RevelioNN: Retrospective Extraction of Visual and Logical Insights for Ontology-based Interpretation of Neural Networks

Anton Agafonov, Andrew Ponomarev
St. Petersburg Federal Research Center of the Russian Academy of Sciences (SPC RAS)
St. Petersburg, Russian Federation
agafonov.a@spcras.ru, ponomarev@iias.spb.su

***Abstract*—The need for AI explainability, which involves helping humans understand why an AI algorithm arrived at a particular decision, is crucial in numerous critical applications. Although deep neural networks play a significant role in modern AI, they inherently lack transparency. Consequently, various approaches have been suggested to clarify their decision-making processes to human users. One promising category of such approaches involves ontology-based methods. These methods have the potential to generate explanations using concepts from an ontology that are familiar to domain experts and the logical connections between these concepts. Specifically, post-hoc ontology-based explanations typically rely on concept extraction, which establishes a link between the internal representations formed by the neural network's inner layers and the domain concepts outlined in the ontology.**

This paper introduces the RevelioNN library, which comprises post-hoc algorithms designed to explain predictions made by deep convolutional neural networks in binary classification tasks, with a focus on leveraging ontologies. The library incorporates cutting-edge concept extraction techniques centered around constructing mapping networks. Furthermore, it provides the capability to form both logical and visual explanations for the predictions of convolutional neural networks by utilizing ontology concepts derived from their internal representations. An essential benefit of this library is its adaptability to interpret predictions from any pre-trained convolutional network implemented using the PyTorch framework.

The code is licensed under the BSD-3-Clause license and is freely available at github.com/cais-lab/revelionn.

I. Introduction

Modern artificial neural networks have demonstrated impressive predictive abilities, occasionally outperforming human-level accuracy in specific tasks. Nevertheless, because of their restricted interpretability, their usefulness in vital fields is constrained, and their outcomes may not consistently instill trust. Consequently, there is an urgent requirement for explainable AI (XAI) that can elucidate the inner workings of neural networks.

Many methods have been suggested to elucidate neural networks, offering insights into the decision-making processes of these networks [1]. However, these approaches are predominantly designed for AI specialists and may not be as easily comprehensible or applicable to domain experts who seek explanations closely aligned with the terminology of their specific problem domain.

One way to enhance the comprehensibility of explanations is by linking them to explicit domain knowledge, represented in the form of a domain ontology. In fact, research has demonstrated that employing such conceptualizations can indeed augment the understandability of explanations [2].

Following the typical structure of XAI research, ontology-based neural explanation methods can be categorized into two main groups: post-hoc methods [2]–[5], where ontologies are employed to elucidate the outcomes of pre-existing black-box neural networks, and self-explainable methods [6]–[8], where ontologies guide the design and/or training of a neural network to create a network whose outputs are easily explainable. In this paper, we focus on post-hoc explanations. The advantage of such approaches is their potential applicability to any neural network, including those built on contemporary high-performance architectures.

It has been demonstrated (e.g., [9]–[11]) that the internal representations within the inner layers of a deep neural network can frequently be aligned with ontology concepts. This alignment process, known as "concept extraction" [9], involves training a mapping neural network. This network takes the output of specific neurons from the "main" network and produces the probability that the sample being processed by the "main" network (triggering a specific activation) is pertinent to the specified concept. In many instances, these mapping networks can achieve a very high level of prediction accuracy, enabling reliable extraction of the set of concepts from a sample. With knowledge of this set of relevant concepts, one can generate an explanation through logical inference, utilizing the definition of the target concept and its associated concepts.

Currently, there exist several algorithms for concept extraction. Search-based algorithms [9], [12] employ various strategies to efficiently locate the neurons (network layers) that best express a particular concept. They extract concepts one by one, i.e. mapping networks are trained for each ontology concept independently. In addition, the process of extracting a concept is complicated by the fact that not every concept can be extracted from a given set of neurons [10], [11]. Thus, the process of searching for the most informative neurons can take a lot of time, especially if the number of concepts of ontology is high.

