or gym addict? semantic lifestyle profiling with wearables and fuzzy knowledge graphs. In *6th Workshop on Automated Knowledge Base Construction, AKBC@NIPS 2017, Long Beach, California*, 2017.

[24] Natalia Díaz-Rodríguez and Galena Pisoni. Accessible cultural heritage through explainable artificial intelligence. In *Adjunct Publication of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, pages 317–324, 2020.

[25] Michelangelo Diligenti, Marco Gori, and Vincenzo Scoca. Learning efficiently in semantic based regularization. In *ECML/PKDD (2)*, volume 9852 of *Lecture Notes in Computer Science*, pages 33–46. Springer, 2016.

[26] Ivan Donadello and Luciano Serafini. Integration of numeric and symbolic information for semantic image interpretation. *Intelligenza Artificiale*, 10(1):33–47, 2016.

[27] Ivan Donadello and Luciano Serafini. Compensating supervision incompleteness with prior knowledge in semantic image interpretation. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[28] Ivan Donadello, Luciano Serafini, and Artur D'Avila Garcez. Logic tensor networks for semantic image interpretation. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*, pages 1596–1602, 2017.

[29] Honghua Dong, Jiayuan Mao, Tian Lin, Chong Wang, Lihong Li, and Denny Zhou. Neural logic machines. In *ICLR*, 2019.

[30] Monireh Ebrahimi, Aaron Eberhart, Federico Bianchi, and Pascal Hitzler. Towards bridging the neuro-symbolic gap: deep deductive reasoners. *Applied Intelligence*, pages 1–23, 2021.

[31] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[32] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2009.

[33] Jerry A Fodor and Ernest Lepore. *The compositionality papers*. Oxford University Press, 2002.

[34] Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

[35] Weifeng Ge, Xiangru Lin, and Yizhou Yu. Weakly supervised complementary parts models for fine-grained image classification from the bottom up. In *Proceedings of the IEEE Conference on Computer Vision and Rattern Recognition*, pages 3034–3043, 2019.

[36] Lise Getoor and Christopher P Diehl. Link mining: a survey. *ACM SIGKDD Explorations Newsletter*, 7(2):3–12, 2005.

[37] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[38] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Rattern Recognition*, pages 770–778, 2016.

[39] Xuehai He, Xingyi Yang, Shanghang Zhang, Jinyu Zhao, Yichen Zhang, Eric Xing, and Pengtao Xie. Sample-Efficient Deep Learning for COVID-19 Diagnosis Based on CT Scans. *medRxiv*, 2020.

[40] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[41] Andreas Holzinger, Markus Plass, Michael Kickmeier-Rust, Katharina Holzinger, Gloria Cerasela Crişan, Camelia-M Pintea, and Vasile Palade. Interactive machine learning: experimental evidence for the human in the algorithmic loop. *Applied Intelligence*, 49(7):2401–2414, 2019.

[42] Daniel Huber, Anuj Kapuria, Raghavendra Donamukkala, and Martial Hebert. Parts-based 3d object classification. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2004.*, volume 2, pages II–II. IEEE, 2004.

[43] Ignacio Huitzil, Umberto Straccia, Natalia Díaz-Rodríguez, and Fernando Bobillo. Datil: learning fuzzy ontology datatypes. In *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 100–112. Springer, 2018.

[44] Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. The compositionality of neural networks: integrating symbolism and connectionism. *arXiv preprint arXiv:1908.08351*, 2019.

[45] Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, 2019.

[46] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Rattern Recognition*, pages 1725–1732, 2014.

[47] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[48] Pieter-Jan Kindermans, Kristof T Schütt, Maximilian Alber, Klaus-Robert Müller, Dumitru Erhan, Been Kim, and Sven Dähne. Learning how to explain neural networks: PatternNet and PatternAttribution. In *6th International Conference on Learning Representations, ICLR*, 2018.

[49] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, ICLR*, 2015.

