Logic explained networks

Gabriele Ciravegna, Pietro Barbiero, Francesco Giannini, Marco Gori, Pietro Liò, Marco Maggini, Stefano Melacci

LOGIC EXPLAINED NETWORKS

Gabriele Ciravegna[*1,2], **Pietro Barbiero**[*3], **Francesco Giannini**[*2],
Marco Gori[2,4], **Pietro Lió**[3], **Marco Maggini**[2], **Stefano Melacci**[2]
[1]Università di Firenze (Italy), [2]Università di Siena (Italy),
[3]University of Cambridge (UK), [4]Universitè Côte d'Azur (France)
gabriele.ciravegna@unifi.it, pb737@cam.ac.uk, francesco.giannini@unisi.it
marco.gori@unisi.it, pl219@cam.ac.uk, marco.maggini@unisi.it, mela@diism.unisi.it

ABSTRACT

The large and still increasing popularity of deep learning clashes with a major limit of neural network architectures, that consists in their lack of capability in providing human-understandable motivations of their decisions. In situations in which the machine is expected to support the decision of human experts, providing a comprehensible explanation is a feature of crucial importance. The language used to communicate the explanations must be formal enough to be implementable in a machine and friendly enough to be understandable by a wide audience. In this paper, we propose a general approach to Explainable Artificial Intelligence in the case of neural architectures, showing how a mindful design of the networks leads to a family of interpretable deep learning models called Logic Explained Networks (LENs). LENs only require their inputs to be human-understandable predicates, and they provide explanations in terms of simple First-Order Logic (FOL) formulas involving such predicates. LENs are general enough to cover a large number of scenarios. Amongst them, we consider the case in which LENs are directly used as special classifiers with the capability of being explainable, or when they act as additional networks with the role of creating the conditions for making a black-box classifier explainable by FOL formulas. Despite supervised learning problems are mostly emphasized, we also show that LENs can learn and provide explanations in unsupervised learning settings. Experimental results on several datasets and tasks show that LENs may yield better classifications than established white-box models, such as decision trees and Bayesian rule lists, while providing more compact and meaningful explanations.

1 Introduction

The application of deep neural networks in safety-critical domains has been strongly limited [1], since neural networks are generally considered black-boxes whose decision processes are opaque or too complex to be understood by users. Employing black-box[2] models may be unacceptable in contexts such as industry, medicine or courts, where the potential economical or ethical repercussions are calling for lawmakers to discourage from a reckless application of non-interpretable models [2, 3, 4, 5]. As a consequence, research in Explainable Artificial Intelligence (XAI) has become strategic and has been massively encouraged, leading to the development of a variety of techniques that aim at explaining black-box models [6, 7] or at designing interpretable models [8, 9].

The notions of *interpretability* and *explainability* were historically used as synonyms. However the distinction between interpretable models and models providing explanations has now become more evident, as recently discussed by different authors [10, 11, 12]. Even if there are no common accepted formal definitions, a model is considered interpretable when its decision process is generally transparent and can be understood directly by its structure and parameters, such as *linear models* or *decision trees*. On the other hand, the way an existing (black-box) model makes

[*]Equal contribution

[2]In the context of this paper, a black-box classifier is any classifier that cannot provide human understandable explanations about its decision.

predictions can be explained by a surrogate interpretable model or by means of techniques providing intelligible descriptions of the model behaviour by e.g. formal rules, saliency maps, question-answering. In some contexts, the use of a black-box model may be unnecessary or even not preferable [13, 14, 15, 9, 16, 17]. For instance, a proof of concept, a prototype, or the solution to a simple classification problem can be easily based on standard interpretable by design AI solutions [18, 19, 20, 21, 22]. However, interpretable models may generally miss to capture complex relationships among data. Hence, in order to achieve state-of-the-art performance in more challenging problems, it may be necessary to leverage black-box models [23, 24, 25, 26] that, in turn, may require an additional explanatory model to gain the trust of the user.

In the literature, there is a wide consensus on the necessity of having an explanation for machine learning models that are employed in safety-critical domains, while there is not even agreement on what an *explanation* actually is, nor if it could be formally defined [13, 11]. As observed by Srinivasan and Chander, explanations should serve cognitive-behavioral purposes such as engendering trust, aiding bias identification, or taking actions/decisions [27]. An explanation may help humans understand the black-box, may allow for a deeper human-machine interaction [28], and may lead to more trustworthy fully automated tasks. All of this is possible as long as the explanations are useful from a human perspective. In a nutshell, an explanation is an answer to a "why" question and what makes an explanation good or bad depends on "the degree to which a human can understand the cause of a decision" [29]. The goodness of an explanation is intimately connected to how humans collect evidences and eventually make decisions. In Herbert Simon's words we may say that a good explanation is *satisficing* when "it either gives an optimal description of a simplified version of the black-box (e.g. a surrogate model) or a satisfactory description for the black-box itself" [30]. However, the notion itself of explanation generally depends on both the application domain and whom it is aimed at [8].

The need for human-understandable explanations is one of the main reasons why concept-based models are receiving ever-growing consideration, as they provide explanations in terms of human-understandable symbols (the *concepts*) rather than raw features such as pixels or characters [31, 32, 28]. As a consequence, they seem more suitable to serve many strategic human purposes such as decision making tasks. For instance, a concept-based explanation may describe a high-level category through its attributes as in "a *human* has *hands* and a *head*". 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 provide synthetic explanations whose validity can be quantitatively assessed [6].

A possible solution to provide human-understandable explanations is to rely on a formal language that is very expressive, closely related to reasoning, and somewhat related to natural language expressions, such as First-Order Logic (FOL). A FOL explanation can be considered a special kind of a concept-based explanation, where the description is given in terms of logic predicates, connectives and quantifiers, such as "$\forall x$: *is_human*$(x) \rightarrow$ *has_hands*$(x) \wedge$ *has_head*(x)", that reads "being human implies having hands and head". However, FOL formulas can generally express much more complex relationships among the concepts involved in a certain explanation. Compared to other concept-based techniques, logic-based explanations provide many key advantages, that we briefly describe in what follows. An explanation reported in FOL is a rigorous and unambiguous statement (clarity). This formal clarity may serve cognitive-behavioral purposes such as engendering trust, aiding bias identification, or taking actions/decisions. For instance, dropping quantifiers and variables for simplicity, the formula "*snow* $\wedge$ *tree* $\leftrightarrow$ *wolf*" may easily outline the presence of a bias in the collection of training data. Different logic-based explanations can be combined to describe groups of observations or global phenomena (modularity). For instance, for an image showing only the face of a person, an explanation could be "(*nose* $\wedge$ *lips*) $\rightarrow$ *human*", while for another image showing a person from behind a valid explanation could be "(*feet* $\wedge$ *hair* $\wedge$ *ears*) $\rightarrow$ *human*". The two local explanations can be combined into "(*nose* $\wedge$ *lips*) $\vee$ (*feet* $\wedge$ *hair* $\wedge$ *ears*) $\rightarrow$ *human*". The quality of logic-based explanations can be quantitatively measured to check their correctness and completeness (measurability). For instance, once the explanation "(*nose* $\wedge$ *lips*) $\vee$ (*feet* $\wedge$ *hair* $\wedge$ *ears*)" is extracted for the class *human*, this logic formula can be applied on a test set to check its generality in terms of quantitative metrics like accuracy, fidelity and consistency. Further, FOL-based explanations can be rewritten in different equivalent forms such as in *Disjunctive Normal Form* (DNF) and *Conjunctive Normal Form* (CNF) (versatility). Finally, techniques such as the Quine–McCluskey algorithm can be used to compact and simplify logic explanations [33, 34, 35] (simplifiability). As a toy example, consider the explanation "(*person* $\wedge$ *nose*) $\vee$ ($\neg$*person* $\wedge$ *nose*)", that can be easily simplified in "*nose*".

