Learning Logic Explanations by Neural Networks

Compendium of Neuro-Symbolic Artificial Intelligence

Pascal Hitzler [a], Md Kamruzzaman Sarker [b], and Aaron Eberhart [a]
[a] *Kansas State University*
[b] *Kansas State University*

Contents

Chapter 1

Learning Logic Explanations by Neural Networks

Gabriele Ciravegna, Universitè Côte d'Azur, Inria, CNRS, Laboratoire I3S, Maasai team, Nice, France

Francesco Giannini, DIISM, University of Siena, Siena, Italy

Pietro Barbiero, University of Cambridge, Cambridge, United Kingdom

Marco Gori, DIISM, University of Siena, Siena, Italy

Pietro Lio, University of Cambridge, Cambridge, United Kingdom

Marco Maggini, DIISM, University of Siena, Siena, Italy

Stefano Melacci, DIISM, University of Siena, Siena, Italy

1.1. Introduction

In the last decade, the development of Deep Neural Networks (DNNs) has driven the Artificial Intelligence (AI) research to unforeseeable achievements in several fields [1, 2, 3]. Nonetheless, the increased complexity of the adopted models has entailed several issues for their applications [4]: an opaque decision process [5], the need for large amounts of training data [6], and the exposure to adversarial attacks [7], to name a few. EXplainable Artificial Intelligence (XAI) aims at whitening the decision process of machine learning models making their predictions more trustworthy for human users. For instance, this is especially important to use these models in safety-critical domains [8, 9]. However, most XAI techniques are not able to explain how DNNs compose the input features to make final predictions, and even less provide concise and unambiguous explanations whose validity can be assessed quantitatively. In order to make some advancements in this direction, in [10, Ciravegna et al.] and [11, Barbiero et al.] we proposed Logic Explained Networks (LENs), a novel class of "explainable-by-design" deep learning models whose predictions are explained by a set of First-Order Logic (FOL) formulas. The FOL explanations may describe the decision process of the LEN itself or of another DNN, e.g. ex-

plaining how the DNN elaborates input samples to provide output predictions or pointing out the relationships among the labels predicted for a certain example.

In the past few years, we have successfully applied LENs: (i) with different learning paradigms, supervised [12] and unsupervised [13]; (ii) for different explanation objectives, explaining existing black-box models or for self-explainable classification [10]; (iii) by using different architectures, ψ-networks, ReLU-networks, μ-networks [10] and Entropy networks [11]; (iv) in different domains, from textual data and computer vision, to natural language processing [14] and graph domains [15, 16, 17]. LENs have been implemented as whole models[1], or as single PyTorch layers that can be integrated in different type of architectures[2]. However, LENs contributions are currently scattered in different papers using slightly different notations adapted according to specific requirements, domains and architectures. After all these contributions, the LENs literature is still short of a unified notation and overview.

For this reason, in this chapter we aim at providing a compact unified vision on LENs framework, providing a common notation for different architectures and domains of applications. Some background on Explainable AI and Concept-Based Explanations is introduced in Section 1.2. Section 1.3 summarizes the different learning criteria to apply according to the use cases (Section 1.3.1), how we can extract different explanations from a LEN (Section 1.3.2), and how to sparsify the network so that the explanations can be a faithful representation of the model behaviour (Section 1.3.3). We also describe some out-of-the-box LENs with different interpretability-vs-accuracy trade-offs (Section 1.3.4). In Section 1.4, we show how LEN can be applied in different domains by modifying the architectural pipeline. Finally, in Section 1.5 we conclude the chapter analysing the limitations of the current proposal and drawing possible future work.

1.2. Background

Explainable AI. In the last few years, the scientific community has developed a wide variety of XAI techniques, with different properties and goals [18, 19]. For instance, XAI methods can be distinguished according to their ROLE (being an interpretable model or explaining an existing black-box), the TYPE of the produced explanation (feature scoring or rule-based) and their SCOPE (local or global). In the literature, existing approaches cover different ROLES, acting as intrinsically *interpretable models* or as *explanation methods*. Interpretable models are white-box models whose decision process is considered transparent. In principle, they are the best models to support decision systems. However, their decision function is often not complex enough, impairing their generalization ability [20]. On the other hand, explanation methods can be applied to get approximated interpretations of state-of-the-art models. These methods are known as "post hoc" techniques (often also model-agnostic), as the explanations are produced once the training procedure is concluded. LENs can be employed to cover both roles, either working as an interpretable model or training them as a model-agnostic explainer that mimic the behaviour of another model. Concerning the type of produced explanation, most of the methods focus on scoring the input features [21, 22, 23, 24, 25]. However, *feature-scoring* techniques may not be so useful to support decision processes. *Rule-based* meth-

[1] `https://github.com/pietrobarbiero/logic_explained_networks`
[2] `https://github.com/pietrobarbiero/pytorch_explain`

ods are generally more appropriate for this objective, since they explain how the selected input features correlate and produce a certain output [26, 27, 28]. For this reason, LENs employ FOL to provide meaningful explanations. Finally, the SCOPE of the provided explanations strongly characterizes explainable methods. Indeed, *local* explanations are valid only for a single sample (and its neighbourhood), while *global* explanations hold on the whole input space. LENs have been designed to provide both local and global explanations according to user preference.

