Towards Ontologically Explainable Classifiers

Grégory Bourguin, Arnaud Lewandowski, Mourad Bouneffa, Adeel Ahmad

Towards Ontologically Explainable Classifiers

G. Bourguin, A. Lewandowski, M. Bouneffa, and A. Ahmad

LISIC, Université du Littoral Côte d'Opale, 62228 Calais, FRANCE
gregory.bourguin@univ-littoral.fr

Abstract. In order to meet the explainability requirement of AI using Deep Learning (DL), this paper explores the contributions and feasibility of a process designed to create ontologically explainable classifiers while using domain ontologies. The approach is illustrated with the help of the Pizzas ontology that is used to create a synthetic image classifier that is able to provide visual explanations concerning a selection of ontological features. The approach is implemented by completing a DL model with ontological tensors that are generated from the ontology expressed in Description Logic.

Keywords: machine learning · ontology · explainability · classifier

1 Introduction

The last years have been characterized by a large democratization of solutions using Machine Learning (ML), and particularly Deep Learning (DL). This wide spread has been accompanied by questions regarding their trustworthiness. Many research papers have recently underlined the problem of the opacity of DL algorithms. This issue is at the heart of the XAI [1] initiative.

As stated in [8], *"clearly explaining a rationale for a classification decision to an end-user can be as important as the decision itself"*. It is thus necessary to create AI solutions that are able to provide explanations regarding their decisions, but moreover, these explanations also need to be understandable by the users, i.e. while using the adequate abstraction level. The users' abstraction level mainly depends on their knowledge, their expertise, or even their viewpoint.

Widely used in the Knowledge Management and Engineering research domains, the ontologies aim at reifying the knowledge of users involved in specific domains, thus allowing algorithms to use it. Taking note of the crucial need for explainable AI, the purpose of this paper is to explore a process for creating automatic classifiers that are able to provide explanations founded on an ontology. We do not focus here on new means for improving classification, but on the contributions of ontologies for explainable AI. We also explore the feasibility of such an approach while using the classical ML tools, and propose a solution that allows to complement a DL model with a graph of tensors that is automatically generated from description logic assertions coming from the targeted ontology.

The 2nd part of the paper proposes a state of the art concerning solutions for providing explainable AI. The 3rd part of the paper illustrates the benefits expected from a process involving ontological reasoning for explainability, and introduces the generic approach we propose. The 4th part presents its application through the implementation of an ontologically explainable image classifier. The 5th part presents our conclusions resulting from this experiment.

2 Explainability

The systems using ML are more and more efficient, but also more and more complex and opaque. They appear as black-boxes [6] making problematic for human to step in and understand their decisions, and to control their deployment, execution, and evolution [9, 1]. As a consequence, the need for transparency, and moreover, AI explainability has been revealed crucial.

2.1 Post-hoc model explanation

The tools for explaining decisions of DL systems mainly follow post-hoc approaches designed to provide explanations about pre-existing models. Most of theses methods are also called agnostic because they can be applied to any DL algorithm.

Most explanation techniques rely on the idea of associating each input feature with a value representing its importance for highlighting the key factors participating in the final prediction. It is thus possible to get explanations concerning a particular prediction, or more global ones represented with different graphics associating features, importance factors and predictions. One of the domains where this type of works is the best represented is Computer Vision (CV). In CV, the raw input features correspond to the image's pixels. The propositions consist in making a correspondence between the predictions and the pixels that leaded to a classification. Diverse approaches have been adopted and one of the most representative is the Grad-CAM [20] method (and derivatives) that uses the gradient of a targeted class to produce a *heatmap* highlighting the image's regions that most participated to its prediction. The techniques using input features to explain a model are not limited to CV. Tools like LIME [15] allow to identify some pixels in an image in a CV problem, but also to highlight the terms most participating in a prediction in a NLP (Natural Language Processing) model. In the same idea, while using different methods, we can also cite SHAP (SHapely Additive exPlanation) [11], or *What-If* [13].

All these techniques and tools have already proved to be really useful for explaining AI models. However, as it was underlined by [10], and as we will show in part 3.2, these approaches do not guarantee that the provided explanations are in fact understandable by the users.