1.1 Contributions

This paper presents a unified framework for XAI allowing the design of a family of neural models, the *Logic Explained Networks* (LENs), which are trained to *solve-and-explain* a categorical learning problem integrating elements from deep learning and logic. Differently from vanilla neural architectures, LENs can be directly interpreted by means

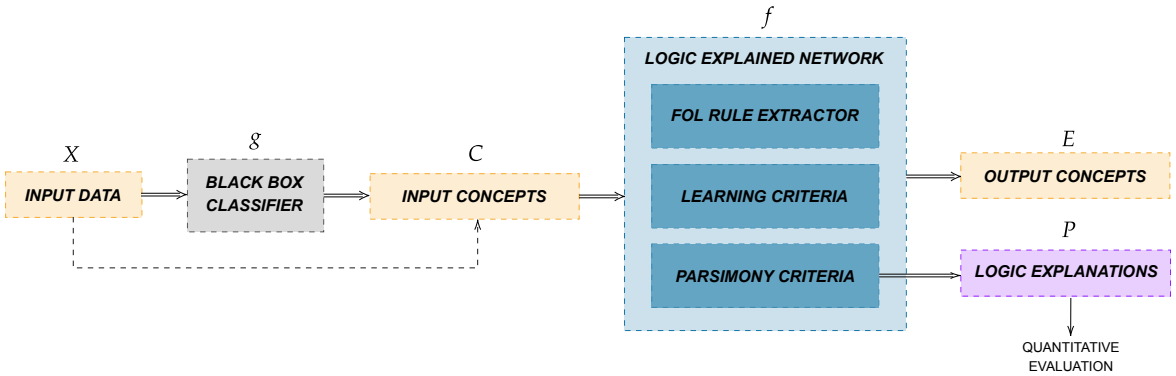


Figure 1: Logic Explained Networks (LENs, f) are neural networks capable of making predictions of a set of output concepts (activation scores belonging to E) and providing First-Order Logic explanations (belonging to P) in function of the LEN inputs. Inputs might be other concepts (activation scores belonging to C) either computed by a neural network (g) or directly provided within the available data (each data sample belongs to X). There are several different ways of instantiating this generic model into real-world problems (Section 3). Within the box labelled Logic Explained Network, the key components of LENs are listed (Section 4).

of a set of FOL formulas. In order to implement such a property, LENs require their inputs to represent the activation scores of human-understandable concepts. Then, specifically designed learning objectives allow LENs to make predictions in a way that is well suited for providing FOL-based explanations that involve the input concepts. In order to reach this goal, LENs leverage parsimony criteria aimed at keeping their structure simple. There are several different computational pipelines in which a LEN can be configured, depending on the properties of the considered problem and on other potential experimental constraints. For example, LENs can be used to directly classify data in an explainable manner, or to explain another black-box neural classifier. Moreover, according to the user expectations, different kinds of logic rules may be provided. Due to this intrinsic versatility of LENs, what we propose can also be thought of as a generic framework that encompasses a large variety of use cases. This framework takes inspiration from our previous works [36] and [37], where a novel type of neural network, the ψ network, has been proposed to extract logic rules from arbitrary neural classifiers. In facts, ψ networks are special instances of LENs, that are also specifically discussed in Section 5.1. However, the framework proposed in this paper significantly extends these previous works by defining a more general setting where different properties of the explained networks can be customized, like their structure and pruning strategy, thus allowing a more suitable trade-off between performances and transparency. In addition, LENs can also be used as explainable-by-design neural architectures for direct classification, whereas [36, 37] were limited to explaining existing black-box models.

Fig. 1 depicts a generic view on LENs, in which the main components are reported. A LEN (blue box – function f) provides FOL explanations (purple box) of a set of output concepts (rightmost yellow-box) in function of the LEN inputs. Inputs might be other concepts (mid yellow box) computed by a neural network classifier (gray box – function g) and/or concepts provided within the available data (leftmost yellow box). This generic structure can be instantiated in multiple ways, depending on the final goal of the user and on the properties of the considered problem. In order to provide the reader with an initial example/use-case (different configurations are explored in the paper), we consider an image classification problem with concepts organized into a two level hierarchy. In Fig. 2 we report an instance of Fig. 1 in which a CNN-based neural classifier gets an input image, predicting the activations of a number of low-level concepts. The LEN f processes such concepts, and it predicts the activation of higher-level output concepts. The LEN can provide a FOL description of each (high-level) output concept with respect to the (low-level) input ones. Another possible instance of the proposed framework consists in using the LEN to directly classify and explain input data (that is basically the case in which the input concepts are immediately available in the data themselves, and not the output of another neural model), thus the LEN itself becomes an interpretable machine. Moreover, LENs can be paired with a black-box classifier operating on the same input data, and forced to mimic as much as possible the behaviour of the black-box, implementing an additional explanation-oriented module.

We investigate three different use-cases that are inspired by the aforementioned instances, comparing different ways of implementing the LEN models. While most of the emphasis of this paper is on supervised classification, we also show how LEN can be leveraged in fully unsupervised settings. Additional human priors could be eventually incorporated into the learning process [36], in the architecture [28], and, following Ciravegna et al. [36, 37], what we propose can be trivially extended to semi-supervised learning (out of the scope of this paper). Our work contributes to the XAI research field in the following ways.

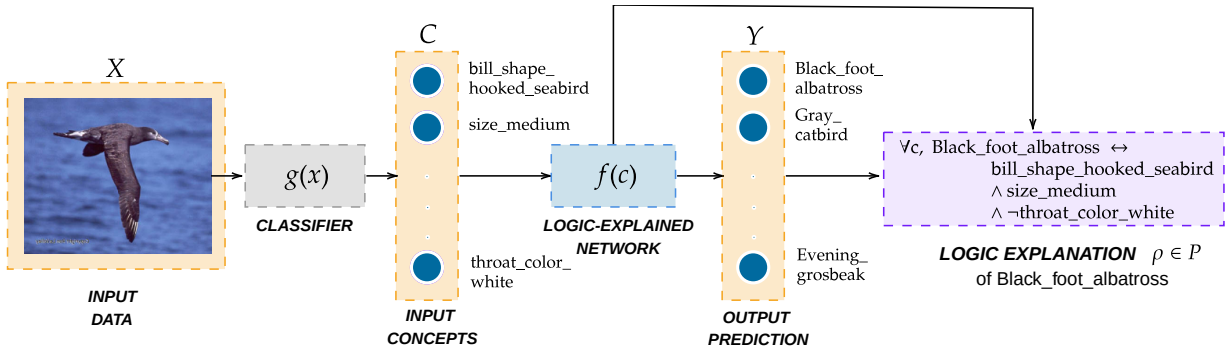


Figure 2: An example of a possible instance of the generic model of Fig. 1, inspired by the CUB 200-2011 fine-grained classification dataset. Classes are divided into a two-level hierarchy. A LEN is placed on top of a convolutional neural network $g(\cdot)$ in order to *(i)* classify the species of the bird in input and *(ii)* provide an explanation on why it belongs to this class. The logic explanation in the example showcases the predicted output class (all the output concepts can be explained), dropping the argument of the predicates for compactness.

- It generalizes existing neural methods for solving and explaining categorical learning problems [37, 36] into a broad family of neural networks i.e., the *Logic Explained Networks* (LENs). In particular, we extend the use of ψ networks also to directly provide interpretable classifications and we introduce other two main instances of LENs, i.e. ReLU networks and μ networks.
- It describes how users may interconnect LENs in the classification task under investigation, and how to express a set of preferences to get one or more customized explanations.
- It shows how to get a wide range of logic-based explanations, and how logic formulas can be restricted in their scope, working at different levels of granularity (explaining a single sample, a subset of the available data, etc.).
- It reports experimental results using three out-of-the-box preset LENs showing how they may generalize better in terms of model accuracy than established white-box models such as decision trees on complex Boolean tasks (in line with Tavares' work [38]).
- It advertises our public implementation of LENs in a GitHub repository[3] with an extensive documentation about LENs models, implementing different trade-offs between intepretability/explainability and accuracy.

The paper is organized as follows (see also Fig. 3). Related works are described in Section 2. Section 3 gives a formal definition of a LEN and it describes its underlying assumptions, key paradigms and design principles. The methods used to extract logic formulas and to effectively train LENs are described in Section 4. Three out-of-the-box LENs, presented in Section 5, are compared on a wide range of benchmarks in terms of classification performance and quality of the explanations, in Section 6, including an evaluation of the LEN rules in an adversarial setting. Finally, Section 7 outlines the social and economical impact of this work as well as future research directions.

2 Related work

In the last few years, the demand for human-comprehensible models has significantly increased in safety-critical and data-sensible contexts. This popularity is justified by the emerging need for unveiling the decision process of pitch-black models like deep neural networks. To this aim, the scientific community has developed a variety of XAI techniques, with different properties and goals. Several taxonomies have been proposed to categorize the XAI models, with partial overlapping and some ambiguities on the referred terminology (such as the distinction between interpretability and explainability). Without pretending to be exhaustive, in the following we focus on some key properties that are relevant to compare LENs with other existing approaches. In particular, we will describe related XAI methods considering the type of RESULT they provide (*feature-scoring* vs. *rule-based*), their specific ROLE (*interpretable models* vs. *explanation methods*), and the SCOPE of the provided explanations (*local* vs. *global*). A brief summary of a few XAI works in terms of these features is reported in Tab. 1 – for more details on existing taxonomies we refer to the recent surveys [39, 8, 12, 22].

[3] https://github.com/pietrobarbiero/logic_explained_networks

XAI models can be differentiated according to the RESULT of the produced explanation. Most of the methods in literature usually focus on scoring or providing summary statistics of features [40, 41, 42, 43, 44, 45, 46]. However, *feature-scoring* techniques can be of scarce utility in decision support cases, whereas a comprehensible language may bring light on the black-box behaviour by identifying concept-based correlations. On the other hand, *rule-based* methods are generally more comprehensible [18, 47, 20], since they usually rely on a formal language, such as FOL. Further, the learned rules can be directly applied to perform the learning task in place of the explained model. Existing approaches have different ROLES in the XAI landscape, acting as intrinsically *interpretable models* or as *explanation methods*. As a matter of fact, the interpretability of a model can be achieved either by constraining the model to be interpretable per se (*interpretable models*) or by applying some methods to get an explanation of an existing model (*explanation methods*). Interpretable models are specific kinds of models whose decision process is considered transparent, while explaining models can be either agnostic with respect to the target model they aim at explaining, or they can be designed to work for a specific type of target model. In principle, interpretable models are the best suited to be employed as decision support systems. However, their decision function often is not smooth which makes them quite sensible to data distribution shifts and impairs their generalization ability on unseen observations [38, 22]. 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, as explanations are produced once the training procedure is concluded. Finally, a fundamental feature to distinguish among explainable methods is the SCOPE of the provided explanations. *Local* explanations are valid for a single sample, while *global* explanations hold on the whole input space. However, several models consider local explanations together with aggregating heuristics to get a global explanation.