Concept-based Explanation. In all domains where the input features are not human-understandable, there is the need to find higher level representation. As an example, in [23] when working with images, LIME is trained to find the super-pixels causing a given classification (rather than the single pixels). However, super-pixels representation do not encode concepts that are understandable by human users. For this reason, concept-based models are receiving ever-growing consideration [29, 30, 31], as they provide explanations in terms of human-understandable symbols (the *concepts*) rather than raw features. For instance, a concept-based explanation may describe a high-level category through its attributes, as in "a *human* has *hands* and a *head*". This makes these methods more suitable for decision-making tasks and allows them to provide also global explanations (which cannot be provided in terms of raw features). However, while concept ranking is a common feature of concept-based techniques, there are very few approaches formulating hypotheses on how black-boxes combine concepts to arrive to a decision and even less providing synthetic explanations whose validity can be quantitatively assessed [32]. LENs, on the contrary, by providing FOL explanations based on concepts can serve both purposes. In Section 1.4, we will show through different pipelines how to obtain concept representations when working with different kind of data.

1.3. Logic Explained Networks

A Logic Explained Network (LEN) is a function $f : X \to Y$ designed according to specific criteria that we illustrate below in this section. We refer to $X = [0,1]^d$ as the *input space* and to $Y = [0,1]^c$ as the *output space*, whereas x_j and f_i denote the j-th component of a certain $x \in \mathscr{X}$ and the i-th component of f. LENs are neural models whose behaviour can be interpreted by FOL explanations ϕ involving symbols associated either to components of the *input* and/or the *output* space. In particular, a LEN f is designed according to a certain role (Section 1.3.1), a FOL rule-extraction algorithm (Section 1.3.2), a pruning criterion (Section 1.3.3), a neural architecture (Section 1.3.4) and a training set $(\mathbb{X}, \mathbb{Y})$ with input data $\mathbb{X} \subset X$ and labels $\mathbb{Y} \subset Y$. The FOL explanations ϕ extracted from a LEN f can correctly explain the prediction $f(x)$ for $x \in \mathbb{X}$, as shown in Fig. 1.

1.3.1. Roles and Learning Criteria

A LEN f can mostly be used in two scenarios, either as an interpretable model per se or to explain a black-box model b. In both cases, f can be trained with *supervised* or *unsupervised* learning criteria.

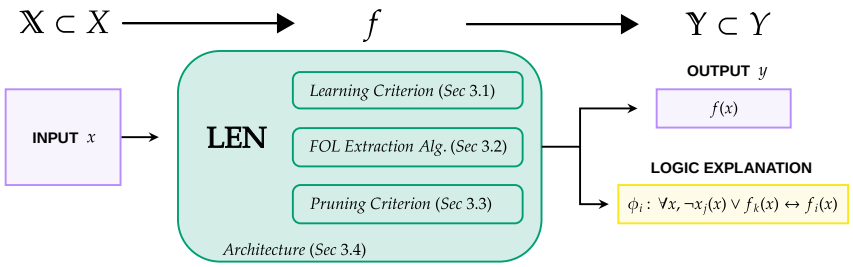

Figure 1. Overall view of the LEN structure.

Interpretable Model. When used as an interpretable model, a LEN is required to both classify and explain its predictions on each input sample. To accomplish this task, we train f to minimize the following loss function:

$$\mathscr{L}(f,\mathbb{X},\mathbb{Y}) = \mathscr{L}_l(f,\mathbb{X},\mathbb{Y}) + \mathscr{L}_p(f,\mathbb{X}) \tag{1}$$

