DEPARTMENT OF
COMPUTER SCIENCE

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

JOÃO MIGUEL DIAS FERREIRA

BSc in Computer Science and Engineering

EXTRACTING ONTOLOGIES FROM NEURAL NETWORKS

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2022

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

EXTRACTING ONTOLOGIES FROM NEURAL NETWORKS

JOÃO MIGUEL DIAS FERREIRA

BSc in Computer Science and Engineering

Adviser: João Alexandre Carvalho Pinheiro Leite
Associate Professor, NOVA University Lisbon

Co-adviser: Ricardo João Rodrigues Gonçalves
Assistant Professor, NOVA University Lisbon

Examination Committee

Chair: Nuno Manuel Robalo Correia
Full Professor, NOVA University Lisbon

Rapporteur: Vitor Manuel Beires Pinto Nogueira
Assistant Professor, Universidade de Évora

Member: João Alexandre Carvalho Pinheiro Leite
Associate Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE AND ENGINEERING

NOVA University Lisbon
September, 2022

Extracting Ontologies From Neural Networks

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.9.4) [30].

Acknowledgements

First, I would like to express my gratitude to my advisors Professor João Leite and Professor Ricardo Gonçalves, for their guidance and patience during all this work.

I would also like to thank my colleagues and friends that were always with me, especially my friend Manuel de Sousa Ribeiro, who always helped and encouraged me.

In addition, a special thank you to my family which have always believed in me and supported me.

Finally, I would like to acknowledge the support provided by Calouste Gulbenkian Foundation through its "New Talents in AI" program, and by FCT through project RIVER (PTDC/CCI-COM/30952/2017) and strategic project NOVA LINCS (UIDB/04516/2020).

Abstract

Artificial neural network-based methods have been growing in popularity, being successfully applied to perform a variety of tasks. As these systems begin to be deployed in domains where it is desired that they have a certain degree of autonomy and responsibility, the need to comprehend the reasoning behind their answers is becoming a requirement. Though, neural networks are still regarded as *black boxes*, since their internal representation do not provide any human-understandable explanation for their outputs. A considerable amount of work has been done towards the development of methods to increase the interpretability of neural networks. However, these methods often produce interpretations are too complex and do not have any declarative meaning, leaving the user with the burden of rationalizing them. Recent work has shown that it is possible to establish mappings between a neural network's internal representations and a set of human-understandable concepts. In this dissertation we propose a method that leverage these mappings to induce an ontology that describes a neural network's classification process, through logical relations between human-understandable concepts.

Keywords: Artificial Neural Networks, Ontologies, Rule Extraction, Inductive Logic Programming

Resumo

Métodos com base em redes neuronais artificiais têm ganho cada vez mais popularidade, e têm sido aplicados na resolução das mais variadas tarefas. À medida que estes sistemas são usados em domínios onde se pretende que tenham um determinado grau de autonomia e responsabilidade, a necessidade de compreender o raciocínio que os conduz às suas respostas torna-se indispensável. No entanto, as redes neuronais são vistas como *caixas negras*, dado que as suas representações internas não constituem uma explicação interpretável para os seus resultados. Tem sido realizada uma quantidade considerável de investigação com o objetivo de desenvolver métodos que permitam o aumento da interpretabilidade de redes neuronais. Todavia, estes métodos tendem a produzir interpretações complexas e a que não possuem nenhum significado declarativo, deixando o utilizador com a responsabilidade as racionalizar. Uma publicação recente mostrou que é possível estabelecer mapeamentos entre as representações internas de uma rede neuronal e conceitos interpretáveis. Nesta dissertação propomos um método que faz uso destes mapeamentos para induzir uma ontologia que reflete o processo de classificação de uma rede neuronal, através de conceitos compreensiveis relacionados logicamente.

Palavras-chave: Redes Neuronais Artificiais, Ontologias, Extração de Regras, Programação em Lógica Indutiva

Contents

1 Introduction

In this chapter, we present the context of the problem we set to address and the motivation as to why our proposition is relevant. We also present a brief description of our main goal, as well as what we set to explore during the realization of this dissertation.

1.1 Context and Motivation

In recent years, artificial neural networks have allowed for breakthroughs in performing various challenging tasks, often being recognized for their achievements in image [48], text [23] and speech [11] recognition. Neural network-based methods have matched or surpassed other state-of-the-art approaches in real-time object detection [35], video classification [19] and segmentation [8], translation [3], even outperforming humans in facial recognition [54] and playing strategic games [43].

These results, along with the versatility of neural network-based methods lead to an increase in their popularity and their applications in the most various domains, such as, medical diagnosis [49] and drug design [36], self-driving vehicles [18], crime prevention [29] and fraud detection [31]. When we consider the application of neural networks in the aforementioned domains, the need for explanations for their results becomes evident. If we take as an example a neural network model trained to predict whether a patient suffers of some disease based on the patient symptoms and lab results, it does not seem reasonable to blindly act on the result given by the model without any further justification for the patient diagnostic.

Although a model is said to have a very high overall accuracy, the possibility of it being wrong together with the lack of an understandable explanation for its outputs, can lead users to lose trust in the model's results. However, having the ability to understand *why* a certain output was obtained allows users to trust and act upon that information, even if they do not fully comprehend *how* the output was obtained [32]. Additionally, having such justifications could help understand why a certain output is right or wrong.

However, neural networks are typically viewed as black boxes [12], i.e., they are considered *opaque* systems that for a given input produce an output, mostly disregarding their internal representations. The opaqueness of these systems is mostly due to their sub-symbolic nature. Their internal representations are generally based on high-dimensional Euclidean spaces, i.e., real-valued tensors that do not allow for any direct human interpretation. Additionally, neural network models have become increasingly larger and more complex, with some architectures having billions of parameters spread across dozens of layers performing non-linear transformations [5]. For these reasons, it is not plausible to provide a meaningful and understandable justification for the outputs given by a neural network based solely on its internal sub-symbolic representations.

If we imagine a neural network model trained to predict whether a patient suffers of some disease, based on its symptoms and lab results, an explanation for the neural network's outputs constructed by providing a list of its internal sub-symbolic representations would have no use. Since these internal reprentations do not possess any obvious declarative meaning, an explanation constructed directly upon them, would also not possess any obvious declarative meaning. In this case, we would like to be able to explain the output of the neural network in a way that is human-understandable and that meets the needs of different users, e.g., according to their level of expertise in the domain. For example, a doctor might deem more reliable an explanation for why a patient has diabetes if it is built upon concepts such as having an *HbA1c* level below 6,5% and having an *IBM* over 30. Conversely, patients might develop greater trust in an explanation built upon concepts such as having high blood sugar and being overweight, since they have a better understanding of these concepts.

A recent proposal [46] takes a step in providing richer justifications for the outputs of neural networks by exploring the idea of establishing mappings between the internal representations of a neural network and the concepts defined in an ontology – a logic-based theory that describes a domain through the definition of concepts and axioms that dictate the relations between them. In [46] the proposed way for establishing these mappings is through the use of what is called *mapping networks*. These *mapping networks* are small neural networks, each built and trained to identify a single human-understandable concept from the activations produced by the *main network*, i.e. the network whose outputs are to be justified. When an input is fed to the main network, we can use the mapping networks to observe which concepts were identified in the main network's activations, gaining additional knowledge about how the main network is processing its input. The justifications are then obtained through logic-based reasoning methods over the selected ontology, together with the observations from the mapping networks for a given input. A justifications consist in a set of minimal axioms from the chosen ontology, that together with the observations from the mapping networks, entail the output given by the main network. Thus, the justifications are symbolic, human-understandable and allow us to reason upon them. While the mapping networks provide a way for interpreting the sub-symbolic internal representations of a neural network, the ontology provides the

language for conveying the justifications.

The fact that the method proposed in [46] is dependent on an existing ontology raises two important questions. On one hand, since we must rely on an existing ontology there is no guarantee that it exactly reflects the way the neural network classifies its inputs, and so, the justifications produced are meant to be taken only as a plausible understandable rationalization for why the neural network gives a certain output. On the other hand, it is fair to assume that such ontology might not be always available. There are domains where one might want to deploy a neural network and where the knowledge required for designing an ontology is not available or have not yet been compiled into one.

1.2 Problem Statement and Main Contributions

Our main objective is to develop a method that increases the transparency of neural networks by providing meaningful, human-understandable explanations for their outputs. To accomplish this, we intend to leverage the work in [46], namely the ability to get additional knowledge about how a neural network is processing its inputs. However, instead of relying on an existing ontology for producing justifications, we intend to leverage the knowledge obtained by using the mapping networks to *generate* an ontology. The mapping networks provide knowledge on how a main network is processing its inputs, through the identification of mapped concepts. So, when an input is fed to the main network, we know not only the output given by the main network, but also which of the mapped concepts were identified when classifying it. By feeding the main network with a set of different inputs and gathering its output along with the knowledge about which concepts were identified for each input, we should be able to *induce* a logic-based theory – an ontology. This ontology describes the outputs of the neural network by logically relating the mapped concepts, such that it reflects the classification process of the neural network. By relying on the mapping networks' observations, which in turn are grounded on the main network's internal representations, the induced ontologies should better reflect the classification process of the main network. Additionally, by selecting appropriate meaningful concepts to be mapped, we are able to induce ontologies that meet the user needs, in terms of the desired level of abstraction.

With this dissertation, we expect to make the following contributions:

1. Present a method that allows for the construction of ontologies that provide a human-understandable description for a neural network's classification process, grounded on the neural network's activations;

2. Provide a formalization of the proposed method;

3. Provide evidence that the ontologies constructed using the proposed method reflect how a neural network classifies an example, by the means of human-understandable concepts;

4. Explore an application of the proposed method using real-world data.

Parts of this dissertation appear in the publication [7]:

- João Ferreira and Manuel de Sousa Ribeiro and Ricardo Gonçalves and João Leite, Looking Inside the Black-Box: Logic-based Explanations for Neural Networks, Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, 2022.

1.3 Document Structure

The remainder of this document is structured as follows:

- In Chapter 3 we introduce work related to ours, that propose different approches for providing explanations for the outputs of neural networks;
- In Chapter 2 we introduce the necessary background;
- In Chapter 4 we present, discuss and formalize our proposed method;
- In Chapter 5 we explore the application of our method in a controlled setting. We present and discuss the results obtain in experiments designed answers some questions related to the characteristics of our proposed method.
- In Chapter 6 we explore the application of our method in a setting close to a real-world scenario and discuss the results obtained.
- In Chapter 7 we make our final conclusions and present some possible avenues for future work.

2 Background

2.1 Artificial Neural Networks

Artificial neural networks, or neural networks for short, are sub-symbolic machine leaning models that can be used to approximate some function f^*. They are composed of multiple interconnected layers, which are in turn composed of multiple units, that compute a function in parallel. In Figure 2.1 we see a representation of a neural network with 3 layers, each with a different number of units. The first layer is called the input layer and the last is called the output layer, these layers are responsible for receiving the neural networks inputs and producing its outputs, respectively. The remaining layers of the network are called hidden layers. The units of a layer, usually called neurons, are vaguely inspired in biological neuros, in the sense that both receive external signals as inputs and output an activation that will serve as input to other neurons. In Figure 2.2 we see a representation of a neuron. The output or activation of the neuron is computed

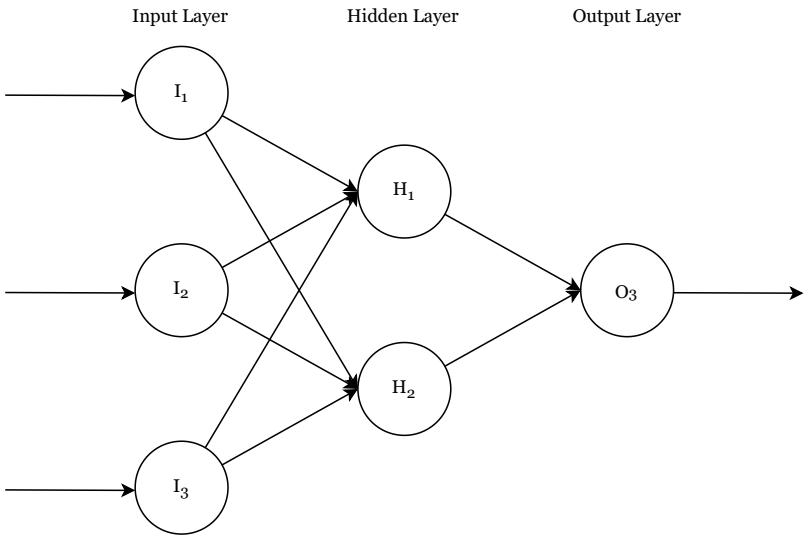


Figure 2.1: Representation of a fully conected neural network with 3 layers.

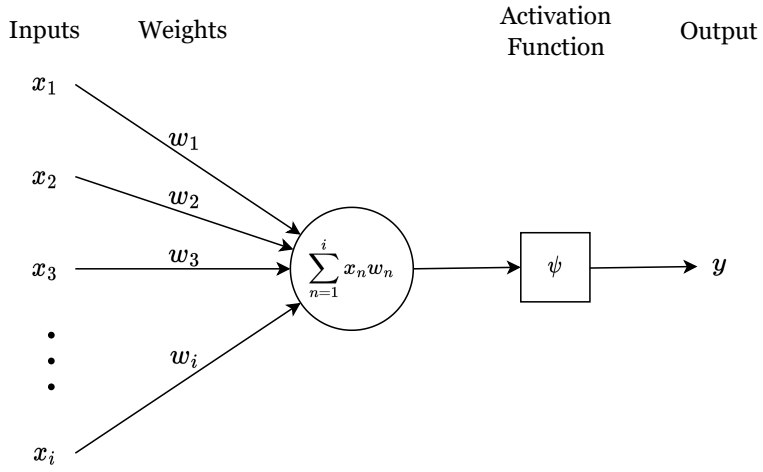


Figure 2.2: Representation of an artificial neuron. Its output is computed by applying a activation function ψ to the sum of its inputs $x_1,...,x_i$ weighted by the corresponding weights.

through the applications of a function φ, the activation function, to the weighted sum of the neuron's inputs.

The process of developing a neural network starts by designing the neural network architecture, from the number of layers, the size of each layer, the type of units in each layer, etc. These decisions are closely dependent on the data it will take as input and the task the neural network will be performing and, usually, involve fine-tuning through experimentation. After designing the neural network architecture, the network is trained and evaluated.

There are several different neural network's architectures, as well as different approaches for training them, where different approaches are more suited for certain tasks. In this Chapter, we will be focusing on introducing feedforward neural networks, namely deep fully connected networks, and convolutional networks, trained using a supervised learning approach, i.e., using labeled data. The distinguished characteristic of feedforward neural networks is in the way information flows through its layers. In feedforward networks, information flows always from one layer to the next. This means that the output of a layers is never fed back into the model. In other terms, if we represent a feedforward neural network as a directed graph where the vertices are the network's layers and the edges between them are the flow of information, the graph would be acyclic. A feedforward network defines a mapping $y = f(x, \Theta)$ where x is the input of the network, y its output and Θ its parameters, such as the neurons weights. The network's parameters Θ are optimized during training so that it results in the best function approximation.

Deep fully connected neural networks are a type of feedforward neural network that is characterized by the units of one layer being connected to every unit of the next layer. This is the case of the neural network representation shown in Figure 2.1. A popular choice for the activation function of the neurons in the hidden layers of a fully connected

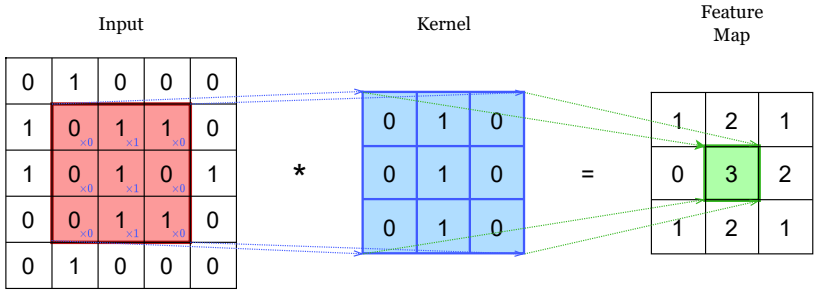


Figure 2.3: Representation of the convolution operation performed in a convolutional layer of a neural network. The output of the convolution is called the feature map, and is computed by combining the input and kernel.

network is the RELU function. The RELU is a piecewise linear function that for negative x_i inputs is zero, and for positive inputs x_i is equal to x_i. With this, the RELU function offers a nonlinear transformation, while still maintaining some characteristics of linear functions that facilitate the optimization process.