Sometimes the easiest way to solve the explanation problem is simply to treat the model as a black-box and, sample-by-sample, determine which are the most important features for a prediction. Prominent examples of algorithms falling in this area accomplish this task by perturbing the input data. Local Interpretable Model-agnostic Explanations (LIME) [44] trains a white-box model (e.g. a logistic regression) to mimic the predictions of a given model in the neighbourhood of the sample to explain. By analyzing the weights of the white-box model it identifies the most important group of pixels (superpixel). Differently, SHapley Additive exPlanations (SHAP) [45] computes the Shapley value of each feature by removing each of them in an iterative manner. Other important techniques provide the same type of explanations through gradient analysis. For instance, Erhan, Courville, and Bengio introduced the Activation Maximization [40] framing this as an optimization problem. They explain the behaviour of a hidden/output unit by slightly modifying a given input data such that it maximizes its activation. An easier way for identifying the most relevant input features consists in computing Saliency Maps [41], which backtrack the classification loss back to the input image. Other post hoc methods have been devised to extract global feature-scoring explanations, often starting from local explanations. A submodular-pick algorithm extends the LIME approach (SP-LIME) to provide global explanations [44]. SP-LIME first finds superpixels of all input samples with the standard LIME procedure. Successively, it identifies the minimum number of common superpixels covering most of the images. Starting from the idea of Activation Maximization, Class Model Visualization [41] searches the input sample maximizing class probability scores in the whole input space. While inheriting several of the above properties, LENs take a different perspective, as they aim at providing human-comprehensible explanations in terms of FOL formulas (see Tab. 1).

Existing XAI methods can also supply FOL explanations. LOcal Rule-based Explanations of black-box decision systems (LORE) [48] extracts FOL explanations via tree induction. Here the authors focus on local rules extracted

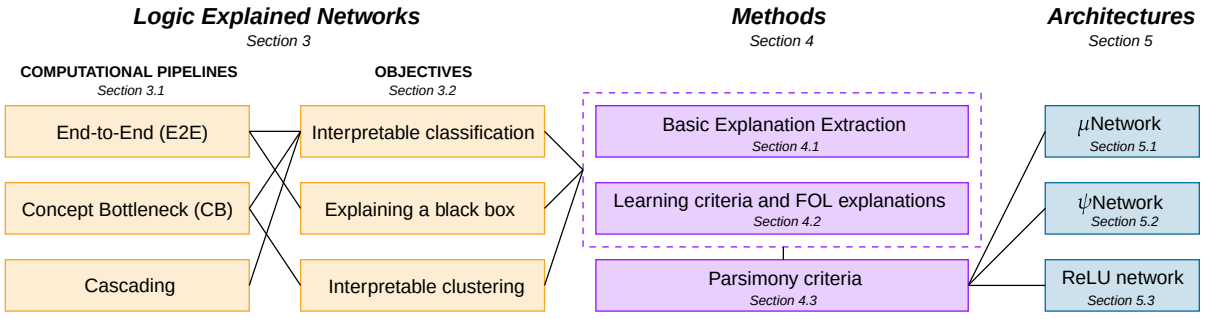


Figure 3: Visual overview of the organization of the paper for those sections that are about describing the whole LEN framework and the specific use-cases we selected. Starting from the nodes on the left, each path that ends to one of the nodes on the right creates a specific instance of the LEN framework. The proposed framework is generic enough to create several other instances than the ones we study in this paper.

	Result Type		Scope		Role	
	feature-scoring	rule-based	local	global	interpret. model	expl. method
LIME	✓		✓			✓
SHAP	✓		✓			✓
Activation Maximization	✓		✓			✓
Saliency Maps	✓		✓			✓
SP-LIME	✓			✓		✓
Class Model Visualization	✓			✓		✓
LORE		✓	✓			✓
Anchors		✓	✓			✓
DeepRED		✓	✓	✓		✓
GAM	✓			✓	✓	
Decision Trees		✓	✓	✓	✓	
BRL		✓	✓	✓	✓	
LENs		✓	✓	✓	✓	✓

Table 1: Summary of related work. The first column lists a number of approaches in the context of XAI. The other columns are about different properties. See the paper text for more details.

through input perturbation. For each sample, a simple decision tree is trained to mimic the behaviour of the black-box model in the neighbourhood of the sample. Both standard and counterfactual explanations can then be extracted following the branches of the tree. Also the Anchors method [49], based on LIME, provides local rules of black-box model predictions via input perturbation. In this case, rules are not extracted from a decision tree but are generated by solving a Multi-Armed-Bandit beam search problem. The rules extracted by Anchors can explain with high precision the behaviour of a classifier on a certain sample point. However, these local explanations are very specific and, for example, they do not provide an overall global explanation of a certain class. Differently, the LEN explanations are way more versatile than the Anchors ones, yielding both class-level explanations and example-level ones. Recent work introduced δ-relevant sets [50], that are shown to be a generalization of both Anchors and Prime Implicant Explanations [51], in order to provide probabilistic explanation of the behaviour of a certain classifier. δ-relevant sets aim at determining a minimum-sized set of features that, if fixed to some predefined value v, ensure that the probability of the prediction on other inputs to be the same as the one for v, is no less than δ. On the other hand, finding a minimal δ-relevant sets is in general intractable, as it has been shown to be NP complete [50], even if for some specific classifiers, such as decision trees, it can be solved with a polynomial number of calls to an NP oracle [52]. LENs follow a related but different strategy. Indeed, LENs determine the relevant subsets of input concepts that provide low-complexity explanations by using ad-hoc pruning strategies and regularization techniques, both of them participating to the learning procedure of neural networks. The Quine–McCluskey algorithm helps LENs keep low the complexity of the formulas, once all the example-level explanations have been aggregated into a class-level explanation. Extending the CRED algorithm [53], DeepRED [54] employs a decomposition strategy to globally explain neural networks. Starting from the output nodes, the predictions of each neuron are explained in terms of the activations of the neurons in the previous layer by training a decision tree. In the end, all rules for each class are merged in a single global formula in terms of input features. For a given sample, a unique local rule is extracted following the firing path. The approach proposed in this paper resembles the DeepRED algorithm. As it will become clear in the following sections, LENs can explain the behaviour of a neural network both end-to-end and layer by layer. However, LENs can be used to explain any black-box model as far as its input and output correspond to human-interpretable categories (Section 3). Furthermore, LENs support different forms of FOL rules, with different scopes and goals.

Interpretable models are capable of providing both local and global explanations. Such explanations may be based either on feature rankings, as in Generalized Additive Models (GAM) [55], or on logic formulas, as in Decision Trees [18] or Bayesian Rule Lists (BRL) [20]. GAMs overcome the linearity assumption of linear regression by learning target categories disjointly from each feature. Caruna et al. proposed GA^{2}M where pairs of features are allowed to interact [56]. A further extension employing neural networks as non-linear functions has recently been proposed by Agarwal et al. [57], which, however, loses interpretability at feature-level. Decision trees [18] are a greedy learning algorithm partitioning input data into smaller subsets until each partition only contains elements belonging to the same class. Different pruning algorithms have been proposed to simplify the final structure to get simple explanations [58].

Each path of a decision tree is equivalent to a decision rule, i.e. an *IF-THEN* statement. Other approaches focus on generating sets of decision rules, either with sequential covering [59] or by selecting the best rules from a pre-mined set via Bayesian statistics [20]. Interestingly, LENs can be used to solve a learning task directly, as they are interpretable per se. Employing an interpretable-by-design neural network has the advantage that the extracted explanations will perfectly match the classifier predictions. In addition, relying on a neural network, LEN generalization performance can be much better than standard interpretable-by-design methods, whose representation capacity is usually limited.

To sum up, the proposed family of neural networks can be used both as interpretable classifiers and as surrogate models. Indeed, the flexibility of the proposed framework allows the user to find an appropriate trade-off between interpretability/explainability and accuracy that is well suited for the task at hand. Concerning the explanation type, LENs provide explanations as FOL formulas. The inner mechanism to extract such explanations is general enough to cover rules with different scopes, from local rules valid on a single sample to global rules that hold for an entire class. It is important to remark that LENs differ from approaches that follow the idea of learning a set of constraints and that use logic as a formal way to express them [60, 61, 62]. For example, we mention the significant amount of work by the Inductive Logic Programming community, focused on learning a set of FOL formulas with the goal of covering all the positive facts and no negative ones [63, 64, 65]. These approaches are generally limited to the case of learning hard-constraints with all the resulting algorithmic complexity issues. Other models, such as Markov Logic Networks [66] and Probabilistic Soft Logic [67], aim at discovering the relevance (weights) of a set of formulas defined within specific templates/structures (with different extensions aimed at automatizing the structure learning procedure [68]), thus allowing to deal with partially violated rules from a statistical perspective. Finally, there exist approaches that aim at learning both clauses (structures) and weights, such as [69, 70, 71]. Of course, LENs include several aspects that are common with the aforementioned approaches (the emphasis on FOL, the importance of learning from data, etc.). However, differently from them, LENs are formulated in the classic learning framework of neural networks, adding some design features to facilitate the extraction of meaningful logic rules. Only at the end of the training stage LENs provide a symbolic interpretation of the target networks.

