Ontologies to build a predictive architecture to classify and explain

Matthieu Bellucci, Nicolas Delestre, Nicolas Malandain, Cecilia Zanni-Merk

Ontologies to build a predictive architecture to classify and explain

Matthieu Bellucci, Nicolas Delestre, Nicolas Malandain, and Cecilia Zanni-Merk

Normandie Université, INSA Rouen, LITIS, Rouen 76000, France
https://www.litislab.fr/

Abstract. Explainable AI is gaining traction because of the widespread use of black box models in the industry. Many explanation methods are proposed to explain models without impacting their design. The literature describes a new architecture where an explainable model interacts with an explanation interface to generate explanations tailored for a user. We propose a novel image classification system that combines an ontology with machine learning models based on this architecture. It uses an ontology to add different labels to the same dataset and generates machine learning models to assess the class of an object and its different properties listed in the ontology. The outputs of these models are added to the ontology to verify that these predictions are consistent, using logical reasoning. The ontology can then be explored to understand the prediction and why it is consistent or not. This system can warn the user when a prediction is uncertain, which will help users to trust it.

Keywords: XAI · Ontology · Classification.

1 Introduction

The Explainable AI (XAI) field is rapidly rising in popularity. Numerous methods and tools are being developed to explain black-boxes [3] and create interpretable and accurate models [2,14]. The main goals of XAI are yet to be clearly defined, but some seem to make a consensus. These goals include a more trustworthy and robust AI [4] or including humans in the loop [1]. Gunning et al. [8] proposed an architecture for an explainable system where an explainable model is trained with a new machine learning process and interacts with an explanation interface to generate explanations. According to them, "systems will have the ability to explain their rationale, characterize their strengths and weaknesses, and convey an understanding of how they will behave in the future".

In this paper, we propose a new machine learning process and an explainable model by adding knowledge to the training and inference processes. This knowledge takes the form of an ontology. We specifically build several image classifiers capable of recognizing some properties in an image. These observed properties are represented in the ontology's TBox. They link two classes together, which

will allow the system to classify an object in an image based on what these models observed. We also use these links to relabel an image and train models with these new labels. An individual that represents the image is added to the ABox. It is linked to the predicted target classes for each observed property. A consistency check is then done using logical reasoning to verify that the classification is consistent with what was observed by the different models. This consistency check allows the system to warn the user when a prediction is uncertain and say why it is uncertain, which is a step towards what Gunning et al. [8] described as new machine learning systems.

In section 2 we review the literature, especially the use of semantic web technologies to build explainable systems. Then in section 3 we describe our proposed architecture. In section 4 we apply and test this system to the case of musical instruments classification. Finally in section 5 we discuss this approach, its limitations and future works.

2 Related works

There exists a plethora of methods that explain black box models, mainly using a technical analysis. Arya et al. [3] categorise them and propose a decision tree to choose the most suitable for the task and the type of explanation needed. Most of these methods function by perturbing the inputs of a model to understand their impact on the output thus determining which features are important for a given model. Rudin [13] advocates against using these methods, arguing that their explanations are not reliable and can be misleading. In the same paper, it is discussed that interpretable models should be preferred. Bellucci et al. [5] discuss the issue of the terminology in the XAI field and the lack of explicit definitions in the literature. They propose some definitions for different terms that we use in this paper. They define interpretability as "the ability of a system to be seen, understood and studied by a user with a reasonable cognitive effort".

Ontologies and specifically semantic web technologies represent knowledge in both a human-understandable and machine-readable way. Using such technologies may help design a system that can be seen, understood and studied by a human, making it interpretable. Seeliger et al. [15] provide a literature review of semantic web technologies for explainable machine learning models. They argue that explainability is dependent on the use of domain knowledge and that semantic web technologies might be a key to achieve truly explainable AI systems. For instance, Geng et al. [7] use a knowledge graph to automatically generate a classification model for an unseen class. It can justify the predictions by combining attention mechanisms with a knowledge graph.

