Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning
Special Session on KR and Machine Learning

Looking Inside the Black-Box: Logic-based Explanations for Neural Networks

João Ferreira , Manuel de Sousa Ribeiro , Ricardo Gonçalves , João Leite
NOVA LINCS, NOVA University Lisbon, Portugal
{jmdi.ferreira, mad.ribeiro}@campus.fct.unl.pt, {rjrg, jleite}@fct.unl.pt

Abstract

Deep neural network-based methods have recently enjoyed great popularity due to their effectiveness in solving difficult tasks. Requiring minimal human effort, they have turned into an almost ubiquitous solution in multiple domains. However, due to the size and complexity of typical neural network models' architectures, as well as the sub-symbolical nature of the representations generated by their neuronal activations, neural networks are essentially opaque, making it nearly impossible to explain to humans the reasoning behind their decisions. We address this issue by developing a procedure to induce human-understandable logic-based theories that attempt to represent the classification process of a given neural network model, based on the idea of establishing mappings from the values of the activations produced by the neurons of that model to human-defined concepts to be used in the induced logic-based theory. Exploring the setting of a synthetic image classification task, we provide empirical results to assess the quality of the developed theories for different neural network models, compare them to existing theories on that task, and give evidence that the theories developed through our method are faithful to the representations learned by the neural networks that they are built to describe.

1 Introduction

In this paper, we investigate how to induce logic-based theories that reflect a given neural network's classification process. Our results suggest that this is achievable by using an inductive reasoning framework and relating the activations of a neural network with human-defined concepts, without the need to retrain or modify the neural network, while requiring few labeled data.

Artificial neural networks have allowed for multiple breakthroughs in different challenging areas in the last few years, being capable of matching or surpassing state-of-the-art approaches in real-time object detection (Ren et al. 2017), video classification (Karpathy et al. 2014) and segmentation (Fu et al. 2021), translation (Bahdanau, Cho, and Bengio 2015), even outperforming humans at facial recognition (Wang and Deng 2021), playing strategic games (Silver et al. 2017), just to name a few. For this reason, they have gained popularity, and are now being applied to solve the most varied tasks in multiple different domains, e.g., in medical diagnosis, (Sun, Zheng, and Qian 2016), self-driving vehicles (Kanagaraj et al. 2021), crime prevention (Lin, Chen, and Yu 2017) and fraud detection (Benchaji et al. 2021).

Despite all the successes and popularity achieved by artificial neural network-based methods, neural network models are typically viewed as black boxes (Guidotti et al. 2019), i.e., their input-output behavior is typically considered of highest importance, while the inner representations learned by these models are largely disregarded.

On the one hand, the opaqueness of neural network models might be attributed to their typical size and complexity, having potentially many billions of parameters spread across several dozens of layers performing non-linear transformations (Brown et al. 2020). On the other hand, these models are subsymbolic in nature, meaning that their internal representations are generally based on a high-dimensional Euclidean space, i.e., real-valued tensors, which do not possess an associated declarative meaning (Hitzler et al. 2020). In practice, when a neural network is used to make a prediction, there is no human-interpretable indication regarding why a particular output was produced.

This shortfall led to the search for methods for explaining how to these models are achieving their results, ultimately contributing to the rise of the field of explainable AI (Ignatiev 2020), which we briefly review in Section 6.

There are many different ways one can look at explanations (Miller 2019), but they are usually taken to be answers to (often interactive sequences of) "why" questions, whose selection and evaluation requires a) a language containing human-understandable concepts and meaningful relations between those concepts, b) some knowledge about the domain, and c) one or more forms of reasoning e.g., deductive, abductive, counterfactual, etc.

One recent proposal (de Sousa Ribeiro and Leite 2021), takes a step towards providing *richer* explanations for the outputs of neural network models by exploring the idea of establishing mappings from the values of the activations produced by the neurons of a neural network model, to human-defined concepts from a selected existing logic-based theory. These mappings are established by what is referred in (de Sousa Ribeiro and Leite 2021) as mapping networks – small neural networks built to predict a single human-defined concept from the activations of a given neural network model. When input is fed to a particular *main network* – the neural network model whose outputs we want to justify – it is possible to use these mapping networks and observe whether

the concepts they were trained to predict were identified in the main network's activations, thus acquiring additional knowledge about how the main network has processed its input. The justifications are then obtained using logic-based reasoning methods over the selected logic-based theory, together with the observations made for a given input regarding each mapped concept. They are minimal sets of axioms from the selected logic-based theory that, together with the mapping networks' observations, entail the output of the main network. They are symbolic, human-understandable, since they are expressed in the language of the chosen logic-based theory, and we can reason about them. While the selected logic-based theory provides the necessary language and background knowledge to adequately convey justifications for the main networks' output, the established mappings provide additional knowledge about how the main network has processed its input, allowing for an interpretation of its internal representations.

However, the fact that the justifications discussed in (de Sousa Ribeiro and Leite 2021) depend not only on the mapping networks' observations of the input but also on the selected existing logic-based theory, raise two important questions. First, because of the dependency on the selected logic-based theory, one cannot really take them to be exact representations of how a neural network reaches its output, but rather plausible and understandable rationalisations of how it might have arrived at its results, based on the selected logic-based theory. The axioms of the theory that are included in the justification might not be warranted by how the neural network is achieving its results. Second, a logic-based theory might not be available, either because our main network solves a task in a domain where there is little background knowledge available, or no one has yet encoded it as a logic-based theory.