[50] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

[51] Alberto Lamas, Siham Tabik, Policarpo Cruz, Rosana Montes, Álvaro Martínez-Sevilla, Teresa Cruz, and Francisco Herrera. MonuMAI: Dataset, deep learning pipeline and citizen science based app for monumental heritage taxonomy and classification. *Neurocomputing*, 420:266–280, 2020.

[52] Yann Le Cun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, pages 396–404, 1989.

[53] Freddy Lecue. On the role of knowledge graphs in explainable AI. *Semantic Web*, 11(1):41–51, 2020.

[54] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[55] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information fusion*, 58:52–68, 2020.

[56] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[57] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[58] Sindy Löwe, David Madras, Richard Zemel, and Max Welling. Amortized causal discovery: Learning to infer causal graphs from time-series data. *arXiv preprint arXiv:2006.10833*, 2020.

[59] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 4765–4774, 2017.

[60] Gianluca Maguolo and Loris Nanni. A critic evaluation of methods for COVID-19 automatic detection from x-ray images. *arXiv preprint arXiv:2004.12823*, 2020.

[61] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Proceedings of the International Conference on Neural Information Processing Systems*, volume 31, pages 3749–3759, 2018.

[62] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

[63] Giuseppe Marra, Michelangelo Diligenti, Francesco Giannini, Marco Gori, and Marco Maggini. Relational neural machines. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 1340–1347. IOS Press, 2020.

[64] Koji Maruhashi, Masaru Todoriki, Takuya Ohwa, Keisuke Goto, Yu Hasegawa, Hiroya Inakoshi, and Hirokazu Anai. Learning multi-way relations via tensor decomposition with neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[65] Pasquale Minervini and Sebastian Riedel. Adversarially regularising neural NLI models to integrate logical background knowledge. In *Proceedings of the SIGNLL Conference on Computational Natural Language Learning (CoNLL 2018)*, pages 65–74. Association for Computational Linguistics, 2018.

[66] Christoph Molnar. *Interpretable Machine Learning*. Lulu.com, 2020.

[67] Grégoire Montavon, Sebastian Lapuschkin, Alexander Binder, Wojciech Samek, and Klaus-Robert Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.

[68] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017. https://distill.pub/2017/feature-visualization.

[69] Galena Pisoni, Natalia Díaz-Rodríguez, Hannie Gijlers, and Linda Tonolli. Human-centred artificial intelligence for designing accessible cultural heritage. *Applied Sciences*, 11(2):870, 2021.

[70] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. ProbLog: A Probabilistic Prolog and Its Application in Link Discovery. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 2462–2467, 2007.

[71] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Proceedings of the International Conference on Neural Information Processing Systems*, pages 91–99, 2015.

[72] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "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*, pages 1135–1144, 2016.

[73] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1527–1535, 2018.

[74] Tim Rocktäschel and Sebastian Riedel. Learning knowledge base inference with neural theorem provers. In *Proceedings of the Automatic Knowledge Base Construction AKBC Workshop @ The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*, pages 45–50. The Association for Computer Linguistics, 2016.

[75] Claudio Saccà, Stefano Teso, Michelangelo Diligenti, and Andrea Passerini. Improved multi-level protein-protein interaction prediction with semantic-based regularization. *BMC Bioinformatics*, 15:103, 2014.

[76] Alberto Sanfeliu and King-Sun Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, pages 353–362, 1983.

[77] Md Kamruzzaman Sarker and Pascal Hitzler. Efficient concept induction for description logics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3036–3043, 2019.

[78] Md Kamruzzaman Sarker, Joshua Schwartz, Pascal Hitzler, Lu Zhou, Srikanth Nadella, Brandon Minnery, Ion Juvina, Michael L Raymer, and William R Aue. Wikipedia Knowledge Graph for Explainable AI. In *Iberoamerican Knowledge Graphs and Semantic Web Conference*, pages 72–87. Springer, 2020.

[79] 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. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 618–626, 2017.