3 Logic Explained Networks

This work presents a special family of neural networks, referred to as Logic Explained Networks (LENs), which are able to both make predictions and provide explanations. LENs may explain either their own predictions or the behaviour of another classifier, being it a neural network or a generic black-box model. Moreover, a LEN might also be used to explain relationships among some given data. In this section, we describe how LENs can be instantiated in the framework of Fig. 1. We start by introducing the notation (following paragraphs). Then we discuss a variety of computational pipelines (Section 3.1) and illustrate different types of explanations LENs can generate in function of different learning objectives (Section 3.2). In Fig. 3 (leftmost part) we provide an overview of the contents of this section. In the same figure, we also show how the following sections will cover the specific methods (Section 4) and the considered neural architectures (Section 5).

A LEN is a function f, implemented with a neural network, that maps data falling within a k-dimensional Boolean hypercube onto data falling within another r-dimensional Boolean hypercube. Each dimension is about the activation strength of what we refer to as a *concept*, with the strict requirement of having a human-understandable description of each *input* dimension/concept [31]. Formally, a LEN is a mapping $f : C \rightarrow E$, where $C = [0, 1]^k$ is the *input concept space*, and $E = [0, 1]^r$ is the so-called *output concept space*. FOL explanations produced by a LEN are about the relationships between the output and the input concepts, and they belong to the generic *rule space* P. In particular, whenever a LEN has been trained, the i-th output f_i can be directly translated into a logic rule φ_i that involves the input concepts and that is leveraged to devise a FOL formula $\rho_i \in P$, such as the one shown in Fig. 2. The FOL extraction process is general enough to offer logic rules with different levels of granularity, from a local to a more global coverage of the available data.

In order to provide a concrete meaning to LEN models, the source of the input concept activations needs to be defined as well as the learning criteria. Amongst a variety of possible configurations that are instance of Fig. 1, we will focus on a limited set of computational pipelines in which the input and output concept spaces are defined with respect to real-world use cases. In some of them, LENs communicate with a black-box classifier, with the goal of providing explanations of its predictions, or with the goal of leveraging its activations as input concepts. For each input sample, each component of the LEN output (r components) corresponds to a FOL explanation that can be directly associated to: (i) a specific category of a classification problem, e.g. to explain how such a category is related to a set of concepts; (ii) an explanation that formalizes discovered relationships involving a set of concepts, e.g. to explain how they are logically connected. In the former case, LENs are trained in a supervised manner, while in the latter case they are trained in an unsupervised setting.

Before going into further details, we introduce the main entities and the notation that will be used throughout the paper, paired with a short description to help the reader in following the paper and to have a quick reference to the main symbols.

f, C, E: The function computed by a LEN is $f\colon C \to E$, where $C = [0,1]^k$ is the space of the activations of the k input concepts, and $E = [0,1]^r$ is about the activations of the r output concepts.

$X, \mathcal{X}, \mathcal{C}$: We consider a scenario in which a finite collection $\mathcal{X}$ of data samples that belong to $X \subseteq \mathbb{R}^d$ is available to train LENs. There exists a mapping from X to the space C of input concept activations. The notation $\mathcal{C}$ indicates the finite set of concept activations obtained by mapping each sample of $\mathcal{X}$ onto C.

q: We use the notation q to indicate the total number of classes in a classification problem. Each data sample can be associated with one or more classes. No strict conditions are posed on the potential mutual exclusivity of the different classes, thus we consider the most generic setting that spans from multi-label to single-label classification. Moreover, classes could also be organized in a structured manner, such as in hierarchy.

$\bar{y}, \bar{y}_i$: Whenever we consider classification problems, classes are assumed to be encoded with binary targets in $\{0,1\}^q$. The function $\bar{y}(\cdot)$ returns the target vector associated to the data sample passed as its argument, while $\bar{y}_i(\cdot)$ returns the i-th component of such a vector, that is about the i-th class. What we propose holds also in the case in which targets are encoded with scores in the unit interval, so that we will frequently refer to generic data encoded in $Y = [0,1]^q$.

$Y^{(a:b)}$: We will use the notation $Y^{(a:b)}$ to indicate a view of Y limited to the dimensions going from index $a \geq 1$ to index $b \leq q$, included.

g: The function $g(x)$ is about a generic black-box neural classifier that computes the membership scores of x to a set of categories. Such categories could be exactly the ones of the considered classification problem (encoded in Y) or a subset of them (encoded in $Y^{(a:b)}$). In detail, $g : X \to Y$ (resp. $g : X \to Y^{(a:b)}$), and $g(x)$ defines the membership scores of x to the considered categories. Of course, $g_i(x) = 1$ when x strongly belongs to class i (resp. $a + i - 1$). No special conditions are enforced in the definition of g.

$\bar{f}, \bar{f}_i$: The notation $\bar{f}$ is used to indicate a Boolean instance of the main LEN function f, in which each output of f is projected to either 0 or 1.[4] In order to refer to a single output of the vector function f (or $\bar{f}$), the subscript will be used, e.g. f_i (or $\bar{f}_i$).

$c_j, \bar{c}_j$: Similarly, for a vector $c \in C$ with activation scores of k concepts, c_j is the j-th score, while $\bar{c}_j$ is the Boolean instance of it (as in the case of $\bar{f}$ with respect to f).

$\bar{c}_j$ (name): Any dimension of the space of input concepts C includes a human-readable label. Logic rules generated by LENs will leverage such labels to give understandable names to predicates. In order to simplify the notation, whenever we report a logic rule, we will use the already introduced symbol $\bar{c}_j$ also to refer to the human-understandable name of the j-th input concept. This notation clash makes the presentation easier.

$\bar{y}_i$ (name): Any dimension of the space of output concepts E might or might not include a human-readable label, whether we consider supervised or unsupervised learning when training LENs, respectively. In the former case, the already introduced notation $\bar{y}_i$ will also refer to the human-understandable name of the i-th output class, following the same simplification we described in the case of $\bar{c}_j$.

φ_i: Any output f_i is associated with a logic rule φ_i given in terms of (the names of) the input concepts.

ρ_i: We indicate with $\rho_i \in P$ the FOL formula that explains the i-th output concept leveraging φ_i. The precise way in which φ_i is used to build ρ_i depends on whether we are considering supervised or unsupervised output concepts.

The listed elements will play a precise role in different portions of the LEN framework and the different figures throughout the paper report concrete examples of the just introduced notation. We notice that we only explicitly report the expression "$\forall c$" in Figure 2 as an example, whereas we drop the quantification on all the other figures for simplicity.

3.1 Computational pipelines

Amongst a large set of feasible input/output configurations of the LEN block, here we consider three different computational pipelines fitting three concrete scenarios/use-cases where LENs may be applied, that consist in what we refer to as END-TO-END, CONCEPT BOTTLENECK, and CASCADING pipelines.

[4] When not directly specified, we will assume 0.5 to be used as threshold value to compute the projection.