What if we do not have a logic-based theory, or we do not want to rely on existing ones because we suspect that they are in some way incorrect or do not properly reflect how our main network might be achieving its results?

To address this issue, in this paper, instead of using the output of the mapping networks as knowledge about how the main network has processed its input to be used together with an existing logic-based theory, we explore its use to *generate* a logic-based theory. We use the mapping networks to provide additional labels to a data set which we then use to induce a logic-based theory. By choosing an appropriate set of human-defined concepts to build the mapping networks, the generated logic-based theory will provide explanations in a language with the required level of abstraction. By relying on the mapping networks, hence on the internal representations of the main neural network, the generated logic-based theory will better reflect the classification process of that neural network model. Hence, following the method described in (de Sousa Ribeiro and Leite 2021), it can be used for the generation of justifications that are more than just rationalisations – they are closer to providing a global description of how a given neural network is achieving its classifications, based on human-defined concepts.

In this paper, after presenting a formal account of neural networks in Section 2, we proceed, in Section 3, with a formalization of the process of extracting concepts from a neural network using mapping networks, only informally described in (de Sousa Ribeiro and Leite 2021). In Section 4, we describe our method to generate the logic-based theories. Then, in Section 5, we describe a series of experiments conducted to empirically test the proposed method, including evaluating the quality of the generated theories; how the generated theories depend on the concepts used to construct the mapping networks, regarding both their level of abstraction and their adequacy; the cost of the method; and whether the generated theories are indeed a good reflection of the classification process of the neural network model. In Section 6 we discuss related work, and conclude in Section 7.

2 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 $L_i(\mathbf{X}; \boldsymbol{\theta}) = L(\mathbf{X}; \boldsymbol{\theta})_i$.

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 $concept_N(\mathbf{Y}, \mathsf{C}) = 1$ indicates that concept C can be identified in output $\mathbf{Y}$.

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, $F^j_{\mathcal{M}}(\mathbf{X}) = L^j(L^{j-1}(\ldots L^1(\mathbf{X}; \boldsymbol{\theta}_1) \ldots; \boldsymbol{\theta}_{j-1}); \boldsymbol{\theta}_j)$. 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 $(N, \overline{\boldsymbol{\theta}})$, the *training process* consists in finding a new model $(N, \overline{\boldsymbol{\theta}}')$, 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.

3 Extracting Concepts from Neural Networks

In this section we formalize the method proposed in (de Sousa Ribeiro and Leite 2021) to extract human-understandable concepts from the activation patterns of a trained neural network, usually called main network.

We assume the main network model $\mathcal{M} = (N, \overline{\boldsymbol{\theta}})$ 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 $\mathcal{M} = (N, \overline{\boldsymbol{\theta}})$ 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 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 $\mathcal{M}_M = (M, \overline{\boldsymbol{\theta}}_M)$ 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 $concept_M(F_{\mathcal{M}_M}(vec(F^1_{\mathcal{M}}(\mathbf{X}) \cdot \mathbf{S}^1, \ldots, F^d_{\mathcal{M}}(\mathbf{X}) \cdot \mathbf{S}^d)))$, 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 Inducing Logical Theories from Neural Networks

In this section we describe our method to induce human-understandable logic-based theories that aim to represent the

[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.

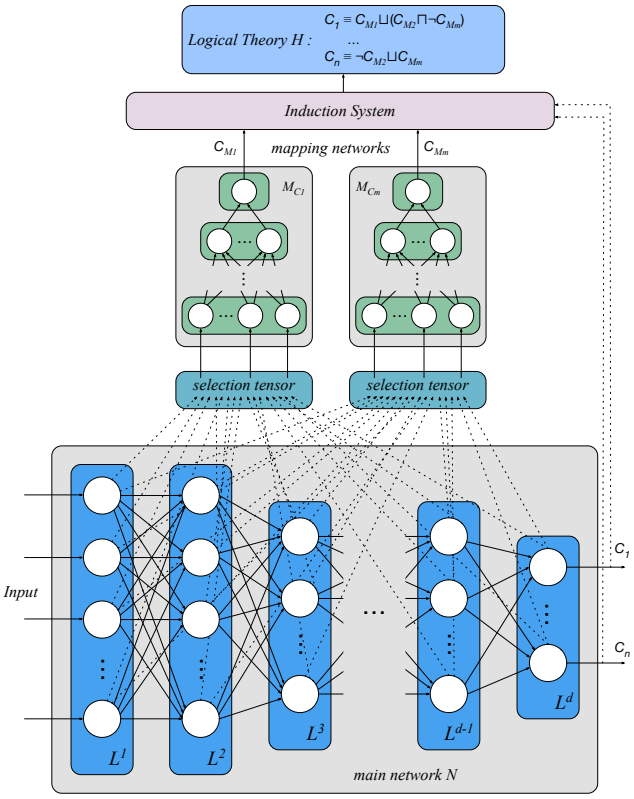

Figure 1: Architecture.