In [13], an algorithm is introduced that accomplishes the

simultaneous extraction of all ontology concepts using a single neural network. This approach eliminates the need for searching for the most informative sets of neurons. Instead, it leverages all activations from the "main" network as input data for the proposed mapping network. The novel architecture not only enables the efficient extraction of all concepts but also possesses the capacity to incorporate prior knowledge. This is made possible by its multiple outputs, each corresponding to a specific concept, facilitating the utilization of logical relationships between these concepts.

In this paper, we present the RevelioNN library of post-hoc algorithms for explaining predictions of deep convolutional neural networks of binary classification using ontologies. It embodies the latest proposed concept extraction algorithms based on the construction of mapping networks. In addition, the library implements the functionality of forming logical and visual explanations of predictions of a convolutional neural network using ontology concepts extracted from its internal representations. An important advantage of the library is that it can be used to interpret the predictions of any pre-trained convolutional network implemented in the PyTorch framework.

The rest of the paper is structured as following. Section II contains problem definition. Section III describes concept extraction approaches and examples of working with them. Section IV describes the principles and examples of the formation of ontology-based explanations.

II. Problem Definition

We make the assumption that there exists a deep neural network denoted as M, an ontology represented by $\mathcal{O}$, and a labeled dataset identified as (X, Y).

The network denoted as M is trained for the binary classification of objects within a specific domain $\mathcal{D}$ (i.e., $x \sim \mathcal{D}$). Typically, such networks produce a probability of an object belonging to a particular class T, denoted as $M(x) \in [0; 1]$. We will also refer to network M as the "main" network, and our objective is to investigate its internal representations to align them with ontology concepts ($M_i(x)$ signifies the output of the i-th layer of M). We consider convolutional neural networks designed for image processing, with $\mathcal{D}$ consisting of images. The network should be constructed as a sequence of layers (e.g., 2D convolutions, activation, batch normalization, pooling, etc.), and it may adhere to popular architectures for object recognition such as ResNet, MobileNet, EfficientNet, among others. We assume that this "main" network has been trained by a third party, and we only require access to the output of the selected network layer produced during the forward pass.

The domain ontology $\mathcal{O}$ outlines several concepts that pertain to the samples within $\mathcal{D}$, including the target class T of the "main" network. Within this ontology, the concept T is defined in terms of other ontology concepts. As stated in [9], [12], we refer to these concepts as being "relevant to T" or simply "relevant," and denote the set of such concepts as $R(T)$. We assume that it is possible to reliably deduce whether a sample belongs to the concept T by discerning its membership in each of the concepts within $R(T)$. Informally, this means that the ontology furnishes a conceptually (terminologically) robust foundation, allowing for symbolic resolution of the classification problem.

Finally, the labeled dataset (X, Y) establishes a connection between samples from the domain $\mathcal{D}$ and the pertinent ontology concepts (i.e., concepts in $R(T)$). Input samples X are drawn from the distribution $\mathcal{D}$ ($X = \{X_i\}, X_i \sim \mathcal{D}$), and they are annotated with each of the relevant ontology concepts as follows: $Y_{i,j} = c_j(X_i), c_j \in R(T)$. It's important to note that the training of the "main" network only requires samples labeled with the target class T, while we assume that the alignment dataset is labeled with all the relevant concepts. Although it may be challenging to annotate samples with all the relevant classes, the size of such a dataset can be relatively small (much smaller than the number of images needed to train the "main" network) [9].

The problem of concept extraction involves finding a model denoted as $E(\{M_i(x)\})$, which, based on the output of the inner layers of M for a sample x drawn from $\mathcal{D}$, produces probabilities of sample x belonging to each of the concepts within $R(T)$, i.e., $E(\{M_i(x)\}) \in [0; 1]^{|R(T)|}$. Since we have assumed that the target class T can be reliably deduced from $R(T)$, these probabilities can be used to infer the relevance of sample x to class T. This inference will also serve as an explanation.