[80] Luciano Serafini and Artur S. d'Avila Garcez. Learning and reasoning with logic tensor networks. In *AI*IA*, volume 10037 of *Lecture Notes in Computer Science*, pages 334–348. Springer, 2016.

[81] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, pages 3145–3153. JMLR. org, 2017.

[82] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2015.

[83] Austin Stone, Huayan Wang, Michael Stark, Yi Liu, D Scott Phoenix, and Dileep George. Teaching compositionality to CNNs. In *Proceedings of the IEEE Conference on Computer Vision and Rattern Recognition*, pages 5058–5067, 2017.

[84] Fabian M Suchanek, Jonathan Lajus, Armand Boschin, and Gerhard Weikum. Knowledge representation and rule mining in entity-centric knowledge bases. In *Reasoning Web. Explainable Artificial Intelligence*, pages 110–152. Springer, 2019.

[85] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

[86] Ilaria Tiddi, Freddy Lécué, and Pascal Hitzler. *Knowledge Graphs for Explainable Artificial Intelligence: Foundations, Applications and Challenges*, volume 47. IOS Press, 2020.

[87] Geoffrey G. Towell and Jude W. Shavlik. Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1-2):119–165, 1994.

[88] Joseph Townsend, Thomas Chaton, and João M. Monteiro. Extracting relational explanations from deep neural networks: A survey from a neural-symbolic perspective. *IEEE Trans. Neural Networks Learn. Syst.*, 31(9):3456–3470, 2020.

[89] Emile van Krieken, Erman Acar, and Frank van Harmelen. Semi-supervised learning using differentiable reasoning. *Journal of Applied Logics - IfCoLog Journal of Logics and their Applications (FLAP)*, 6(4):633–652, 2019.

[90] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017.

[91] Joseph D Viviano, Becks Simpson, Francis Dutil, Yoshua Bengio, and Joseph Paul Cohen. Saliency is a possible red herring when diagnosing poor generalization. In *International Conference on Learning Representations*, 2021.

[92] Haofan Wang, Mengnan Du, Fan Yang, and Zijian Zhang. Score-cam: Improved visual explanations via score-weighted class activation mapping. In *CVPR 2020 Workshop on Fair, Data Efficient and Trusted Computer Vision*, 2020.

[93] Sarah Wiegreffe and Yuval Pinter. Attention is not not explanation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 11–20, Hong Kong, China, November 2019. Association for Computational Linguistics.

[94] Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Van den Broeck. A semantic loss function for deep learning with symbolic knowledge. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 5498–5507. PMLR, 2018.

[95] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the International Conference on Machine Learning*, pages 2048–2057. PMLR, 2015.

[96] Yuan Yang and Le Song. Learn to explain efficiently via neural logic inductive learning. In *ICLR*, 2020.

[97] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[98] Matthew D Zeiler, Graham W Taylor, and Rob Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *2011 International Conference on Computer Vision*, pages 2018–2025. IEEE, 2011.

[99] X. Zhao, W. Zeng, J. Tang, W. Wang, and F. Suchanek. An experimental study of state-of-the-art entity alignment approaches. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[100] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE Conference on Computer Vision and Rattern Recognition*, pages 2921–2929, 2016.

8. Appendix

In this appendix we extend the results obtained with MonuMAI dataset to a second dataset, PASCAL-Part, to further validate our results. We detail such datasets and the results obtained in the following sections.

8.1. Additional results: PASCAL-Part Dataset

In order to validate results with more than one part-based dataset, we expanded experiments to use an adapted version of PASCAL-Part [15] which provides two level of annotations: element annotations for the detection level (*object-parts*), and the macro (*whole*) level with image level labels.

PASCAL-Part Dataset

PASCAL VOC 2010 dataset is a popular dataset for the task of object detection. It is organized into 20 object classes [31]. PASCAL-Part dataset extends PASCAL VOC-2010 dataset with additional annotations by providing segmentation masks for each object part [15].

In this work, we use a curated version of the PASCAL-Part provided by [26][11]. The idea is to reduce the number of elements of the original PASCAL-Part by collapsing categories together such as "upper arm" and "lower arm" inside a single category "arm".