where $\mathscr{L}_l$ is the learning criterion and $\mathscr{L}_p$ is a parsimony criterion that allows us to regularize the network and extract FOL explanations. Hereafter, we focus on $\mathscr{L}_l$, while $\mathscr{L}_p$ will be discussed in Section 1.3.3. According to whether we are facing a supervised or an unsupervised learning task, different loss functions can be employed. In case of classification tasks, the learning criterion may be expressed by the standard cross-entropy loss $\mathscr{L}_l(f,\mathbb{X},\mathbb{Y}) = \sum_{x\in\mathbb{X},y\in\mathbb{Y}}\sum_{i=1}^{c} y_i \log(f_i(x))$. In this case, the explanation ϕ_i will consist in the mutual implication between the class i and some input features, like "$\neg$longPetal$(x) \Leftrightarrow$ irisSetosa(x)" if we consider the case of the Iris classification task. On the other hand, when facing an unsupervised learning task, we need to employ different learning techniques, like maximizing the Mutual Information $\mathscr{L}_l(f,\mathbb{X}) = \mathscr{H}(f) - \mathscr{H}(f|X)$, where we respectively indicated with $\mathscr{H}(f)$ and $\mathscr{H}(f|X)$ the entropy of the output predictions and the conditional entropy of the predictions when the input data are known. In this latter cases, the explanations will point out the common characteristics of the samples belonging to a certain cluster, like "longPetal$(x) \wedge \neg$widePetal(x)".

Explaining a Black-Box Model. When explaining an existing model, a LEN can be trained following the same criteria, with some minor modifications. Indeed, if we want to explain some class predictions of a black-box model b, we can replace the supervision labels y with the black-box network predictions $b(x)$. More precisely, the supervised learning criteria become $\mathscr{L}_l(f,b,\mathbb{X}) = \sum_{x\in\mathbb{X}}\sum_{i=1}^{c} b(x)\log(f_i(x))$. Instead, if we do not have any preference on the class to explain, the LEN takes as input all the predictions of b and tries to find significant correlations by using an unsupervised learning criterion. In this case, we can maximize the Mutual Information between the explained network predictions $b(x)$ and the output of the LEN $f(b(x))$, i.e., $\mathscr{L}_l(f,b,\mathbb{X}) = \mathscr{H}(f) - \mathscr{H}(f|b)$.

1.3.2. Extracting Logic Explanations

After the training stage, LENs are capable of providing accurate logic explanations that can be used to make predictions at test time. In the following, we describe how LENs

provide explanations both for single samples and for the entire f_i, that can represent either a class or a cluster[3] in case of supervised or unsupervised learning, respectively.

Example-level Explanations. For a given sample x, the prediction $f_i(x) = 1$ is locally explained by the conjunction $\phi_i^l(x)$ of the most relevant input features for the class i on the current example:

$$\text{LEN Local Explanation:} \quad \phi_i^l(x) = \bigwedge_{j \in \mathscr{A}_i(x)} \mathbf{x_j}(x) \tag{2}$$

where we indicated with $\mathbf{x_j}(x)$ the logic predicate (or its negation) associated to the j-th input feature and with $\mathscr{A}_i(x)$ the set of relevant features for the i-th task on the current sample x. The way in which a LEN finds $\mathscr{A}_i(x)$ strongly depends on the selected parsimony criterion and it will be better clarified in Section 1.3.3. Each $\mathbf{x_j}(x)$ can be either a positive or a negative literal, according to a given threshold, e.g. $\mathbf{x_j}(x) = [x_j > 0.5]$. For any f_i, the set of all its local explanations is denoted by Φ_i^l, i.e. $\Phi_i^l = \{\phi_i^l : \ \phi_i^l(x)$ is the local explanation of $f_i(x)$ for some $x \in \mathscr{X}$ with $f_i(x) = 1\}$.

Class-level Explanations. To globally explain f_i, a LEN considers the disjunction of the most important local explanations:

$$\text{LEN Global Explanation:} \quad \phi_i^g = \forall x : \bigvee_{\phi_i^l \in \mathscr{B}_i} \phi_i^l(x) \leftrightarrow \mathbf{f_i}(x) \tag{3}$$

Here $\mathbf{f}_i$ indicates the logic predicate associated to f_i and $\mathscr{B}_i$ collects the k-most frequent local explanations of the training set that is computed as $\mathscr{B}_i = \{\phi_i^l \in \arg\max^k_{\phi_i^l \in \Phi_i^l} \mu(\phi_i^l)\}$, where we indicated with $\mu(\cdot)$ the frequency counting operator. However, the way $\mathscr{B}_i$ is determined can be refined by employing a greedy strategy gradually aggregating frequent local explanations only if they improve the validation accuracy, as in [10, Ciravegna et al.] and [11, Barbiero et al.], or by employing an exhaustive search and selecting the set of local explanations providing the highest accuracy, as in [14, Jain et al.]. Finally, the kind of implication depends on the learning criteria: if we use the losses introduced in Sec. 1.3.1 we extract formulas with the double implication $\leftrightarrow$, but others can be used (see [12, Ciravegna et al.]).