Semantic web technologies can also be used to generate explanations. Futia and Vetrò [6] examine the advantages of integrating knowledge graphs into deep learning models. They claim that combining them enables the creation of interactive and cross-disciplinary explanations. Phan et al. [11] propose a system that learns user representation from health ontologies; explanations consist in showing which characteristics extracted from the ontologies have the largest influence

on the prediction. Halliwel et al. [9] propose a method to find the most intuitive explanations using facts of an ontology and evaluate the quality of these explanations. Rožanec et al. [12] propose feature-based explanations to extract the most important features and then use an ontology to get the context of these features and retrieve higher level concepts that describe these features. The context and higher level concepts are then combined to create an explanation.

Gunning et al. [8] propose a framework to design explainable systems, as we have already discussed. However, they do not mention the use of semantics to design these explainable models or explanation interfaces. From this review, we observe that existing systems cannot warn users when there is an uncertainty in their prediction, which prevents them from being trusted. In this paper, we propose a design for an explainable model and a machine learning process that incorporates semantic web technologies, in the form of an ontology, to advance towards an interpretable model. A logical reasoner can then be used to look for inconsistencies in the ontology which will detect problematic predictions and warn the user.

3 Ontology-based Image Classifier

We propose an ontology-based image classifier, which combines machine learning with an ontology. As we mentioned in section 1, we intend to use the architecture described in [8], that is to say, an explainable model interacting with an explanation interface. The following system is the explainable model and its training process. The goals of this classifier are to propose an explainable system for image classification that is trustworthy and robust. It is able to tell when a prediction is uncertain and inform the user if the prediction can be trusted.

The system trains multiple machine learning models, each capable of detecting one property visible in the image. For instance, one model will be tasked with detecting the texture in the image and another with detecting particular shapes. Then, a machine learning model that we call the *global* classifier is trained to directly classify the data. To make an inference, an image is given to every model. An individual is created and added to the ABox. This individual is linked by observable properties to instances of target classes that were detected in the image by the corresponding models. Finally, a logical reasoner is used to verify the consistency of the ontology. If the ontology is consistent, the prediction is considered valid. Otherwise, the user is warned that the prediction is uncertain.

In this section, we first define the vocabulary used in the rest of the paper to describe ontology elements. Then, we discuss the ontology architecture needed to make this system work. Finally, we describe the training and inference processes.

3.1 Vocabulary

In the following paper, the classes and object properties of an ontology are written in the following style: `MyClass` for classes and *`myObjectProperty`* for object properties. Class restrictions are presented as such: "*`myObjectProperty`* constraint `TargetClass`", where `TargetClass` is part of the range of *`myObjectProperty`*. In this restriction, we refer to *`myObjectProperty`* as the property of

the restriction and `TargetClass` as its target or target class. Negative restrictions are restrictions that mention what a class is not. They take the form "Not *myObjectProperty* constraint `TargetClass`". The other ontology related terms such as domain and range are defined in the W3C recommendation for the Web Ontology Language (OWL) [16] Model refers to any machine learning model, such as a neural network or a decision tree. We use the terms dataset class and dataset label interchangeably. We do so to align the dataset vocabulary with the ontology vocabulary. We use the definitions proposed in [5] for XAI terms.

3.2 Ontology building

This system uses the open-world assumption and the Web Ontology Language version 2 [16] (OWL 2). It requires the ontology to formally describe the data available. In order to exploit the data, object properties that describe observable characteristics in the data should be added to the TBox of the ontology. Their domain and range should not be empty sets. If that is the case, new classes should be declared and added to the empty domain or range. These observable characteristics or properties are declared as subproperties of *observableProperty* in the ontology, so that the system can identify and exploit them. The domain of these properties is the classes present in the dataset. The range can be any class or set of classes defined in the ontology.

The dataset classes must match with ontology classes. These classes must be defined with restrictions whose properties are subproperties of *observableProperty*. If a class does not have a restriction with this kind of property, it can not be used to train the models, rendering a part of the dataset unused. The class definitions should be as precise as possible to maximize the accuracy and diversity of the labels. Negative restrictions should also be used whenever possible to help the system find inconsistencies.