We created a second curated version of the dataset, as in the PASCAL-Part there can be several "macro" objects labelled within a single image, whereas we want to consider, to ease evaluation purposes, only images with one image-level label[12]. The total of 1448 remaining images include 20 macro categories and 44 different parts, whose distribution and some samples are shown in previous sections of the Appendix.

PASCAL-Part dataset classes and parts are in the following list. The first element represents each class, and it is followed by its corresponding part classes[13]:

- Bird: Torso, Tail, Neck, Eye, Leg, Beak, Animal Wing, Head
- Aeroplane: Stern, Engine, Wheel, Artifact Wing, Body
- Cat: Torso, Tail, Neck, Eye, Leg, Ear, Head
- Dog: Torso, Muzzle, Nose, Tail, Neck, Eye, Leg, Ear, Head
- Sheep: Torso, Tail, Muzzle, Neck, Eye, Horn, Leg, Ear, Head
- Train: Locomotive, Coach, Headlight
- Bicycle: Chain Wheel, Saddle, Wheel, Handlebar
- Horse: Hoof, Torso, Muzzle, Tail, Neck, Eye, Leg, Ear, Head
- Bottle: Cap, Body
- Person: Ebrow, Foot, Arm, Torso, Nose, Hair, Hand, Neck, Eye, Leg, Ear, Head, Mouth
- Car: License plate, Door, Wheel, Headlight, Bodywork, Mirror, Window
- DiningTable: DiningTable
- Pottedplant: Pot, Plant

[11] Available online at github.com/ivanDonadello/semantic-PASCAL-Part/.
[12] Therefore, we discarded all images where there was more than one macro object.
[13] In OWL language, the latter would be placed in their *hasPart* object property range

- Motorbike: Wheel, Headlight, Saddle, Handlebar
- Sofa: Sofa
- Boat: Boat
- Cow: Torso, Muzzle, Tail, Horn, Eye, Neck, Leg, Ear, Head
- Chair: Chair
- Bus: License plate, Door, Wheel, Headlight, Bodywork, Mirror, Window
- TvMonitor: Screen, TvMonitor

Labels for PASCAL-Part dataset used are shown in Figs. 14 and 15, and image samples are in Figs. 16 - 17.