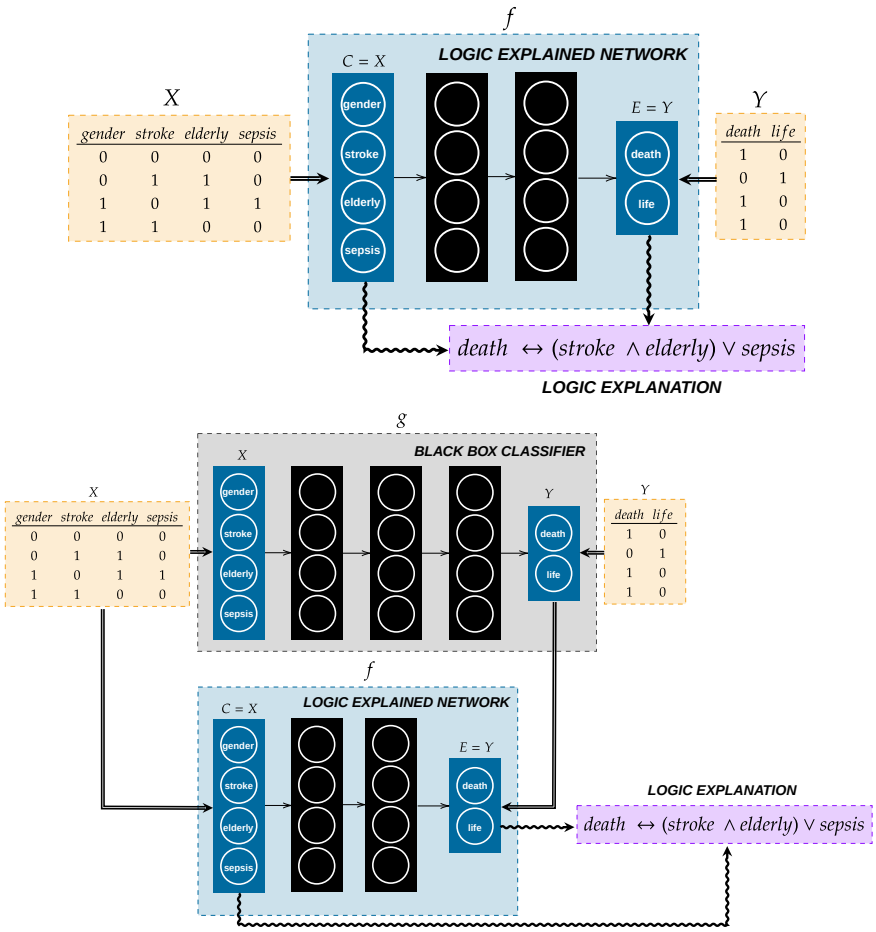


Figure 4: End-to-end (E2E) LENs directly work on input data that are interpretable per se and treated as concepts ($C = X$), while the output concepts are the activation scores of the classes of the dataset ($r = q = 2$). This is a real-world example from the MIMIC II dataset (see the experimental section), where patient features are used to classify if the patient will survive 28 days after hospital recovery. Top: the LEN solves the classification problem and provides explanation, also referred to as *interpretable classification*. Bottom: the LEN *provides explanations* of a black-box classifier. The universal quantifier and the argument of the predicates have been dropped for simplicity.

END-TO-END (E2E). The most immediate instance of the LEN framework is the one in which the LEN block aims at directly explaining, and eventually solving, a supervised categorical learning problem. In this case, we have $C = X$, and $r = q$, as shown in Fig. 4. In order to make the first equality valid with respect to the LEN requirements, the d input features of the data in X must score in $[0, 1]$ (or in $\{0, 1\}$ in the most extreme case), so that they can be considered as activations of d human-interpretable concepts. In order to create these conditions in different types of datasets, generic continuous features can be either discretized into different bins or Booleanized by comparing them to a reference ground truth (e.g. for gene expression data, the gene signature obtained from control groups can be used as a reference). (i) An E2E LEN can perform a fully *interpretable classification* task, having the double role of main classifier and explanation provider (Fig. 4, top), or (ii) it can work in parallel with another neural black-box classifier g, thus providing explanations of the predictions of g (Fig. 4, bottom) – what we refer to as *explaining a black-box*. In both the cases (i) and (ii) we have $E = Y$. The first solution is preferred when a white-box classifier is of utmost importance, while a loss in terms of classification accuracy is acceptable, due to the constrained nature of the LENs. The second solution is useful in contexts where the classification performance plays a key role, and approximate explanations of black-box decisions are sufficient.

CONCEPT-BOTTLENECK (CB). A Concept-Bottleneck LEN is a computational pipeline in which the LEN f aims at explaining, and eventually solving, a categorical learning problem whose features do not correspond to human-interpretable concepts and, as a consequence, they are not suitable as LEN inputs. In this case, a black-box model g computes the activations of an initial set of z concepts out of the available data $\mathcal{X}$. The LEN solves the problem of

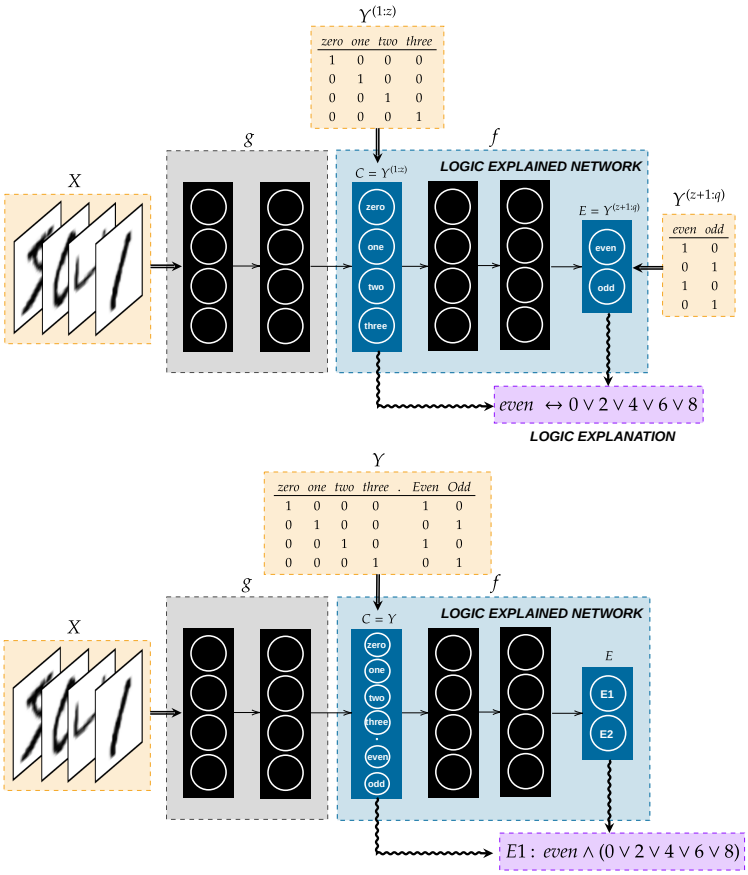


Figure 5: Concept-bottleneck (CB) LEN. The LEN is placed on top of a black-box model g which maps the input data into a first set of interpretable concepts. An MNIST-based experiment is shown (see the experimental section). Top: Handwritten digits are first classified by g. A LEN is then employed to classify and explain whether the digit is even or odd – *interpretable classification*. Bottom: MNIST digits are classified as belonging to one of the 10 classes and whether they are even of odd by a black-box g. A LEN groups these predictions in an unsupervised manner, and analyzes the relations within each cluster – *interpretable clustering*. In the latter, supervision labels are not provided for the LEN output. The universal quantifier and the argument of the predicates have been dropped for simplicity.

predicting r new concepts from the activations of the z ones, as shown in Fig. 5 (top/bottom). Differently from the E2E case, there is always a neural network g processing the data, and the outcome of such processing is the input of the LEN. Formally, we have $g : X \to Y^{(1:z)}$, while $f : C \to E$ with $C = Y^{(1:z)}$, with $z \leq q$,[5] and where the meaning of E varies in function of what we describe in the following. (i) This two-level scenario can be implemented in a way that is coherent with the already discussed example of Fig. 2, that is what we also show in Fig. 5 (top). In this case, each example is labeled with q binary targets divided into two disjoint sets, where the black-box network g predicts the first $z < q$ ones, and the LEN f predicts the remaining $r = q - z$, i.e. $E = Y^{(z+1:q)}$. The LEN will be both responsible of predicting the higher level concepts and of explaining them in function of the lower level concepts yielded by the black-box model, thus still falling withing the context of *interpretable classification*. (ii) We also consider the case in which the outcome of the LEN is not associated to any known categories. In this case, the LEN is trained in an unsupervised manner, as shown in Fig. 5 (bottom), thus implementing a form of *interpretable clustering*. Formally, $g : X \to Y$, $C = Y$, and $E = [0, 1]^r$, with customizable $r \geq 1$. In both the cases (i) and (ii), the black-box classifier g is trained in a supervised manner.

CASCADING. Cascading LENs can be used to provide explanations by means of a hierarchy of concept layers. In particular, multiple LENs are used to map concepts into a higher level of abstraction (without the need of any black-box g), as shown in Fig. 6. Each LEN provides explanations of its outputs with respect to its inputs, allowing

[5] We implicitly assumed the axes of Y to be sorted so that the first z dimensions are the ones we want to predict with g, and we will make this assumption in the whole paper.

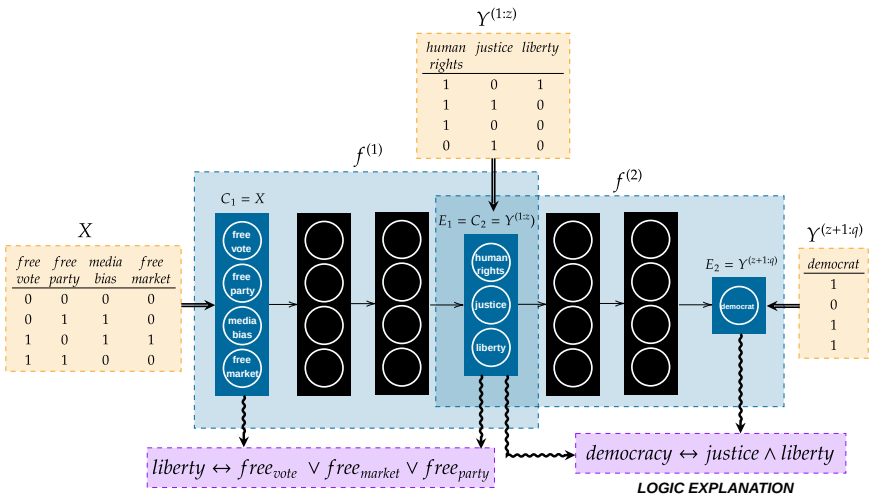


Figure 6: Cascading LENs, an example taken from the V-Dem classification dataset (see the experimental section). The final classification on the status of the democracy is divided into two steps. First, the input data/concepts C_1 are mapped into high-level concepts C_2. High-level concepts are then used to compute the final classification. Cascading LENs can provide explanations at different levels of granularity, implementing multiple *interpretable classifications*. The universal quantifier and the argument of the predicates have been dropped for simplicity.