1.3.3. Parsimony Criteria and Sparsification (Pruning)

Parsimony criteria influence the learning process towards specific solutions. These configurations are generally those achieving higher generalization accuracy, and reducing the bias of the network. The most commonly employed regularizations are the $L_1(W)$ and $L_2(W)$ regularizations applied to the network weight W with reuse strategies [10]. In alternative, in [11, Barbiero et al.] we have shown that minimizing the Entropy of the input weights is also an effective regularizer, producing sharper configurations. Once the model is converging, the effort can be accelerated and finalized by pruning the neural network [33, 34] i.e., removing connections whose likelihood of carrying important in-

[3] In unsupervised learning tasks, with $f_i(x) = 1$ we indicate that the example x belongs to the i-th cluster.

formation is low. Furthermore, by forcing the network to rely on few connections, we can better understand which are the relevant features for a certain class, and determine $\mathscr{A}_i(x)$. We notice that the choice of the connections to be pruned may have a profound impact on the quality of the explanations, but also on the classification performance [35]. In the following, we propose three different strategies with decreasing impacts on the network and we refer for convenience to the underlying graph structure of a LEN f as $\mathscr{G} = (N, E)$.

Node Pruning. A straightforward pruning strategy considers each neuron independently. This strategy allows to set a maximum *fan-in* $\zeta \in \mathbb{N}$ for each neuron of the network, i.e., the number of non-pruned incoming connections that each neuron can have. More precisely, the pruning strategy iteratively removes the connections associated to the smallest weights until the required fan-in is reached. Let us consider the i-th output neuron of a LEN. We define as $\mathscr{G}_i = (N_i, E_i)$ the sub-portion of $\mathscr{G}$ composed of the neurons and connections participating to just valid paths (i.e. paths involving only non-zero weights) whose destination is the i-th output neuron. In this case, the set of important features $\mathscr{A}_i$ is fixed for every sample and is defined as $\mathscr{A}_i = \{j \in [1,d] \mid j \in N_i\}$. This strategy yields higher interpretability, because limiting the number of neuron connections can allow to extract compact local and global explanations not only for the entire model but also for each hidden neuron. On the contrary, since this pruning entails each neuron, it significantly reduces the learning capability of the network.

Network Pruning. To select a small set of important input features $\mathscr{A}_i$, pruning all the neurons can be unnecessary. Indeed, it is sufficient to iteratively prune only the weights connected to the first layer of hidden neurons, i.e. for each i we have $\mathscr{A}_i = \mathscr{A} = \{j \in [1,d] \mid \exists k \in [1,h_1] : \ w_{j,k} \neq 0\}$, where h_1 is the number of hidden neurons at the first layer of the network. However, as a consequence the set of important input features would be the same for all the classes (i.e. it does not depend on the class i), which is quite unlikely. To overcome this issue in [11, Barbiero et al.], we introduced the Entropy Layer, that is composed of a 3-dimensional input layer providing different weights for each class. It therefore allows us to prune a different set of weights for each class i, and to define $\mathscr{A}_i = \{j \in [1,d] \mid \exists k \in [1,h_1] : \ w_{j,k,i} \neq 0\}$. This strategy allows us to retain a network-level interpretability without losing much performance.

Example-level Pruning. When employing a LEN whose activation functions in all hidden layers are Rectified Linear Units (ReLU Network), $\mathscr{G}$ can be reduced to $\hat{\mathscr{G}}$, the subgraph only retaining the units corresponding to active neurons (i.e. the ones for which the ReLU activation is non-zero) and the corresponding edges, also referred to as "firing path". Since all neurons operate in linear regime (affine functions), in $\hat{\mathscr{G}}$ the multi-layer feed-forward ReLU network can be simplified with a single affine function, leading to $\forall x \in X, f(x) = \sigma(\hat{W}^{(x)}x + \hat{b}^{(x)})$, being σ the activation of the output layer and $\hat{W}^{(x)}$, $\hat{b}^{(x)}$ the simplified weight matrix and biases, respectively. We refer the reader to [10, Ciravegna et al.] for the theorem proving this result. For each sample x, this allows us to only consider the simplified weights $\hat{w}^{(x)}$ to locally explain the prediction of the model. More precisely, in this case we define $\mathscr{A}_i(x) = \{j \in [1,d] \mid \hat{w}^{(x)}_{j,i} \neq 0\}$.

1.3.4. Architectures

The creation of an explainable-by-design neural network comes in general at the cost of a reduced learning capability. However, the methods introduced in Section 1.3.3 allow to

reach different accuracy vs. interpretability trade-offs. In the following, we show three out-of-the-box Logic Explained Networks (LENs).