To summarize, dataset classes must be defined in the ontology. Their definitions should use subproperties of *observableProperty* in restrictions and be as precise as possible, using negative restrictions when feasible. Algorithm 1 illustrates a class definition for wooden chairs where its texture is an observable property. Here, wooden chairs are made of wood and cannot be made of metal. This algorithm is an excerpt of a larger ontology that can be found in our code[1].

Algorithm 1 Example of class definition, using OWL2 Manchester Syntax [17]

ObjectProperty: *hasTexture*, **SubPropertyOf**: *observableProperty*
Class: `WoodenChair`, **SubClassOf**: `Chair` that *hasTexture* some `Wood`,
SubClassOf: not *hasTexture* `Metal`

3.3 Training

A model per property The training process builds a model per subproperty of *observableProperty* They use the same unique dataset purposely relabelled for

[1] https://git.litislab.fr/s4xai/ontology-based-image-classifier/ contains the code for the whole system and the ontology used for this paper.

each model using the ontology and all take the same image as input. The labels for each model correspond to the classes in the range of the object property. A class can be defined by multiple restrictions with the same object property but different target classes. The model needs to be able to reflect this in its prediction; therefore, the labels are one-hot encoded, and the output of each model is a vector containing the probability of having each class as target of a restriction with the selected object property. This is a multi-label classification problem. For instance, for the object property *hasTexture*, a chair of class `Chair` with a class definition containing "*hasTexture* some `Metal`" and "*hasTexture* some `Wood`" has the label $(1, 1)$ where the first item corresponds to the class `Wood` and the second to `Metal`. Based on the definitions described in algorithm 1, this chair couldn't be classified as a `WoodenChair` even if the chair is mostly made of wood with some metallic parts. The definition should be refined to take this case into account. A plastic chair would have the label $(0, 0)$ since it is neither made of wood nor metal.

Some object properties are functional, meaning that for each individual x, there can be at most one distinct individual y such that x is connected by a functional property to y [18]. Thus, we simplify the training problem into a multi-class classification problem, where the output's sum should equal 1. To handle the case where the label is a zero vector, we need to add a temporary class that will be predicted when the image does not contain any target class of the object property. Finally, a model is built to classify the images directly. This model that we call *global* classifier in the following is trained using the original labels of the dataset.

Labelling the data Each model predicts the classes that are targets of an object property for a given image. Therefore, the dataset needs to be relabelled accordingly which is done by using the class definitions in the ontology. We describe how the system finds the new labels for a given class named `MyClass` and a given object property named *myProperty*.

In the class definition of `MyClass`, the system searches for restrictions containing the object property *myProperty*, i.e. restrictions of the form "`MyClass` *myProperty* some `TargetClass`". If some are found, the target classes of these restrictions will be the new labels for the image. This process is done for each subproperty of *observableProperty* and each class of the dataset.

When a subproperty of *observableProperty* is found in a class definition, the system maps this class to the targets of this property, e.g. if class `MyClass` is a subclass of "*hasTexture* some `Wood`" and "*hasTexture* some `Metal`", the system will store that if the label of the image is $MyClass$, then the label for the model of *hasTexture* is $(Wood, Metal)$.

3.4 Inference

The inference part of the system uses the machine learning models discussed in section 3.3 to predict the class and object properties of a data point. Algorithm 2 describes the different steps of an inference. Functions with an uppercase first

Algorithm 2 Inference algorithm

function INFER($x, onto, observablePropertiesSet, threshold^+, threshold^-$) ▷ x a data point, *onto* the ontology
Ensure: $isConsistent(onto)$
    $class \leftarrow globalClassifier(x)$ ▷ Corresponds to the class predicted by the *global* classifier
    $Declaration\,(NamedIndividual(indiv))$
    $ClassAssertion\,(class, indiv)$
    for all *property* in *observablePropertiesSet* **do**
        $y \leftarrow propertyClassifier(property, x)$ ▷ y is the vector of predictions.
        for $i \leftarrow 0, length(y)$ **do**
            $Declaration\,(NamedIndividual(target_i))$
            $ClassAssertion\,(class_i, target_i)$ ▷ $class_i$ is the i-th class in the labels for the classifier of *property*.
            if $y_i \geq threshold^+$ **then**
                $ObjectPropertyAssertion(property, indiv, target_i)$
            else if $y_i \leq threshold^-$ **then**
                $ClassAssertion\,(Not(property, class_i), indiv)$
            end if
            if not $isConsistent(onto)$ **then**
                return $False$
            end if
        end for
    end for
    return $True$
end function