Convolutional neural networks are a type of feedforward network that arouse to deal with some shortfalls of deep fully connected networks, namely their struggle to deal with large input data, such as images. A deep fully connected network taking as input an image with a resolution of 250×250 with 3 color channels (RGB) requires each neuron of the first fully connected layer to have $250 \times 250 \times 3 = 187500$ parameters. By having several of neurons in this layer, not just the computational cost of performing the operations becomes impractical, but the neural network aslo quickly starts to suffer from overfitting and the curse of dimensionality. Convolutional neural networks have convolutional layers that apply a convolution operation. In the context of neural networks, a convolution is an operation between the input and the kernel that is done in one or more axis, and that outputs what is sometimes referred to as the feature map. The input is usually a multidimensional array of data, and the kernel is a multidimensional array of parameters that are adjusted during training by the learning algorithm. In Figure 2.3 we can see a representation of a convolution operation performed by a convolutional unit. By making the kernel smaller that the input, convolutional networks have sparse connections between units, since the kernel does not interact the entire input. Convolutional neural networks also employ parameter sharing. Each member of the kernel is used at every position of the input (with exception to the boundaries, depending on design decisions). This means that rather than having to learn a separate set of parameters for each location in the input, it is possible to learn only one. An important property of parameter sharing in the case of convolutional layers is their equivariance to translation, which means that if the input changes, the output changes in the same way. This allows convolutional layers to detect input features independently of their position, e.g., in the context of images, if a convolutional layer detects an object in an image and that object is moved, the object representation will move by the same amount in the output. In convolutional neural

networks it is also common to use pooling layers. In these layers, the units implement a pooling function where each one summarizes a part of the input.A commonly used pooling operation is one called max pooling, which takes a part of the input and outputs the maximum of the features. Pooling layers improve the computational efficiency of the neural network, since by outputting a summary of their input, the next layer in the network will have a reduced input while maintaining the most relevant features. Another advantage of using pooling layers is to help maintain the representations invariant to small changes in the input.

When designing a neural network architecture, an important aspect to take into consideration for the is the output layer. The type of the output layer, its number of units and their activation functions is closely dependent on the task the neural network is to perform. If a neural network is to be performing a regression task, i.e., we want the network to output a continuous value, it is common for the activation function of the neurons in the output layer to have a linear activation function. When the network is to perform a classification task, i.e., an input belongs to some class, typically, the number of neurons in the output layer is equal to the number of classes considered. In classification tasks, the activation of the neurons in the output layer is, usually the sigmoid function which outputs a value between $[0, 1]$ indicating the likelihood of an example belong to the corresponding class. In multi-class classifications tasks, i.e., an input belongs to one class among multiple classes, the size of the output layer is equal to the number of classes considered. In this case, the softmax function, an extension of the sigmoid function for multiple classes, is used as the activation function of the neurons.

In supervised learning, neural networks are trained through using a set of labeled examples. These examples are fed to the network and when the output does not match the corresponding label, the value of each weight in the neural network is changes to reduce the error. This is done by an optimization algorithm that defines how the values of the parameters of a neural network should be changed based on the difference between the obtained output and the example label. This difference is calculated using a loss function. The most used optimization algorithms are Stochastic Gradient Descent and its variants. They work by iteratively updating the parameters of the network in the direction of the gradient of the loss function, using a batch of randomly selected training examples. This randomness helps avoid local minima and improves the chances of converging to better solutions. In Stochastic Gradient Descent learning rate, i.e., how much to update the network's parameters in each iteration, is constant and must be defined when designing the network. The learning rate has a major impact in the ability of the optimizer to find good solutions, and it's a difficult to task to determine a value only by trial and error. Variants of Stochastic Gradient Descent can automatically adjust the learning parameter during training to aid with convergence. A popular choice among those algorithms is Adam [21]. After training, a neural network should be tested using a set of examples different from the ones used in training, this way, we can get an unbiased estimation of the performance of the neural network. If the neural network performs well in the set of

testing examples, it is said that the network was able to generalize.

In this section, we provided only a breif overview of neural network, focusing on some different architectures and the trainning of the networks. For a more comprehensive exploration of neural networks, we refer to [10].

2.2 Description Logics

Descriptions Logics are a family of knowledge representation languages, consisting of decidable fragments of First Order Logic (FOL), which are equipped with a formal, logic-based semantics. Description Logics can be used to represent the knowledge of a domain through the definition of concepts about the domain, and then using them to describe the proprieties of the objects and individuals in the domain. Besides providing the means for describing the knowledge of a domain, these languages allow us to infer knowledge about the domain, that is not explicitly contained in the knowledge base. The 3 main ingredients for representing a domain's knowledge using a Description Logics are: *atomic concepts*, which denote a set of individuals in the domain of interest, that share some characteristics; *atomic roles*, which denote binary relations between individuals of the domain; and *individuals*, which are instances in the knowledge base, representing a single object in the domain of interest. For representing the knowledge of a domain using a Description Logics, the first step is to define the vocabulary, a set of *atomic concepts* and a set of *atomic roles*. Atomic concepts and atomic roles correspond to unary and binary predicates in FOL, respectively, while individuals in Description Logics correspond to constants in FOL. The sets atomic concepts and atomic roles provide the necessary syntax for representing the domain knowledge. Different Descriptions Logics provide different constructors that can be used built *complex concepts*, and there is a trade of between the expressiveness of the language and the complexity of reasoning upon such language [4]. In this dissertation, we will be focusing on the $\mathcal{ALCQ}$ description logic, which provides the following constructors for building complex concepts:

$$
\begin{array}{rcll}
C, D & \longrightarrow & A \quad | & (\textit{atomic concept}) \\
 & & \top \quad | & (\textit{univeral concept}) \\
 & & \bot \quad | & (\textit{bottom concept}) \\
 & & \neg A \quad | & (\textit{atomic negation}) \\
 & & C \sqcap D \quad | & (\textit{intersection}) \\
 & & C \sqcup D \quad | & (\textit{union}) \\
 & & \forall R.C \quad | & (\textit{value restriction}) \\
 & & \exists R.C \quad | & (\textit{existential quantification}) \\
 & & \geqslant n\, R.C \quad | & (\textit{qualified at-least restriction}) \\
 & & \leqslant n\, R.C & (\textit{qualified at-most restriction})
\end{array}
$$

Where C and D denote concept descriptions, A denotes an atomic concept, R denotes a role, and n denotes an integer number. Using these constructors one can express complex concepts through the relation of atomic concepts and roles. As an example, one could describe individuals of the male gender that have a daughter using the complex concept: $\mathsf{Person} \sqcap \mathsf{Male} \sqcap \exists \mathsf{parent}.(\mathsf{Person} \sqcap \mathsf{Female})$.

Description Logics use a set-theoretic interpretation for concepts. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, consists of a non-empty set $\Delta^{\mathcal{I}}$ called the interpretation domain, and an interpretation function $\cdot^{\mathcal{I}}$) which maps each atomic concept to a subset of $\Delta^{\mathcal{I}}$ and each atomic role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function can be extended for complex concepts according to the following definitions:

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
\neg A^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus A^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\forall R.C)^{\mathcal{I}} &= \{\, a \in \Delta^{\mathcal{I}} \mid \forall b.\, (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}} \} \\
(\exists R.C)^{\mathcal{I}} &= \{\, a \in \Delta^{\mathcal{I}} \mid \exists b.\, (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}} \} \\
(\geqslant nR.C)^{\mathcal{I}} &= \{\, a \in \Delta^{\mathcal{I}} \mid \#\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \geq n \} \\
(\leqslant nR.C)^{\mathcal{I}} &= \{\, a \in \Delta^{\mathcal{I}} \mid \#\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \leq n \}
\end{aligned}
$$

Complex concepts can be used to describe other concepts in the knowledge base, through what is called axioms. There two kinds of axioms, one expresses the equality between concepts (or roles), and the other expresses inclusion between concepts (or roles).

$$\begin{aligned}
\text{Person} &\sqsubseteq \text{Male} \\
\text{Person} &\sqsubseteq \text{Female} \\
\text{Father} &\equiv \text{Male} \sqcap \exists \text{parent}.(\text{Person}) \\
\text{Mother} &\equiv \text{Male} \sqcap \exists \text{parent}.(\text{Person}) \\
\text{GrandParent} &\equiv \text{Person} \sqcap \exists \text{parent}.(\text{Father} \sqcup \text{Mother})
\end{aligned}$$

$$\begin{aligned}
&\text{Person(GEORGE)} \\
&\text{Person(SUSAN)} \\
&\text{Father(GEORGE)} \\
&\text{parent(SUSAN, GEORGE)}
\end{aligned}$$

Figure 2.4: Example of an ontology in Description Logics by the TBox (top) and the ABox (bottom).

We say that two concepts are equivalent, $C \equiv D$, if $C^{\mathcal{I}} = D^{\mathcal{I}}$, and we say that two roles are equivalent $R \equiv S$, if $R^{\mathcal{I}} = S^{\mathcal{I}}$, for all interpretations $\mathcal{I}$. The same notion applies to the axioms expressing inclusion, for two concepts C and D, $C \sqsubseteq D$, if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and for two roles R and S, $R \sqsubseteq S$, if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, for all interpretations $\mathcal{I}$. Description Logics use assertions for expressing the relation between individuals and concepts (or roles). Considering a and b as individuals, a concept assertion $C(a)$ satisfies some interpretation $\mathcal{I}$, $\mathcal{I} \models C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and a role assertion $R(a, b)$ satisfies some interpretation $\mathcal{I}$, $\mathcal{I} \models R(a, b)$.

An ontology in Description Logics is tipically composed by a TBox and an ABox. The Tbox defines the terminology of the domain through axioms that describe how the concepts in the domain are related. The ABox consists in a set of assertions on the individuals of the domain. In Figure 2.4 we see an ontology in discription logics, the TBox is represented by the top-most axioms, and the ABox is represented at the bottom.

This was only a brief overview on Description Logics, for a more in dept analizes, we refer to [51].

2.3 Inductive Logic Programming

Inductive Logic Programming (ILP) is a form of logic-based machine learning and has the goal of inducing an hypothesis from examples and produce new knowledge from experience [6]. ILP can be described as a supervised learning technique since it generalizes from data examples and learns relational rules from data [15]. A common use of ILP is to perform concept learning, i.e. inducing a concept definition from examples which captures the semantics of the data. Usually, training examples are represented as

logical atoms and are divided in a set of positive and a set of negative examples, Pos and Neg respectively. The background knowledge BK is represented as a logic program and provides the semantic backbone for the ILP algorithm. The algorithm has the goal of searching for a consistent hypothesis, H, such that:

- $BK \cup H \models Pos$
- $BK \cup H \not\models Neg$

For getting a better grasp of how an ILP system can be applied to formulate an hypothesis, we can imagine a simplified example where we want to obtain a hypothesis that accurately describes the concept of a Passenger Train, based on its characteristics such as having a certain type of carriage. Consider that we have one positive example and one negative, trains A and B, where train A is a Passenger Train and train B is not. We can represent these examples using the logic atom $passengerTrain(x)$ where $passengerTrain$ is the target concept and x is a train identifier. Applying this to trains A and B we can write:

$$passengerTrain(A)$$
$$\neg passengerTrain(B)$$

The background knowledge BK contains the knowledge about the trains which will be used as the basis to induce the hypothesis H. In this example we have information about the type of carriages each train has, and it can be represented as:

$$hasPassengerCar(A)$$
$$\neg hasFreightWagon(A)$$
$$hasPassengerCar(B)$$
$$hasFreightWagon(B)$$

In this conditions, an example of an hypothesis induced by an ILP algorithm could be:

$$PassengerTrain \equiv \neg hasFreightWagon$$

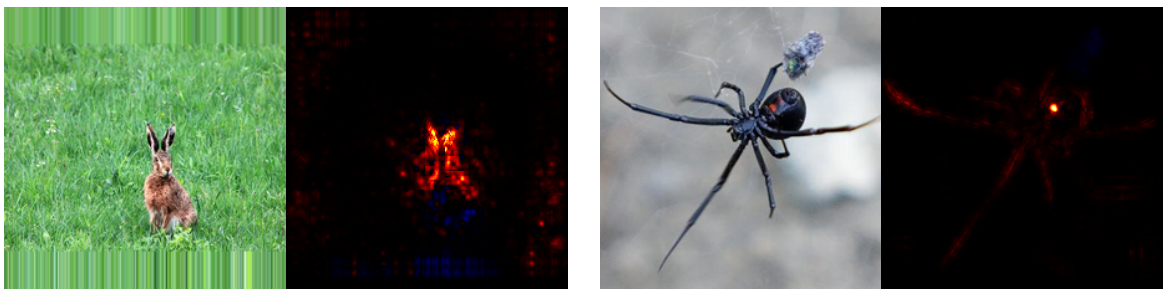
In this instance, a Passenger Train is described as train that does not have a freight wagon. In this example, we can see that all positives examples can be logically derived from $BK \cup H$ and none of the negatives examples can be logically derived from $BK \cup H$. It is worth noticing that in this simplified example we have a very reduced number of training examples and it could be the case that the hypothesis would not generalize outside our training set. Accordingly, the examples in the training set must be enough to capture the semantic of the concept we aim to describe, so that induced hypotheses can generalize to new examples.

3

Related Work

In this chapter, we review some work that is related to our objective, point the main diferences and explore how we can build uppon what has already been proposed.

The increase in popularity of neural network-based methods and their application in domains where the need for explanations is essential, has led to a considerable amount of work being done towards increasing the transparency of neural networks. Different approaches have been proposed for increasing the interpretability of neural networks, one is through saliency and attribution methods where the explanation for a neural network's behavior is given in terms of the contribution of each input feature for a given prediction. Another approach is through proxy-based methods that attempt to substitute the neural network model with another model with a similar behavior and that is inherently interpretable. In this chapter, we present work proposed for augmenting the interpretability of neural networks, grouping them by their approach.



(a) Hare (b) Black widow

Figure 3.1: Examples of heatmap visualizations of attribuition maps generated using LPR, for a neural network classifying an input as an hare (on the left) and as a black widow (on the right). The images were obtained from [16].

3.1 Saliency and attribution methods

Saliency and attribution methods for explaining the outputs of neural networks [27, 50, 55] work by determining the importance, or contribution, of each input feature of a neural network. These methods attempt to explain why a neural network classified an input, through highlighting the features of the input, according to their relevance, i.e., their influence in the classification obtained. The explanations provided by saliency and attributions methods are called attribution maps, that for each feature of the input indicate its importance. They are typically visualized as heat maps, showing the input highlighted according to the attribution map, as showed in Figure 3.1.

Among the methods that follow this approach, we have back-propagation based methods [34], which compute an attribution map using a set of propagation rules to propagate the output backwards in the neural network, to determine the relevance of each input feature. Layer-wise Relevancy Propagation (LRP) [2] is an example of a back-propagation based method, where a rule is attributed to each layer of a neural network, defining how the to propagate the relevancy of its neurons to the neurons of the previous layer, based on their contribution. LRP allows for the attribution of different rules to different layers to better deal with their characteristics, e.g., their size and position in the neural network. The LRP algorithm stops once the relevance is propagated all the way to the input, resulting in an attribution map, describing the contribution of each feature. In Figure 3.1 we show a heat map visualization of attributions maps obtained using LRP.