classification process of a neural network model, based on the output of mapping networks, illustrated in Figure 1. We assume a fixed trained main network model $\mathcal{M} = (N, \overline{\boldsymbol{\theta}})$. To represent the classification process of $\mathcal{M}$ and the domain of interest, we need a logical language $\mathcal{L}$ that is rich enough to allow for 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 (Baader et al. 2003), Answer-Set Programming (Brewka, Eiter, and Truszczynski 2011), 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 $\mathcal{C}_N$ and a set of concepts of interest $\mathcal{C}$, which are concepts relevant to the domain of the main network N that would normally be obtained from domain experts, and be dependent on the end-users' preferences or level of expertise (e.g., a health practitioner would probably require a set of concepts different than those required by a patient). For example, for a neural network trained on the CIFAR-10 dataset (Krizhevsky and Hinton 2009) we could use concepts such as 'wheels', 'horns', or 'water', while for one trained on the HAM10000 dataset (Tschandl, Rosendahl, and Kittler 2018), we could use concepts such as 'asymmetric', or 'patchy'. We also assume a fixed set of constants $c_N = \{c_{\mathbf{X}} : \mathbf{X} \in \mathbb{I}_N\}$, 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 $\mathsf{C}(c) \in Pos$, and $BK \cup H \not\models \mathsf{C}(c)$ 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 characterizing 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 $\mathbf{X} \in \mathbb{X}$, we check using the main network model, if the concepts of $\mathcal{C}_N$ are identified or not in $\mathbf{X}$, adding this information to the positive or negative examples, accordingly. Formally, for each $\mathsf{C} \in \mathcal{C}_N$, we define: $Pos^{\mathsf{C}} = \{\mathsf{C}(c_{\mathbf{X}}) : \mathbf{X} \in \mathbb{X} \text{ and } concept_N(F_{\mathcal{M}}(\mathbf{X}), \mathsf{C}) = 1\}$; $Neg^{\mathsf{C}} = \{\mathsf{C}(c_{\mathbf{X}}) : \mathbf{X} \in \mathbb{X} \text{ and } concept_N(F_{\mathcal{M}}(\mathbf{X}), \mathsf{C}) = 0\}$. 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 Sect. 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 $BK = \{\mathsf{C}(c_{\mathbf{X}}) : \mathbf{X} \in \mathbb{X} \text{ and } \mathsf{C} \in \mathcal{C}_\alpha \text{ and } label_{\mathcal{M}_{\mathsf{C}}}(\mathbf{X}) = 1\}$. 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 paper.

Given the sets Pos, Neg and BK defined above, the logic-based theory 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.

5 Empirical Evaluation

In this section, we empirically evaluate our proposed method using the XTRAINS Dataset (de Sousa Ribeiro, Krippahl, and Leite 2020). After introducing the dataset in Section 5.1 and the experimental setting in Section 5.2, we discuss the

Figure 2: Sample images from the XTRAINS dataset.

direct application of our method to this dataset in Section 5.3, followed by a set of experiments to highlight different aspects of our proposal: –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 a theory 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 a theory for the classification process of a neural network model.

5.1 Dataset and Main Networks

The XTRAINS is an image dataset containing representations of trains, such as those in Figure 2. The images are labeled according to their visual features, which correspond to human-understandable concepts. Having an accompanying ontology, this dataset provides a controlled environment for the experiments. The use of images is challenging since human-understandable concepts cannot be directly extracted from pixel representations.

The dataset images can be seen as $152 \times 152 \times 3$ tensors, and are highly diverse. The trains 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, but also in the distance between each wagon, the thickness of the wagons' walls or the height of the trains' couplers. Also, noise is present in the form of missing pixels from the trains' representations.

The ontology accompanying the dataset is represented using Description Logics, which are a family of logic based knowledge representation languages, consisting of decidable fragments of First-Order Logic.[2] Figure 3 presents a subset of this ontology, which agrees with the labels of the XTRAINS dataset. The ontology represents how different concepts are related, e.g., a train has a wagon or 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 our experiments, we consider three different main network models – referred to as $\mathcal{M}_{\mathsf{A}}$, $\mathcal{M}_{\mathsf{B}}$, $\mathcal{M}_{\mathsf{C}}$. Each of these models was trained to identify, respectively, trains with the following descriptions, each illustrated in Figure 2, and described in the accompanying ontology, cf. Figure 3:

- 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;
- 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;
- 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.

Each of the three main networks[3] was trained using a set of 25 000 images and achieved an accuracy of about 99% on a test set of 10 000 images. The three main networks possess different architectures, but all start with a set of convolutional, batch normalization, pooling, and dropout layers, followed by a varying number of fully connected and batch normalization layers, with a single output unit at the end.

5.2 Experimental Setting

The mapping networks used in the following experiments all have the same architecture, being composed solely by an input layer, receiving the main network's activations, and an output layer with a single neuron using the sigmoid activation function. Similarly to the main networks, the mapping networks were trained using the optimization algorithm Adam (Kingma and Ba 2015), with a learning rate of 0.001, the binary cross entropy as loss function, and early stopping with a patience value of 20 (30 for main networks). Each mapping network was trained using a balanced set of 800 samples and tested using 1 000 samples, with the exception of those in Section 5.6, where we discuss the effects of varying the amount of available training samples.

The accuracy threshold α, used to select which mapping networks should be considered when building the background knowledge, is set to be $\alpha = 90\%$.

We choose ontologies over Description Logics as our logical language, a standard choice for taxonomic classification, which also allow us to compare our induced theories with the accompanying ontology of the dataset. As for the induction system, we use DL-Learner framework (Lehmann 2009), which supports supervised ontology induction. Other alternatives for learning logic-based theories could have been used, such as ILASP (Law, Russo, and Broda 2014) or even differentiable logic machines (Zimmer et al. 2021). We use the CELOE algorithm (Lehmann et al. 2011), which is biased to minimize the induced theory, and consider a positive-negative learning problem using closed world reasoning with a maximum execution time of 60 seconds. In all experiments, the theories are induced using a set of 3 000 samples, and tested with a set of 1 000 samples.