2.2 Explainablity, Semantics and Ontologies

As recalled by [2]: *"Concept-based explanation approach is a popular model interpertability tool because it expresses the reasons for a model's predictions in terms* *of concepts that are meaningful for the domain experts"*. The benefits from ontologies for providing explanations have already been proven. For many authors, it is obvious that ontologies can help in providing adequate explanations about the decision process of DL models [3].

Following this idea, the concept of *semantic bottleneck* has recently been formalized in [10] and further developed in [12]: a classifier is built while integrating semantic layers in its very conception in order to extract *semantic features*. These latter are then used to compute the final classification. The pondered contribution of each semantic feature helps in providing explanations regarding a prediction, and in understanding misclassification errors. However, it should be noted that even if these interesting works speak about semantics, none of them use an ontological approach.

Research works like [4] are dedicated to image interpretations while using ontologies. A DL process (e.g. object detection) is used to extract features corresponding to ontological concepts. These features are then used to infer predictions at a higher abstraction level. Such approach implies reasoning while using description logic for enabling complex classification tasks. Even if they do not explicitly focus on the explainability issue, such reasoning is intrinsically explainable, and these solutions should indeed be able to provide explanations at the users' ontological abstraction level.

2.3 Positioning

As in Grad-CAM or LIME, our goal is to provide explanations while highlighting in the raw data the features that leaded to a classification. However, our approach is not agnostic: the explanations we want to provide are intimately bound to the targeted domain, and the features we want to highlight need to be at the user's abstraction level, i.e., from our viewpoint, ontological features.

Our approach is neither post-hoc because we need the ontology to be directly involved in the very process of the classifier creation. We follow a *semantic bottleneck* approach, with the difference that the semantics is here provided by an ontology which moreover is itself directly used to compute the predictions.

From this viewpoint, we are inspired by the research works implying ontologies for high abstraction level image interpretation. Our approach is however somehow different while definitely focusing on the explainability issue, and by advocating for explainable classifiers that involve ontologies in classification tasks that would *a priori* not need it.

3 Ontological Explainability Approach

3.1 Illustration Domain: Pizzas

To illustrate our approach, we choose to reuse the Pizzas ontology (Manchester University). The reasons are multiple, but the main one is the fact that this ontology is accessible and really famous, and thus already known by many researchers.

The Pizzas ontology defines a set of pizzas classes (e.g. Napoletana), that are subclasses of the NamedPizza class (being itself a subclass of Pizza). The pizzas definitions mainly involve the hasTopping object property whose domain is the Pizza class, and whose range is the PizzaTopping class which is the superclass of diverse toppings concepts like anchovies (AnchoviesTopping), etc.

We will then for example find the Napoletana pizza defined by:

$$\begin{aligned}\text{Napoletana} \equiv\ & \text{Pizza} \\ & \sqcap (\exists\ \text{hasTopping}\ .\ \text{AnchoviesTopping}) \\ & \sqcap (\exists\ \text{hasTopping}\ .\ \text{OliveTopping}) \\ & \sqcap (\forall\ \text{hasTopping}\ .\ (\text{AnchoviesTopping} \sqcup \text{OliveTopping}))\end{aligned}$$

As our aim is to link the results of an image classifier with the ontological definitions, we built a dataset in which the samples are labeled with the subclasses of NamedPizza. Other searchers have already created some pizzas images datasets [14]: to our knowledge, none of them corresponds to the definitions of the Pizzas ontology. Moreover, our goal in this experiment is not to enhance the classification performance (in terms of accuracy, etc.), but rather to study the benefits and the feasibility of an approach that implies an ontology for enhancing a classifier's explainability. Inspired by [14] in which the authors generate synthetic pizzas images to constitute a controlled dataset, we generated synthetic images of ontological pizzas by combining toppings cliparts (cf. Figure 1).

The Pizza ontology defines 22 subclasses of NamedPizza while using 36 subclasses of PizzaTopping. To simplify the construction of our dataset, we decided to focus on 14 subclasses of NamedPizza involving 16 subclasses of PizzaTopping, thus keeping diversity while removing the pizzas made of toppings like TobascoPepperSauce that are hard do illustrate with cliparts, and that are not essential to our demonstration.

The generated images voluntary use the same "pizza base" : only the toppings distribution is varying in number, position and orientation, in order to force any (non-ontological) classifier to focus on the toppings for classifying the pizzas. The classification task for these synthetic pizza images being relatively simple, we only generated a "small" and totally balanced dataset containing 200 pizzas per each of the 14 NamedPizza subclasses.