III. Concept Extraction Approaches

Concept extraction algorithms are based on the construction of mapping networks linking the internal representations of a convolutional neural network with ontology concepts. Let's consider the aspects of the implementation of concept extraction using the example of the proposed library.

A. Main Networks

As noted earlier, the "main" network is a convolutional neural network of binary classification, at the output of which there is a sigmoid activation function. This sigmoid activation function produces the probability of the target class (concept).

Convolutional network class must be described in a separate file in which the following variables must also be declared:

- variable storing the number of channels of the image fed to the network;
- variable storing the size of the image fed to the network;
- the `torchvision.transforms` module object, which represents a transformation over images.

The network class must be inherited from the `torch.nn.Module` class, that is, the network must be implemented using PyTorch. An example of a description of the "main" network is given in Listing 1.

RevelioNN can interpret convolutional binary classification networks that have already been trained without using this library. To do this, the specified model must be converted to RevelioNN format. To convert to the RevelioNN format, the

model file should contain only `state_dict`, which is typical for most cases.

Listing 1. Description of the main network

```
import torch
from torch import nn
from torchvision import transforms


def ResNet18():
    model = torch.hub.load(
        'pytorch/vision:v0.10.0',
        'resnet18', pretrained=True)

    for param in model.parameters():
        param.requires_grad = True

    model.fc = nn.Sequential(nn.Linear(512,1),
                             nn.Sigmoid())
    return model


NUM_CHANNELS = 3
IMG_SIDE_SIZE = 224
transformation = transforms.Compose([
    transforms.Resize(size=(IMG_SIDE_SIZE,
                            IMG_SIDE_SIZE)),
    transforms.ToTensor(),
    transforms.Normalize(
        [0.485, 0.456, 0.406],
        [0.229, 0.224, 0.225]
    )
])
```

B. Mapping Networks

The library provides two types of mapping networks. Both networks receive the values of activations produced by the "main" network during the inference as input.

1) Single Mapping Network: It is a full-connected neural network, the number of input neurons of which is determined by the number of activations of neurons of the specified convolutional network layers. It has a ReLU activation function in its hidden layers and a sigmoid in its output.

It is reasonable to use it to extract only one concept from one or more given convolutional network layers, since when using a larger set of layers, the number of parameters of this network will increase dramatically.

The user can vary the number of layers and the number of neurons in each layer of this mapping network (see Listing 2).

Listing 2. Initialization of the single mapping network

```
from revelionn.mapping_nets.single_mapping_net \
    import SingleMappingNet

single_net = SingleMappingNet(
    in_features=in_features,
    num_neurons_list=[10, 5, 1]
)
```

2) Simultaneous Mapping Network: It was proposed in [13] and is a mapping network that allows you to extract the entire set of relevant concepts simultaneously, receiving activations of all specified layers of the convolutional network at once. This mapping network also shows good results in semi-supervised learning using semantic loss [14], which strengthens the relationship between concepts.

The architecture of the simultaneous mapping network is shown in Fig. 1. Outputs of each layer of the "main" network are fed into separate decoder blocks. These decoder blocks take into account the structure of the 2D image and significantly reduce the tendency of the network to overfit.

Two types of decoder blocks are considered:

- for 2D spatial feature maps (the outputs of convolutional layers), the decoder blocks consist of 1×1 convolutions followed by global average pooling. The number of output channels for all 1×1 convolutions in the decoder blocks is kept the same.
- for 1D feature maps (the outputs of fully-connected layers), the decoder blocks consist of fully-connected layer.

The outputs of the decoder blocks are concatenated and forwarded as input to an internal representation block, which is realized as a series of fully-connected layers.

Subsequently, there exists a distinct output block for each concept, responsible for extracting the probability associated with that specific concept from the shared internal representation. These concept blocks are constructed with a few fully-connected layers, followed by a sigmoid activation function.