The main metric to quantitatively evaluate the quality of our induced theories is the fidelity measure (F_{Main}), which computes how well a given theory *imitates* the classifications of the main network model 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 theory together with the knowledge obtained from the outputs of the mapping networks.

[2] We assume basic familiarity with Description Logics (see (Baader et al. 2003)).

[3] The networks are available at https://bit.ly/XTrainsModels

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

Figure 3: A subset of the XTRAINS dataset's ontology.

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 theories 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 mapped 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 theories logically compare with the existing XTRAINS' ontology. However, whereas 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 was run 20 times, using different balanced sets of samples for training and testing purposes. Throughout the paper, the induced theories that we present and logically analyze are the mode of the 20 runs, while the discussed fidelity scores correspond to the runs' averages.

5.3 Inducing a Main Network's Theory

In this section, we test our main hypothesis, namely that we can leverage on the classifications provided by mapping networks, trained to identify concepts of interest, and use them as basis to induce a human-understandable theory describing how a neural network processes its inputs.

To that end, for each main network, we trained mapping networks to identify the following 11 concepts[4] from the ontology in Figure 3: EmptyTrain, FreightTrain, LongTrain, MixedTrain, PassengerTrain, RuralTrain, WarTrain, $\exists$has.FreightWagon, $\exists$has.LongWagon, $\exists$has.OpenRoof, $\exists$has.ReinforcedCar. In this process, we treat each complex concept, e.g., $\exists$has.FreightWagon, as an atomic concept.

Using our method, we induced the following theories for each of the three main networks:

TypeA $\equiv$ WarTrain $\sqcup$ EmptyTrain
TypeB $\equiv$ (FreightTrain $\sqcap$ LongTrain) $\sqcup$ (PassengerTrain $\sqcap$ $\neg$EmptyTrain)
TypeC $\equiv$ MixedTrain $\sqcup$ RuralTrain

For example, the theory induced for $\mathcal{M}_\mathsf{A}$ 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, whereas the theory induced for $\mathcal{M}_\mathsf{B}$ 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.

It is worth noting that, despite the usage of 11 concepts when inducing each theory, only 3 of those concepts were used on average. 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 theories with the ontology that accompanies the XTRAINS dataset allows us to observe that both the definitions of concepts TypeA and TypeC in the induced theories for $\mathcal{M}_\mathsf{A}$ and $\mathcal{M}_\mathsf{C}$, respectively, are equivalent to the respective definitions of TypeA and TypeC in the datasets' ontology. The concept of TypeB in the induced theory for $\mathcal{M}_\mathsf{B}$ is a subclass of TypeB as defined in the XTRAINS ontology, meaning that any individual of TypeB in the induced theory is also considered to be of TypeB according to the XTRAINS ontology.

Table 1 shows the fidelity scores of the induced theories. The resulting high F_{Main} scores provide evidence that the resulting theories are properly reflecting the main network's internal classification process, strongly supporting our main hypothesis. Furthermore, one can observe that the induced theories 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.

	F_{Main}	$F_{XTrains}$
$\mathcal{M}_\mathsf{A}$	$99.72 \pm 0.18\%$	$99.92 \pm 0.35\%$
$\mathcal{M}_\mathsf{B}$	$98.71 \pm 0.31\%$	$99.83 \pm 0.76\%$
$\mathcal{M}_\mathsf{C}$	$99.33 \pm 0.32\%$	$99.52 \pm 1.52\%$

Table 1: Fidelity scores for the theories of each main network.

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, we may want to

[4]We selected the same 11 concepts that were explored in (de Sousa Ribeiro and Leite 2021)

understand how a neural network is classifying some type of train based on other known 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.

To investigate how our method performs when faced with concepts at different levels of abstraction, we crafted the following sets of concepts:

Train-level:

$\{\mathsf{EmptyTrain}, \mathsf{LongFreightTrain}, \mathsf{MixedTrain}, \mathsf{PassengerTrain}, \mathsf{RuralTrain}, \mathsf{WarTrain}\}$

Wagon-Level:

$\{\exists \mathsf{has.EmptyWagon}, \exists \mathsf{has.FreightWagon}, \exists \mathsf{has.LongWagon}, \exists \mathsf{has.}(\mathsf{LongWagon} \sqcap \mathsf{PassengerCar}) \exists \mathsf{has.PassengerCar}, \exists \mathsf{has.ReinforcedCar}, \geq 2\mathsf{has.FreightWagon}, \geq 2\mathsf{has.LongWagon}, \geq 2\mathsf{has.PassengerCar}, \geq 3\mathsf{has.Wagon}\}$

The application of our method to the three main networks, using the train-level concepts, resulted in the theories:

$\mathsf{TypeA} \equiv \mathsf{EmptyTrain} \sqcup \mathsf{WarTrain}$

$\mathsf{TypeB} \equiv \mathsf{LongFreightTrain} \sqcup \mathsf{PassengerTrain}$

$\mathsf{TypeC} \equiv \mathsf{MixedTrain} \sqcup \mathsf{RuralTrain}$

While with the wagon-level concepts we obtained:

$\mathsf{TypeA} \equiv (\exists \mathsf{has.PassengerCar} \sqcup \neg\exists \mathsf{has.FreightWagon}) \sqcap (\exists \mathsf{has.ReinforcedCar} \sqcup \neg\exists \mathsf{has.PassengerCar})$

$\mathsf{TypeB} \equiv \exists \mathsf{has.}(\mathsf{LongWagon} \sqcap \mathsf{PassengerCar}) \sqcup (\geq 3\mathsf{has.Wagon} \sqcap \geq 2\mathsf{has.FreightWagon})$