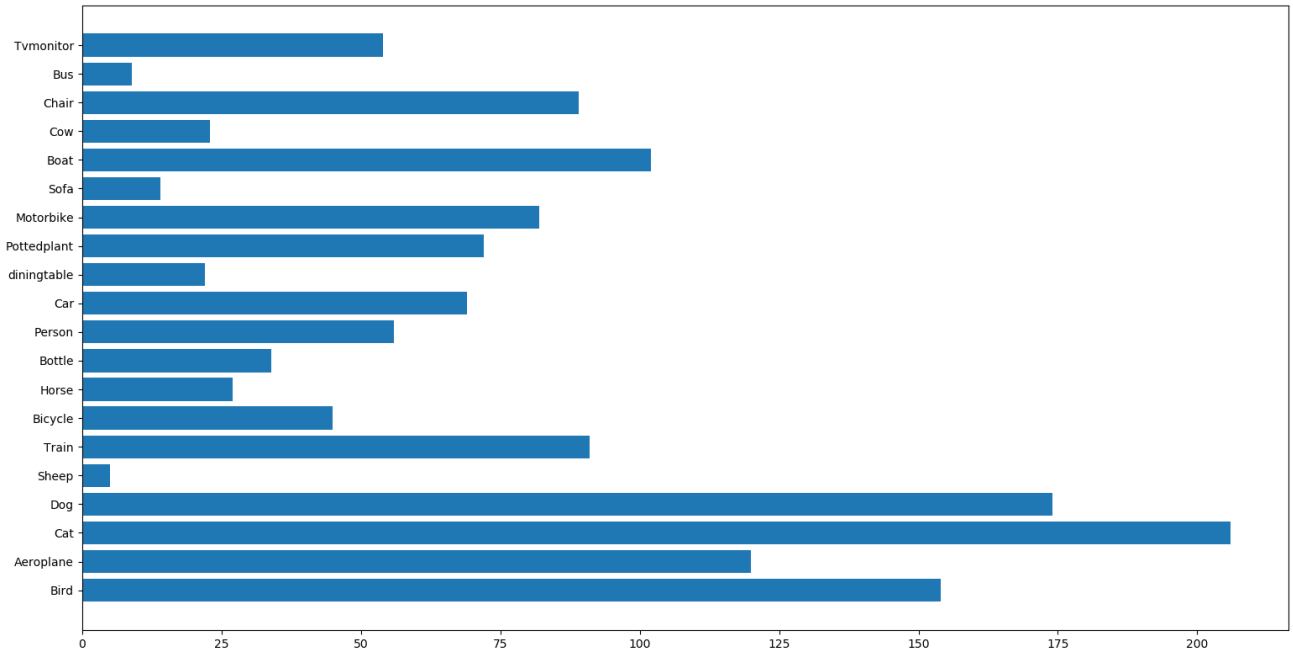


Figure 14: Distribution of classes in PASCAL-Part dataset.

Since explicitly using the ontology built for MonuMAI, in the case of object classification yielded no significant advantage,we did not pursue this direction further for PASCAL-Part dataset. A thorough work to convert PASCAL-Part into an ontology could be done, but the variety of elements inside it could make it difficult to 1) group them in meaningful categories, 2) extend the additional data and object properties of such richer ontology to a KG that can be compared with an attribution graph. Since such extension to an ontology can complicate the ontology - misattribution matching process, we leave such extension to future work.

PASCAL-Part's Knowledge Graph

In analogy to MonuMAI previous application, where architectural elements play the role of object parts and macro labels correspond to the architectural styles, we also used the KG provided by the PASCAL-Part