The values of all the listed parameters of the simultaneous mapping network are determined by the user (see Listing 3).

Listing 3. Initialization of the simultaneous mapping network

```
from revelionn.mapping_nets. \
    simultaneous_mapping_net \
    import SimultaneousMappingNet

simultaneous_net = SimultaneousMappingNet(
    activation_extractor=act_extractor,
    decoder_channels=20,
    num_shared_neurons=[160, 80, 40, 20],
    num_output_neurons=[20, 1],
    num_outs=num_relevant_concepts
)
```

C. Concept Extraction Algorithms

RevelioNN provides the following set of concept extraction algorithms:

1) Exhaustive search: In this approach, single mapping networks are trained for each layer of the "main" net and for each relevant concept (resulting in $L \times |RELEVANT(T)|$ networks, where L is the number of layers of the "main" net). The advantages of this method include the fact that as a result we will have complete information about the localization of each of the concepts. However, it is very time-consuming and computationally intensive.

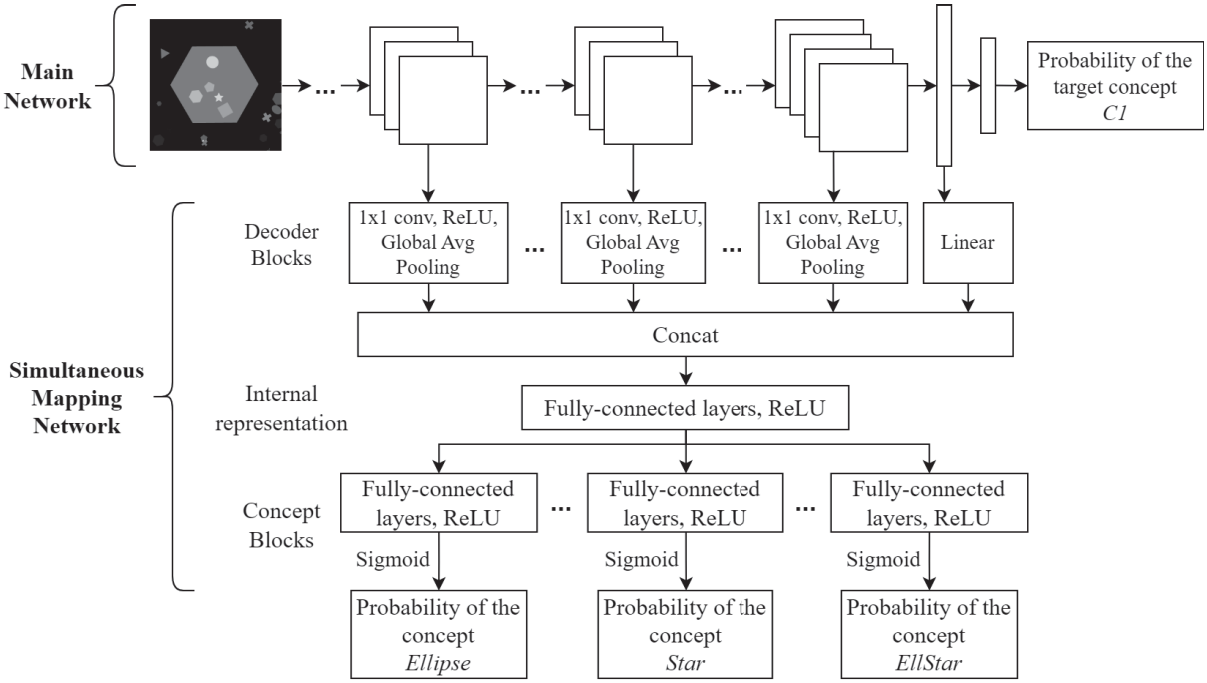

Fig. 1. Architecture of simultaneous mapping network on the example of the SCDB dataset

2) Heuristic search (proposed in [12]): Here, as in the previous case, single mapping networks are used, however, due to heuristic pruning such mapping networks are trained not for every layer-concept combination. The disadvantages include the fact that, due to the heuristics used, the most informative set of neurons (layer) cannot always be found.