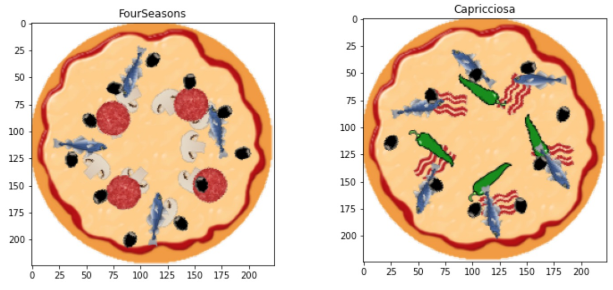

Fig. 1. Generating synthetic images of ontological pizzas.

3.2 Problems of a non-ontological approach

To illustrate our thoughts about the problems of post-hoc tools for explainability, we built and trained a “classical” image classifier based on a VGG19 [19] CNN architecture, we used transfer learning by reusing the weights of VGG19 pretrained on Imagenet, and complemented it by a Dense (256) and a SoftMax (14 pizzas classes) layers. The images data being simple, we were able to train this classifier while achieving 100% accuracy on a test set constituted by 20% of our samples.

We then used tools implementing the LIME [15] and Grad-CAM [18] methods which both explain a specific prediction by generating a heatmap highlighting the image’s pixels that most participated to the classification. As our images were generated in a way to let the toppings being the sole elements that can help in differentiating the pizzas classes, we can expect the heatmaps to focus on the toppings’ corresponding pixels.

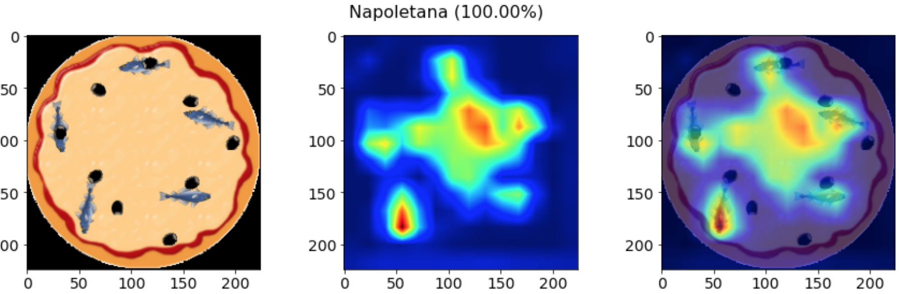

Fig. 2. Grad-CAM explanations for a specific pizza classification.

The Figure 2 shows the explanations provided by Grad-CAM for a pizza predicted as Napoletana, i.e. a pizza only containing olive and anchovies toppings (cf. definition in part 3.1). We only show Grad-CAM’s results here, but the explanations provided by LIME and Grad-CAM are similar. We can notice that the CNN focuses well on the anchovies. However, it ignores the olives while also focusing on a part of the pizza base (empty of toppings). We can thus consider that for this DL model, this pizza is a Napoletana because it has some anchovies and some void: this of course does not correspond to the definition we expected.

Nevertheless, the Pizzas ontology can help in explaining this phenomenon if it is used to generate an ontological correlation matrix regarding the toppings. This matrix reveals to what extent the toppings are correlated in the pizzas definitions: we can notice that the anchovies (AnchoviesTopping) always appear with olives (OliveTopping). On the other hand, the olives frequently appear with other toppings. As a result, for the CNN, on a Napolitana only made of anchovies and olives, the discriminant is the presence of anchovies.

The purpose of these remarks is not to discredit tools like Grad-CAM and LIME at all. As we just showed it, they are truly useful for explaining a classifier. However, these explanations can generally only be interpreted by an AI specialist and, as in works trying to associate some semantics to CNN filters [7], this example demonstrates that the abstraction level of the discriminants emerging from the training of a CNN does not coincide with the abstraction level of a pizza specialist. As a result, these tools do not seem to be the most adequate for providing explanations that are easily understandable by domain experts.

3.3 Proposed Approach

Our proposition aims at creating classifiers that are able to provide explanations at the domain experts' abstraction level, i.e. using the terms of their ontology. The steps for this realization are:

(a) Build a set C of the classes from the ontology that will be predicted as output of the classifier.