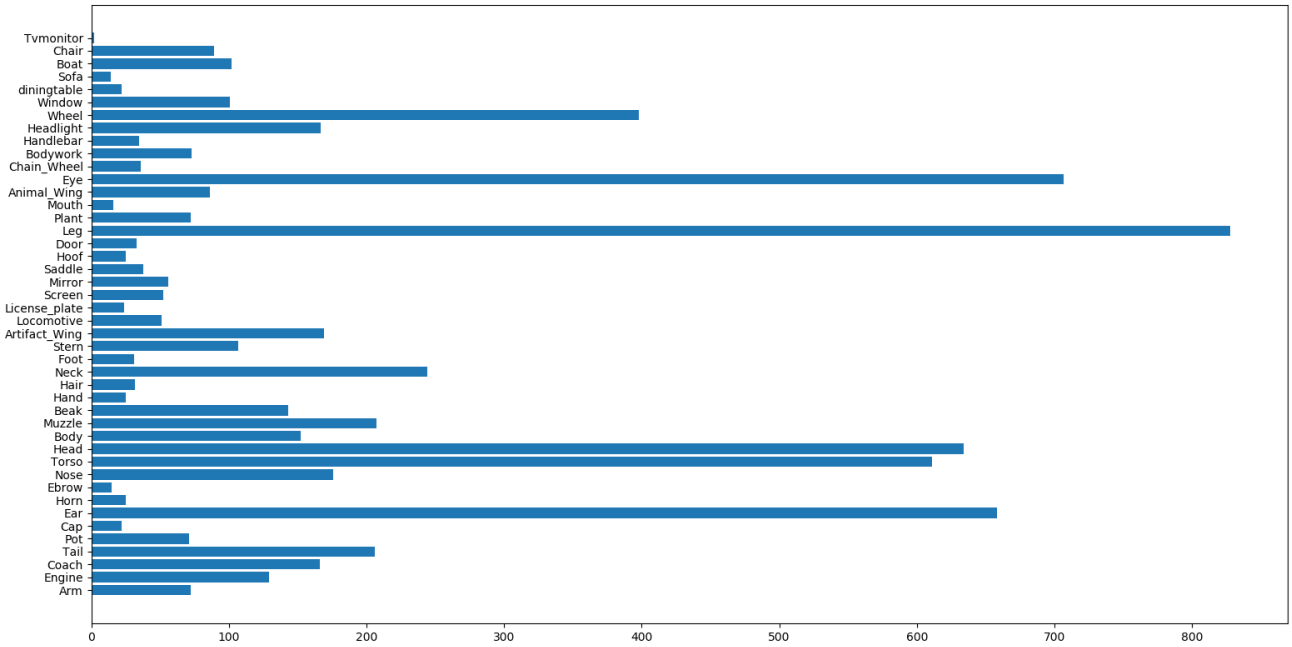


Figure 15: Distribution of object-parts in PASCAL-Part dataset.

dataset [26][14].

8.2. *Results for PASCAL-Part Dataset*

Results for *PASCAL-Part* dataset are compiled in Table 7. Applying SHAP-Backprop has almost no effect on the accuracy and interpretability, but it had detrimental effect on the detector mAP. This could be explained due to this particular KG being very sparsely populated, and the fact that object parts have large overlap in the object labels they theoretically contribute to. For instance, consider the following concrete example: We have a person in an image, for which we detect the legs, and let us assume the background is such that sheep legs are detected. According to our expert KG data, detecting legs makes sense toward predicting a person, and thus the sheep legs detection inside the background would not be discouraged. Future work should consider the distinction among syntactically equal (e.g. in image captioning tasks, the word *leg*) but semantically different parts of objects (an animal vs a human leg). In other words, the *isPartOf* relationship could be further specialized in our KG to a) have as range the class Leg, with subclasses *AnimalLeg* and *HumanLeg* (instead of just Leg as PASCAL-Part dataset has it now), or b) having *isPartOfAnimal, isPartOfHuman* as extra specialized object properties in our ontology (right now PASCAL-Part and MonuMAI only have one kind of object property, *hasPart*).

The overall lower score in mAP we obtain for *PASCAL-Part* stems from the fact that this dataset makes it harder for smaller objects to be detected, and the Faster R-CNN model we used was not fine-tuned to be fairly comparable in both settings.

When considering the different weighting schemes of SHAP-Backprop, for the PASCAL-Part, the vanilla ResNet classifier baseline performs better than that one for EXPLANet, which means that the object-part-based classifier EXPLANet underperforms in this case. It can be explained by several factors,

[14]Curated PASCAL-Part Dataset and KG available github.com/ivanDonadello/semantic-PASCAL-Part/. We do not provide a visualization of this KG as it would be unreadable.