3) Simultaneous extraction (proposed in [13]): Uses a simultaneous mapping network, a detailed description of which was given earlier.

Examples of using algorithms for extracting concepts are given Listing 4. The `ConceptExtractor` class provides these algorithms. To initialize it, you must first define an instance of `MappingTrainer` (this is an interface class that facilitates interaction with the library) and an ontology graph (`networkx.DiGraph`).

It is worth noting that if you need to train one particular network to extract the concepts of a certain layer (or layers), then for simplicity, you can use methods of the `MappingTrainer` class such as `train_single_model()`, `train_simultaneous_model()`, etc.

Listing 4. Concept extraction

```
from revelionn.mapping_trainer import
    MappingTrainer
from revelionn.concept_extraction import
    ConceptExtractor

extractor = ConceptExtractor(trainer,
                             onto_graph)

best_layer, best_auc = extractor.\
    exhaustive_search(relevant_concept,
                      layer_names,
                      number_neurons)

best_layers = extractor.heuristic_search(
    target_concept, top_layer_num,
    patience, number_neurons)

concepts_auc, all_auc = extractor.\
    simultaneous_extraction(target_concept,
                            decoder_channels,
                            num_shared_neurons,
                            num_output_neurons)
```

IV. Ontology-based Explanations

After the necessary relevant concepts have been extracted (the corresponding mapping networks have been trained), it becomes possible to form ontology-oriented explanations. First you need to load a model of a trained mapping network, as shown in Listing 5.

A. Logical Explanations

To form logical explanations through ontological inference the ontology must be represented in the OWL 2 language. Also, in the ontology, it is necessary to define an individual with the name `__input__`, which will be used in the future to form ontological explanations. In addition, it is necessary to describe the so-called concepts map, which is a dictionary whose keys are the names of the attributes of the dataset, and the values are the corresponding concepts of the ontology.

One must first extract the concepts relevant to the target concept from the image, and then transfer the extracted concepts and their probabilities to the reasoning module along with the ontology (see Listing 6). The formation of logi-

Listing 5. Loading the mapping model

```
from revelionn.utils.model import
    load_mapping_model

main_module, mapping_module, act_extractor,
transformation, img_size = load_mapping_model(
    mapping_model_filepath,
    main_models_directory,
    main_net_modules_directory, device)
```

cal explanations is carried out using the reasoning module BUNDLE [15]. BUNDLE provides reasoning for probabilistic ontologies through the use of various reasoners. It uses the DISPONTE approach, where each axiom is assigned a probability value indicating the likelihood that a certain axiom will be accepted as true.

Listing 6. Formation of logical explanations

```
from revelionn.utils.explanation import \
    extract_concepts_from_img, \
    explain_target_concept
from PIL import Image

image = Image.open(image_path)
main_concepts, extracted_concepts, \
    mapping_probabilities = \
    extract_concepts_from_img(main_module,
                              mapping_module,
                              image,
                              transformation)

justifications = explain_target_concept(
    extracted_concepts, mapping_probabilities,
    concepts_map, target_concept,
    path_to_owl_ontology, path_to_temp_files)

print(justifications)
```

By extracting relevant concepts, it is possible to form logical explanations about the belonging of the sample to the target concept, accompanied by a set of axioms of ontology. The input image presented in Fig. 1 was taken from the SCDB dataset [16], with which the "main" network and mapping networks were trained. A fragment of the ontology developed for the SCDB dataset is shown in Fig. 2. This image belongs to class $C1$. An example of a logical explanation by ontological inference for this sample is given in Fig. 3.