$\mathsf{TypeC} \equiv \exists \mathsf{has.EmptyWagon} \sqcap (\neg\exists \mathsf{has.LongWagon} \sqcup (\geq 3\mathsf{has.Wagon} \sqcap \exists \mathsf{has.PassengerCar}))$

Comparing the induced theories with the dataset's ontology, the induced definitions for the train-level theories of TypeA, TypeB, and TypeC are all logically equivalent to the ones in the dataset's ontology. Turning to the wagon-level induced theories, we can 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 theories, shown in Table 2, we can observe that the train-level theories 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 theories using concepts at different levels of abstraction. The slight difference in fidelity between the results achieved by the train-level and wagon-level induced theories 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 – amplified by the fact that the wagon-level induced theories use more concepts.

5.5 Insufficient Concepts

Having shown that the proposed method is able to induce theories that represent a neural network's classification process, even when using concepts with different levels of abstraction, we now turn our attention to the case where the chosen concepts of interest are somehow insufficient to describe the neural networks' classification process. This is relevant since, in real-life, it may occur that we are unable to determine a set of concepts that is adequate, or sufficient, to describe a neural networks' classification process.

		F_{Main}	$F_{XTrains}$
Train-level	$\mathcal{M}_\mathsf{A}$	$99.76 \pm 0.19\%$	$100.00 \pm 0.0\%$
	$\mathcal{M}_\mathsf{B}$	$98.82 \pm 0.39\%$	$100.00 \pm 0.0\%$
	$\mathcal{M}_\mathsf{C}$	$99.46 \pm 0.20\%$	$100.00 \pm 0.0\%$
Wagon-level	$\mathcal{M}_\mathsf{A}$	$94.55 \pm 10.14\%$	$94.44 \pm 11.12\%$
	$\mathcal{M}_\mathsf{B}$	$97.50 \pm 0.52\%$	$96.73 \pm 1.18\%$
	$\mathcal{M}_\mathsf{C}$	$98.16 \pm 0.36\%$	$99.02 \pm 0.56\%$

Table 2: Fidelity scores of the train- and wagon-level theories.

We expect the fidelity of the resulting theories to reflect the adequacy of the concepts $\mathcal{C}$ being mapped. If the chosen concepts are insufficient to properly describe a main network's $\mathcal{M}$ classification process, then the induced theory would have a diminished F_{Main} fidelity. Otherwise, it would mean that our method could be picking up on spurious correlations to describe $\mathcal{M}$'s classification process.

To test this, we sampled 20 random sets of 5 concepts among all concepts defined in the XTRAINS ontology, and induced theories for each of the three main networks. The average fidelity scores F_{Main} and $F_{XTrains}$ of the resulting theories were, respectively, 72.6% and 71.9%, which are considerably lower than the ones obtained in Sections 5.3 and 5.4. These results back our hypothesis that the inadequacy of the selected concepts negatively reflects in the quality of the resulting theories.

As an example, using the set $\{\mathsf{LongFreightTrain}, \exists \mathsf{has.LongWagon}, \exists \mathsf{has.PassengerCar}, \geq 3\mathsf{has.Wagon}, \geq 2\mathsf{has.PassengerCar}\}$, resulted in the following theories:

$\mathsf{TypeA} \equiv \top$

$\mathsf{TypeB} \equiv \geq 3\mathsf{has.Wagon} \sqcup \mathsf{LongFreightTrain}$

$\mathsf{TypeC} \equiv (\geq 3\mathsf{has.Wagon} \sqcap \exists \mathsf{has.PassengerCar}) \sqcup (\neg(\geq 2\mathsf{has.PassengerCar}) \sqcap \neg\exists \mathsf{has.LongWagon})$

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 or superclasses. The poor quality of these theories is reflected in their fidelity scores, shown in Table 3. The high fidelity of the theory obtained for $\mathcal{M}_\mathsf{B}$ can be explained by the use of concepts $\geq 3\mathsf{has.Wagon}$ and $\mathsf{LongFreightTrain}$, which, based on the experiments of Sections 5.3 and 5.4, are adequate to describe the classification process of $\mathcal{M}_\mathsf{B}$.

5.6 Theory 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 mapped concepts are sufficient to describe a neural network's classification process. However, in order to verify the method's feasibility, its cost needs to be assessed.

	F_{Main}	$F_{XTrains}$
$\mathcal{M}_{\mathsf{A}}$	$50.16 \pm 0.26\%$	$50.00 \pm 0.00\%$
$\mathcal{M}_{\mathsf{B}}$	$92.53 \pm 2.75\%$	$92.70 \pm 1.43\%$
$\mathcal{M}_{\mathsf{C}}$	$76.78 \pm 1.99\%$	$76.99 \pm 0.94\%$

Table 3: Fidelity scores of the theories with randomly selected concepts.

Figure 4: Sample images of the designed classification problem.

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 $\mathcal{C}$ to be mapped. However, this data can be repurposed to induce the theory, by relabeling it according to the mapping networks' classifications.

Hence, if the quality of our resulting theories 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 (de Sousa Ribeiro and Leite 2021), the cost of our method would be relatively low, depending mostly on the amount of concepts $\mathcal{C}$ to be mapped.