LENs to handle multiple levels of granularity. This structure provides a hierarchy of explanations and upholds human interventions at different levels of abstraction, providing a far deeper support for human-machine interactions. We indicate with $(C_1, E_1), (C_2, E_2), \ldots, (C_s, E_s)$ the input/output concept spaces of the s cascading LENs, with $C_j = E_{j-1}$, $j > 1$. In the experiments, we will consider the case in which LENs are trained in a supervised manner, thus implementing multiple *interpretable classifications*.

3.2 Objectives

In Section 3.1, when describing the selected computational pipelines, we made an explicit distinction among INTERPRETABLE CLASSIFICATION, EXPLAINING A BLACK-BOX, and INTERPRETABLE CLUSTERING, as they represent three different "objectives" the user might consider in solving the problem at hand. These objectives impose precise constraints in the selection of the learning criteria and on the form of the FOL formulas that can be obtained by the LENs. The form of FOL formulas will be described in detail in Section 4.1 (generic process of logic rule extraction), and in Section 4.2 (learning criteria and FOL rules). The key difference among the aforementioned objectives is about the criterion that drives the learning dynamics of f, as each LEN module can be trained in a supervised or an unsupervised fashion.

INTERPRETABLE CLASSIFICATION. Whenever LENs are leveraged to both solve the classification problem and provide explanations, i.e. in INTERPRETABLE CLASSIFICATION, learning is driven by supervised criteria. Following Ciravegna et al. [37], supervised criteria leverage the available data so that: (i) f learns class labels as in classic supervised learning, and (ii) f is constrained to be coherent with the target type of FOL rules. Each output neuron of the LEN is associated to a target category named $\bar{y}_i$ of the considered learning problem. LENs can provide a FOL explanation $\rho_i \in P$ of such category, so that the class-predicate $\bar{y}_i(\cdot)$ will be involved in the extracted FOL formula. For example, if *person* is one of the output categories, LENs can explain the reasons behind the prediction of class $\bar{y}_i = person$ by means of an automatically discovered relationship φ_i involving the activations/not-activations of some (m_i) of the k input concepts ($m_i < k$). If such input concepts are $\bar{c}_j = head$, $\bar{c}_z = hands$, $\bar{c}_h = body$, the system could learn that $\rho_i = \forall c \in C\colon \bar{y}_i(c) \leftrightarrow \varphi_i(c)$, where $\varphi_i(c) = \bar{c}_j(c) \wedge \bar{c}_z(c) \wedge \bar{c}_h(c)$, i.e. (discarding the quantifier) *person* $\leftrightarrow$ *head* $\wedge$ *hands* $\wedge$ *body*. Interestingly, in the case of interpretable classification, LENs basically become white-box classifiers, as is showcased by the concrete examples of Fig. 4 (top), Fig. 5 (top), and Fig. 6.

EXPLAINING A BLACK-BOX. In case the user aims at EXPLAINING A BLACK-BOX, LENs act in parallel to black-box classifiers with the goal of explaining the decisions of such black-box models. In this scenario, LENs are constrained to mimic the predictions of the black-boxes, i.e. f is forced to be close to g when evaluated on the available data samples. This is what is shown in Fig. 4 (bottom) and it may not require any supervision, since learning can be driven by a coherence criterion between the outputs of g and those of f when processing the same data. The type of FOL rules LENs can extract are the same of the case of interpretable classification, thus the same principles are followed.

INTERPRETABLE CLUSTERING. Differently, LENs can be trained using unsupervised criteria, whenever the user is not aiming at explaining the target classes of a classification problem, but is interested in discovering generic relations among input concepts. We refer to this objective as INTERPRETABLE CLUSTERING. The user might be interested in discovering co-occurrences φ_i of input concept activations in not-defined-before subsets O_i of the concept space. This leads to FOL rules such as $\rho_i = \forall c \in O_i \colon \varphi_i(c)$, where, for example, $\varphi(c) = \bar{c}_e(c) \vee \bar{c}_t(c)$ and $\bar{c}_e = soccer\_ball$, $\bar{c}_t = foot$, and the set O_i can be thought as a cluster. In this scenario, LEN's output neurons are not associated to any known categories (in contrast to previous objectives), but to unspecified generic symbols, as shown in Fig. 5 (bottom). The activation score of the output concept represents a cluster membership score and it is used to define whether an input pattern belongs to the subset O_i or not.

In all the discussed objectives, logic formulas are extracted from LENs using the same principles. Rules are then instantiated into FOL formulas that well cope with objective-specific learning criteria. As it will become clear in the following section, due to such generality of the extraction mechanisms, the user might decide to automatically discover definite regularities focused on single examples, groups of data points, or, more generally, the whole dataset, moving from local to more global explanations.

4 Methods

This section presents the fundamental methods used to implement Logic Explained Networks introduced in Section 3. We start by describing the procedure that is used to extract logic rules out of LENs for an individual observation or a group of samples (Section 4.1). This procedure is common to all LENs' objectives/use-cases. Then, we provide the formal definition of the learning objectives constraining LENs to provide required types of FOL explanations (Section 4.2). Finally, we discuss how to constrain LENs to yield concise logic formulas. To this aim, ad-hoc parsimony criteria (Section 4.3) are employed in order to bound the complexity of the explanations. In Fig. 3 (middle) we provide an overview of the contents of this section.

4.1 Extraction of logic explanations

Once LENs are trained, a logic formula can be associated to each output concept f_i. As it will become clear shortly, extracting logic formulas out of trained LENs can be done by inspecting its inputs/outputs accordingly to their Boolean interpretation. We have already introduced the notation φ_i to indicate the logic explanation of the output concept i. This generic notation will be properly formalized in the following. We overload the symbol φ_i to explicitly indicate, when needed, the data subset where the logic explanation holds true, using the notation $\varphi_{i,\cdot}$. Here the second subscript can refer either to a single data sample c, $\varphi_{i,c}$, or to a set S of data samples, $\varphi_{i,S}$. In practice, S denotes the region of the concept space that is covered by the i-th explanation, i.e. the set of concept tuples for which the formula $\varphi_{i,S}$ is true. By aggregating over multiple samples, the scope of the logic formula may be tuned from strictly local EXAMPLE-LEVEL EXPLANATIONS ($S = \{c\}$) to SET-LEVEL EXPLANATIONS ($S \subseteq \mathcal{C}$), where the latter can be focused on a precise class, i.e., CLASS-LEVEL EXPLANATIONS. Eventually, for $S = \mathcal{C}$, global logic formulas holding on the whole concept space $\mathcal{C}$ can be extracted.

To allow the extraction of FOL formulas, any LEN $f = (f_1, \ldots, f_r)$ requires both its inputs and outputs to belong to the real-unit interval. This apparent limitation allows any f_i, for $i = 1, \ldots, r$, to correspond to a logic formula. First, f maps the data in $\mathcal{C}$ into the rule space E. After this forward pass, both the input data $\mathcal{C}$ and the predictions of f are thresholded, e.g. with respect to 0.5, to obtain their Boolean values. Then, for each output neuron i, an *empirical truth-table* $\mathcal{T}^i$ is built by concatenating the k-columns of Booleanized input concept tuples $\{\bar{c} : \ c \in \mathcal{C}\}$, with the column of the corresponding LEN's predictions $\bar{f}_i(c)$ (left-side of Fig. 7). The truth-table $\mathcal{T}^i$ can be converted into a logic formula φ_i in Disjunctive Normal Form (DNF) as commonly done in related literature [72]. However, the rationale behind LENs is to extract formulas that are compact, emphasizing the most relevant relationships among the input concepts, according to specific parsimony indexes (that will be the subject of Section 4.3). Thus, any f_i will depend only on a proper subset of $m_i \leq k$ concepts, and the formula φ_i will be built according to the restriction of $\mathcal{T}^i$ to $m_i \leq k$ columns (see e.g. Fig. 7). Notice that, for convenience in the notation, we assumed the first m_i columns to be the ones playing a role in the explanation, even if they could be any set of m_i columns of $\mathcal{T}^i$. In order to give more details about the rule extraction, we formally introduce the set $O_i = \{c \in \mathcal{C} : \bar{f}_i(c) = 1\}$ as the set of all the sampled concept tuples that make true the i-th output explanation, i.e. the *support* of $\bar{f}_i$.