Each of the extracted concepts corresponds to a certain probability, which is then used to calculate the degree of confidence of the justifications. The list of possible justifications is ranked by the degree of trust. If any concept has not been extracted, then we can say that the opposite concept has been extracted, the name of which is automatically formed by adding the prefix "Not".

In Fig. 3 shows one of the explanations from the list of possible explanations. It can be interpreted as follows. The concepts of $Star$ and $Ellipse$ were extracted from the image. Therefore, based on the axiom of ontology that the conjunction of the concepts $Star$ and $Ellipse$ is a subclass of $EllStar$, we can conclude that the image also represents $EllStar$. And according to another axiom, the $C1$ target concept is equivalent to $EllStar$. Thus, the prediction of the neural network was confirmed by ontological reasoning.

B. Visual Explanations

Visual explanations mean highlighting positively extracted concepts in the image. Currently, visual explanations are formed using the occlusion method [17]. Its essence lies in the fact that the input image is systematically overlapped by a square of a given size with a given step. At each step, the overlapped image is run through the "main" network, and its activations are run through the mapping network. Thus, by obtaining output probabilities at each step, a saliency map can be formed. Visual explanations can be formed as shown in Listing 7.

Listing 7. Formation of visual explanations

```
import matplotlib.pyplot as plt
from revelionn.occlusion import \
    perform_occlusion

perform_occlusion(main_module, mapping_module,
                  activation_extractor,
                  transformation, img_size,
                  image_path, window_size=20,
                  stride=5, threads=0)
plt.show()
```

An example of visual explanations for our image is presented in Fig. 4. It can be seen that the extracted concepts (shapes) were correctly identified on the corresponding fragments of the image.

The library also shows good results when working with real datasets, for example, on a pizza dataset [18]. We have trained the "main" network to classify images of pizzas of the $MeatOrVegetables$ class. Pizza belongs to this class if it contains $Meat$ ($Bacon$ or $Pepperoni$) or $Vegetables$ ($Tomatoes$ and $Onions$). In Fig. 5, it can be seen that visual explanations are still being formed reliably and the concept $Pepperoni$, and therefore $Meat$, has been successfully extracted.

V. Conclusion

In this paper, we introduced RevelioNN, library designed to address the challenge of explaining predictions made by deep convolutional neural networks in binary classification tasks while leveraging ontologies. This library incorporates cutting-edge concept extraction algorithms, particularly those centered around the creation of mapping networks. Furthermore, it offers the capability to generate both logical and visual explanations for the predictions of convolutional neural networks by utilizing ontology concepts extracted from their internal representations. Notably, the library's versatility shines through, as it can seamlessly interpret predictions from any pre-trained convolutional network built within the PyTorch framework.

$$
\begin{aligned}
\text{C1} &\equiv \text{EllStar} \sqcup \text{HexStar} \sqcup \text{TEStarmarker} \\
\text{EllStar} &\sqsupseteq \exists \text{has.Ellipse} \sqcap \exists \text{has.Star} \\
\text{HexStar} &\sqsupseteq \exists \text{has.Hexagon} \sqcap \exists \text{has.Star} \\
\text{TEStarmarker} &\sqsupseteq \exists \text{has.Triangle} \sqcap \exists \text{has.Ellipse} \sqcap \exists \text{has.Starmarker}
\end{aligned}
$$

Fig. 2. Fragment of the SCDB dataset ontology.

```
The image is classified as ['C1'].

The following concepts were extracted from the image:
['HexStar', 'EllStar', 'NotTEStarmarker', 'Hexagon', 'Star', 'Ellipse', 'NotTriangle', 'NotStarmarker']
with the following probabilities:
[0.99938893, 0.99976605, 0.9937676684930921, 0.99947304, 0.9999995, 0.99962604, 0.9861229043453932, 0.9810010809451342]

Justification for '__input__ Type C1': (Degree of Belief: 0.99963)
       __input__ Type has some Star    ("0.9999995")
       __input__ Type has some Ellipse ("0.99962604")
       (has some Ellipse) and (has some Star) SubClassOf EllStar
       C1 EquivalentTo EllStar or HexStar or TEStarmarker
```