Another approach for computing attribution maps is through perturbation-based methods [17]. These methods determine the relevance of each input feature applying a mask to the input and measuring the changes in the output of the network. When dealing with images, a popular approach for masking the input is through Occlusion [56], where the input is perturbated by sliding a grey square over the imaging. In this case, the relevance of input features is typically measured according to increase in the error of the neural network's output, when those inputs feature is occluded.

Although salience and attribution methods contribute to augmenting the transparency of neural networks, the explanations they produce do not possess any declarative meaning. The heatmaps generated by these methods provide a visualization of the importance of each feature of the input, but the burden of rationalizing why those inputs were important to the network's task is left to the user, and in most cases, this is not obvious.

3.2 Proxy-based methods

Proxy-based methods for increasing the transparency of neural networks attempt to replace the neural network with another model that is inherently interpretable, while showing a similar behavior to the neural network.

One popular example of a proxy-based method is LIME [37], a model-agnostic method for producing local explanations for the classification of an example. The idea behind

LIME stems from the fact that linear models are easier to interpret than non-linear models, and despite being less powerful, linear models may be used to approximate the behavior of non-linear models, locally. To explain the classification of an example, LIME probes the example's neighborhood for gathering new examples, and then fit a linear model, using the non-linear model's classifications, for classifying these examples. LIME does not provide a global explanation for the outputs of a model, instead it provides a local explanation, in the neighborhood of an example. Moreover, the interpretation of the linear models used for producing the explanations, is usually limited to the features of the example, which may not have a declarative meaning, e.g., an image.

Another example of proxy-based approache are rule extraction methods, which attempt do describe how a model classifies its inputs, through a set of logic-based rules. Rule extraction methods are usually divided into three categories – Pedagogical, Decompositional and Eclectic – according to their interpretation of neural networks. Pedagogical methods treat neural networks as black boxes and attempt to produce rules by focusing on their input-output behavior. Decompositional work by examining the network at the neuron level, taking into account their activation and weights, and the result obtained from each neuron is then aggregated to represent the network as a whole [13]. Eclectic methods, often call hybrid methods, combine the approaches of Pedagogical and Decompositional methods.

3.2.1 Rule Extraction Methods

Rule extraction methods are a popular type of proxy method that attempt to increase the transparency of neural networks by constructing a set o logical rules that aim to replicate the neural network's behavior. The proposed methods for rule extraction are usually divided into three categories – Pedagogical, Decompositional and Eclectic – according to their interpretation of neural networks. Pedagogical methods treat neural networks as black boxes and attempt to produce rules by focusing on their input-output behavior. Decompositional work by examining the network at the neuron level, taking into account its the activation and weights and the result obtained from each neuron is then aggregated to represent the network as a whole [13]. Eclectic methods, often call hybrid methods, combine the approaches of pedagogical and decompositional methods. In this section we review some proposed rule extraction methods, gruping according to the category they fall under.

3.2.2 Pedagogical Rule Extraction Methods

RxREN [1] (Rule extraction by Reverse Engineering the Neural networks) is a pedagogical rule extraction method that works by first applying a pruning procedure that removes the insignificant input neurons of a neural network based on the amount of misclassified examples when a neuron is removed. This pruning procedure ends once the accuracy of the neural networks decreases under a given threshold. The next phase of this method

consists in finding the data range of each significant input neuron, for each output class. This is done by analyzing the set of examples of each class misclassified when the input neuron is absent. If there are no misclassified examples of a given class it is said that that neuron is insignificant to that class and will not figure in the rule that describes it. Then, for each class, a rule is constructed based on the data ranges of the significant neurons. The rules are then pruned by removing conditions and updating data ranges such that it does not affect the accuracy of rule.

The ANN-DT [42] (Artificial Neural-Network Decision Tree) algorithm, extracts binary decision trees from trained neural networks. In a first phase the algorithm uses the neural network to generate outputs for samples interpolated from the training data set, which are guaranteed to be representative of the training set by only allowing the sampling to be done on the neighborhood of the examples in the training set by imposing a limit on the distance (or other similarity metric) between the examples and the interpolated samples. A binary three is then constructed by recursively splitting the set of samples into two until the standard deviation or the variance between two newly created branches is zero, or when some stopping criterion is met. This prevents the creation of sub-branches whose outcome would not be significantly different from one another.

The method proposed in [40], proposes the construction of a set of rules that aim to describe how a neural network classifies its inputs, using an Inductive Logic Programing approach. In this work, the first step for inducing the rules describing the network's classification process is to label a set of examples according to a set of concepts that will be used to formulate the rules. Then, they use an induction framework for inducing the rules, such that the classifications given by the set of rules is in accordance with the classification given by the neural network. The main difference between this method and the other pedagogical rule extraction methods presented, is that this method does not construct rules based directly on the features of the neural network input, but instead, attributes to this features a concept interpretation, through labelling.

Pedagogical rule extraction methods are not limited by the architecture of the neural network. Since these methods treat neural networks as black boxes, they are agnostic to the model's architecture and its internal representations. However, this can also be pointed as their main limitation. By focusing only on the input-output behavior of a neural network, the rules extracted may not provide a faithful rationalization of the network's classification process since two different networks with different internal representations might exhibit the same input-output behavior.

3.2.3 Decompositional Rule Extraction Methods

CRED (a Continuous/discrete Rule Extractor via Decision tree induction) [41] is a decompositional rule extraction method that uses decision trees to describe a neural network with a single hidden layer. The main difference between this method and the ANN-DT algorithm discussed in Section 3.2.2 is that it also considers the activations and inputs

of the hidden layer of the neural network for extracting the rules. CRED uses the C4.5 algorithm [33] to transform each output unit of a neural network into a decision tree [57]. The inner nodes of those trees are considered tests on the values of the hidden layer's neurons and the leaves represent the class that each example would belong to.

DeepRED (Deep neural network Rule Extraction via Decision tree induction) [57] is an extension of the CRED method for neural networks with an arbitrary number of inner layers, by deriving additional decision trees and intermediate rules for every additional hidden layer. DeepRED also uses the pruning procedure employed in the RxREN algorithm.

Both these methods contribute to increasing the transparency of neural networks, enabling the extraction of rules that take into account the network's internal representations and provide a accurate description for the classification process of a neural network. However, these methods often generate rules that despite interpretable are often too complex to be human-understandable, since they rely on the ranges of values of the networks internal representations. Furthemore, this methods are often closely dependent on the networks architecture and can not be easily adapted to work with state of the art neural network architectures such as convolutional networks.

3.2.4 Eclectic Rule Extraction Methods

The MofN method [52] extracts a set of rules that represent the neural network at the neuron level. The rules explaining a single neuron are obtain by first clustering the neuron inputs, based on their weights, using the join algorithm [14]. Then the weights of all neurons in the same cluster is set to the average of their weights. After this step, the algorithm attempts to identify and remove insignificant clusters based on the value of the weights, clusters that cannot change whether the net input exceeds the bias of the neuron are eliminated. Additionally, clusters whose absence has no impact on the ability of the network to provide accurate outputs are removed, i.e. removing them does not result in a qualitative change in the activation of the neuron receiving activations from the cluster. After the iterative process of identifying and removing the unnecessary clusters, the network is retrained and the rules are created by directly translating the bias and weights of each neuron into a rule that is true if the sum of the weighted inputs exceeds the bias. The final step of the algorithm attempts to simplify the rules whenever it is possible.

Similarly to the discussed decompositional rule extraction methods, the rules extrated by this method do not explicitly present a human-understandable explanation for what leads a particular output to be produced from a given input. Since the rules obtained are expressed using numerical values that by themselves have no real meaning, they leave to the user the burden of understanding why the explanation, justifies the output of a neural network.

3.3 TCAV

Testing with Concept Activation Vectors (TCAV) [20] is a method that aims to increase the transparency of neural networks, by providing an interpretability of their internal states using human-understandable concepts and these concepts importance for the classification of an input. TCAV determines the importance of concepts by finding concept activation vectors (CAVs). To find a concept activation vector the first step is to select a set of input samples that represent the concept, then a linear classifier is trained to separate the activations produced by the examples representing the concept and the activations of other examples. The concept activation vector is then defined as the vector normal to the hyperplane that separates the space of activations according to the concept, which is defined by the trained linear classifier. The importance value of a concept for an output of the neural network, i.e., the neural networks sensibility to a concept, can be obtained through CAV's directional derivates.

Although TCAV provides a concept based interpretation of a neural network by determining the importance of concepts for the output of the neural network, no account is given on how these concepts relate to each other and output of the neural network.

4

Ontology Extraction

In this chapter we formalize and describe our proposed method for increasing the explanaibility of trained neural networks through the construction of ontologies that describe their classification process.

Our main objective is to develop a method that provides the ability to construct ontologies for describing the classification process of trained neural networks. These ontologies should be constructed upon human-understandable and meaningful concepts and be grounded on the internal representations of the neural network which classification process it describes. We intend to take advantage of the *mapping networks* proposed in [46] and their ability to extract human-understandable concepts from the activations of a trained neural network, to obtained the knowledge necessary for inducing an ontology. In the following sections, we formalize and describe our proposed method. We start by providing a formalization of neural networks and a formalization of the concept extraction technique proposed in [46], to introduce the necessary notions and notation for formalizing our method. After this, we present a formalization of our proposed method, detailing the reasoning behind the approaches taken. Finally, we provide a summary of our proposal.

4.1 Formalizing Neural Networks

In this section, we formally introduce the necessary notions and notation on neural networks.

We start by defining the mathematical objects that neural networks manipulate, *tensors*. Formally, a *tensor space* is a subset $\mathbb{A}$ of $\mathbb{R}^{n_1 \times \cdots \times n_s}$, with $s \geq 1$, and each $\mathbf{T} \in \mathbb{A}$ is called a tensor. Vectors and matrices are particular cases of tensors, namely elements of $\mathbb{R}^n$ and $\mathbb{R}^{n \times m}$, respectively. We denote by $size(\mathbb{A}) = \Pi_{i \in \{1,\ldots,s\}} n_i$ the size of $\mathbb{A}$, that is, the number of components of $\mathbb{A}$. Given a tensor $\mathbf{T} \in \mathbb{A}$ and an index $j \in \{1,\ldots,n_1\} \times \cdots \times \{1,\ldots,n_s\}$, by $\mathbf{T}_j$ we denote the j-component of $\mathbf{T}$.

A useful operation on tensors is the vectorization of a sequence of tensors, also known as flattening, which corresponds to the representation of such sequence using a vector that preserves the components of the tensors. We abstract the details of possible vectorizations, and assume that for each sequence $(\mathbb{A}_1,\ldots,\mathbb{A}_k)$ of tensors spaces, there is a function $vec : \mathbb{A}_1\times\ldots\times\mathbb{A}_k \to \mathbb{R}^n$, with $n = \sum_{i\in\{1,\ldots,k\}} size(\mathbb{A}_i)$, that gives the vectorization of each sequence of tensors.

The basic elements of neural networks are called *units* or *neurons*, and are defined as functions $u\colon \mathbb{I}_u \times \mathbb{S}_u \to \mathbb{R}$, where $\mathbb{I}_u$ and $\mathbb{S}_u$ are tensor spaces, representing the input and the parameters space of u, respectively. Units that share input and parameters spaces can be combined to form *layers*. Formally, a layer is a function $L\colon \mathbb{I}_L \times \mathbb{S}_L \to \mathbb{O}_L$, where $\mathbb{I}_L$, $\mathbb{S}_L$, and $\mathbb{O}_L$ are tensor spaces, where $\mathbb{O}_L$ is called the output tensor space. The units of a given layer L, one for each component j of the output tensor space $\mathbb{O}_L$, are given by the functions $L_i\colon \mathbb{I}_L \times \mathbb{S}_L \to \mathbb{R}$, where $L_i(\mathsf{X};\boldsymbol{\theta}) = L(\mathsf{X};\boldsymbol{\theta})_i$.

As an example, a fully connected sigmoid layer with two units can be defined as $L\colon \mathbb{R}^k \times \mathbb{R}^{2(k+1)} \to [0,1]^2$ such that $L(\boldsymbol{x};(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)) = (\phi(\boldsymbol{x}\boldsymbol{w}_1 + b_1), \phi(\boldsymbol{x}\boldsymbol{w}_2 + b_2))$, where ϕ is the sigmoid activation function, k the size of the input $\boldsymbol{x}$, $\boldsymbol{\theta}_i = (\boldsymbol{w}_i, b_i)$, where $\boldsymbol{w}_i$ and b_i are the weights and biases of the corresponding unit i. The output of each unit is the projection of the output of the layer into the respective component, i.e., $L_i(\boldsymbol{x};(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2)) = L(\boldsymbol{x};(\boldsymbol{\theta}_1,\boldsymbol{\theta}_2))_i = \phi(\boldsymbol{x}\boldsymbol{w}_i + b_i)$.

A *neural network* N is a sequence $(L^1,\ldots,L^d)$ of layers such that the input of one layer is the same as the output of the previous layer, that is, $\mathbb{I}_{L^j} = \mathbb{O}_{L^{j-1}}$, for all $j \in \{2,\ldots d\}$. The outputs of all layers of N are also usually called the activations of N.

Given a neural network $N = (L^1,\ldots,L^d)$, its input space is equal to the input space of the first layer, that is, $\mathbb{I}_N = \mathbb{I}_{L^1}$, its output space is equal to the output space of the last layer, that is, $\mathbb{O}_N = \mathbb{O}_{L^d}$, its parameter space is the product of the parameter spaces of each layer, that is, $S_N = \mathbb{S}_{L^1} \times \cdots \times \mathbb{S}_{L^d}$, and its size is the number of units it has, that is, $size(N) = \sum_{i\in\{1,\ldots,d\}} size(\mathbb{O}_{L^i})$.

A neural network N is said to be for *classification* if there exists a set of concepts $\mathcal{C}_N$ and a function $concept_N\colon \mathbb{O}_N \times \mathcal{C}_N \to \{0,1\}$, such that $concept_N(\mathsf{Y},\mathsf{C}) = 1$ indicates that concept C can be identified in output Y.

As a simple example of a classification neural network, take N with a single output, that is, $\mathbb{O}_N = [0,1]$, and a single concept $\mathcal{C}_N = \{\mathsf{C}\}$. Then, $concept_N$ can be defined as $concept_N(y,\mathsf{C}) = 1$ iff $y > 0.5$, that is, concept C is only identified when the output value is larger than 0.5.

A *neural network model* is pair $\mathcal{M} = (N,\overline{\boldsymbol{\theta}})$, where N is a neural network and $\overline{\boldsymbol{\theta}}$ is a choice of parameters for each layer of N, that is, $\overline{\boldsymbol{\theta}} \in S_N$. Given a model $\mathcal{M} = (N,\overline{\boldsymbol{\theta}})$, with $N = (L^1,\ldots,L^d)$ and $\overline{\boldsymbol{\theta}} = (\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_d)$, the output of a layer j can be obtained using a function $F^j_{\mathcal{M}}\colon \mathbb{I}_N \to \mathbb{O}_{L^j}$ defined as the composition of the layers up to layer j, that is,

$$F^j_{\mathcal{M}}(\mathsf{X}) = L^j(L^{j-1}(\ldots L^1(\mathsf{X};\boldsymbol{\theta}_1)\ldots;\boldsymbol{\theta}_{j-1});\boldsymbol{\theta}_j)$$

The function associated with the entire model $F_{\mathcal{M}}\colon \mathbb{I}_N \to \mathbb{O}_N$ is exactly the function of the last layer, that is, $F_{\mathcal{M}} = F^d_{\mathcal{M}}$.

A *training set* for a neural network N is a set of labeled examples, that is, a set of pairs $\{(\mathbf{X}^1,\mathbf{Y}^1),\ldots,(\mathbf{X}^k,\mathbf{Y}^k)\}$, with $\{\mathbf{X}^1,\ldots,\mathbf{X}^k\} \subseteq \mathbb{I}_N$ and $\{\mathbf{Y}^1,\ldots,\mathbf{Y}^k\} \subseteq \mathbb{O}_N$.