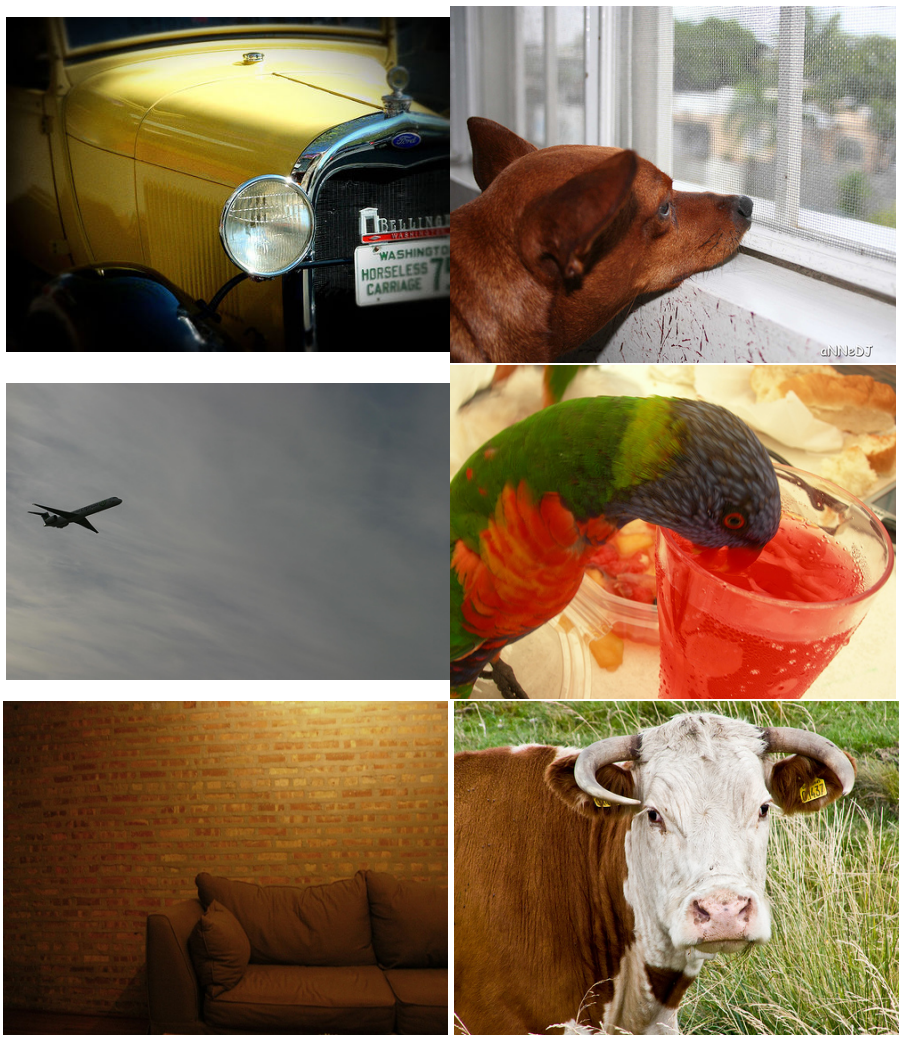


Figure 16: Extract from PASCAL-Part dataset (I).

	mAP (detection)	**Accuracy** (classification)	**SHAP GED** (interpretability)
EXPLANet using Faster R-CNN backbone detector	36.5	82.4	**0.45**
EXPLANet using RetinaNet backbone detector	**39.3**	80.3	0.59
ResNet-101 baseline classifier	N/A	**87.2**	N/A

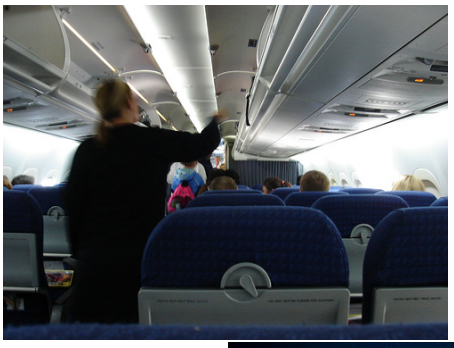
Table 6: Results of two backbone variants of EXPLANet (using Faster R-CNN and RetinaNet) on PASCAL-Part dataset.

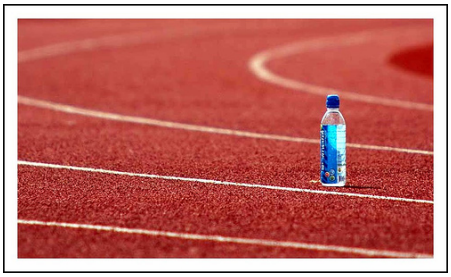


Figure 17: Extract from PASCAL-Part dataset (II).