To test whether the quality of the resulting theories mostly depends on the accuracy of the mapping networks, for each of the three main networks used throughout this paper, we induced theories using the 11 concepts selected in Section 5.3, while varying the amount of data used for their training between 50 and 1200 samples. The amount of data used to induce the theories remained constant at 3000 samples. A Pearson's correlation test on fidelity F_{Main} and the average accuracy of the mapping networks for $\mathcal{C}_{\alpha}$, 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 theories 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 theories, the resulting theories were typically quite accurate, with an average fidelity F_{Main} of about 96%.

5.7 Importance of the Mappings

The goal of our proposed method is to develop a human-understandable theory representing the classification process of a given neural network model. Given that our theories are induced based on the results of mapping networks, which in turn were trained based on labeled data, one might wonder why they should bother with using mapping networks, instead of inducing the theories directly from the labels used to train the mapping networks. We hypothesize that theories induced directly from 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 theories induced through our method, being reliant on the internal representations of the model obtained through the mapping networks, will 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 4. A traffic signal is said to be *on* if it contains any symbol inside it, 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: $\mathsf{TopLeftOn}$, $\mathsf{TopRightOn}$, $\mathsf{BottomLeftOn}$, $\mathsf{BottomRightOn}$, 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. A neural network is then tasked with identifying whether some traffic signal is on in a given image. Notice that, 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 with 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), the same theory was always obtained: $\{\mathsf{On} \equiv \mathsf{BottomLeftOn} \sqcup \mathsf{BottomRightOn}\}$.[5] This observation 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 networks classification process.

However, if we apply our proposed method to the same 50 main networks, which considers the outputs of the mapping networks, we observe diverse induced theories. 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 theory $\{\mathsf{On} \equiv \mathsf{TopLeftOn} \sqcup \mathsf{TopRightOn}\}$, while 42% *looked at* the two bottom signals, resulting in the induced theory $\{\mathsf{On} \equiv \mathsf{BottomLeftOn} \sqcup \mathsf{BottomRightOn}\}$. This indicates that different main networks learned to classify their inputs differently, which was captured by the induced theories using the outputs of the mapping networks, but not when only the

[5] Despite the existence of two minimal theories 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 – the induced theory depends on the order in which the concepts were presented.

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 Related Work

The success of artificial neural network-based methods, along with their characteristic opaqueness, led to the development of many methods with the goal of increasing the interpretability of artificial neural network models. One of the most popular approaches being saliency and attribution methods (Li et al. 2021; Sundararajan, Taly, and Yan 2017; Wang et al. 2020), where the explanation for a neural network's behavior is given in terms of the contribution of each input feature for a given prediction. This is the case of gradient-based methods (Ancona et al. 2018), which compute the gradient of the output with respect to the input to approximate the input features' contributions, and backpropagation-based methods (Montavon et al. 2019; Rebuffi et al. 2020), where a set of propagation rules is used to propagate the output backwards, in order to compute the relevancy of each input feature. There also exist saliency and attribution methods based on perturbation (Zeiler and Fergus 2014; Ivanovs, Kadikis, and Ozols 2021), where the input features' contributions are estimated by measuring how the output changes when masking different parts of the input, or based on abduction (Ignatiev, Narodytska, and Marques-Silva 2019), where inputs are selected based on an encoding of the model as a set of constrains.

Other methods, known as proxy-based methods (Gilpin et al. 2018), attempt to substitute the model being interpreted for another one that has a similar behavior and is inherently interpretable. One popular proxy-based method is LIME (Ribeiro, Singh, and Guestrin 2016), where local linear models are used as a simplified proxy for the full model. Automatic rule extraction algorithms (Guidotti et al. 2019) are a large subgroup of proxy-based methods, where different approaches can be found. Pedagogical rule extraction algorithms (Augasta and Kathirvalavakumar 2012; Schmitz, Aldrich, and Gouws 1999) treat the neural network as a black-box and attempt to produce rules based only on its input-output behavior, decompositional rule extraction algorithms (Zilke, Mencía, and Janssen 2016) consider the inner structure of a neural network model to generate its rules, and eclectic rule extraction algorithms (Towell and Shavlik 1993) apply both elements of pedagogical and decompositional rule extraction algorithms. Recently, multilayer perceptrons were mapped into quantitative bipolar argumentation frameworks (Potyka 2021), although the meaning of the resulting arguments, and the added-value, is yet unclear.

While the above mentioned methods increase the interpretability of neural network models in some way, most only do so in terms of the inputs of the model being interpreted, with explanations consisting only of sets of input features and their corresponding contribution values. However, user studies (Adebayo et al. 2020; Chu, Roy, and Andreas 2020; Shen and Huang 2020) have shown that such explanations typically end up being ignored or unhelpful to end users. We attribute this to the fact that this kind of explanations do not explicitly present any kind of clarification regarding the underlying phenomena that lead to the production of a particular output from a given input to the model, leaving to the user the burden of understanding why the explanation, e.g., a particular set of input features and contribution values, justifies the output of a neural network. Works such has TCAV (Kim et al. 2018) use a notion of "human-interpretable concepts", but no account is given regarding how these concepts relate to the output of the neural network.

The field of neuro-symbolic AI (Besold et al. 2017) and inductive logic-based explainable AI (Schmid 2018) have also contributed to the increase of the interpretability of subsymbolic models. Works such as LIME-FOLD (Shakerin and Gupta 2019) and SHAP-FOIL (Shakerin and Gupta 2020) induce a theory for explaining a model's input-output behavior, by leveraging local explanation methods to determine input feature importance, while (Sarker et al. 2017) attempts to induce such a theory based on labels describing a model's inputs and outputs. (D'Asaro et al. 2020) employs a similar approach, but addresses preference learning tasks, using ILASP (Law, Russo, and Broda 2014) and its ability to learn weak constraints to explain a model. However, given that these methods are pedagogical rule-extraction methods, they are limited to only describing a model's input-output behavior, and thus might fail to be truthful to a neural networks' internal representations.