Given a neural network model $(N,\overline{\theta})$, the *training process* consists in finding a new model $(N,\overline{\theta}')$, by optimizing some objective function with respect to the parameters space S_N over a training set for N. The *accuracy* of a classification model $\mathcal{M}$ over a set of samples $\mathbb{X}$, denoted by $acc_{\mathbb{X}}(\mathcal{M})$, is the proportion of well-classified examples from the sample set. We may omit $\mathbb{X}$ when it is implicit.

4.2 Concept Extraction

In this section we formalize the method proposed in [46] to extract human-understandable concepts from the activation patterns of a trained neural network, usually called main network.

We assume the main network model $\mathcal{M} = (N,\overline{\theta})$ to be fixed, where $N = (L^1,\ldots,L^d)$ is a classification network for a set $\mathcal{C}_N$ of concepts. We also assume a vectorization $vec : \mathbb{O}_{L^1} \times \ldots \times \mathbb{O}_{L^d} \to \mathbb{R}^k$, with $k = size(N)$, that transforms the activations of N into a vector form.

The fundamental elements of the method are the *mapping networks*, each being a classification network M that has as input the activations of the main network and is associated with a single concept C_M. Formally, a mapping network over $\mathcal{M}$ is a classification network M, with $\mathbb{I}_M = \mathbb{R}^k$, where $k = size(N)$, $\mathbb{O}_M = [0,1]$, and $\mathcal{C}_M = \{\mathsf{C}_M\}$.

To allow mapping networks to consider only a subset of the set of all units of the main network N, we associate to each a sequence of selection tensors, $sel_M = (\mathbf{S}^1,\ldots,\mathbf{S}^d)$, one for each layer of N, such that each $\mathbf{S}^i$ has the same dimension as $\mathbb{O}_{L^i}$ but is restricted to $0's$ and $1's$[1]. The positions of $\mathbf{S}^i$ with a 1 correspond to the units of the main network that are to be considered as input for the mapping network M. Given a possible output of the main network, $(\mathbf{O}^1,\ldots,\mathbf{O}^d) \in \mathbb{O}_{L^1} \times \ldots \times \mathbb{O}_{L^d}$, the corresponding input for a mapping network M is $vec(\mathbf{O}^1 \cdot \mathbf{S}^1,\ldots,\mathbf{O}^d \cdot \mathbf{S}^d)$, where $\cdot$ is the point-wise multiplication of tensors.

As an example, let $\mathcal{M} = (N,\overline{\theta})$ be a main network model where N has two layers, one with 3×2 units and the other with 4 units. The vectorization transforms the output of the two layers in a vector of size 10, the size of the input vector of every mapping network for $\mathcal{M}$. The selection sequence of each mapping network M is a pair $sel_M = (\mathbf{S}^1,\mathbf{S}^2)$, where $\mathbf{S}^1$ is a 3×2 binary matrix and $\mathbf{S}^2$ is a size 4 binary vector.

A mapping network is trained to identify its associated concept C_M, given the activations of the main network. Therefore, the training set of a mapping network is a set of pairs, each composed by the activations of the main network together with a label that

[1] For the sake of presentation, we assumed that the size of the input vector of each mapping network is equal to the number of units of N. In practice, the size of the input vector of each mapping network is the size of the considered subset of units of N.

indicates whether C_M was identified on the input to the main network that generated such activations. To build such training sets, we first need to label samples of the main network with the concepts associated with the mapping networks. Given a concept C and a set of samples $\mathbb{X} \subseteq \mathbb{I}_N$, a *labeling function* for C is of the form $label_{\mathsf{C}} \colon \mathbb{X} \to \{0,1\}$, such that $label_{\mathsf{C}}(\mathbf{X}) = 1$ indicates whether concept C is present in a given sample $\mathbf{X} \in \mathbb{X}$. A training set for a mapping network M can then be obtained by labeling the output of the units of the main network model generated by an input sample $\mathbf{X}$ with the corresponding label $label_{\mathsf{C}_M}(\mathbf{X})$. Formally, given a set of samples $\mathbb{X} \subseteq \mathbb{I}_N$, the training set for M generated by $\mathbb{X}$ is the set of pairs $(vec(F^1_{\mathcal{M}}(\mathbf{X}) \cdot \mathbf{S}^1, \ldots, F^d_{\mathcal{M}}(\mathbf{X}) \cdot \mathbf{S}^d), label_{\mathsf{C}_M}(\mathbf{X}))$, $\mathbf{X} \in \mathbb{X}$. The cost of training a mapping network is labeling the samples, which usually needs domain expert knowledge.

After training a mapping network M, the resulting model $\mathcal{M}_M = (M, \overline{\theta}_M)$ can be used to predict its associated concept C_M from the activations of the main network. Therefore, we can associate to each $\mathcal{M}_M$ a labeling function $label_{\mathcal{M}_M} \colon \mathbb{I}_N \to \{0,1\}$ where $label_M(\mathbf{X})$ is given by,

$$label_M(\mathbf{X}) = concept_M(F_{\mathcal{M}_M}(vec(F^1_{\mathcal{M}}(\mathbf{X}) \cdot \mathbf{S}^1, \ldots, F^d_{\mathcal{M}}(\mathbf{X}) \cdot \mathbf{S}^d)))$$

for each $\mathbf{X} \in \mathbb{I}_N$. Intuitively, this labeling function is the application of the function associated with $\mathcal{M}$ to the activations of N generated by $\mathbf{X}$. Then, $concept_M$ indicates whether the concept C_M was identified in the activations of N. It is important to note that, once a mapping network is trained, its labeling function has almost negligible cost (one forward pass of both the main and the mapping network), specially when compared with the cost of manually labeling data.

4.3 Inducing the Ontology

In this section we describe our method to induce human-understandable logic-based theories that aim to represent the classification process of a neural network model, based on the output of mapping networks, illustrated in Figure 4.1. We assume a fixed trained main network model $\mathcal{M} = (N, \overline{\theta})$. Since a neural network model $\mathcal{M}$ rely on sub-symbolic representations for performing their task, to describe such task in human-understandable way, we need a logical language $\mathcal{L}$ that is rich enough for representing the network model's classification process and its domain of interest, i.e., the representation of concepts, relations between them, and knowledge specific to the individuals of the domain of discourse. Our method is general in the sense that it is parametric on the choice of a logical language (e.g. Description Logics [51], Answer-Set Programming [28], etc.) and an induction framework defined over that language. We just assume that $\mathcal{L}$ is defined over a given set of concepts that includes the concepts in $\mathcal{C}_N$, which are the concepts associated with the outputs of N, and a set of concepts of interest $\mathcal{C}$, which are concepts relevant to the domain of the main network N. The concepts $\mathsf{C} \in \mathcal{C}_N \cup \mathcal{C}$, are the vocabulary of the ontology we aim to construct, i.e., the set of atomic concepts that will serve as the building blocks of the

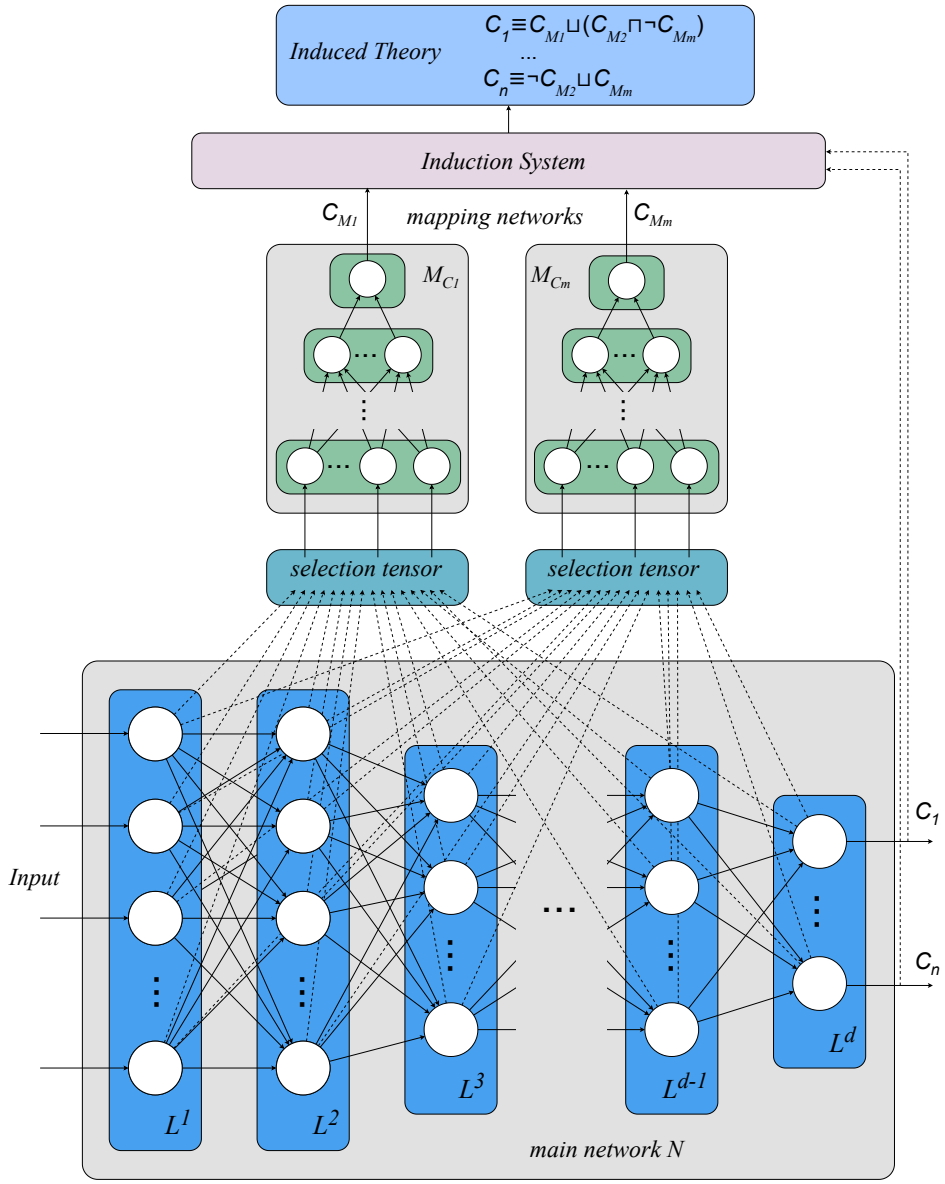


Figure 4.1: Schematic representation of our proposed method for inducing ontologies that describe the classification process of trained neural network.

ontology. The concepts of interest $\mathcal{C}$, would be normally obtained from experts in the domain where the neural network operates, according to their knowledge and intuitions, and what they believe could be enough for describing the task being performed by the network. In addition to this, the concepts should meet the needs of the end-users. One of the purposes for having the ability of explaining the outputs of a neural network is to allow users to trust them, so it is important that the explanations are according to their preferences or level of expertise. It is reasonable to assume that a health practitioner would probably require an explanation given based on a set of concepts different from the ones required by a patient. As an example, for a neural network trained on the CIFAR-10 dataset [22] we could use concepts such as 'wheels', 'horns', or 'water', while for one trained on the HAM10000 dataset [53], we could use concepts such as 'asymmetric', or 'patchy'.

We also assume a fixed set of constants $c_N = \{c_{\mathsf{X}} : \mathsf{X} \in \mathbb{I}_N\}$, representing the individuals of the domain of interest, corresponding to the possible input elements of the main network, i.e., its input space.

The elements of $\mathcal{L}$ are called formulas, and should include as basic elements the atoms of the form $\mathsf{C}(c)$, for $\mathsf{C} \in \mathcal{C} \cup \mathcal{C}_N$ and $c \in c_N$. We assume given a consequence relation $\models \subseteq 2^{\mathcal{L}} \times \mathcal{L}$ over $\mathcal{L}$. Then, an inductive framework over $(\mathcal{L}, \models)$ has three ingredients: the so called background knowledge, denoted by BK, which is a subset of $\mathcal{L}$ and represents the known knowledge; a set Pos of atoms, usually called the positive examples; and a set Neg of atoms, called the negative examples. The goal of the inductive framework is to induce an *hypothesis*, which is a set of formulas that together with the background knowledge entails all the positive examples and none of the negative examples. Formally, a set $H \subseteq \mathcal{L}$ is called an inductive hypothesis if:

- $BK \cup H \models \mathsf{C}(c)$ for every $\mathsf{C}(c) \in Pos$;
- $BK \cup H \not\models \mathsf{C}(c)$ for every $\mathsf{C}(c) \in Neg$.

In our method, the positive and negative examples will refer to the concepts in $\mathcal{C}_N$, while the background knowledge will contain knowledge about the concepts of interest in $\mathcal{C}$. This way, we can obtain a logical theory representing the classification process of N by describing the concepts in $\mathcal{C}_N$ in terms of those in $\mathcal{C}$. To construct the sets BK, Pos and Neg, we fix a set of samples $\mathbb{X} \subseteq \mathbb{I}_N$. For each sample $\mathsf{X} \in \mathbb{X}$, we check using the main network model, if the concepts of $\mathcal{C}_N$ are identified or not in X, adding this information to the positive or negative examples, accordingly. Formally, for each $\mathsf{C} \in \mathcal{C}_N$, we define:

- $Pos^{\mathsf{C}} = \{\mathsf{C}(c_{\mathsf{X}}) : \mathsf{X} \in \mathbb{X} \text{ and } concept_N(F_{\mathcal{M}}(\mathsf{X}), \mathsf{C}) = 1\}$;
- $Neg^{\mathsf{C}} = \{\mathsf{C}(c_{\mathsf{X}}) : \mathsf{X} \in \mathbb{X} \text{ and } concept_N(F_{\mathcal{M}}(\mathsf{X}), \mathsf{C}) = 0\}$.

The sets of positive and negative examples are defined as $Pos = \cup_{\mathsf{C} \in \mathcal{C}_N} Pos^{\mathsf{C}}$ and $Neg = \cup_{\mathsf{C} \in \mathcal{C}_N} Neg^{\mathsf{C}}$, respectively.

The distinctive characteristic of our method is the use of mapping networks to obtain such background knowledge. The fact that mapping networks learn to classify using the activations of the main network is fundamental to assure that this classification is intrinsic to the main network. For each concept $\mathsf{C} \in \mathcal{C}$ we train a mapping network M_{C} as described in Section 4.2, resulting in a mapping network model $\mathcal{M}_{\mathsf{C}}$. Since some of these mapping networks may not learn to correctly identify its associated concept, we assume some threshold α for the accuracy of the mapping networks, below which we do not use them to build the background knowledge. Then, only the concepts in $\mathcal{C}_\alpha = \{\mathsf{C} \in \mathcal{C} : acc(\mathcal{M}_{\mathsf{C}}) > \alpha\}$ will be used to build the background knowledge. Formally, we have,

$$BK = \{\mathsf{C}(c_{\mathsf{X}}) : \mathsf{X} \in \mathbb{X} \text{ and } \mathsf{C} \in \mathcal{C}_\alpha \text{ and } label_{\mathcal{M}_{\mathsf{C}}}(\mathsf{X}) = 1\} \tag{4.1}$$

Note that the set BK could also include additional background knowledge over $\mathcal{C}$, e.g. provided by a domain expert, but we do not further explore that possibility in this dissertation.

Given the sets Pos, Neg and BK defined above, the ontology $\mathcal{O} = H$ where H is the hypothesis given by the induction framework, can be seen as a representation over $\mathcal{L}$ of the classification process of the main network, describing in a human-understandable logical language the characterization of the concepts of the main network in terms of the concepts of interest.