letter are functions that modify the ontology, as described in functional syntax [18]. The other functions are designed by us.

The inference process computes the predictions of each model, adds the corresponding restrictions to a new individual in the ABox of the ontology and checks the consistency of the ontology with this new individual. The class of the individual is given by the *global* classifier. When a prediction is inconsistent, the system can warn the user when it believes it is wrong, which is a requirement of explainable systems highlighted by Gunning et al. [8]. Two parameters, $threshold^+$ and $threshold^-$ are introduced. They take advantage of the open-world assumption. Indeed, if a probability is in-between these thresholds, the system does not add a restriction and considers that the information is missing.

This architecture allows the generation of different types of explanations. Indeed, counterfactual explanations can be provided by adding or modifying restrictions in the ontology or modifying the thresholds. Using a logical reasoner to check the consistency of the ontology allows the construction of explanations based on this logical reasoning since it uses concepts defined in the ontology, which are human-understandable. Finally, any machine learning model can be used with this system. The ideal case would be to use interpretable models so that the entire system can be explained.

4 Experimentation

4.1 Problem description

We evaluate this system on a musical instrument classification task. We have designed an ontology based on a simple hierarchy of instruments families. The selected families of instruments are brass, woodwinds and strings. We also added unrelated objects such as chairs, tables, and pipes to highlight this system's ability to classify a variety of items within the same ontology.

This task is well-suited for our system because there already exists a taxonomy of musical instruments which helped us design the ontology. Instruments also have particular observable properties that enables us to distinguish them. Woodwinds and brass instruments all have a single mouthpiece, which is visible on most images. The different mechanisms of instruments can also be used to differentiate them. For instance, most woodwinds use keys while most brass use pistons. However, there are some overlaps, the serpent is from the brass family because it has a brass mouthpiece but it has a wood texture and keys as mechanism. A single classifier would probably classify the serpent as a woodwind because of its visual similarity with woodwinds.

We have gathered and handpicked images from Google Images that correspond to some ontology classes for a total of 20 classes. The dataset contains 5642 images with an average of 250 images per class, resized to 224×224 pixels. This dataset is imbalanced; for instance, the class `Serpent` which corresponds to an ancestor of the tuba, has 151 images, whereas `Saxophone` has 476 images. We have defined 5 subproperties of *`observableProperty`*: *`hasTexture`*, *`hasMechanism`*, *`hasMouthpiece`*, *`hasShape`* and *`hasApparentStrings`*. These observable properties are based on what is visible in the majority of the images. To build the different models of the system, we use convolutional neural networks, with the ResNet50 architecture [10]. These models are pretrained on the ImageNet dataset, and only the last layer is modified and finetuned with our dataset, using the labelling process described in section 3.3. The activation function for the last layer depends on the property. For functional object properties, a softmax function is used. Otherwise, it is a sigmoid.

4.2 Evaluation and results

We want to evaluate the capacity of our system to accurately tell when a prediction is correct or incorrect and compare it to the *global* classifier's performance. The correctness of a prediction corresponds to whether the class prediction from the *global* classifier is correct or not. Therefore, we divide the predictions into 4 categories: Correct Consistent (CC), Correct Inconsistent (CI), Incorrect Consistent (IC) and Incorrect Inconsistent (II). A 5-fold cross-validation was performed on the system to compute evaluation metrics. Table 1 shows a few metrics computed for each class then aggregated with the mean value, the maximum and minimum values and the standard deviation to observe the dispersion of the results for each class. The code and dataset is available at https://git.litislab.fr/s4xai/ontology-based-image-classifier/ .