ψ Network [13, 12]. The ψ Network is a fully interpretable model, but with limited learning capacity. It employs a node-level pruning strategy, with the same fan-in across all neurons. The fan-in should be sufficiently low (suggested values between 2 and 9) since it is directly proportional to the number of terms involved in the explanations. The pruning process is conducted during the training by progressively zeroing the least important weights in input to each neuron. The explanation extraction process is conducted for all neurons in the network. This allows to reach high interpretability, even if at the cost of low classification performances. Indeed, it requires to employ strong L_1-regularization and $[0,1]$-valued activation functions (like e.g. sigmoids) in all the layers. The final explanations can then be extracted by composing the layer-level explanations. This reduces the explanation quality as well, since it sums the approximation errors, but it can be avoided by directly extracting explanations at network-level.

Entropy Network [11]. The Entropy Network utilizes a network-level pruning strategy to provide high classification accuracy and still retains good explainability and interpretability. More precisely, it employs an entropy layer as the first layer of the network. This layer is trained to minimize the entropy of the importances of the input features, in such a way to rely on few features. For each feature j, the importance for a class i is represented by the attention score α_j^i, which is computed as:

$$\alpha_j^i = \frac{e^{\gamma_j^i/\tau}}{\sum_{l=1}^{k} e^{\gamma_l^i/\tau}} \tag{4}$$

where τ is a user-defined temperature parameter and $\gamma_j^i = ||W_j^i||_1$. The entropy $\mathscr{H}(\alpha)$ is minimized when a single α_j^i is one, thus representing the extreme case in which only one concept matters, while it is maximum when all concepts are equally important. Each score α^i is used to further weight the input features $\tilde{x}^i = x \odot \tilde{\alpha}^i$.

ReLU Network [10]. The ReLU network is a LEN, providing a slightly different accuracy vs. interpretability trade-off. This model is based on three design principles: (*i*) all activation functions of hidden neurons are rectified linear units; (*ii*) a mild L_1-regularization is applied to all the weights associated to each layer of the network; (*iii*) an example-level pruning strategy, which can be applied due to the presence of rectified linear unit activation functions. What makes a ReLU Network significantly different from previous LENs is that the pruning strategy does not alter the network structure at all. This is due to the fact that pruning is applied to the weights that belong to $\hat{W}^{(x)}$ and are only computed for rule-extraction purposes. This means that the original capacity of the model is fully preserved, eventually leading to state-of-the-art classification performances. However, this type of pruning does not provide general insights about the model behaviour, as it is only about the considered example x. To mitigate this issue, a mild L_1-regularization is imposed to encourage the network to employ the same connection patterns to classify similar samples and, in turn, provide explanations.

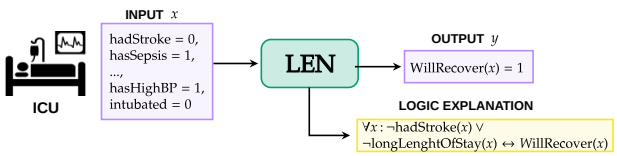

Figure 2. Tabular data pipeline

1.4. Applications

LENs are not restricted to a specific neural architecture and therefore can perform pretty well on very different scenarios and kind of data. So far, LENs have been successfully applied in a wide variety of learning frameworks ranging from standard classification tasks on tabular data to natural language processing, computer vision and graph analysis. Since LENs are meant to provide explanations in terms of human understandable *concepts*, their only requirement is to take input features in $X = [0,1]^d$. This means that in case of, e.g., real-valued inputs $X' = \mathbb{R}^d$, we may need to map the original raw data into $[0,1]$-valued vectors before applying LENs. This may be necessary with slight variations also for other application cases, so basically different pipelines may need to be defined to apply LEN in the different contexts. More precisely, it can be necessary to map the raw input features $x' \in X'$ to the LEN input space X. This can be achieved by means of standard input preprocessing or by employing an additional model $g : X' \to X$, e.g. another neural network, that can be directly co-trained with the LEN $f : X \to Y$.