(b) Consider D: the set of definitions in the ontology such as
$D = \{d \mid \exists\ c \in C,\ d \equiv c$ is an axiom of the ontology$\}$.
Consider P: the set of properties in the ontology involved in D.
Consider R: the set of ranges such as $R = \{r \mid r=\text{range}(p),\ p \in P\}$
Build the set F of the ontological features $f \in F$, i.e. the triplets $(c,\ p,\ r)$ involved in D and that will be used while explaining the predictions.

(c) Implement a DL technique to build the set $FI \subseteq F$ of the identified ontological features (satisfied assertions) in a data sent to the classifier such as $FI = \{fi \in F \mid fi \equiv \exists\ p.r\ \}$

(d) Implement an ontological reasoning using D and FI for calculating $CI \subseteq C$, the set of the ci classes predicted for a data.

(e) Use the set $DI \subseteq D$ such as $DI = \{di \equiv ci\}$ and the set FI for explaining the CI classification.

One can note that the (b) and (c) steps are tightly linked because it would be useless to build F with ontological features that cannot be extracted from the data. In our example, we focus on the hasTopping object property because it defines the pizzas, but also because the presence of toppings (elements of R) can be deduced from the image.

We also want to underline that the abstraction level of the explanations is intrinsically linked to the abstraction level of the ontological features. Indeed, if in our example it will be possible to explain that an image represents a Napoletana because it contains anchovies and olives, the classifier will not be able to explain how it decided that an image region corresponds to a specific topping. We need to recall that any approach for explainability is facing the fact that, at some abstraction level, one considers not having to provide deeper explanations. We for example can cite [8] that proposes a birds species classifier while marrying a CNN and NLP for providing explanations: the system can explain that an image represents an Albatross because it contains a yellow beak, etc., but it does not try to demonstrate what is a yellow beak.

4 Ontological Classifier

This section presents the implementation of our approach with the pizzas example. This implementation is mainly constituted by 2 modules (Figure 3): a semantic segmentation (DL) module designed to extract the ontological features from an image, and an ontological reasoning module called OntoClassifier designed to compute the CI classes that can be deduced from FI, while being able to provide explanations. These 2 modules are implemented in Tensorflow 2.

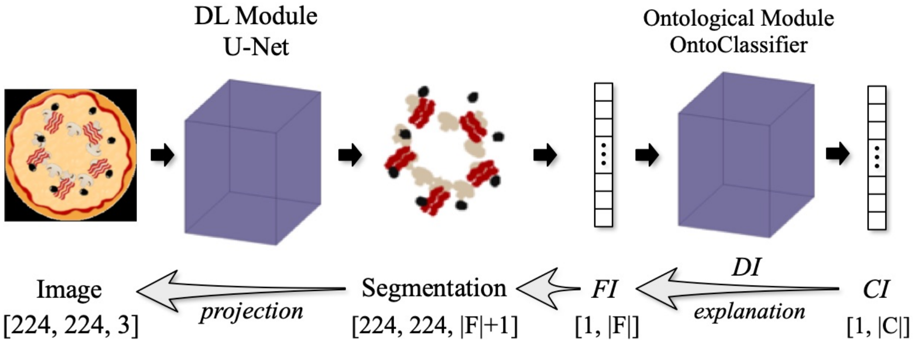

Fig. 3. Architecture of the ontologically explainable classifier.

4.1 DL Module: Semantic Segmentation

The first part of our classification pipeline aims at extracting the ontological features from an image, i.e. the satisfiability of the assertions corresponding to the triplets of F, considering that here:

$$F = \{(c \sqsubseteq \text{Pizza}, \text{hasTopping}, r \sqsubseteq \text{PizzaTopping})\}$$

We choose to use a semantic segmentation technique whose purpose is to label each pixel of an image with the classes from $R = \{r \sqsubseteq \text{PizzaTopping}\}$. We use a model architecture based on U-Net [16] and, as our dataset is fully controlled, we generate the segmentation masks (that are necessary to train the model) during the image generation process.

This U-Net implementation (based on MobileNetV2 [17] with the Imagenet weights) has for input 3 channels (RGB) pizzas images (224x224x3), and for output an image segmentation with 17 channels (224x224x17): each channel corresponds to one of the 16 PizzaTopping subclasses, excepted for 1 channel that is intended to receive the pixels not corresponding to any topping.

The central part of Figure 3 shows how an image sent to the DL module is segmented: to represent this segmentation here, we have overlaid the different channels while associating each of them with a different color.