Fig. 3. Example of a logical explanation

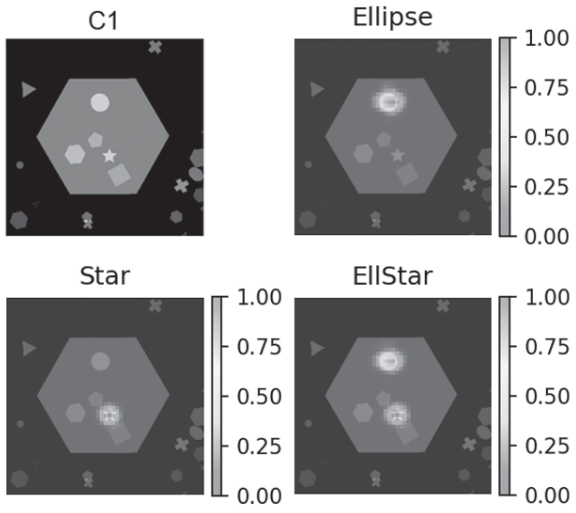

Fig. 4. Example of a visual explanation (SCDB dataset)

ACKNOWLEDGMENT

The research is funded by the Russian Science Foundation (project 22-11-00214).

REFERENCES

[1] N. Burkart and M. F. Huber, "A survey on the explainability of supervised machine learning," *Journal of Artificial Intelligence Research*, vol. 70, pp. 245–317, 2021.

[2] R. Confalonieri, T. Weyde, T. R. Besold, and F. Moscoso Del Prado Martín, "Trepan reloaded: A knowledge-driven approach to explaining black-box models," *Frontiers in Artificial Intelligence and Applications*, vol. 325, pp. 2457–2464, 2020.

[3] R. Confalonieri, T. Weyde, T. R. Besold, and F. M. d. P. Martín, "An Ontology-based Approach to Explaining Artificial Neural Networks," 2019. [Online]. Available: http://arxiv.org/abs/1906.08362

[4] R. Confalonieri, T. Weyde, T. R. Besold, and F. Moscoso del Prado Martín, "Using ontologies to enhance human understandability of global post-hoc explanations of black-box models," *Artificial Intelligence*, vol. 296, p. 103471, 2021. [Online]. Available: https://doi.org/10.1016/j.artint.2021.103471

[5] C. Panigutti, A. Perotti, and D. Pedreschi, "Doctor XAI: An ontology-based approach to black-box sequential data classification explanations," in *FAT* 2020 - Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, 2020, pp. 629–639.

[6] G. Bourguin, A. Lewandowski, M. Bouneffa, A. Ahmad, and T. Ontologically, "Towards Ontologically Explainable Classifiers," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2021, vol. LNCS 12892, pp. 472–484. [Online]. Available: https://link.springer.com/10.1007/978-3-030-86340-1{_}38

[7] Z. A. Daniels, L. D. Frank, C. Menart, M. Raymer, and P. Hitzler, "A framework for explainable deep neural models using external knowledge graphs," in *Proc. SPIE 11413, Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II*, T. Pham, L. Solomon, and K. Rainey, Eds. SPIE, apr 2020, p. 73. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/11413/2558083/A-framework-for-explainable-deep-neural-models-using-external-knowledge/ 10.1117/12.2558083.full

[8] V. Bourgeais, F. Zehraoui, M. Ben Hamdoune, and B. Hanczar, "Deep GONet: self-explainable deep neural network based on Gene Ontology for phenotype prediction from gene expression data," *BMC Bioinformatics*, vol. 22, pp. 1–24, 2021. [Online]. Available: https://doi.org/10.1186/s12859-021-04370-7

[9] M. de Sousa Ribeiro and J. Leite, "Aligning Artificial Neural Networks and Ontologies towards Explainable AI," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 6, 2021, pp. 4932–4940. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/16626