7 Conclusions

In this paper, we proposed a method for inducing logic-based theories that represent the classification process of a neural network model, providing a human-understandable description of how a neural network is achieving its results based on human-defined concepts.

We provide a formalization of our method along with an experimental evaluation. We were able to show that the method is capable of inducing theories that are faithful to a neural network's classifications. The method has shown to be able to deal with unnecessary concepts, and select the ones that are adequate, depending on the neural network model to which it is being applied. Moreover, it was able to induce theories at different levels of abstraction, and shown to be applicable even when few labeled data is available. Our results indicate that it is indeed possible to obtain logic-based theories that reflect the internal classification process of a given neural network, by leveraging the knowledge hidden in its internal representations to extract symbolic human-defined concepts. This allows us to better understand how a neural network model is performing its classifications, at an adequate level of abstraction, and thus better interpret that model.

Regarding future work, an interesting avenue of research is to explore how to induce probabilistic theories based on the accuracy of the mapping networks, in order to obtain theories that better represent the internal classification process of neural network models.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments, and acknowledge the support provided by Calouste Gulbenkian Foundation through its "New Talents in AI" program, and by FCT through PhD grant (UI/BD/151266/2021), project FORGET (PTDC/CCI-INF/32219/2017), project RIVER (PTDC/CCI-COM/30952/2017) and strategic project NOVA LINCS (UIDB/04516/2020).

References

Adebayo, J.; Muelly, M.; Liccardi, I.; and Kim, B. 2020. Debugging tests for model explanations. In *Procs. of NeurIPS'20*.

Ancona, M.; Ceolini, E.; Öztireli, C.; and Gross, M. 2018. Towards better understanding of gradient-based attribution methods for deep neural networks. In *Procs. of ICLR'18*. OpenReview.net.

Augasta, M. G., and Kathirvalavakumar, T. 2012. Reverse engineering the neural networks for rule extraction in classification problems. *Neural Process. Lett.* 35(2):131–150.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *Procs. of ICLR'15*.

Benchaji, I.; Douzi, S.; Ouahidi, B. E.; and Jaafari, J. 2021. Enhanced credit card fraud detection based on attention mechanism and LSTM deep model. *J. Big Data* 8(1):151.

Besold, T. R.; d'Avila Garcez, A. S.; Bader, S.; Bowman, H.; Domingos, P. M.; Hitzler, P.; Kühnberger, K.; Lamb, L. C.; Lowd, D.; Lima, P. M. V.; de Penning, L.; Pinkas, G.; Poon, H.; and Zaverucha, G. 2017. Neural-symbolic learning and reasoning: A survey and interpretation. *CoRR* abs/1711.03902.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Commun. ACM* 54(12):92–103.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language models are few-shot learners. In *Procs. of NeurIPS'20*.

Chu, E.; Roy, D.; and Andreas, J. 2020. Are visual explanations useful? A case study in model-in-the-loop prediction. *CoRR* abs/2007.12248.

D'Asaro, F. A.; Spezialetti, M.; Raggioli, L.; and Rossi, S. 2020. Towards an inductive logic programming approach for explaining black-box preference learning systems. In *Procs. of KR'20*, 855–859.

de Sousa Ribeiro, M., and Leite, J. 2021. Aligning artificial neural networks and ontologies towards explainable AI. In *Procs. of AAAI'21*, 4932–4940. AAAI Press.

de Sousa Ribeiro, M.; Krippahl, L.; and Leite, J. 2020. Explainable abstract trains dataset. *CoRR* abs/2012.12115.

Fu, Y.; Yang, L.; Liu, D.; Huang, T. S.; and Shi, H. 2021. Compfeat: Comprehensive feature aggregation for video instance segmentation. In *Procs. of AAAI'21*, 1361–1369. AAAI Press.

Gilpin, L. H.; Bau, D.; Yuan, B. Z.; Bajwa, A.; Specter, M. A.; and Kagal, L. 2018. Explaining explanations: An overview of interpretability of machine learning. In *Procs. of DSAA'18*, 80–89. IEEE.

Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; and Pedreschi, D. 2019. A survey of methods for explaining black box models. *ACM Comput. Surv.* 51(5):93:1–93:42.

Hitzler, P.; Bianchi, F.; Ebrahimi, M.; and Sarker, M. K. 2020. Neural-symbolic integration and the semantic web. *Semantic Web* 11(1):3–11.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019. Abduction-based explanations for machine learning models. In *Procs. of AAAI'19*, 1511–1519. AAAI Press.

Ignatiev, A. 2020. Towards trustable explainable AI. In *Procs. of IJCAI'20*, 5154–5158. ijcai.org.

Ivanovs, M.; Kadikis, R.; and Ozols, K. 2021. Perturbation-based methods for explaining deep neural networks: A survey. *Pattern Recognit. Lett.* 150:228–234.

Kanagaraj, N.; Hicks, D.; Goyal, A.; Tiwari, S. M.; and Singh, G. 2021. Deep learning using computer vision in self driving cars for lane and traffic sign detection. *Int. J. Syst. Assur. Eng. Manag.* 12(6):1011–1025.

Karpathy, A.; Toderici, G.; Shetty, S.; Leung, T.; Sukthankar, R.; and Fei-Fei, L. 2014. Large-scale video classification with convolutional neural networks. In *Procs. of CVPR'14*, 1725–1732. IEEE Computer Society.