In the work presented in this dissertation, we use an inductive framework based on the ideas of Inductive Logic Programming (ILP) [6], which allows for learning a concept description from examples. The main particularity of this inductive framework, compared to a general framework described above, is that the descriptions for the main network's output concepts $\mathcal{C}_N$ must be learned separately, i.e., for each concept $\mathsf{C} \in \mathcal{C}_N$ we must instantiate and solve an ILP problem, where the Pos and Neg sets are,

- $Pos = Pos^{\mathsf{C}}$,
- $Neg = Neg^{\mathsf{C}}$,

and the BK set is defined the same as in 4.1. In this case, the resulting ontology $\mathcal{O}$ will be the set containing all learnt concept descriptions H^{C}, such that $\mathcal{O} = \cup_{\mathsf{C} \in \mathcal{C}_N} H^{\mathsf{C}}$.

4.4 Method Overview

In Figure 4.1 is a schematic representation of our method for constructing an ontology that provides a human-understandable description for the classification process of a trained neural network. Our method, using an ILP based framework for inducing the ontology, can be summarized into the following steps:

1. Choose a language $\mathcal{L}$ for conveying the ontology, that offers the desired and necessary level of expressiveness;

2. Choose a set of concepts of interest $\mathcal{C}$, that together with the concepts associated with the main networks outputs $\mathcal{C}_N$, will be the vocabulary of the ontology. The concepts of interest $\mathcal{C}$ must possess meaning in the domain where the neural network operates, should meet the end user's needs, and must be adequate to describe the network's classification process, according to a domain expert;

3. Build and train mapping networks to extract the concepts of interest $\mathcal{C}$ from the activations of the main network, as described in 4.2, such that whenever an input is fed to the main network each mapping network can provide the knowledge regarding the identification of their respective concept;

4. Induce a description for each concept in $\mathcal{C}_N$ associated with the outputs of the main network, by instantiating an ILP problem as follows:

    a) Choose a set of learning examples, which represent samples from the input space of the main network. In the context of the ILP problem, these examples are represented using an identifying constant;

    b) Construct the sets of positive and negative examples, Pos and Neg, by dividing the examples according to the classification of the main network;

    c) Construct the background knowledge by instantiating concepts of interest $\mathcal{C}$, according to the mapping network's observations for each example, e.g., if a mapping network identified a concept C for an example x, we add $\mathsf{C}(x)$ to the background knowledge;

    d) Use an inductive framework to solve the ILP problem instantiated.

5. The ontology $\mathcal{O}$ describing the classification process of the main network is the set containing the learnt concept descriptions H^{C}, for each $\mathsf{C} \in \mathcal{C}_N$.

5

Inducing Ontologies from Neural Networks

In this chapter, we explore the application of our method in a controlled setting, using a synthetic dataset. We present and discuss the results obtained when applying our method directly in this setting, as well as the results obtain in experiments that highlight different aspects of our proposal.

Our method allows for the induction of ontologies that provide a human-understandable description for the classification process of a trained neural network. In this chapter, we explore its application in a controlled setting, using the XTRAINS dataset [45]. We will be presenting and discussing the results of the application of our method to neural networks trained to perform a classification task in the domain of this dataset.

In the following section, Section 5.1, we describe the XTRAINS dataset and the trained neural networks – the *main networks*. In Section 5.2, we detail the experimental setting in which we conducted our experiments. In Section 5.3 we examine the direct application of our method for inducing ontologies for each of the main networks, and in the following sections we explore experiments that highlight different aspects of our method: Section 5.4 discusses how the induced theories are affected when using concepts with different levels of abstraction; Section 5.5 analyzes how our method behaves when provided with insufficient concepts to induce an ontology for a neural network; Section 5.6 evaluates the cost of applying our method, and how it performs with different amounts of available data; and Section 5.7 examines the importance of using mapping networks when inducing an ontology for describing the classification process of a neural network.

5.1 Dataset and Main Networks

The XTRAINS dataset is a synthetic dataset, containing images of representations of trains, such as those in Figure 5.1. These images have a resolution of 152×152 pixels with 3 color channels (RGB), and are highly diverse. The trains representations vary in the number, size, and shape of their wagons and wheels, and in the quantity, size, and relative position of the geometric shapes inside each wagon. They also vary in the

(a) TypeA train. (b) TypeB train. (c) TypeC train.

Figure 5.1: Sample images from the XTRAINS dataset.

$$\exists \text{has.}(\text{Wagon} \sqcup \text{Locomotive}) \equiv \text{Train}$$
$$\text{WarTrain} \sqcup \text{EmptyTrain} \equiv \text{TypeA}$$
$$\text{PassengerTrain} \sqcup \text{LongFreightTrain} \equiv \text{TypeB}$$
$$\text{RuralTrain} \sqcup \text{MixedTrain} \equiv \text{TypeC}$$
$$\text{LongTrain} \sqcap \text{FreightTrain} \equiv \text{LongFreightTrain}$$
$$\forall \text{has.}(\text{EmptyWagon} \sqcup \text{Locomotive}) \sqcap \exists \text{has.EmptyWagon} \equiv \text{EmptyTrain}$$
$$\exists \text{has.EmptyWagon} \sqcap \exists \text{has.}(\text{PassengerCar} \sqcup \text{FreightWagon}) \sqcap \neg \exists \text{has.LongWagon} \sqsubseteq \text{RuralTrain}$$
$$\exists \text{has.FreightWagon} \sqcap \exists \text{has.PassengerCar} \sqcap \exists \text{has.EmptyWagon} \sqsubseteq \text{MixedTrain}$$
$$\exists \text{has.}(\text{PassengerCar} \sqcap \text{LongWagon}) \sqcup (\geq 2\ \text{has.PassengerCar}) \sqsubseteq \text{PassengerTrain}$$
$$\exists \text{has.ReinforcedCar} \sqcap \exists \text{has.PassengerCar} \sqsubseteq \text{WarTrain}$$
$$(\geq 2\ \text{has.LongWagon}) \sqcup (\geq 3\ \text{has.Wagon}) \sqsubseteq \text{LongTrain}$$
$$(\geq 2\ \text{has.FreighWagon}) \sqsubseteq \text{FreightTrain}$$

Figure 5.2: Subset of the XTRAINS dataset's ontology.

distance between each wagon, the thickness of their walls, the height of the couplers, etc. Also, noise is present in the images in the form of missing pixels from the trains' representations. Each image in the dataset is labeled according to their visual features, which correspond to human-understandable concepts. The use of images is challenging since human-understandable concepts cannot be directly extracted from pixel representations. The dataset is accompanied by an ontology represented in Description Logics. In Figure 5.2, is a subset of this ontology, which agrees with the labeling of the images in the XTRAINS dataset. The ontology describes the domain of the trains through concepts whose meaning is related to the train's visual features. For example, the concept PassengerCar is represented by a wagon containing at least a circle inside its walls and the concept FreightWagon is represented by a wagon containing geometric figures inside its walls, which are not circles. The ontology represents how different concepts are related, e.g., a train has a wagon and a locomotive and how trains are classified in this domain,

e.g., a type A train is either a war train or an empty train. In the ontology we can also identify at different levels of abstraction, where the concepts at a higher level of abstraction are described by logically relating the concepts at a lower level of abstraction. For instance, we have concepts such as MixedTrain which is described as a train having an empty wagon, a passenger car, and a freight wagon.

Due to being accompanied by a rich ontology and providing images labeled for a large set of human-understandable concepts, the XTRAINS dataset provides a controlled setting. Besides the direct application of our method, this controlled setting allows us to conduct other experiments explore and highlight different aspects of our method, which would be difficult to conduct in other circumstances, e.g., in the context of using real-world data, we could be limited by the scarcity of data or labelling.

In our experiments, we consider three different main network models – referred to as $\mathcal{M}_A$, $\mathcal{M}_B$, $\mathcal{M}_C$. Each of these models was trained to identify, respectively, trains with the following descriptions, each illustrated in Figure 5.1, and described in the accompanying ontology, cf. Figure 5.2:

- TypeA – trains having either, a wagon with at least a circle inside and a wagon with two walls in each side, or no wagons with geometric figures inside them, cf. 5.1(a);

- TypeB – trains having a long wagon, or two wagons, with at least a circle inside, or trains having at least two long wagons, or three wagons, with at least two of which with a geometric figure inside that is not a circle, cf. 5.1(b);

- TypeC – trains having a wagon with no geometric figure inside and either, a wagon with a circle inside and a wagon with a geometric figure inside that is not a circle, or no long wagons and a wagon with a figure inside, cf. 5.1(c).

The three main networks possess are convolutional networks and they are composed by an input layer for receiving images with a resolution of 152×152 pixels and 3 color channels, followed by the layers that make up the convolutional part of the networks, in this order: two sets of convolutional and batch normalization layers; a max-pooling layer and a dropout layer; two more sets of convolutional, batch normalization and max pooling layers; a dropout layer; and a flatten layer. After these layers, follows the fully connected part of the networks composed by interleaved fully connected and batch normalization layers, which is different for each main network: $\mathcal{M}_A$ has three sets of these layers, $\mathcal{M}_B$ has two, and $\mathcal{M}_C$ has four. The output layer of the three main networks is composed of a single neuron with the sigmoid activation function. The main networks were training with a balanced set of 25000 images, using early stopping with a patience of 30, Adam [21] with a learning rate of 0.001 as the optimizer, and the binary crossentropy loss function. After training, the three networks achieved an accuracy of about 99% when tested on a balanced set of 10000 images.

5.2 Experimental Setting

In the experiments explored in the following sections, we choose Description Logics as the language for conveying the induced ontologies. Description Logics is a standard choice for taxonomic classification and, choosing it as our language allows us to draw comparisons between the induced ontologies and the ontology that accompanies the dataset.

The mapping networks developed for extracting the concepts chosen in each experiment all have the same architecture. They are composed of an input layer receiving the corresponding main network's activations and an output layer composed of a single neuron using the sigmoid activation function. For the input of the mapping networks, we are only considering the activations of the neurons in the batch normalization layers, of the fully connected part of the main networks. The mapping networks were trained using the optimization algorithm Adam [21] with a learning rate of 0.001, the binary crossentropy loss function, and early stopping with a patience of 20. Each mapping network was trained using a balanced set of 800 samples and tested using a balanced set of 1000 samples, with the exception of those in Section 5.6, where we discuss the effects of varying the amount of training samples. We use an accuracy threshold α of 90% to select which mapping networks should be considered when building the background knowledge.

For inducing the ontologies, we use DL-Learner framework [25], although other alternatives for leaning ontologies could be have been used, such as ILASP [24] or differential logic machines [58]. DL-Learner is a very flexible and configurable system that extends the ideas of ILP and can be applied to several ontology learning and engineering tasks. In the following experiments we configure DL-Learner to use the CELOE algorithm proposed in [26] which is biased to minimize the induced ontology. This algorithm uses a refinement operator for searching in the space of expressions. The space of expressions is generated using a top-down approach that starts with the most general expression, $\top$, which is then mapped to a set of more specific expressions by means of a downward refinement operator. The refinement operator can be applied to the obtained expressions again, spanning a search tree, where the search will be guided using a heuristic. We configure DL-Learner to solve a positive-negative learning problem using closed world reasoning with a maximum execution time of 60 seconds. In all the experiments below, the ontologies are induced using a balanced set of 3000 examples and are tested using a balanced set of 1000 examples.

The main metric to quantitatively evaluate the quality of our induced ontologies is the fidelity measure (F_{Main}), which computes how well a given theory imitates the classifications of the main network it was induced from. This metric is computed as the ratio of samples where the classifications of the main network coincide with those obtained from the induced ontology together with the knowledge obtained from the outputs of the mapping networks. Given the high accuracy of the three main networks being studied, we consider another fidelity metric ($F_{XTRAINS}$), which measures the concordance

of the induced ontologies with respect to the XTRAINS dataset labeling. This metric is computed as the ratio of samples where the classifications of the XTRAINS labels (for the main network's output concepts $\mathcal{C}_N$) coincide with those obtained from the induced theory together with the knowledge obtained from the labels of the extracted concepts ($\mathcal{C}_\alpha$). This metric allows us to compare whether the induced theories are better suited to the classifications of our main and mapping networks, or to the labeling of the dataset. Even when obtaining high F_{Main} scores, it is relevant to also examine $F_{XTRAINS}$, since if it is significantly greater than F_{Main}, it could mean that our method was overfitting to the dataset labels. Additionally, we also inspect how the induced ontologies logically compare with the ontology accompanying the XTRAINS dataset. However, although an existing logical relation between both could serve as confirmation of the quality of the induced theory, it is worth noting that the absence of such a relation should not be seen as evidence of an induced theory with poor quality since the main networks may have learned to perform the same classifications through a different internal process.

Each experiment in the following sections, was run 20 times, using different balanced sets of samples for training and testing purposes. The induced ontologies that we present and logically analyze are the mode of the 20 runs, while the discussed fidelity scores correspond to the average of the fidelity scores obtained in the 20 runs.

5.3 Inducing a Main Network's Ontology

In this section, we explore our main hypothesis, namely that it is possible induce a human-understandable ontology that describes the classification process of a trained neural network. Also, that we can induce such ontology by leveraging the classifications provided by mapping networks trained to identify a set of concepts of interest from the activations of said neural network. To that end we applied our proposed method for inducing ontologies for the main networks models presented in Section 5.1 - $\mathcal{M}_A$, $\mathcal{M}_B$, and $\mathcal{M}_C$. For each main network, we trained mapping networks to extract the following 11 concepts [1] from the ontology accompanying the XTRAINS dataset, cf. 5.2: EmptyTrain, FreightTrain, LongTrain, MixedTrain, PassengerTrain, RuralTrain, WarTrain, ∃has.FreightWagon, ∃has.LongWagon, ∃has.OpenRoofCar, ∃has.ReinforcedCar. We point out that for the purposes of our method, we treat the complex concepts, e.g., ∃has.ReinforcedCar, as atomic concepts. When presenting our results, we maintain the representation of these concepts as complex concepts, to facilitate the comparison between the induced ontologies and the ontology accompanying the dataset.

The ontologies that resulted from the application of our method to each main network model, $\mathcal{M}_A$, $\mathcal{M}_B$, $\mathcal{M}_C$, can be seen bellow:

[1] We selected the same 11 concepts that were explored in [46].

$$\begin{aligned}
\mathsf{TypeA} &\equiv \mathsf{EmptyTrain} \sqcup \mathsf{WarTrain} \\
\mathsf{TypeB} &\equiv (\mathsf{LongTrain} \sqcap \mathsf{FreightTrain}) \\
&\quad \sqcup (\mathsf{PassengerTrain} \sqcap \neg \mathsf{EmptyTrain}) \\
\mathsf{TypeC} &\equiv \mathsf{MixedTrain} \sqcup \mathsf{RuralTrain}
\end{aligned}$$

The ontology induced for $\mathcal{M}_A$ states that a sample is classified as being a TypeA train if and only if it is either a war train or an empty train. The ontology induced for M_B states that a sample is classified as being a TypeB train if and only if it is either a freight train and a long train, or a passenger train and not an empty train. The ontology induced for $\mathcal{M}_C$ states that a sample is classified as being a TypeC train if and only if it is a mixed train and a rural train.

It is worth noting that, despite the usage of 11 concepts for inducing each ontology, we see that the ontology induced for $\mathcal{M}_A$ only contains 2 of those concepts, the ontology induced for $\mathcal{M}_B$ contains only 4, and the ontology induced for $\mathcal{M}_C$ contains only 2 concepts. This is due to only considering the concepts that could be extracted using the mapping networks, i.e., their testing accuracy was not under 90%, and the learning framework's ability to induce minimal ontologies. This suggests that the proposed method was able to properly select the concepts better suited to describe each neural network's classification process.

A logical comparison of the induced ontologies with the ontology that accompanies the XTRAINS dataset allows us to observe that both the definitions of concepts TypeA and TypeC in the induced ontologies for $\mathcal{M}_A$ and $\mathcal{M}_C$, respectively, are equivalent to the respective definitions of TypeA and TypeC in the datasets' ontology. The concept of TypeB in the induced ontology for $\mathcal{M}_B$ is a subclass of TypeB as defined in the XTRAINS ontology, meaning that any individual of TypeB in the induced ontology is also considered to be of TypeB according to the XTRAINS ontology.