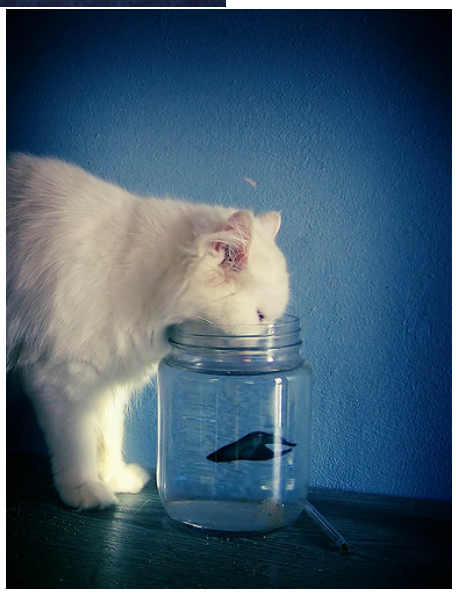


Figure 18: Extract from PASCAL-Part dataset (III).



Figure 19: Extract from PASCAL-Part dataset (IV).

but the predominant one is probably that images contain valuable information that the part-label model does not. It becomes quite clear when studying the PASCAL-Part KG, as we do in Section 8.1, since several labels are made of the same part names, but represent distinct things, i.e., parts from different object provenance (e.g. *leg* of a person and leg of an animal; both *car* and *bus* have the same object-parts). The part-based model has thus trouble differentiating such categories whereas a purely image-based model (and not attribute based) would have no issue with those.

	mAP (detection)	**Accuracy** (classification)	**SHAP GED** (interpretability)
Standard procedure (baseline, no SHAP-Backprop)	**36.5**	*82.4*	*0.45*
Linear BBox-level weighting	33.0	79.7	0.65
Exponential BBox-level weighting	32.0	81.0	0.55
Linear instance-level weighting	33.4	80.3	**0.42**
Exponential instance-level weighting	*34*	**82.7**	0.56

Table 7: Impact of the new SHAP-Backprop training procedure on the mAP (detection), accuracy (classification) and SHAP GED (interpretability). Results on PASCAL-Part dataset with EXPLANet architecture. GED is computed between the SAG and its projection in the expert KG. (Standard procedure: sequential typical pipeline of 1) train detector, 2) train classifier without SHAP-Backprop)

While the linear weighting appears to have a more positive effect on improving explainability of the model, it may not be significant, given that it does not always improve interpretability when applied on the more specific (but with lack of signal) PASCAL-Part dataset.

The discordance in performance (for mAP and Accuracy in the detector task) in Table 5 and 7 for RetinaNet being superior than Faster R-CNN only in MonuMAI but not for PASCAL-Part can be explained due to PASCAL-Part dataset labelling procedure (joining elements not unifiable with the same identifier). For instance: the same label is used for object parts of different semantic and visual meaning: *sheep*'s legs and *cow*' legs, or car's wheel and bike's wheel. Therefore, it is worth highlighting the differences in the labelling process of both datasets, as the classification based in parts with the same name but different semantics and visual appearance in PASCAL-Part is not designed for a neural network that only takes attributes as input. Thus, as PASCAL-Part lacks highly discriminative features, accuracy and SHAP GED are not obviously nor directly connected, specially in RetinaNet where, due to its inherent architecture aggregation function, all probabilities are used (i.e., summed) to perform a prediction, not just the highest one.

As micro and macro level labels are not appropriate (as designed in MonuNet dataset), the interpretability metric fails to reflect reality, independently of the quality of the detector.

The take away message from the ablation study assessing the impact of the object detector on EXPLANet (Table 6) is that even if X-NeSyL methodology showed slightly differently results in datasets designed with different purpose[15], having worse default results when using EXPLANet with RetinaNet instead of with Faster R-CNN could be due to 1) hyperparameter choice, since Faster R-CNN uses pretraining on MS-COCO while RetinaNet uses pretraining on ImageNet, and 2) both coarse grained MonuMAI dataset and fine-grained PASCAL-Part are of different nature in terms of the overlap among *part* classes. Lacking discriminative dataset labels in PASCAL-Part results in insufficient signal for EXPLANet to leverage. Thus, dataset label noise is not appropriate to capture the compositional part-based hierarchy in the data nor to 100% assess SHAP-Backprop.

[15] Independently assessed outside EXPLANet, RetinaNet is slightly superior to Faster R-CNN.