EXAMPLE-LEVEL EXPLANATIONS. Given a sample $c \in O_i \subseteq \mathcal{C}$, the Booleanization $\bar{c}$ of its continuous features may provide a natural way to get an example-level logic explanation $\varphi_{i,c}$. To make logic formulas more interpretable, the notation $\tilde{c}$ denotes human-interpretable strings representing the concept names or their negation,

$$\varphi_{i,c} = \tilde{c}_1 \wedge \ldots \wedge \tilde{c}_{m_i} \qquad \text{where } \tilde{c}_j := \begin{cases} \bar{c}_j, & \text{if } c_j \geq 0.5 \\ \neg \bar{c}_j, & \text{if } c_j < 0.5 \end{cases}, \text{ for } j = 1, \ldots, m_i \tag{1}$$

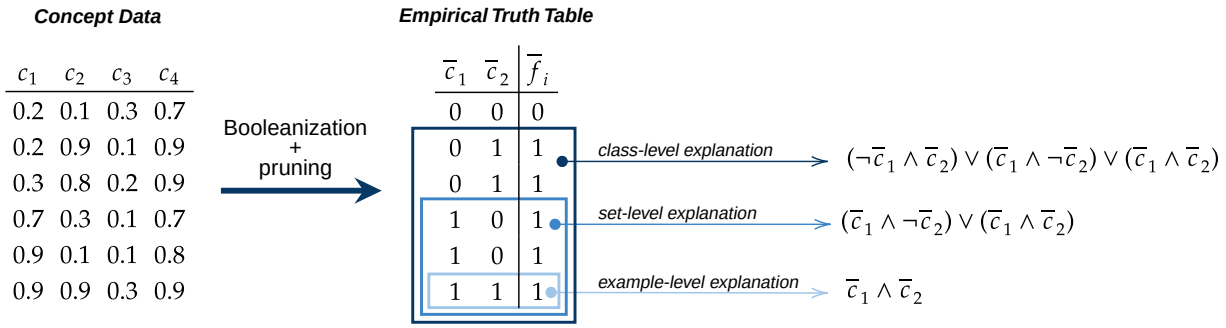


Figure 7: Empirical truth table $\mathcal{T}^i$ of the i-th LEN output f_i, with $k = 4$ and $m_i = 2$ (assuming that only the first two input concepts are kept). The aggregation of concept tuples with the same Booleanization yields a common example-level explanation, and do not complicate the class-level explanation. For any example-level explanation, we may count how many samples it explains, to discard the most infrequent cases.

SET-LEVEL AND CLASS-LEVEL EXPLANATIONS. By considering Eq. 1 for all $c \in S$, with $S \subseteq O_i$, and aggregating all the example-level explanations, an explanation for a set of samples can be generated as follows:

$$\varphi_{i,S} = \bigvee_{c \in S} \varphi_{i,c} = \bigvee_{c \in S} \tilde{c}_1 \wedge \ldots \wedge \tilde{c}_{m_i} \tag{2}$$

As some $\varphi_{i,c}$'s might be equivalent for different c's, repeated instances can be discarded keeping only one of them, without loss of generality. In case $S = O_i$, we simply write φ_i in place of $\varphi_{i,S}$ and we refer to such set-level explanation as the CLASS-LEVEL EXPLANATION of the i-th output concept f_i.

In the rest of this section we will explore further details of what has been described so far. We will start by the following example.

Example 1. *Let's consider the Boolean XOR function, defined by $xor(0,0) = xor(1,1) = 0$, $xor(1,0) = xor(0,1) = 1$. Let $f = [f_1]$ be a LEN that has been trained to approximate the XOR function in the input space $C = [0,1]^2$. Then, if we consider, e.g. the inputs $c^1 = (0.2, 0.7)$, $c^2 = (0.6, 0.3)$, we get $\bar{c}^1 = (0,1)$, $\bar{c}^2 = (1,0)$, and therefore $\bar{f}_1(c^1) = \bar{f}_1(c^2) = 1$. These examples yield the example-level explanations $\varphi_{1,c^1} = \neg\bar{c}_1 \wedge \bar{c}_2$ and $\varphi_{1,c^2} = \bar{c}_1 \wedge \neg\bar{c}_2$ respectively. As a result, the class-level explanation for f_1 is given by $\varphi_1 = (\neg\bar{c}_1 \wedge \bar{c}_2) \vee (\bar{c}_1 \wedge \neg\bar{c}_2)$, which correctly matches the truth-table of the Boolean XOR function.*

It is worth noting that both example and set-level explanations can be recursively applied in case of LENs with multiple $[0,1]$-valued hidden layers or in case of cascading LENs. For instance, given a cascading LEN, as the one in Fig. 6, we may get both example and class-level explanations of the concepts in $E_1 = C_2$ with respect to the ones in C_1, so as the concepts in E_2 with respect to the ones in C_2 and, in turn, in C_1. The modularity of logic formulas allows the composition of categories at different levels that may express arbitrary complex relationships among concepts.

Logic explanations $\varphi_{i,S}$ generally hold only on a sub-portion S of the sampled concept space $\mathcal{C}$. However, we may get a logic formula providing an explanation holding on the whole $\mathcal{C}$ by means of the disjunction $\varphi = \varphi_1 \vee \ldots \vee \varphi_r$, if we assume $O_1 \cup \ldots \cup O_r = \mathcal{C}$. Since φ can turn out to be of little significance, if we are interested in simpler (and clearer) global explanations we may convert φ into an equivalent φ' in *Conjunctive Normal Form* (CNF). In particular, there always exist r' and some clauses $\varphi'_1, \ldots, \varphi'_{r'}$ such that:

$$\varphi = \bigvee_{i=1}^{r} \varphi_i \equiv \bigwedge_{i=1}^{r'} \varphi'_i = \varphi' \tag{3}$$

As a result, we get a set of r' explanations holding on the whole $\mathcal{C}$, indeed $\varphi'_i(c) = 1$ for every $c \in \mathcal{C}$, $i = 1, \ldots, r'$. Unfortunately, converting a Boolean formula from DNF into CNF can lead to an exponential explosion of the formula. However, after having converted φ_i in CNF, the conversion can be computed in polynomial time with respect to the number of minterms in φ_i [73].

The methodologies described so far illustrate how logic-based explanations can be aggregated to produce a wide range of explanations, from the characterization of individual observations to formulas explaining model predictions for all the samples leading to the same output concept activation. The formula for a whole class can be obtained by aggregating all the minterms corresponding to example-level explanations of all the observations having the same concept

output. In theory, this procedure may lead to overly long formulas as each minterm may increase the complexity of the explanation. In practice, we observe that many observations share the same logic explanation, hence their aggregation may not change the complexity of the class-level formula (right-side Fig. 7). In general, "*satisficing*" class-level explanations can be generated by aggregating the most frequent explanations for each output concept, avoiding a sort of "explanation overfitting" with the inclusion of noisy minterms which may correspond to outliers [74]. The criterion we followed to prune noisy terms consists in including in a global explanation only the local formulas that increase the accuracy of the explanation when measured on a validation set, sorting formulas by their support (largest support first).

As a final remark, a possible limitation of LENs can be the readability of logic rules. This may occur when (i) the number of input concepts (the length of any minterm) $k \gg 1$, or (ii) the size of the support $|O_i|$ is very large (possibly getting too many different minterms for any f_i). The greater k is, the more different concepts are available for example-level explanations. As a consequence, it will be less probable to find common minterms for different examples and aggregate them into concise class-level explanations. In these scenarios, viable approaches to generate shorter logic rules are needed to provide interpretable explanations. More details on how to generate concise explanations are in Section 4.3.

4.2 Learning criteria

In this section, we describe some of the loss functions allowing LENs to provide FOL explanations $\rho_i \in P$, according to the objectives introduced in Section 3.2 (INTERPRETABLE CLASSIFICATION, EXPLAINING A BLACK-BOX and INTERPRETABLE CLUSTERING). For simplicity, here we will not distinguish among the different computational pipelines, and we will refer to a generic LEN with r output units. We just saw how a logic rule φ_i can be associated to an output concept f_i. However to improve the expressiveness of logic explanations, in this section we will promote φ_i to a FOL formula ρ_i, depending on the selected learning criterion. This follows the usual approach adopted when a model combining logic and machine learning, trained on a finite collection of data, is then applied to out-of-sample inputs, following related studies [36, 37, 75]. As a result, any $\bar{c}_j$ that composes φ_i is thought of as a logic predicate defined on the concept space C, and such that $\bar{c}_j(c) = 1$ if and only if $c_j > 0.5$, for any $c \in C$. In addition, if for instance $\varphi_i = \bar{c}_2 \wedge \neg\bar{c}_5$, we will write $\varphi_i(c)$ for $\bar{c}_2(c) \wedge \neg\bar{c}_5(c)$. Then, the specific choice on the loss function that drives the learning criteria of LENs introduces a link between $\varphi_i(c)$ and the final FOL formulas ρ_i that is produced by the network.

INTERPRETABLE CLASSIFICATION. Supervised learning is needed to extract explanations for specific categories from LENs. This learning approach binds a logic explanation φ_i to a specific output class of a classification problem. By denoting with $\bar{y}_i$ the binary predicate associated to the output class i, we will consider three kinds of FOL explanations ρ_i, between φ_i and $\bar{y}_i$, expressed as the universal closure of an IF, Only IF or IFF rule. These rules can be imposed in case of interpretable classification according to the following learning criteria.