Kim, B.; Wattenberg, M.; Gilmer, J.; Cai, C. J.; Wexler, J.; Viégas, F. B.; and Sayres, R. 2018. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (TCAV). In *Procs. of ICML'18*, volume 80 of *Proceedings of Machine Learning Research*, 2673–2682. PMLR.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Procs. of ICLR'15*.

Krizhevsky, A., and Hinton, G. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Law, M.; Russo, A.; and Broda, K. 2014. Inductive learning of answer set programs. In *Procs. of JELIA'14*, volume 8761 of *Lecture Notes in Computer Science*, 311–325. Springer.

Lehmann, J.; Auer, S.; Bühmann, L.; and Tramp, S. 2011. Class expression learning for ontology engineering. *J. Web Semant.* 9(1):71–81.

Lehmann, J. 2009. Dl-learner: Learning concepts in description logics. *J. Mach. Learn. Res.* 10:2639–2642.

Li, X.; Shi, Y.; Li, H.; Bai, W.; Cao, C. C.; and Chen, L. 2021. An experimental study of quantitative evaluations on saliency methods. In *Procs. of KDD'21*, 3200–3208. ACM.

Lin, Y.; Chen, T.; and Yu, L. 2017. Using machine learning to assist crime prevention. In *Procs. of IIAI-AAI'17*, 1029–1030. IEEE Computer Society.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.* 267:1–38.

Montavon, G.; Binder, A.; Lapuschkin, S.; Samek, W.; and Müller, K. 2019. Layer-wise relevance propagation: An overview. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*. Springer. 193–209.

Potyka, N. 2021. Interpreting neural networks as quantitative argumentation frameworks. In *Procs. of AAAI'21*, 6463–6470. AAAI Press.

Rebuffi, S.; Fong, R.; Ji, X.; and Vedaldi, A. 2020. There and back again: Revisiting backpropagation saliency methods. In *Procs. of CVPR'20*, 8836–8845. Computer Vision Foundation / IEEE.

Ren, S.; He, K.; Girshick, R. B.; and Sun, J. 2017. Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 39(6):1137–1149.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Procs. of KDD'16*, 1135–1144. ACM.

Sarker, M. K.; Xie, N.; Doran, D.; Raymer, M. L.; and Hitzler, P. 2017. Explaining trained neural networks with semantic web technologies: First steps. In *Procs. of NeSy'17*, volume 2003 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Schmid, U. 2018. Inductive programming as approach to comprehensible machine learning. In *Procs. of DKB'18 and KIK'18*, volume 2194 of *CEUR Workshop Proceedings*, 4–12. CEUR-WS.org.

Schmitz, G. P. J.; Aldrich, C.; and Gouws, F. S. 1999. ANN-DT: an algorithm for extraction of decision trees from artificial neural networks. *IEEE Trans. Neural Networks* 10(6):1392–1401.

Shakerin, F., and Gupta, G. 2019. Induction of non-monotonic logic programs to explain boosted tree models using LIME. In *Procs. of AAAI'19*, 3052–3059. AAAI Press.

Shakerin, F., and Gupta, G. 2020. White-box induction from SVM models: Explainable AI with logic programming. *Theory Pract. Log. Program.* 20(5):656–670.

Shen, H., and Huang, T. K. 2020. How useful are the machine-generated interpretations to general users? A human evaluation on guessing the incorrectly predicted labels. *CoRR* abs/2008.11721.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; Chen, Y.; Lillicrap, T. P.; Hui, F.; Sifre, L.; van den Driessche, G.; Graepel, T.; and Hassabis, D. 2017. Mastering the game of go without human knowledge. *Nat.* 550(7676):354–359.

Sun, W.; Zheng, B.; and Qian, W. 2016. Computer aided lung cancer diagnosis with deep learning algorithms. In *Procs. of MICAD'16*, volume 9785 of *SPIE Proceedings*, 97850Z. SPIE.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic attribution for deep networks. In *Procs. of ICML'17*, volume 70 of *Proceedings of Machine Learning Research*, 3319–3328. PMLR.

Towell, G. G., and Shavlik, J. W. 1993. Extracting refined rules from knowledge-based neural networks. *Mach. Learn.* 13:71–101.

Tschandl, P.; Rosendahl, C.; and Kittler, H. 2018. The HAM10000 dataset: A large collection of multi-source dermatoscopic images of common pigmented skin lesions. *CoRR* abs/1803.10417.

Wang, M., and Deng, W. 2021. Deep face recognition: A survey. *Neurocomputing* 429:215–244.

Wang, Z.; Mardziel, P.; Datta, A.; and Fredrikson, M. 2020. Interpreting interpretations: Organizing attribution methods by criteria. In *Procs. of CVPR'20 Workshops*, 48–55. Computer Vision Foundation / IEEE.

Zeiler, M. D., and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *Procs. of ECCV'14*, volume 8689 of *Lecture Notes in Computer Science*, 818–833. Springer.

Zilke, J. R.; Mencía, E. L.; and Janssen, F. 2016. Deepred - rule extraction from deep neural networks. In *Procs. of DS'16*, volume 9956 of *Lecture Notes in Computer Science*, 457–473.

Zimmer, M.; Feng, X.; Glanois, C.; Jiang, Z.; Zhang, J.; Weng, P.; Hao, J.; Li, D.; and Liu, W. 2021. Differentiable logic machines. *CoRR* abs/2102.11529.