4.2 Ontological Module: OntoClassifier

The presence of pixels in a segmentation layer can be interpreted as the presence of an ontological feature (∃ hasTopping . topping), topping ∈ R, thus letting to deduce the set FI of the satisfied assertions for each image treated by this model. It then remains to reason from FI while using the set of definitions D to deduce the set of classes CI that can label the image.

This reasoning process using properties extracted from an image is similar to those that can be found in diverse works merging DL and ontologies to propose interpretations at high abstraction level. A classical approach would be to populate the ontology with instances representing the samples to be classified (using their identified ontological features), and then to start an ontological reasoner like Jena, Hermit or Pellet to obtain a classification. However, as it was underlined in [5], this process is costly because it needs to complement the DL model with external tools, and these tools that are designed to reason in globality about an ontology are much slower that the DL pipelines nowadays used to create classifiers.

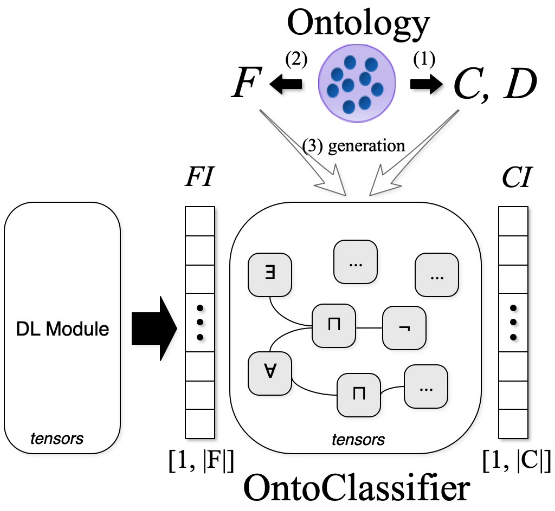

Fig. 4. Generating an OntoClassifier.

In our proposed approach, we do not need the full power of a classic ontological reasoner to deduce CI from FI and D. We thus created the OntoClassifier module whose constructor generates a set of tensors directly from the ontology, and in particular from C, D and F. This process is illustrated in Figure 4: after the selection of the targeted classes and definitions constituting C and D (1), and of the ontological features constituting F (2), a graph of tensors is automatically generated (3): these tensors are typed and interconnected thanks to the decomposition of the OWL (Web Ontology Language) definitions found in

D. The resulting OntoClassifier is then ready to complement the classification pipeline directly after the output of the DL module (right part of Figure 3). This assemblage is able to compute the satisfiability of ontological assertions like the definition of a Napoletana (cf. 3.1), or more complex assertions for example implying toppings' superclasses (using inheritance) like in:

$$\begin{aligned}\text{CheesyPizza} &\equiv \exists\ \text{hasTopping}\ .\ \text{CheeseTopping}\\ \text{VegetarianPizza} &\equiv\ \neg\ (\exists\ \text{hasTopping}\ .\ \text{FishTopping}) \sqcap\\ &\quad\ \neg\ (\exists\ \text{hasTopping}\ .\ \text{MeatTopping})\end{aligned}$$

Moreover, the OntoClassifier being implemented as a graph of tensors that represents the decomposition of the elements of D, this module allows to trace back the graph to identify the elements of FI that satisfied each assertion for a given data.

4.3 Results

Generating the OntoClassifier under the form of tensors allows the integration of the ontological dimension directly inside the classification pipeline. The resulting global model is then truly faster than in the case where the ontological reasoning is delegated to an external "classic" inference engine. For instance, using our semantic segmentation (DL) module combined with the Hermit reasoner on our computers (I9-10850K 3.6 GHz, 32 Go DDR4 3200MHz, GPU RTX 3080), the classification of 100 pizzas images takes on average 130s. With the OntoClassifier, on the same computers, this classification only takes on average 1,6s. As in 3.2, we were able to train this pipeline to achieve 100% accuracy on the test set.

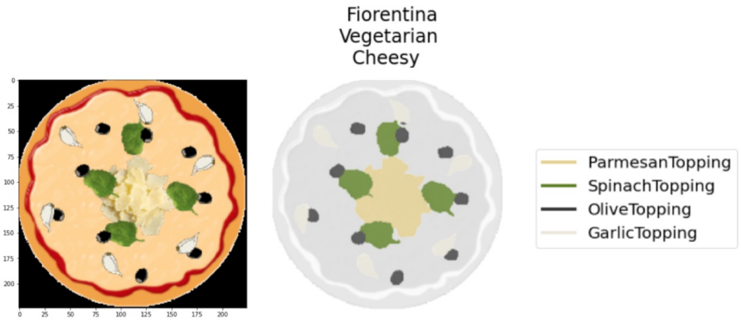

Fig. 5. Classification and ontological segmentation.