Table 1. Results of the experiment

Metric	Mean	Max	Min	Standard Deviation
Global classifier accuracy	0.80	0.97	0.34	0.19
System accuracy	0.75	0.95	0.33	0.18
Adjusted System accuracy	0.79	0.96	0.40	0.17
False Positive Rate	0.72	0.94	0.35	0.17
False Negative Rate	0.05	0.22	0	0.06

The system accuracy is computed as $\frac{CC}{n}$ because inconsistent predictions are marked as wrong. It is not a fair comparison with the *global* classifier accuracy because the latter is computed as $\frac{CC+CI}{n}$ which is always greater than the system's accuracy. The system's goal is to detect correct and incorrect predictions. This is why the adjusted system accuracy shown in this table is computed with the formula $\frac{CC+II}{n}$. The system accuracy is similar to the *global* classifier accuracy, which means the impact of the consistency check is minimal on performance. The accuracy is biased since a majority of the predictions are correct. That is why we study the False Positive Rate (FPR) and False Negative Rate (FNR), considering IC as false positives and CI as false negatives. The values of FPR and FNR show that the system classifies most predictions as consistent, regardless of the prediction's correctness. This issue may come from two factors.

The first factor is the ontology's design. Since the consistency is checked using the open-world assumption, the predictions are unlikely to be inconsistent if class definitions do not have strong restrictions, such as negative restrictions. The open-world assumption is crucial for our system because it does not consider that missing properties are proof of their absence, thus allowing the system to handle the uncertainty of some predictions. Some classes also have the same definitions that only use observable properties because of the low amount of observable properties and the granularity of property ranges. The issue then becomes the amount of data available to train the property models since the smaller the granularity, the less data is available for training per target class.

A second factor is the values of the thresholds described in section 3.4. They have a direct impact on FPR and FNR. Indeed, these thresholds decide whether to add a restriction or not. When both thresholds are at 0.5, a restriction will always be added to the individual, leading to a spike in inconsistent predictions because the system will be highly sensitive to the performance of the property models leading to a high FNR and low FPR. Threshold values at respectively 0 and 1 mean no restrictions will be added, making every prediction consistent leading to a high FPR and low FNR.

In this experiment, these thresholds were 0.7 and 0.3 for $threshold^+$ and $threshold^-$ respectively. From our observations, it is clear that the thresholds should be different per model to account for the variation in their performance. We have a high FPR and low FNR, which means that not enough restrictions are added; thus, we should probably decrease $threshold^+$ and increase $threshold^-$.

This experiment highlighted the challenges to make these consistency checks as accurate as possible. A first challenge is the choice of thresholds. Many factors

need to be taken into account to decide these thresholds, such as the performance of each model, the sensitivity of the task and the cost of false positives and false negatives. Another challenge is the definitions in the ontology's TBox. There is a trade-off in the precision of the definitions. Too precise definitions imply fewer data for the training, but too broad definitions mean inconsistencies are unlikely to be detected. Hence, data availability is an essential factor in the system's quality. We chose to relabel a dataset, but it is possible to gather different datasets corresponding to the different object properties, which we believe would lead to better performances. However, finding such specialized datasets is not always possible, which is why we decided to showcase the ability of our system to relabel existing data.

5 Conclusion and future work

This paper introduced a new system that combines ontologies with machine learning models as well as a novel training process that exploits the ontology to relabel a dataset. It is a step towards trustworthy and robust predictive systems since it can detect and warn the user when predictions are inconsistent, based on the ontology. We believe that the capacity of warning the user when a prediction is inconsistent renders the system more transparent than a classic machine learning model which makes it more trustworthy. Likewise, we argue that the system is more robust than a single predictive model because it aggregates the results of concurrent statistical models using logic. This system requires minimal additional work to be implemented with regards to building a new ontology and statistical models for a new task. Indeed, any existing ontology can be utilized, only observable properties should be added and used in some class definitions. Likewise, any statistical model architecture can be used, allowing the usage of already trained models instead of training new ones. The system was evaluated with a musical instruments classification task. The system's capacity of detecting incorrect predictions is not yet satisfying, but it demonstrated the feasibility and should be seen as a proof of concept.