1.4.1. Tabular Data

In case of tabular data, the pipeline is pretty straightforward, as the input features can be already regarded as concepts. There are mostly two options: (i) $X' = [0,1]^d$, (ii) $X' = \mathbb{R}^d$. In case (i), we trivially have $X' = X$. As an example, in Fig. 2 we can appreciate a LEN application in the Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II) [36] medical domain. Here the input features were already binary $X = \{0,1\}^d$ and therefore no preprocessing was required. A LEN has been used to predict and explain whether patients entering intensive care units will recover or not, according to their starting condition. An example of explanation provided by the network in this case is $\phi^g = \forall x :$ liverImpairment$(x) \wedge \neg$Stroke$(x) \wedge \neg$cancerAggressive$(x) \leftrightarrow$ Recover(x). In case (ii), each input sample $x' \in \mathscr{X}' \subset X'$ needs to be rescaled into a $[0,1]$-value. This can be done by discretizing the feature values with opportune thresholds, like e.g. $[0 \leq age(x) < 60]$ and $[age(x) \geq 60]$, or by fuzzyfication in the unit interval, like e.g. mapping $highSalary(x) \in [0,1]$ with values near 0 corresponding to the minimum wage and 1 to the maximum salary. You can find further examples in [10, Ciravegna et al.] and [11, Barbiero et al.]. In [37, Ciravegna et al.], we have also shown that the explanation provided by LENs in this scenario can be used to select the data labels within an active learning scenario.

1.4.2. Natural Language Processing

Like in the previous case, when dealing with textual data, we can regard each word directly as an understandable concept certifying the presence/absence of the word it-

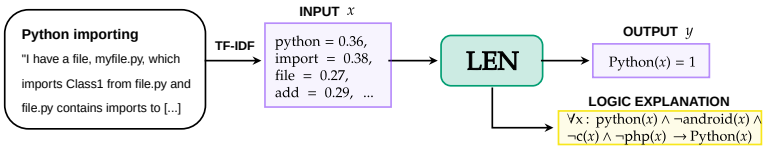

Figure 3. Natural Language Processing pipeline

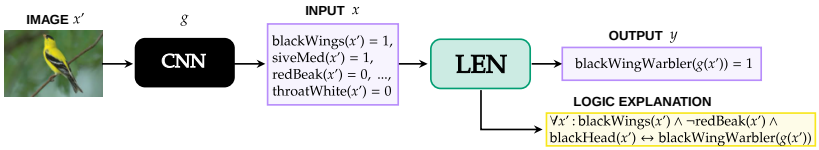

Figure 4. Convolutional Neural Networks pipeline

self in an input stream of text and, therefore, we only need to map the words into the unit interval. This can be easily achieved in different ways, like e.g. by using TF-IDF as shown in Fig. 3, showing a LEN application on the "StackSample: 10% of Stack Overflow Q&A" [38] tag classification task. The LEN receives in input the preprocessed text and output the corresponding tag and an explanation, in this case: $\phi^g = \forall x:\ \text{python}(x) \wedge \neg\text{android}(x) \wedge \neg\text{c}(x) \wedge \neg\text{php}(x) \rightarrow \text{Python}(x)$. By means of a human survey, in [14, Jain et al.] we have also showed that the explanations provided by LENs can be useful in a variety of tasks (e.g. in detecting biased models), and that better serve these purposes than feature-based explanations provided by LIME.

1.4.3. Computer Vision

In Computer Vision tasks, pixels cannot be properly considered as meaningful concepts to provide logic explanations. For this reason, we employ the Concept-bottleneck Model pipeline introduced in [31], which considers a Convolutional Neural Network (CNN) g mapping the input images $x' \in X'$ into a set of concepts $x \in X$. In Fig. 4, we reported an example of this pipeline on the Caltech-UCSD Birds 200 (CUB200) [39] according to the following steps. First, the CNN g predicts a set of low-level bird attributes for the given input image. Then these attributes are fed in input to a LEN f. Finally, the LEN detects e.g. blackwingWarbler$(g(x'))$ from the input image and provides an explanation in terms of the bird attributes, i.e., $\phi^g = \forall x':\ \text{blackWings}(x') \wedge \neg\text{redBeak}(x') \wedge \text{blackHead}(x') \leftrightarrow \text{blackWingWarbler}(g(x'))$. Further examples of LEN application in the computer vision domains can be found in [13, 12, Ciravegna et al.] [11, Barbiero et al.]. Furthermore, in [10, Ciravegna et al.], we have shown that the explanations provided by the LEN can be effectively employed to detect adversarial data.