The Figure 5 shows the example of an image of Fiorentina. The classifier predicts the set of classes that can label this image (Fiorentina, Vegetarian, Cheesy). The segmentation mask highlights the set of identified toppings. One can notice that in contrast to the example presented in 3.2, the abstraction level of this heatmap is in line with the ontology: no topping is ignored by the classifier, the highlighted entities correspond to the class definitions, and each topping is differentiated and identifiable thanks to a color code.

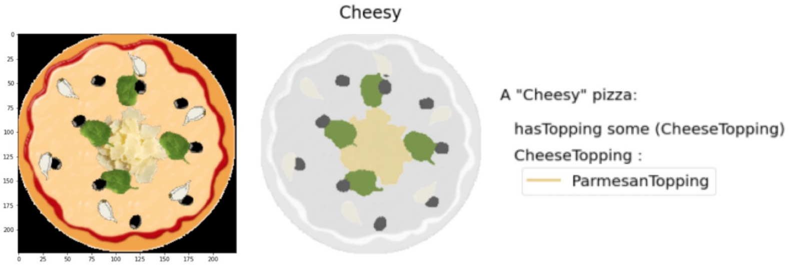

Fig. 6. Classification and visual ontological explanations.

The Figure 6 illustrates the fact that this system also allows to focus on a specific identified class (here focusing on the Cheesy class identified for the image classified in Figure 5), and to use the OntoClassifier's introspection mechanism to explain this classification. This focus reuses the ontological segmentation mask, and adds (in the right part), an explanation that binds the OWL definition of the targeted class with the parts of the image that satisfied the (sub-)assertions. Here, the binding is represented using a heatmap color code, but we can imagine more interactive means like dynamically highlighting the original image while moving the cursor over parts of the definition.

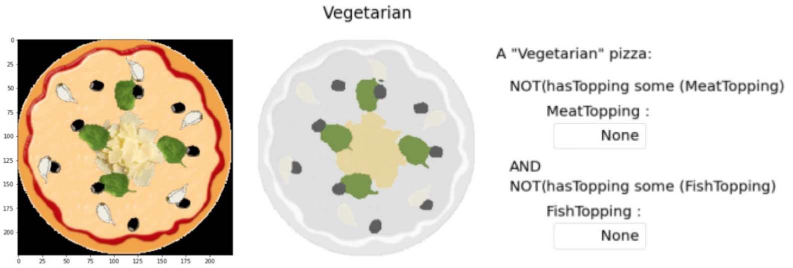

Fig. 7. Explanations for classification due to missing features.

The Figure 7 shows that the OntoClassifier can focus on an identified class and also explain a classification due to the absence of elements.

5 Conclusion

Accounting the need for explainable AI, we explored the benefits and the feasibility of a process for creating classifiers that are ontologically explainable, i.e. providing explanations at the abstraction level for their domain users. We proposed a generic approach that results in an architecture mainly constituted by 2 modules: a DL module dedicated to the ontological features extraction, and another module named OntoClassifier and dedicated to ontological reasoning. In order to integrate the ontological dimension in the very heart of the classifier without burdening the resulting classification pipeline, we have introduced a tool for automatically generating the OntoClassifier as a graph of tensors directly built from the class definitions provided in the ontology. We have exemplified our approach by creating an image classifier and have illustrated the possibilities for ontological classification, as well as for visual explanation.