Table 5.1 shows the fidelity scores of the induced theories. The resulting high F_{Main} scores provide evidence that the resulting ontologies are properly reflecting the main network's internal classification process, strongly supporting our main hypothesis. Furthermore, one can observe that the induced ontologies are also faithful to the dataset's labels, achieving similar $F_{XTRAINS}$ and F_{Main} scores. This is an interesting indication that the main networks have learned to classify their samples in similar fashion to how they are classified by the ontology accompanying the XTRAINS dataset.

5.4 Levels of Abstraction

It is often the case that different scenarios require different levels of abstraction and detail to interpret a neural networks' classification process. For example, a user may want to understand how a neural network is classifying some type of train based on other known

	F_{Main}	$F_{XTRAINS}$
M_A	$99.72 \pm 0.18\%$	$99.92 \pm 0.35\%$
M_B	$98.71 \pm 0.31\%$	$99.83 \pm 0.76\%$
M_C	$99.33 \pm 0.32\%$	$99.52 \pm 1.52\%$

Table 5.1: Fidelity scores for the ontologies induced for each main network.

types of trains, or otherwise wish to be more specific and understand how particular kinds of wagons are influencing the output of the neural network. We hypothesize that it is possible to induce different ontologies, at different levels of abstraction, which are faithful to a neural network's classification process. This should be possible through the selection of different set of concepts of interest, at the desired level of abstraction, for inducing each ontology. To investigate the ability to use our proposed method for inducing ontologies at different levels of abstraction, we formulated two sets of concepts. One, is a set of train-level concepts, i.e., concepts at a higher level of abstraction which are related to different types of trains, and contains the concepts: EmptyTrain, LongFreightTrain, MixedTrain, PassengerTrain, RuralTrain, WarTrain. The other, is a set of wagon-level concepts, i.e., concepts at a lower-level of abstraction which are related to the types of wagons a train can possess, and contains the concepts: ≥ 3 has.Wagon, $\exists$has.EmptyWagon, $\exists$has.FreightWagon, $\exists$has.LongWagon, $\exists$has.(LongWagon $\sqcap$ PassengerCar), $\exists$has.ReinforcedCar, ≥ 2 has.FreightWagon, ≥ 2 has.LongWagon, ≥ 2 has.PassengerCar, $\exists$has.PassengerCar.

The application of our method to the three main networks - M_A, M_B, and M_C - using the train-level concepts, resulted in the ontologies:

$$\begin{aligned}
&\mathsf{TypeA} \equiv \mathsf{EmptyTrain} \sqcup \mathsf{WarTrain} \\
&\mathsf{TypeB} \equiv \mathsf{LongFreightTrain} \sqcup \mathsf{PassengerTrain} \\
&\mathsf{TypeC} \equiv \mathsf{MixedTrain} \sqcup \mathsf{RuralTrain}
\end{aligned}$$

While with the wagon-level concepts we obtained the following ontologies:

$$\begin{aligned}
\mathsf{TypeA} \equiv\ & (\exists \mathsf{has.PassengerCar} \sqcup \neg \exists \mathsf{has.FreightWagon}) \\
& \sqcap (\exists \mathsf{has.ReinforcedCar} \sqcup \neg \exists \mathsf{has.PassengerCar}) \\
\mathsf{TypeB} \equiv\ & \exists \mathsf{has.(LongWagon} \sqcap\ \mathsf{PassengerCar}) \\
& \sqcup (\geq 3\ \mathsf{has.Wagon} \sqcap \geq 2\ \mathsf{has.FreightWagon}) \\
\mathsf{TypeC} \equiv\ & \exists \mathsf{has.EmptyWagon} \sqcap (\neg \exists \mathsf{has.LongWagon} \\
& \sqcup (\geq 3\ \mathsf{has.Wagon} \sqcap \exists \mathsf{has.PassengerCar}))
\end{aligned}$$

Comparing the induced ontologies with the dataset's ontology, the induced definitions for the train-level ontologies of TypeA, TypeB, and TypeC are all logically equivalent to the ones in the dataset's ontology. Turning to the wagon-level induced ontologies, we can

		F_{Main}	$F_{XTRAINS}$
Train-level	M_A	$99.76 \pm 0.19\%$	$100.00 \pm 0.0\%$
	M_B	$98.82 \pm 0.39\%$	$100.00 \pm 0.0\%$
	M_C	$99.46 \pm 0.20\%$	$100.00 \pm 0.0\%$
Wagon-level	M_A	$94.55 \pm 10.14\%$	$94.44 \pm 11.12\%$
	M_B	$97.50 \pm 0.52\%$	$96.73 \pm 1.18\%$
	M_C	$98.16 \pm 0.36\%$	$99.02 \pm 0.56\%$

Table 5.2: Fidelity scores for the ontologies induced using the train-level and wagon-level concepts.

observe that the induced definition of TypeB is a subclass of the corresponding definition in the dataset's ontology, while neither the induced definitions of TypeA and TypeC are subclasses or superclasses of their corresponding definitions in the dataset's ontology. However, when we turn to the fidelity scores for both groups of induced ontologies, shown in Table 5.2, we can observe that the train-level ontologies achieve, on average, a slightly higher value (99.35%) than the wagon-level ones (96.74%), but they are all high, showing that our method is able to generate high-quality ontologies at different level of abstraction. The slight difference in fidelity between the results achieved by the train-level and wagon-level induced ontologies can be attributed to the difference in accuracy of the mapping networks – 97.43% on average for the train-level concepts and 96.07% for the wagon-level ones. Since we rely on the mapping networks for identifying these concepts, this difference in accuracy, albeit small, means more samples are wrongly classified when using the wagon-level concepts that when using the train-level concepts. This effect is amplified by the fact that the wagon-level ontologies use more concepts than the train-level ontologies.

5.5 Insufficient Concepts

So far, we have seen that our method can be used to induce human-understandable ontologies for that represent the classification process of a neural network. Also, we have seen that we can induce faithful ontologies at different levels of abstraction, by selecting concepts of interest that capture the desired level of abstraction. In this section, we turn our attention to the case where the concepts of interest selected for inducing the ontologies are somehow insufficient or inappropriate for describing the neural network's classification process. The main reason for exploring this possibility is tied with the application of our method in a real-world scenario, since it may occur that we are unable to determine a set of concepts that is adequate, or sufficient, to describe a neural network's classification process. We expect the fidelity of the ontologies induced using our method to reflect the adequacy of the chosen concepts of interest ($\mathcal{C}$). So, if the chosen concepts are inadequate to describe the classification process of a given neural network, the fidelity score - F_{Main} - of an ontology induced using those concepts should be diminished. If this were not to

	F_{Main}	$F_{XTRAINS}$
M_A	$67.75 \pm 5.01\%$	$67.79 \pm 4.87\%$
M_B	$74.90 \pm 3.46\%$	$73.56 \pm 3.51\%$
M_C	$75.14 \pm 3.14\%$	$74.45 \pm 3.40\%$

Table 5.3: Average fidelity scores of the ontologies induced using 20 sets of randomly selected concepts.

happen, it would mean that our method could be picking up on spurious correlations between the inadequate concepts, to describe the neural network's classification process. To test this, we randomly sampled 20 sets of 5 concepts among all the concepts defined in the XTRAINS ontology, and for each set of concepts, we applied our method to induce ontologies for the three main networks models - $\mathcal{M}_A$, $\mathcal{M}_B$, and $\mathcal{M}_C$. The average fidelity scores of the induced ontologies are shown in Table 5.3. For the three main networks, both fidelity scores, F_{Main} and $F_{XTRAINS}$, are considerably lower that the ones obtained in Section 5.3 and Section 5.4. These results back our hypothesis that the inadequacy of the selected concepts is reflected in the quality decrease oh the resulting ontologies. As an example of this is the ontologies induced when using the set of concepts: LongFreightTrain, ∃has.LongWagon, ∃has.PassengerCar, ≥ 3 has.Wagon, ≥ 2 has.PassengerCar. For these set of concepts, were induced the following ontologies:

$$\begin{aligned}
&\mathsf{TypeA} \equiv \top \\
&\mathsf{TypeB} \equiv \geq 3\ \mathsf{has.Wagon} \sqcup \mathsf{LongFreightTrain} \\
&\mathsf{TypeC} \equiv (\geq 3\ \mathsf{has.Wagon} \sqcap \exists \mathsf{has.PassengerCar}) \\
&\qquad \sqcup (\neg(\geq 2\ \mathsf{has.PassengerCar}) \sqcap \neg\exists \mathsf{has.LongWagon})
\end{aligned}$$

A logical comparison of these theories with the dataset's ontology shows that the definition of TypeA in the dataset's ontology is a subclass of the induced definition of TypeA, although this is hardly interesting since that would be the case for any concept. The definitions of TypeB and TypeC are not directly comparable with their respective definitions in the dataset's ontology, given that they are neither their subclasses nor superclasses. The poor quality of these ontologies is reflected in their fidelity scores, shown in Table 5.4. The high fidelity of the ontology obtained for $\mathcal{M}_B$ can be explained by the presence of concepts ≥ 3 has.Wagon and LongFreightTrain, which, based on the experiments of Sections 5.3 and 5.4, are adequate to describe the classification process of $\mathcal{M}_B$.

5.6 Ontology Induction's Cost

The experiments presented so far provide evidence that our method is able to perform properly in multiple scenarios, as long as the extracted concepts are sufficient to describe

	F_{Main}	$F_{XTRAINS}$
M_A	$50.16 \pm 0.26\%$	$50.00 \pm 0.00\%$
M_B	$92.53 \pm 2.75\%$	$92.70 \pm 1.43\%$
M_C	$76.78 \pm 1.99\%$	$76.99 \pm 0.94\%$

Table 5.4: Fidelity scores of the ontologies induced using the set of concepts: LongFreightTrain, ∃has.LongWagon, ∃has.PassengerCar, ≥ 3 has.Wagon, ≥ 2 has.PassengerCar

a neural network's classification process. However, in order to verify the method's feasibility, its cost needs to be assessed.

The method requires the training of multiple mapping networks, and a set of samples to induce the theory. The mapping networks' development adds little computational overhead, given their simple architectures, but requires data labeled with respect to the concepts to be extracted ($\mathcal{C}$). However, this data can be repurposed to induce the ontology, by relabeling it according to the mapping networks' classifications.

Hence, if the quality of our resulting ontologies would mostly depend on the accuracy of the mapping networks, and assuming the availability of enough unlabeled data, then the main cost of applying our method would be in labeleling the data required to train the mapping networks. Consequently, since mapping networks are known to require few labeled data to train, as shown in [46], the cost of our method would be relatively low, depending mostly on the amount of concepts to be extracted ($\mathcal{C}$).

To test whether the quality of the resulting ontologies mostly depends on the accuracy of the mapping networks, for each of the three main networks models - $\mathcal{M}_A$, $\mathcal{M}_B$, and $\mathcal{M}_C$- we induced ontologies using the 11 concepts selected in Section 5.3, while varying the amount of data used for their training. The mapping networks were trained using 50, 100, 200, 400, 600, 800, and 1200 samples. The amount of data used to induce the ontologies remained constant at 3000 samples. A Pearson's correlation test on fidelity F_{Main} and the average accuracy of the mapping networks used to build the background knowledge (BK) for induncing the ontologies, with $\alpha = 90\%$, shows a significant strong correlation ($r = 0.8161$, $p < 0.0001$), thus indicating that when the mapping networks' accuracy increases, the quality of the induced ontologies increases as well. Since mapping networks are able to achieve high accuracies even with few training data, and that this data is typically the limiting factor, the cost of our method can be considered to be quite moderate. Even when considering only 50 samples to both train the mapping networks and induce the ontologies, the resulting ontologies were typically quite accurate, with an average fidelity F_{Main} of about 96%.

5.7 Importance of Mappings

The goal of our proposed method is to develop a human-understandable ontology representing the classification process of a given neural network model. Our method proposes that, for inducing the ontologies we should rely on the results of mapping networks for

(a) (b) (c)

Figure 5.3: Sample images from the XTRAINS dataset with a signal added in each corner of the image. A signal is said to be *on* if it has a symbol inside, and *off* if it has a stripped background or it has no symbol inside.

$$\mathsf{On} \equiv \mathsf{TopLeftOn} \sqcup \mathsf{TopRightOn}$$
$$\mathsf{On} \equiv \mathsf{BottomLeftOn} \sqcup \mathsf{BottomRightOn}$$

Figure 5.4: Ontology describing the labelling of images acording to the presence of signals *on*.

constructing the background knowledge (BK) that will serve as the syntactic and semantic back bone for constructing the ontologies. However, since the mapping networks are trained based on labeled data, one might wonder why they should bother with using mapping networks, instead of using the labels needed to train the mapping networks to define the background knowledge for inducing the ontologies. We hypothesize that ontologies induced directly from using labeled data, even if faithful to the data classification, can misrepresent the classification process of the neural network model they were built to represent. Conversely, we hypothesize that the ontologies induced through our method, being reliant on the internal representations of the model obtained through the mapping networks, will result in ontologies that more faithfully represent its internal process. To test our hypotheses, we designed a classification problem where four traffic signals are added to each image of the XTRAINS dataset, as shown in Figure 5.3. A traffic signal is said to be *on* if it contains any symbol inside it, e.g., a question mark, and *off* if it does not contain any symbol inside, or if it only contains diagonal stripes inside. The images were then labeled regarding 5 different concepts: TopLeftOn, TopRightOn, BottomLeftOn, BottomRightOn, with obvious meaning, and On whenever some traffic signal is on. The dataset was designed such that whenever one of the top traffic signals is on, at least one of the bottom traffic signals is on as well, and vice versa. In Figure 5.4 is an ontology

that describes the labelling of the dataset, using the concepts defined above. Consider a neural network tasked with identifying whether some traffic signal is *on* in a given image. Due to the way the dataset was designed, it is enough to look at either the top, or bottom signals to be able to identify if any traffic signal is on. Thus, one would expect that neural networks trained to perform this task would sometimes learn to identify the top signals, sometimes learn to identify the bottom ones, and sometimes learn to identify them all. In this setting, we trained 50 different main networks, which achieved a test accuracy higher than 90%. When inducing a theory for each of the developed main networks, using their outputs and the dataset labels (instead of the mapping network outputs), we obtained the same ontology for all 50 main networks:

$$\mathsf{On} \equiv \mathsf{BottomLeftOn} \sqcup \mathsf{BottomRightOn}$$

The fact that this ontology is not the same as the ontology describing the labelling of the dataset, c.f. 5.4 is due to DL-Learner inducing minimal ontologies. Moreover, despite the existence of two minimal ontologies that describe the dataset's labels,

$$\mathsf{On} \equiv \mathsf{BottomLeftOn} \sqcup \mathsf{BottomRightOn}$$

and,

$$\mathsf{On} \equiv \mathsf{TopLeftOn} \sqcup \mathsf{TopRightOn}$$

DL-Learner always presents the same one because it is not non-deterministic, in the sense that, the resulting ontology depends on the order in which the concepts were defined in the background knowledge. Having obtained the same ontology for the 50 main networks considered, strongly supports our first hypothesis, suggesting that due to the high accuracy of the main networks' outputs, when using the dataset labels, the induced theories are instead describing how the dataset was labeled. Hence, having no relation to the main network's classification process.