1.4.4. Graph Domain

In relational domains, LENs have been applied on top of different neural architectures with the aim of providing global concept-based logic explanations of Graph Neural Networks (GNNs, [40]), as in [16, Magister et al.] and [17, Azzolini et al.], or for explain-

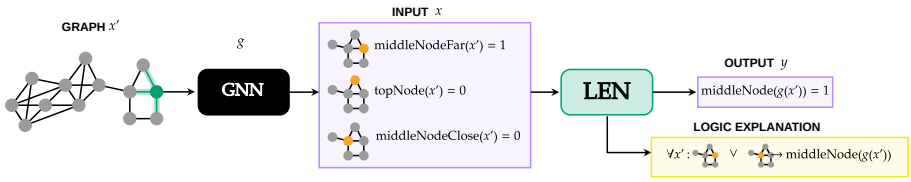

Figure 5. Graph Neural Networks pipeline

ing Neural Algorithmic Reasoning (NAR, [41]) and sub-symbolic heuristics as in [15, Georgiev et al.]. To explain GNNs, both [16, Magister et al.] and [17, Azzolin et al.] use message passing to generate a set of concepts corresponding to graph motifs. LENs are then used to solve typical relational tasks (e.g. node and graph classification) providing logic explanations in terms of the motifs identified by the GNN in the input graph. To explain sub-symbolic heuristics, [15, Georgiev et al.] trained NAR to learn a set of relevant concepts (e.g. node colours for graph colouring or visited edges for minimum spanning trees). At each NAR iteration, LENs are used to predict and provide a logic explanation for nodes and edges states at next iteration.

1.5. Conclusions and Future Work

This chapter presents a concise view of the Logic Explained Network (LEN) framework and some existing applications. LENs are "explainable-by-design" neural networks that can be trained to provide FOL explanations of their predictions or of other black-box models. One of the main advantages of LENs is their versatility in realizing different accuracy vs. interpretability trade-offs, that can be forced by means of a few learning principles according to user preferences. Experimentally, LENs have been shown to achieve performances close to state-of-the-art black-boxes on a wide variety of learning scenarios, while also providing concept-based logic explanations. As we discussed in the chapter, achieving high accuracy while providing human-understandable explanations is fundamental for decision support problems, especially in safety-critical applications.

Open Challenges. One weaker point for LENs application is that they rely on concept-based input features, that may require the instantiation of ad hoc pipelines to deal with raw input data. Even if we propose different alternatives to face this issue, for future work would be interesting to automatically extract from a DNN low-level concepts on the original inputs, and use these newly-devised concepts as ingredient to provide logic explanations of the predictions of the final classes. Another interesting application field that we plan to investigate is the use of LENs in a continual learning setting, where predictions and explanations may depend on a temporal coordinate and can be revised as time goes by. In this regard, LENs contribution could be twofold, as they can be used both as a neural model learning in a dynamic environment and to produce different set of formulas in different timestamps. In this context, a fundamental role could be played by temporal logic that allows formulas to be referred as valid into specific time intervals. Finally, within the interactive Machine Learning (iML) scenario, it would be interesting to progressively train and explain a model at the same time. By iteratively feeding the network with new samples, we could select those that allows the network to address

mistakes and mitigate the biases that it might have learnt. In this context, translating the logic formulas into natural language may facilitate the interaction between the human and the machine.

Acknowledgments

This work was partially supported by TAILOR and by HumanE-AI-Net, projects funded by EU Horizon 2020 research and innovation programme under GA No 952215 and No 952026, respectively. It was also partially supported by the EU Horizon 2020 project AI4Media, under contract no. 951911 and by the French government, through Investments in the Future projects managed by the National Research Agency (ANR), 3IA Cote d'Azur with the reference number ANR-19-P3IA-0002. Finally, this work was partially supported by the PRIN 2017 project RexLearn (Reliable and Explainable Adversarial Machine Learning), funded by the Italian Ministry of Education, University and Research (grant no. 2017TWNMH2).