It is true that this approach involves additional work around the creation of an ontology that reifies the users' abstraction level, the construction of the sets C, D, and F, and also the use of a DL technique that is more complex than for a "simple" classification. One can however also notice that this approach not only results in an ontologically explainable classifier, but also presents other benefits. As long as the set of ontological features (F) does not change, it is easily possible to evolve the classifier, for example by adding new classes/definitions, and to integrate this evolution into the pipeline without having to retrain the DL model. The sole thing to do is to automatically re-generate the OntoClassifier. As underlined in [21], the notion of viewpoint is important too, even while considering ontologies. Our proposed approach and tools can offer a solution to the need for multi-viewpoints because it is easily possible to generate different OntoClassifier modules, each one dedicated to a specific viewpoint on the same domain. Our approach also allows to introduce the notion of viewpoints in the explanations themselves. Indeed, for the same C set, it is possible to build different D sets, and then to generate different explainable classifiers focusing on different ontological definitions for the same classes. For example, in the Pizza ontology, a Vegetarian pizza can be defined as:

$$\begin{aligned}
&(1)\quad \text{VegetarianPizza} \equiv \neg\,(\exists\ \text{hasTopping}\ .\ \text{FishTopping}) \sqcap \\
&\qquad\qquad\qquad\qquad\quad\ \neg\,(\exists\ \text{hasTopping}\ .\ \text{MeatTopping}) \\
&(2)\quad \text{VegetarianPizza} \equiv \forall\ \text{hasTopping}\ .\ \text{VegetarianTopping}
\end{aligned}$$

If these 2 definitions lead to the same classification, the associated visual explanations correspond to different viewpoints focusing on (or highlighting) (1) the fish or meat toppings (cf. Figure 7), or (2) the vegetarian toppings.

These approach and tools are still in development. In this paper, the visual explanations directly use the OWL expressions: this representation needs to be improved, and then evaluated through experiments while involving end-users. We also are working on the human-machine interfaces that will allow end-users to manipulate and explore the provided explanations. The presented elements however already let imagine functionalities that are promising, and even necessary in the frame of projects involving actors with different cultures and viewpoints.

References

1. Arrieta, A., Díaz-Rodríguez, N., Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F.: Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI. ArXiv **abs/1910.10045** (2020)
2. Bahadori, M.T., Heckerman, D.: Debiasing concept bottleneck models with a causal analysis technique (2020)

3. Confalonieri, R., Besold, T.R.: Trepan reloaded: A knowledge-driven approach to explaining black-box models. In: ECAI (2020)
4. Conigliaro, D., Ferrario, R., Hudelot, C., Porello, D.: Integrating computer vision algorithms and ontologies for spectator crowd behavior analysis. In: Group and Crowd Behavior for Computer Vision (2017)
5. Ding, Z., Yao, L., Liu, B., Wu, J.: Review of the application of ontology in the field of image object recognition. In: ICCMS 2019 (2019)
6. Dosilovic, F.K., Brčič, M., Hlupic, N.: Explainable artificial intelligence: A survey. 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) pp. 0210–0215 (2018)
7. Gonzalez-Garcia, A., Modolo, D., Ferrari, V.: Do semantic parts emerge in convolutional neural networks? International Journal of Computer Vision **126**, 476–494 (2017)
8. Hendricks, L.A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., Darrell, T.: Generating visual explanations. In: ECCV (2016)
9. Lipton, Z.C.: The mythos of model interpretability. Queue **16**, 31 – 57 (2018)
10. Losch, M., Fritz, M., Schiele, B.: Interpretability beyond classification output: Semantic bottleneck networks. ArXiv **abs/1907.10882** (2019)
11. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: NIPS (2017)
12. Marcos, D., Lobry, S., Tuia, D.: Semantically interpretable activation maps: what-where-how explanations within cnns. 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW) pp. 4207–4215 (2019)
13. Martens, D., Provost, F.: Explaining data-driven document classifications. MIS Q. **38**, 73–99 (2014)
14. Papadopoulos, D.P., Tamaazousti, Y., Ofli, F., Weber, I., Torralba, A.: How to make a pizza: Learning a compositional layer-based gan model. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 7994–8003 (2019)
15. Ribeiro, M.T., Singh, S., Guestrin, C.: "why should i trust you?": Explaining the predictions of any classifier. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2016)
16. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. ArXiv **abs/1505.04597** (2015)
17. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 4510–4520 (2018)
18. Selvaraju, R.R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. International Journal of Computer Vision **128**, 336–359 (2019)
19. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR **abs/1409.1556** (2015)
20. Zhang, Q., Wu, Y., Zhu, S.: Interpretable convolutional neural networks. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 8827–8836 (2018)
21. Zhitomirsky-Geffet, M., Erez, E.S., Bar-Ilan, J.: Toward multiviewpoint ontology construction by collaboration of non-experts and crowdsourcing: The case of the effect of diet on health. Journal of the Association for Information Science and Technology **68** (2017)