This paper presents this system as an image classifier, but we believe it could be applied to any kind of data. Only observable properties should change based on what information the data contains. We will implement this system to different data types in future work. We highlighted challenges in the system's design, such as choosing the adequate thresholds. In our experiment, we arbitrarily chose a single architecture for every model in our system. We would like to study and improve our system by investigating the impact of the thresholds, how to find their best values and how the system behaves with different model architectures.

We designed this system with explainability in mind. We believe that this architecture enables the generation of counterfactual explanations. Indeed, it is possible to modify the ontology or the thresholds to see how the system behaves. To fully verify this hypothesis and further explore the explainability of our architecture, we intend to create an explanation interface that is capable of interacting with the user to provide satisfying explanations.

References

1. Adadi, A., Berrada, M.: Peeking inside the black-box: A survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018)
2. Alvarez-Melis, D., Jaakkola, T.S.: Towards robust interpretability with self-explaining neural networks. Advances in neural information processing systems **31** (Jun 2018)
3. Arya, V., Bellamy, R.K., Chen, P.Y., Dhurandhar, A., Hind, M., Hoffman, S.C., Houde, S., Liao, Q.V., Luss, R., Mojsilović, A., et al.: One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques. arXiv:1909.03012 (2019)
4. Beaudouin, V., Bloch, I., Bounie, D., Clémençon, S., d'Alché Buc, F., Eagan, J., Maxwell, W., Mozharovskyi, P., Parekh, J.: Flexible and context-specific AI explainability: A multidisciplinary approach. SSRN Electronic Journal (2020)
5. Bellucci, M., Delestre, N., Malandain, N., Zanni-Merk, C.: Towards a terminology for a fully contextualized XAI. Procedia Computer Science **192**, 241–250 (2021)
6. Futia, G., Vetrò, A.: On the integration of knowledge graphs into deep learning models for a more comprehensible AI—three challenges for future research. Information **11**(2), 122 (feb 2020)
7. Geng, Y., Chen, J., Jimenez-Ruiz, E., Chen, H.: Human-centric transfer learning explanation via knowledge graph. arXiv:1901.08547 (Jan 2019)
8. Gunning, D., Aha, D.: DARPA's explainable artificial intelligence (XAI) program. AI Magazine **40**(2), 44–58 (jun 2019)
9. Halliwell, N., Gandon, F., Lecue, F.: User scored evaluation of non-unique explanations for relational graph convolutional network link prediction on knowledge graphs. In: Proc. of the 11th on Knowledge Capture Conf. ACM (dec 2021)
10. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) (June 2016)
11. Phan, N., Dou, D., Wang, H., Kil, D., Piniewski, B.: Ontology-based deep learning for human behavior prediction with explanations in health social networks. Information Sciences **384**, 298–313 (apr 2017)
12. Rožanec, J.M., Mladenić, D.: Semantic xai for contextualized demand forecasting explanations. arXiv:2104.00452 (Apr 2021)
13. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (may 2019)
14. Sachan, S., Yang, J.B., Xu, D.L., Benavides, D.E., Li, Y.: An explainable AI decision-support-system to automate loan underwriting. Expert Systems with Applications **144**, 113100 (apr 2020)
15. Seeliger, A., Pfaff, M., Krcmar, H.: Semantic web technologies for explainable machine learning models: A literature review. PROFILES/SEMEX@ ISWC **2465**, 1–16 (2019)
16. W3C: OWL 2 web ontology language document overview (Dec 2012), https://www.w3.org/TR/owl2-overview/, accessed on 15/03/2022
17. W3C: OWL 2 web ontology language manchester syntax (Dec 2012), https://www.w3.org/TR/owl2-manchester-syntax/, accessed on 15/03/2022
18. W3C: OWL 2 web ontology language structural specification and functional-style syntax (Dec 2012), https://www.w3.org/TR/owl2-syntax/, accessed on 15/03/2022