References

[1] LeCun Y, Bengio Y, Hinton G. Deep learning. nature. 2015;521(7553):436–444.
[2] Litjens G, Kooi T, Bejnordi BE, et al. A survey on deep learning in medical image analysis. Medical image analysis. 2017;42:60–88.
[3] Kamilaris A, Prenafeta-Boldú FX. Deep learning in agriculture: A survey. Computers and electronics in agriculture. 2018;147:70–90.
[4] Marcus G. Deep learning: A critical appraisal. arXiv preprint arXiv:180100631. 2018;.
[5] Dayhoff JE, DeLeo JM. Artificial neural networks: opening the black box. Cancer: Interdisciplinary International Journal of the American Cancer Society. 2001;91(S8):1615–1635.
[6] Deng J, Dong W, Socher R, et al. Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition; Ieee; 2009. p. 248–255.
[7] Szegedy C, Zaremba W, Sutskever I, et al. Intriguing properties of neural networks. arXiv preprint arXiv:13126199. 2013;.
[8] Goddard M. The eu general data protection regulation (gdpr): European regulation that has a global impact. International Journal of Market Research. 2017;59(6):703–705.
[9] Willers O, Sudholt S, Raafatnia S, et al. Safety concerns and mitigation approaches regarding the use of deep learning in safety-critical perception tasks. In: International Conference on Computer Safety, Reliability, and Security; Springer; 2020. p. 336–350.
[10] Ciravegna G, Barbiero P, Giannini F, et al. Logic explained networks. Artificial Intelligence. 2022;in press (available online). Available from: `https://doi.org/10.1016/j.artint.2022.103822`.
[11] Barbiero P, Ciravegna G, Giannini F, et al. Entropy-based logic explanations of neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 36; 2022. p. 6046–6054.
[12] Ciravegna G, Giannini F, Gori M, et al. Human-driven fol explanations of deep learning. In: IJCAI-PRICAI 2020-29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence; International Joint Conferences on Artificial Intelligence Organization; 2020. p. 2234–2240.
[13] Ciravegna G, Giannini F, Melacci S, et al. A constraint-based approach to learning and explanation. In: Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 34; 2020. p. 3658–3665.
[14] Jain R, Ciravegna G, Barbiero P, et al. Extending logic explained networks to text classification. In: Findings of the Association for Computational Linguistics: EMNLP 2022, "(accepted)"; 2022. p. NA.
[15] Georgiev D, Barbiero P, Kazhdan D, et al. Algorithmic concept-based explainable reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 36; 2022. p. 6685–6693.
[16] Magister LC, Barbiero P, Kazhdan D, et al. Encoding concepts in graph neural networks. arXiv preprint arXiv:220713586. 2022;.

[17] Azzolin S, Longa A, Barbiero P, et al. Global explainability of gnns via logic combination of learned concepts. arXiv preprint arXiv:221007147. 2022;.

[18] Adadi A, Berrada M. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). IEEE access. 2018;6:52138–52160.

[19] Carvalho DV, Pereira EM, Cardoso JS. Machine learning interpretability: A survey on methods and metrics. Electronics. 2019;8(8):832.

[20] Molnar C. Interpretable machine learning. Lulu. com; 2020.

[21] Hastie T, Tibshirani R. Generalized additive models: some applications. Journal of the American Statistical Association. 1987;82(398):371–386.

[22] Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:13126034. 2013;.

[23] Ribeiro MT, Singh S, Guestrin C. " 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; 2016. p. 1135–1144.

[24] Lundberg SM, Lee SI. A unified approach to interpreting model predictions. Advances in neural information processing systems. 2017;30.

[25] Selvaraju RR, Cogswell M, Das A, et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision; 2017. p. 618–626.

[26] Letham B, Rudin C, McCormick TH, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. Annals of Applied Statistics. 2015;9(3):1350–1371.

[27] Guidotti R, Monreale A, Ruggieri S, et al. Local rule-based explanations of black box decision systems. arXiv preprint arXiv:180510820. 2018;.

[28] Ribeiro MT, Singh S, Guestrin C. Anchors: High-precision model-agnostic explanations. In: Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 32; 2018. p. 1527–1535.

[29] Kim B, Wattenberg M, Gilmer J, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In: International conference on machine learning; PMLR; 2018. p. 2668–2677.

[30] Ghorbani A, Wexler J, Zou JY, et al. Towards automatic concept-based explanations. Advances in Neural Information Processing Systems. 2019;32.

[31] Koh PW, Nguyen T, Tang YS, et al. Concept bottleneck models. In: International Conference on Machine Learning; PMLR; 2020. p. 5338–5348.

[32] Das A, Rad P. Opportunities and challenges in explainable artificial intelligence (xai): A survey. arXiv preprint arXiv:200611371. 2020;.

[33] LeCun Y, Denker JS, Solla SA, et al. Optimal brain damage. In: NIPs; Vol. 2; Citeseer; 1989. p. 598–605.

[34] Hassibi B, Stork DG. Second order derivatives for network pruning: Optimal brain surgeon. Morgan Kaufmann; 1993.

[35] Frankle J, Carbin M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:180303635. 2018;.

[36] Saeed M, Villarroel M, Reisner AT, et al. Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database. Critical care medicine. 2011;39(5):952.

[37] Ciravegna G, Precioso F, Gori M. Knowledge-driven active learning. arXiv preprint arXiv:211008265. 2021;.

[38] Overflow S. Stacksample: 10% of stack overflow q&a ; 2019. Available from: `https://www.kaggle.com/datasets/stackoverflow/stacksample`.

[39] Wah C, Branson S, Welinder P, et al. The caltech-ucsd birds-200-2011 dataset; 2011.

[40] Scarselli F, Gori M, Tsoi AC, et al. The graph neural network model. IEEE transactions on neural networks. 2008;20(1):61–80.

[41] Veličković P, Blundell C. Neural algorithmic reasoning. Patterns. 2021;2(7):100273.