[10] A. Agafonov and A. Ponomarev, "Localization of Ontology Concepts in Deep Convolutional Neural Networks," in *2022 IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*. IEEE, nov 2022, pp. 160–165. [Online]. Available:

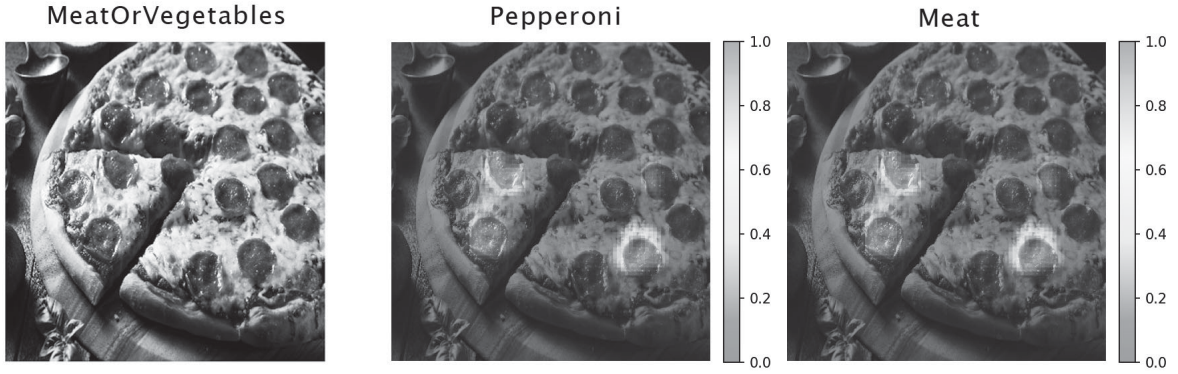

Fig. 5. Example of a visual explanation (Real pizza dataset)

https://ieeexplore.ieee.org/document/10016932/

[11] ——, "An Experiment on Localization of Ontology Concepts in Deep Convolutional Neural Networks," in *The 11th International Symposium on Information and Communication Technology*. New York, NY, USA: ACM, dec 2022, pp. 82–87. [Online]. Available: https://dl.acm.org/doi/10.1145/3568562.3568602

[12] A. Ponomarev and A. Agafonov, "Ontology Concept Extraction Algorithm for Deep Neural Networks," in *2022 32nd Conference of Open Innovations Association (FRUCT)*. IEEE, nov 2022, pp. 221–226. [Online]. Available: https://ieeexplore.ieee.org/document/9953838/

[13] ——, "Ontology-based post-hoc neural network explanations via simultaneous concept extraction," in *Intelligent Systems and Applications. Proceedings of the 2023 Intelligent Systems Conference (IntelliSys)*. Springer International Publishing.

[14] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Van Den Broeck, "A semantic loss function for deep learning with symbolic knowledge," *35th International Conference on Machine Learning, ICML 2018*, vol. 12, pp. 8752–8760, nov 2018. [Online]. Available: http://arxiv.org/abs/1711.11157

[15] F. Riguzzi, E. Bellodi, E. Lamma, and R. Zese, "BUNDLE: A reasoner for probabilistic ontologies," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7994 LNCS, pp. 183–197, 2013.

[16] A. Lucieri, M. N. Bajwa, A. Dengel, and S. Ahmed, "Explaining ai-based decision support systems using concept localization maps," in *Neural Information Processing*. Cham: Springer International Publishing, 2020, pp. 185–193.

[17] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8689 LNCS, no. PART 1, pp. 818–833. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10590-1_53

[18] D. P. Papadopoulos, Y. Tamaazousti, F. Ofli, I. Weber, and A. Torralba, "How to make a pizza: Learning a compositional layer-based gan model," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June. IEEE, jun 2019, pp. 7994–8003. [Online]. Available: https://ieeexplore.ieee.org/document/8953956/