However, when we apply our proposed method to the same 50 main networks, which considers the outputs of the mapping networks to construct the background knowledge for inducing the ontologies, we observe diverse induced ontologies. Our results reveal that 22% of the main networks learned to classify their outputs just by considering whether the two top signals were *on*, resulting in the induced ontology:

$$\mathsf{On} \equiv \mathsf{TopLeftOn} \sqcup \mathsf{TopRightOn}$$

while 42% were just considering the two bottom signals, resulting in the induced ontology:

$$\mathsf{On} \equiv \mathsf{BottomLeftOn} \sqcup \mathsf{BottomRightOn}$$

This indicates that different main networks learned to classify their inputs differently, and that this was captured by the ontologies induced by relying on the outputs of the mapping networks, which was not the case when only the dataset labels were used. The remaining main networks learned to classify their inputs based on some other combination of signals, e.g., by observing all signals. Interestingly, our results hint that some of these networks did not seem to have learned the concepts that we considered, but instead had learned to classify their inputs based on some other spurious correlations present in the dataset.

6

Inducing Ontologies In a Real-World Scenario

In this chapter, we explore the application of our method in a real-world scenario, using a dataset with fotographic images. We begin by presenting the dataset used in our experiments and the neural network considered. Following this, we provide the experimental setting in which we applied our method. Finally, we present and discuss the results obtained in our experiments.

6.1 Dataset

Our experiments are conducted in the setting of the *German Traffic Sign Recognition Benchmark Dataset* (*GTSRB*) [47]. This dataset is composed of photographic images, as the ones in Figure 6.1. These images are of real-world traffic signals and are labeled according to the category of each signal:

- Category A – Danger signals, c.f. Figure 6.1(a);
- Category B – Priority signals, c.f. Figure 6.1(b);
- Category C – Prohibitory signals, c.f. Figure 6.1(c);
- Category D – Mandatory signals, c.f. Figure 6.1(d).

(a) Category A (b) Category B (c) Category C (d) Category D

Figure 6.1: Sample images from the GTSRB dataset.

- Bar, BlackBar, WhiteBar,
- Border, BlackBorder, RedBorder, WhiteBorder,
- RedGround, WhiteGround, YellowGround,
- EndProhibition, StartProhibition,
- CircularShape, DiamondShape, TriangularShape, OctagonalShape,
- Symbol, BlackSymbol, WhiteSymbol, SymbolNoEntryGoods, SymbolOvertaking, SymbolOvertakingGoods, SymbolStop, SymbolSpeed,
- Blue, Post.

Figure 6.2: Concepts associated with the visual carachteristics of the signals in the GTSRB dataset, grouped according to their meaning.

In addition to the labels regarding the category of each signal, the dataset was labels according to a set of concepts as described in [44]. For the purposes of our experiments, we will only be considering the labelling according to concepts that are associated to the visual characteristics of the traffic signals which meaning can be easily inferred from their designation. These concepts are related to the following characteristics:

- Bar – a signal may have a white bar, a black bar, or have no bar.
- Border – a signal may have a white boarder, a black boarder, red boarder, or have no boarder.
- Background – a signal may have a white, yellow, or red ground;
- Prohibition indication – a signal might indicate the start or the end of a prohibition.
- Shape – a signal might have a triangular, circular, rectangular, or octagonal shape.
- Symbol – a signal might have a black symbol, a white symbol, a symbol indicating no entry goods, a symbol indicating overtaking, a symbol indicating overtaking goods, a *STOP* symbol, a symbol referring to a speed limit (numerical digits), or no symbol.

Notice that, despite the concepts associated with the features described above being related to the traffic signals, in the context of our method, the individuals they refer to are the images in the dataset. For example, the concept CircularShape, in this context, should be interpreted as an image containing a traffic signal with a circular shape. Since each image contains only one signal, we can omit the full interpretation and assume that these concepts are associated to the signal directly. In addition to these concepts that can be directly associated with the traffic signals, we also consider two more concepts that are associated to the image: the presence of the color blue and the presence of a pole. Both these concepts should not be relevant to the task of a neural network classifying the signals according to their category. Thus, if they appear in an induced ontology, we can

investigate whether our method is not using the most appropriate concepts, or the neural network has some biases in its classification process. The full list of concepts considered can be seen in Figure 6.2.

6.2 Main Network

The main network used in the experiment of this chapter, which we denote by $\mathcal{M}_{GTSRB}$, is the same used in [44] and was designed based on the popular *MobileNetV2* architecture. The main network is composed of an input layer receiving 128×128 images with 3 color channels (RGB). This layer is followed by a convolutional layer, 17 residual bottleneck layers [39], an average pooling layer, and another convolutional layer. This network was pretrained using the ImageNet dataset [38]. For adapting the network to the GTSRB dataset, its output layer was replaced by a fully connected layer with 4 neurons, using the softmax activation functions, and the network was retrained to classify the images of the GTSRB dataset as belonging to one of the 4 category of signals. The retraining of the network was done using early stopping with a patience of 30, the Adam [21] optimizer with a learning rate of 0.0001, and using the categorical crossentropy loss function. The resulting neural network is reached a testing accuracy of 99,9%.

6.3 Experimental Setting

In the experiments explored in this chapter, we choose Description Logics as the language for conveying the induced ontologies, as we did in the experiments explored in Chapter 5. For all the concepts described in Section 6.1, we developed mapping networks with 2 different architectures. One of these is the same used in the experiments of Chapter 5, composed of an input layer receiving the main network's activations and an output layer composed of a single neuron using the sigmoid activation function. The other, is composed of an input layer receiving the main network's activations, followed by a fully connected layer of 10 neurons, and another fully connected layer of 5 neurons. The neurons in both fully connected layers are using the RELU activation function. The output layer is composed of a single neuron using the sigmoid activation function. Both types of mapping networks receive as input the main network's activations from the neurons of the 17 residual bottleneck layers. The mapping networks were trained using the optimization algorithm Adam [21] with a learning rate of 0.001, the binary crossentropy loss function, and early stopping with a patience of 20. Each mapping network was trained using a balanced set of 350 samples and tested using a balanced set of 100 samples. For each extracted concept, we consider the mapping network which achieved a higher testing accuracy, between the two different architectures considered. Among these, we use a testing accuracy threshold α of 90% to select which mapping networks should be considered when building the background knowledge.

Since the main network considered in the experiments presented in the chapter, $\mathcal{M}_{GTSRB}$, is trained to classify an image as showing a signal as being one of 4 categories, the ontologies induced for describing the network's classification process should contain 4 axioms. Each axiom is the description of the concept associated with a category of signals, using the concepts extracted by the mapping networks. Hence, we use DL-Learner framework [25] for learning the description of each signal category. DL-Learner is configured with the same configurations as described in Section 5.2, with the exception of the maximum execution time – it is configured to solve a positive-negative learning problem using closed world reasoning with a maximum execution time of 120 seconds. Each axiom of the ontology describing a category of signals is induced using a balanced set of 1000 examples and is tested using a balanced set of 500 examples. Considering that each axiom of the ontology is induced separately, it is sensible to perform an axiom-by-axiom analysis of the induced ontology, instead of just considering its overall quality. So, to quantitatively evaluate the quality of our induced ontologies we use the fidelity measure F_{Main}, similar to the one used in the experiments explored in Chapter 5, on an axiom-by-axiom basis. Thus, F_{Main} computes the ratio of samples where the classifications of the main network for a given signal category coincide with those obtained from the corresponding axiom of the ontology, together with the knowledge obtained from the outputs of the mapping networks.

Each experiment is run 20 times, using different balanced sets of samples for training and testing purposes. The induced ontologies that we present are the mode of the 20 runs, while the discussed fidelity scores correspond to the average of the fidelity scores obtained in the 20 runs.

6.4 Inducing the Ontology

In this section, we test if our method can be used in a setting as close as close as possible to a real-world scenario, for inducing an ontology that describes a neural network's classification process. For this, we apply our method to the main network $\mathcal{M}_{GTSRB}$, using the concepts showed in Figure 6.2 as the concepts of interest. Using the mapping networks for extracting these concepts, we construct the ontology by inducing a description for each concept associated with the main network's outputs: Danger, Priority, Prohibitory, Mandatory. The resulting ontology can be seen below:

$$\begin{aligned}
\mathsf{Danger} &\equiv \mathsf{Boarder} \sqcap \mathsf{Symbol} \sqcap \mathsf{TriangularShape} \sqcap \neg \mathsf{StartProhibition} \\
\mathsf{Priority} &\equiv \mathsf{SymbolStop} \sqcup ((\mathsf{WhiteBoarder} \sqcup \neg \mathsf{CircularShape}) \sqcap \mathsf{Symbol}) \\
\mathsf{Prohibitory} &\equiv \mathsf{CircularShape} \sqcap (\mathsf{StartProhibition} \sqcup \neg \mathsf{WhiteSymbol}) \\
\mathsf{Mandatory} &\equiv \mathsf{CircularShape} \sqcap \mathsf{WhiteSymbol} \sqcap \neg \mathsf{SymboStop}
\end{aligned}$$

Table 6.1: Fidelity scores of the axioms contained in the ontology describing the classification of traffic signals.

	F_{Main}
Danger	$99.63 \pm 0.42\%$
Priority	$97.94 \pm 0.64\%$
Prohibitory	$99.27 \pm 0.50\%$
Mandatory	$98.38 \pm 0.60\%$

The axiom describing danger signals states that a sample is classified as a danger signal if the signal has a boarder, contain a symbol, has triangular shape, and is not indicating the start of a prohibition. The axiom describing priority signals states that a sample is classified as a priority signal if the signal has the symbol *STOP*, or, if it has either a white boarder or does not have a circular shape and it contains some symbol. The axiom describing prohibitory signals states that a sample is classified as a prohibitory signal if the signal has a circular shape, or if it is either a signal indicating the start of a prohibition or if it does not have a white symbol. The axiom describing mandatory signals states that a sample is classified as a mandatory signal if the signal has a circular shape, has a white symbol, and does not have the symbol *STOP*.

When observing the induced ontology, we see that it only contains 8 concepts from the 26 considered for inducing the ontology. Among the concepts that are not contained in the induced ontology, we point out concepts such as DiamondShape and SymbolSpeed, that following our intuition, should be relevant for describing priority and prohibitory signals, respectively. However, we can still draw similarities between the description of the different categories of signals and our intuition on how they should be described, e.g., danger signals must have a triangular shape, having a *STOP* symbol is a defining characteristic of some priority signals, prohibitory signals have a circular shape, and mandatory signals have a circular shape and a white symbol. Moreover, we should keep in mind that our goal is not for the ontology to agree with our intuitions, but instead, it should be a faithful description of the main network's classification process. The ontologies induced showed an overall fidelity F_{Main} of around 98.8%. In Table 6.1, we show the axiom-by-axiom fidelities obtained by the induced ontologies. The slight difference in the fidelity scores obtained for the axioms describing each category of signals, could be due to the difference in the number of signals that are exceptions to the general description of a category of signals, e.g., most danger signals could be unequivocally described as having a triangular shape and a symbol inside. Nevertheless, this difference is small and all the induced axioms show very high fidelity scores. This provides strong evidence that, although the ontology does not describes the signals exactly according to our intuitions, it is a faithful and understandable description of the main network's classification process.

7

Conclusion

In this chapter, we present the main conclusions reached in this dissertations, followed by the presentation of possible avenues for future work.

In recent years, neural networks have grown in popularity, mostly due to their performance and versatility for performing a large variety of tasks. This has led neural network-based methods to become an almost ubiquitous solution in many domains, and among them, domains where they are required to have a certain degree of autonomy and responsibility. The application of neural networks in these critical domains has made evident the need for explanations for the results provided by these systems, whether for increasing trust and allow users to act upon these results, or to help understand when and why these results are wrong. The problem of obtaining explanations for the results of a neural network is tied with the sub-symbolic nature of their internal representations. Neural networks are typically considered *black boxes*, since the size and sub-symbolic nature of their internal representations makes the process of trying to reason upon these representations almost impossible. Hence, it is not viable to *directly* use the internal representations of a neural network for explaining its outputs. A recent work [46], proposes obtaining justifications for the outputs of a neural network, grounded in the network's internal representations, by leveraging an ontology as the background knowledge for producing the justifications. This is accomplished by using small neural networks, called mapping networks, to establish mappings between the activations of the network which outputs are to be explained – the main network – and a set of concepts from the ontology. When an input is fed to the main network, the observations made by the mapping networks regarding the concepts that were identified for that input are used together with the ontology for computing the justifications. Although promising, this work raises a question regarding the unavailability of an ontology to be used as the background for computing the justifications. Moreover, even if an ontology is available there is no guarantee that it properly reflects the internal processes of the neural network.

In this dissertation, we address this issue by developing a method that leverages the knowledge obtained using the mapping networks, for inducing ontologies from neural

networks. Hence, we expect that the induced ontologies are grounded in the neural networks' internal representations, and that they provide a faithful human-understandable description of the networks' internal classification process. We divide our method in 3 main steps:

1. Choosing a language for representing the ontology and the set of concepts of interest that will serve as the building blocks of the ontology. The language should be expressive enough for defining the concepts and relations necessary for describing the classification process of the neural network. The concepts of interest should possess meaning in the domain where the neural network operates and be enough to capture the task being performed by the network;

2. Developing and training mapping networks for extracting the concepts of interest from the main network's activations, so that, when an input is fed to the main network we know both its classification and which concepts were identified when classifying the input, according to the mapping network's observations;

3. Using an Inductive Logic Programming framework for inducing a description for each output class of the main network, based on a set of examples. These examples refer to samples being fed to the main network and are divided into a set of positive and and a set of negative examples according to main network's classification. The background knowledge for inducing the description is given by the observations made by the mapping networks, regarding the identification of concepts of interest when the main network was classifying each sample.

In this dissertation, we started by exploring the application of the proposed method in a controlled setting, using a dataset of synthetic images with an accompanying ontology describing how the dataset was labeled. From the experiments conducted in this setting, we conclude that it is possible to use our method for inducing ontologies that are understandable and faithful to a neural network's classifications. Also, we showed that it is possible to induce faithful ontologies at different levels of abstraction, by selecting the appropriate concepts of interest to induce the ontology. Additionally, we explored the case where the chosen concepts of interest are not adequate or enough for capturing a neural network's classification process, and we conclude that the inadequacy of the chosen concepts is reflected in the fidelity of the induced ontologies. Furthermore, we investigated the feasibility of our method and conclude that in this setting, the main cost is bound with the development of the mapping networks, and since they do not require much data to develop, the cost of our method should be relatively low. In addition to this, we investigated the importance of using the mapping networks and their observations for the inducing the ontologies. We induced ontologies using both the observations of the mapping networks and dataset labels, and we conclude that when using the dataset

labels the induced ontologies are independent of the main network's classification process. Conversely, we conclude that the ontologies induced using the mapping network's observations reflect the internal mechanisms of the neural networks.

We further tested our proposed method by applying it in a setting as close as possible to a real-world application, using a dataset of photographic images and a neural network based on a popular architecture. We conducted experiments in this setting and conclude that are reasonable, understandable, and faithful to the neural network's classification process.

7.1 Future Work

A possible avenue for future work concerns the main limitation of the method we propose, the definition of a set of concepts of interest for inducing the ontologies. As of now, we assume these concepts to be given by domain experts, and despite this being feasible in some domains, in others it can be a complicated task. A possibility to address this is to determin a set of relevant concepts for a neural network's classification task, by looking at its activations and using unsupervised learning methods to find patterns that could be correlated with concepts present in its inputs. An example of a method which takes a similar approach for identifying relavant concepts from neural networks performing image classification is ACE [9].

Another aspect that is worth exploring is related with the choice of which mapping networks to consider when inducing an ontology, i.e., how should one determine if a mapping network is indeed identifying the correct concept. In the experiments we present, we addressed this by establishing a threshold on the testing accuracy of the mapping networks (90%). However, this might not always be feasible. Since the accuracy of the mapping networks could be influenced by other variables, e.g., the number of available training examples, there might be domains where establishing a threshold could lead to the omission of mapping networks, and in turn concepts of interest, that could be useful for inducing the ontology. A possible approach to deal with this, could be to consider the testing accuracy of the mapping networks as a *degree of belief* and attempt to induce fuzzy ontologies with their observations.

In this dissertation, we apply our method to neural networks with testing accuracies close to 99%. For future work it would be interesting to explore the application of our method to a neural network with greater uncertainty, as it could expose the deficiencies in its classification process. Also, it would be interesting to apply our method to a neural network tasked to output a probability distribution of an example belonging to a set of classes and explore if it is possible to induce a probabilistic ontology that is faithful to the neural network's outputs.

Bibliography

[1] M. G. Augasta and T. Kathirvalavakumar. "Reverse Engineering the Neural Networks for Rule Extraction in Classification Problems". In: *Neural Process. Lett.* 35.2 (2012), pp. 131–150. DOI: 10.1007/s11063-011-9207-8. URL: https://doi.org/10.1007/s11063-011-9207-8 (cit. on p. 15).

[2] S. Bach et al. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS one* 10.7 (2015), e0130140 (cit. on p. 14).

[3] D. Bahdanau, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2015. URL: http://arxiv.org/abs/1409.0473 (cit. on p. 1).

[4] R. J. Brachman and H. J. Levesque. "The Tractability of Subsumption in Frame-Based Description Languages". In: *Proceedings of the National Conference on Artificial Intelligence. Austin, TX, USA, August 6-10, 1984*. Ed. by R. J. Brachman. AAAI Press, 1984, pp. 34–37. URL: http://www.aaai.org/Library/AAAI/1984/aaai84-036.php (cit. on p. 9).

[5] T. B. Brown et al. "Language Models are Few-Shot Learners". In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165 (cit. on p. 2).

[6] A. Cropper et al. "Inductive logic programming at 30". In: *Mach. Learn.* 111.1 (2022), pp. 147–172. DOI: 10.1007/s10994-021-06089-1. URL: https://doi.org/10.1007/s10994-021-06089-1 (cit. on pp. 11, 25).

[7] J. Ferreira et al. "Looking Inside the Black-Box: Logic-based Explanations for Neural Networks". In: *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel. July 31 - August 5, 2022*. Ed. by G. Kern-Isberner, G. Lakemeyer, and T. Meyer. 2022. URL: https://proceedings.kr.org/2022/45/ (cit. on p. 4).

[8] Y. Fu et al. "CompFeat: Comprehensive Feature Aggregation for Video Instance Segmentation". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 1361–1369. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16225 (cit. on p. 1).

[9] A. Ghorbani et al. "Towards Automatic Concept-based Explanations". In: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*. Ed. by H. M. Wallach et al. 2019, pp. 9273–9282. URL: https://proceedings.neurips.cc/paper/2019/hash/77d2afcb31f6493e350fca61764efb9a-Abstract.html (cit. on p. 47).

[10] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on p. 9).

[11] A. Graves, A. Mohamed, and G. E. Hinton. "Speech Recognition with Deep Recurrent Neural Networks". In: *CoRR* abs/1303.5778 (2013). arXiv: 1303.5778 (cit. on p. 1).

[12] R. Guidotti et al. "A Survey of Methods for Explaining Black Box Models". In: *ACM Comput. Surv.* 51.5 (2019), 93:1–93:42. DOI: 10.1145/3236009. URL: https://doi.org/10.1145/3236009 (cit. on p. 2).

[13] T. Hailesilassie. "Rule Extraction Algorithm for Deep Neural Networks: A Review". In: *CoRR* abs/1610.05267 (2016). arXiv: 1610.05267. URL: http://arxiv.org/abs/1610.05267 (cit. on p. 15).

[14] J. Hartigan. *Clustering Algorithms*. Out-of-print Books on demand. Wiley, 1975. ISBN: 9780471356455. URL: https://books.google.pt/books?id=cDnvAAAAMAAJ (cit. on p. 17).

[15] Y. Hua and B. Hein. "Concept Learning in AutomationML with Formal Semantics and Inductive Logic Programming". In: *14th IEEE International Conference on Automation Science and Engineering, CASE 2018, Munich, Germany, August 20-24, 2018*. IEEE, 2018, pp. 1542–1547. DOI: 10.1109/COASE.2018.8560541. URL: https://doi.org/10.1109/COASE.2018.8560541 (cit. on p. 11).

[16] H. H. Institute. *Explaining Artificial Intelligence*. URL: https://lrpserver.hhi.fraunhofer.de/image-classification (cit. on p. 13).

[17] M. Ivanovs, R. Kadikis, and K. Ozols. "Perturbation-based methods for explaining deep neural networks: A survey". In: *Pattern Recognit. Lett.* 150 (2021), pp. 228–234. DOI: 10.1016/j.patrec.2021.06.030. URL: https://doi.org/10.1016/j.patrec.2021.06.030 (cit. on p. 14).

[18] N. Kanagaraj et al. "Deep learning using computer vision in self driving cars for lane and traffic sign detection". In: *Int. J. Syst. Assur. Eng. Manag.* 12.6 (2021), pp. 1011–1025. DOI: 10.1007/s13198-021-01127-6. URL: https://doi.org/10.1007/s13198-021-01127-6 (cit. on p. 1).

[19] A. Karpathy et al. "Large-Scale Video Classification with Convolutional Neural Networks". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE Computer Society, 2014, pp. 1725–1732. DOI: 10.1109/CVPR.2014.223. URL: https://doi.org/10.1109/CVPR.2014.223 (cit. on p. 1).

[20] B. Kim et al. "Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by J. G. Dy and A. Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2673–2682. URL: http://proceedings.mlr.press/v80/kim18d.html (cit. on p. 18).

[21] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Y. Bengio and Y. LeCun. 2015. URL: http://arxiv.org/abs/1412.6980 (cit. on pp. 8, 29, 30, 42).

[22] A. Krizhevsky, G. Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009) (cit. on p. 24).

[23] S. Lai et al. "Recurrent Convolutional Neural Networks for Text Classification". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*. Ed. by B. Bonet and S. Koenig. AAAI Press, 2015, pp. 2267–2273 (cit. on p. 1).

[24] M. Law, A. Russo, and K. Broda. "Inductive Learning of Answer Set Programs". In: *Logics in Artificial Intelligence - 14th European Conference, JELIA 2014, Funchal, Madeira, Portugal, September 24-26, 2014. Proceedings*. Ed. by E. Fermé and J. Leite. Vol. 8761. Lecture Notes in Computer Science. Springer, 2014, pp. 311–325. DOI: 10.1007/978-3-319-11558-0_22. URL: https://doi.org/10.1007/978-3-319-11558-0_22 (cit. on p. 30).

[25] J. Lehmann. "DL-Learner: Learning Concepts in Description Logics". In: *J. Mach. Learn. Res.* 10 (2009), pp. 2639–2642. URL: https://dl.acm.org/citation.cfm?id=1755874 (cit. on pp. 30, 43).

[26] J. Lehmann et al. "Class expression learning for ontology engineering". In: *J. Web Semant.* 9.1 (2011), pp. 71–81. DOI: 10.1016/j.websem.2011.01.001. URL: https://doi.org/10.1016/j.websem.2011.01.001 (cit. on p. 30).

[27] X. Li et al. "An Experimental Study of Quantitative Evaluations on Saliency Methods". In: *KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021*. Ed. by F. Zhu, B. C. Ooi, and C. Miao. ACM, 2021, pp. 3200–3208. DOI: 10.1145/3447548.3467148. URL: https://doi.org/10.1145/3447548.3467148 (cit. on p. 14).

[28] V. Lifschitz. "What Is Answer Set Programming?" In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*. Ed. by D. Fox and C. P. Gomes. AAAI Press, 2008, pp. 1594–1597. URL: http://www.aaai.org/Library/AAAI/2008/aaai08-270.php (cit. on p. 22).

[29] Y. Lin, T. Chen, and L. Yu. "Using Machine Learning to Assist Crime Prevention". In: *6th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2017, Hamamatsu, Japan, July 9-13, 2017*. IEEE Computer Society, 2017, pp. 1029–1030. DOI: 10.1109/IIAI-AAI.2017.46. URL: https://doi.org/10.1109/IIAI-AAI.2017.46 (cit. on p. 1).

[30] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. ii).

[31] A. Mehbodniya et al. "Financial Fraud Detection in Healthcare Using Machine Learning and Deep Learning Techniques". In: *Secur. Commun. Networks* 2021 (2021), 9293877:1–9293877:8. DOI: 10.1155/2021/9293877. URL: https://doi.org/10.1155/2021/9293877 (cit. on p. 1).

[32] W. Pieters. "Explanation and trust: what to tell the user in security and AI?." In: *Ethics and Information Technology* 13.1 (2011), pp. 53–64. ISSN: 1572-8439. DOI: 10.1007/s10676-010-9253-3. URL: https://doi.org/10.1007/s10676-010-9253-3 (cit. on p. 1).

[33] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993. ISBN: 1-55860-238-0 (cit. on p. 17).

[34] S. Rebuffi et al. "There and Back Again: Revisiting Backpropagation Saliency Methods". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 8836–8845. DOI: 10.1109/CVPR42600.2020.00886. URL: https://openaccess.thecvf.com/content_CVPR_2020/html/Rebuffi_There_and_Back_Again_Revisiting_Backpropagation_Saliency_Methods_CVPR_2020_paper.html (cit. on p. 14).

[35] S. Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.6 (2017), pp. 1137–1149. DOI: 10.1109/TPAMI.2016.2577031. URL: https://doi.org/10.1109/TPAMI.2016.2577031 (cit. on p. 1).

[36] M. A. Rezaei et al. "Deep Learning in Drug Design: Protein-Ligand Binding Affinity Prediction". In: *IEEE ACM Trans. Comput. Biol. Bioinform.* 19.1 (2022), pp. 407–417. DOI: 10.1109/TCBB.2020.3046945. URL: https://doi.org/10.1109/TCBB.2020.3046945 (cit. on p. 1).

[37] M. T. Ribeiro, S. Singh, and C. Guestrin. ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. Ed. by B. Krishnapuram et al. ACM, 2016, pp. 1135–1144. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778 (cit. on p. 14).

[38] O. Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y (cit. on p. 42).

[39] M. Sandler et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474. URL: http://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html (cit. on p. 42).

[40] M. K. Sarker et al. "Explaining Trained Neural Networks with Semantic Web Technologies: First Steps". In: *Proceedings of the Twelfth International Workshop on Neural-Symbolic Learning and Reasoning, NeSy 2017, London, UK, July 17-18, 2017*. Ed. by T. R. Besold, A. S. d'Avila Garcez, and I. Noble. Vol. 2003. CEUR Workshop Proceedings. CEUR-WS.org, 2017. URL: http://ceur-ws.org/Vol-2003/NeSy17_paper4.pdf (cit. on p. 16).

[41] M. Sato and H. Tsukimoto. "Rule extraction from neural networks via decision tree induction". In: *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*. Vol. 3. 2001, 1870–1875 vol.3. DOI: 10.1109/IJCNN.2001.938448 (cit. on p. 16).

[42] G. P. J. Schmitz, C. Aldrich, and F. S. Gouws. "ANN-DT: an algorithm for extraction of decision trees from artificial neural networks". In: *IEEE Trans. Neural Networks* 10.6 (1999), pp. 1392–1401. DOI: 10.1109/72.809084. URL: https://doi.org/10.1109/72.809084 (cit. on p. 16).

[43] D. Silver et al. "Mastering the game of Go without human knowledge". In: *Nat.* 550.7676 (2017), pp. 354–359. DOI: 10.1038/nature24270. URL: https://doi.org/10.1038/nature24270 (cit. on p. 1).

[44] M. de Sousa Ribeiro. "Neural and Symbolic AI - mind the gap! Aligning Artificial Neural Networks and Ontologies". In: (2021). URL: http://hdl.handle.net/10362/113651 (cit. on pp. 41, 42).

[45] M. de Sousa Ribeiro, L. Krippahl, and J. Leite. "Explainable Abstract Trains Dataset". In: *CoRR* abs/2012.12115 (2020). arXiv: 2012.12115. URL: https://arxiv.org/abs/2012.12115 (cit. on p. 27).

[46] M. de Sousa Ribeiro and J. Leite. "Aligning Artificial Neural Networks and Ontologies towards Explainable AI". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 4932–4940. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16626 (cit. on pp. 2, 3, 19, 21, 31, 36, 45).

[47] J. Stallkamp et al. "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition". In: *Neural Networks* 0 (2012), pp. –. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2012.02.016. URL: http://www.sciencedirect.com/science/article/pii/S0893608012000457 (cit. on p. 40).

[48] F. Sultana, A. Sufian, and P. Dutta. "Advancements in Image Classification using Convolutional Neural Network". In: *CoRR* abs/1905.03288 (2019). arXiv: 1905.03288 (cit. on p. 1).

[49] W. Sun, B. Zheng, and W. Qian. "Computer aided lung cancer diagnosis with deep learning algorithms". In: *Medical Imaging 2016: Computer-Aided Diagnosis, San Diego, California, United States, 27 February - 3 March 2016*. Ed. by G. D. Tourassi and S. G. A. III. Vol. 9785. SPIE Proceedings. SPIE, 2016, 97850Z. DOI: 10.1117/12.2216307. URL: https://doi.org/10.1117/12.2216307 (cit. on p. 1).

[50] M. Sundararajan, A. Taly, and Q. Yan. "Axiomatic Attribution for Deep Networks". In: *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*. Ed. by D. Precup and Y. W. Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 3319–3328. URL: http://proceedings.mlr.press/v70/sundararajan17a.html (cit. on p. 14).

[51] *The Description Logic Handbook: Theory, Implementation and Applications*. 2nd ed. Cambridge University Press, 2007. DOI: 10.1017/CBO9780511711787 (cit. on pp. 11, 22).

[52] G. G. Towell and J. W. Shavlik. "Extracting Refined Rules from Knowledge-Based Neural Networks". In: *Mach. Learn.* 13 (1993), pp. 71–101. DOI: 10.1007/BF00993103. URL: https://doi.org/10.1007/BF00993103 (cit. on p. 17).

[53] P. Tschandl. *The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions*. Version V3. 2018. DOI: 10.7910/DVN/DBW86T. URL: https://doi.org/10.7910/DVN/DBW86T (cit. on p. 24).

[54] M. Wang and W. Deng. "Deep face recognition: A survey". In: *Neurocomputing* 429 (2021), pp. 215–244. DOI: 10.1016/j.neucom.2020.10.081. URL: https://doi.org/10.1016/j.neucom.2020.10.081 (cit. on p. 1).

[55] Z. Wang et al. "Interpreting Interpretations: Organizing Attribution Methods by Criteria". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR Workshops 2020, Seattle, WA, USA, June 14-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 48–55. DOI: 10.1109/CVPRW50498.2020.00013. URL: https://openaccess.thecvf.com/content_CVPRW_2020/html/w1/Wang_Interpreting_Interpretations_Organizing_Attribution_Methods_by_Criteria_CVPRW_2020_paper.html (cit. on p. 14).

[56] M. D. Zeiler and R. Fergus. "Visualizing and Understanding Convolutional Networks". In: *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*. Ed. by D. J. Fleet et al. Vol. 8689. Lecture Notes in Computer Science. Springer, 2014, pp. 818–833. DOI: 10.1007/978-3-319-10590-1_53. URL: https://doi.org/10.1007/978-3-319-10590-1_53 (cit. on p. 14).

[57] J. R. Zilke, E. L. Mencía, and F. Janssen. "DeepRED - Rule Extraction from Deep Neural Networks". In: *Discovery Science - 19th International Conference, DS 2016, Bari, Italy, October 19-21, 2016, Proceedings*. Ed. by T. Calders, M. Ceci, and D. Malerba. Vol. 9956. Lecture Notes in Computer Science. 2016, pp. 457–473. DOI: 10.1007/978-3-319-46307-0_29. URL: https://doi.org/10.1007/978-3-319-46307-0_29 (cit. on p. 17).

[58] M. Zimmer et al. "Differentiable Logic Machines". In: *CoRR* abs/2102.11529 (2021). arXiv: 2102.11529. URL: https://arxiv.org/abs/2102.11529 (cit. on p. 30).