IF rules mean that, in the extreme case, for each sample of class i we want the i-th output of the LEN to score 1 (but not necessarily the opposite). In other words, the set of concept tuples c belonging to the i-th class, i.e. such that $\bar{y}_i(c) = 1$ has to be included in the support of $\bar{f}_i$, while no conditions are imposed when $\bar{y}_i(c) = 0$ (recall that $f_i(c) \in [0, 1]$). This behavior can be achieved by minimizing a hinge loss,

$$L_{\rightarrow}(\bar{y}_i, f_i, \mathcal{C}) = \sum_{c \in \mathcal{C}} \max\{0, \bar{y}_i(c) - f_i(c)\} \qquad i \in [1, r] \tag{4}$$

Following a symmetric approach, in Only IF rules a class is explained in terms of lower-level concepts. This principle is enforced by swapping the two terms in the loss function in Eq. 4,

$$L_{\leftarrow}(\bar{y}_i, f_i, \mathcal{C}) = \sum_{c \in \mathcal{C}} \max\{0, f_i(c) - \bar{y}_i(c)\} \qquad i \in [1, r] \tag{5}$$

Both in IF and Only IF rules, further conditions on f must be included in order to make the learning problem well posed. To this aim, we need to define the behavior of the model in regions of the concept space not covered by training samples in order to avoid trivial solutions with constant f, e.g. $f_i = 1$ or $f_i = 0$ can trivially minimize Eq. 4 or Eq. 5, respectively. For instance, these conditions could require that the f_i's have a fixed bias equal to 1 or no biases at all.

Finally, LENs can learn double implication rules (IFF) which completely characterize a certain class. Our experiments are focused on this type of explanations. Any function penalizing points for which $f_i(c) \neq \bar{y}_i(c)$ may be employed in this scenario, such as the the classic Cross-Entropy loss,

$$L_{\leftrightarrow}(\bar{y}_i, f_i, \mathcal{C}) = \sum_{c \in \mathcal{C}} \bar{y}_i(c) \log(f_i(c)) + (1 - \bar{y}_i(c)) \log\left(1 - (f_i(c))\right) \tag{6}$$

For all the above loss functions (Eq. 4-6), LENs provide logic explanations in First-Order Logic by means of the following equations:

$$\text{IF-rule}: \quad \rho_i = \forall c \in C: \ \bar{y}_i(c) \rightarrow \varphi_i(c) \tag{7}$$

$$\text{Only IF-rule}: \quad \rho_i = \forall c \in C: \ \varphi_i(c) \rightarrow \bar{y}_i(c) \tag{8}$$

$$\text{IFF-rule}: \quad \rho_i = \forall c \in C: \ \bar{y}_i(c) \leftrightarrow \varphi_i(c) \tag{9}$$

where each ρ_i, for $i = 1, \ldots, r$ corresponds to a FOL formula, ranging on the whole concept space C, thus generalizing the relationships discovered on the data samples. In the same way, in case of a concept-bottleneck pipeline, a FOL explanation can be derived from the function $g : X \rightarrow C$ extracting LEN's input concepts from raw features. For instance for the IFF-rule, we will get

$$\rho_i = \forall x \in X: \ \bar{y}_i(g(x)) \leftrightarrow \varphi_i(g(x)) \tag{10}$$

Before considering the remaining learning criteria, a couple of comments are in order here. First, we remark that the logic predicate $\bar{y}_i$ appearing in the Eq. 7-10 is simply a *virtual* predicate denoting the membership of a certain concept tuple $c \in C$ to the i-th output class and that coincides with the available class label of concepts when evaluated on tuples in $\mathcal{C}$. Second, we notice that the correspondence between f_i and $\bar{y}_i$ can be enforced only on the sampled concept space $\mathcal{C}$, ensuring that each formula ρ_i from Eq. 7-9 is satisfied when limiting its quantifier's range to the sampled space $\mathcal{C}$. We recall that also φ_i has been built by considering the empirical truth-table $\mathcal{T}^i$, that only contains the boolean configurations associated to the available data of the problem and not all the possible configurations in C. Indeed, as standard machine learning models do, we assume that the statistical regularities collected by the rules on the visible data are also representative for unseen samples belonging to the same data distribution, but with no general guarantees on the satisfaction of the rules. This assumption, common in machine learning models, is what allows our rules to be associated to the universal quantification used in Eq. 7-9, in line with previous work [76, 75, 36, 37]. Analogous considerations apply to the concept-bottleneck pipeline expressed by Eq. 10, with respect to the input space X and samples in $\mathcal{X}$.

EXPLAINING A BLACK-BOX. In this case, LEN's outputs are forced to mimic the predictions of a black-box $g : C \rightarrow Y$, instead of ground-truth labels. This behaviour can be imposed by considering loss functions analogous to the ones in Eq. 4-6, with g_i in place of $\bar{y}_i$. For instance, for the IFF rule the coherence loss of Eq. 6 can be leveraged, allowing LENs to mimic the behaviour of the black-box,

$$L_{\leftrightarrow}(g_i, f_i, \mathcal{C}) = \sum_{c \in \mathcal{C}} g_i(c) \log(f_i(c)) + (1 - g_i(c)) \log\left(1 - (f_i(c))\right) \tag{11}$$

As in the case of interpretable classification, IF, Only IF and IFF rules can be expressed according to Eq. 7-10. However, here FOL explanations will hold assuming that $\bar{y}_i(c)$ is not a virtual predicate, but it is explicitly associated to the (Booleanized) black-box predictions $g_i(c)$.

INTERPRETABLE CLUSTERING. Generic explanations can be obtained by means of fully unsupervised principles we borrow from Information Theory. As a matter of fact, the maximization of the Mutual Information (MI) index between the input concept space C and the output concept space E allows LENs to be trained in a fully unsupervised way [36]. More specifically, a max-MI criterion (see [77] for further details) leads to LENs leaning towards 1-hot activation scores, such that $\forall c \in \mathcal{C}$ only one $f_i(c) \simeq 1$, while the others are close to zero. This encourages LENs to cluster input data such that each input sample belongs to a single cluster. Intuitively, the MI index measures the mutual dependence between two probability distributions, and this is why we aim at maximizing it to devise meaningful correlations between concepts. In order to define the MI index, we have to model the probability distribution of each f_i to be active (close to 1) on a given sample c, that we implemented using the softmax operator on LENs' outputs. In order to train LENs in this setting, we minimize a loss that is the MI index with the opposite sign (since we aim at maximizing MI),

$$L_{MI}(f, \mathcal{C}) = -H_E(f, \mathcal{C}) + H_{E|C}(f, \mathcal{C}) \tag{12}$$

where H_E and $H_{E|C}$ denote the entropy and the conditional entropy functions associated to the aforementioned probability distribution, respectively, and measured over the whole $\mathcal{C}$, as described by Ciravegna et al. [36]. In this case, the support O_i of $\bar{f}_i$ is exactly the cluster of data points where the i-th output of the LEN is active. Leaving φ_i free to relate concepts in a purely unsupervised manner, we naturally get the FOL explanation

$$\rho_i = \forall c \in O_i: \ \varphi_i(c) \tag{13}$$

As a final remark, in all the computational pipelines in which a classifier g is employed, the available supervision is enforced on g as well, e.g. by means of the Cross-Entropy loss. As a side note, we mention that what we are describing in a fully supervised setting can be trivially extended to the semi-supervised one, as investigated in previous works [36, 37].

4.3 Parsimony

When humans compare a set of explanations outlining the same outcomes, they tend to have an implicit bias towards the simplest one [78, 79]. In the case of LENs, the notion of simplicity is implemented by reducing the dependency of each output unit by all of the k input concepts, encouraging only a subset of them to have a major role in computing LENs' outputs. Such subset is of size $m_i \leq k$ for the i-th output unit.

Over the years, researchers have proposed many approaches to integrate "*the law of parsimony*" into learning machines. These approaches are considered as potential solutions to implement parsimony criteria in LENs, in order to find a valid way to fulfill the end-user requirements on the quality of the explanations and on the classification performance. For instance, Bayesian priors [80] and weight regularization [81] are two of the most famous techniques to put in practice the Occam's razor principle in the fields of statistics and machine learning. Among such techniques, $L1$-regularization [82] has been recently shown to be quite effective for neural networks providing logic-based explanations [37] as it encourages weight sparsity by shrinking the less important weights to zero [83]. This allows the model to ignore some of the input neurons, that, in the case of the first layer of a LEN, corresponds to discarding or giving negligible weight to some of the input concepts, opening to simplified FOL explanations. If W collects (a subset of) the weights of the LEN that are subject to this regularization, the learning criterion is then augmented by adding $\lambda \|W\|_1$. Notice that the precise weights involved in W might vary in function of the selected neural architecture to implement the LEN, that is the subject of Section 5. All the out-of-the-box LENs of the next section are trained using this parsimony criterion, that acts in different portions of the network in the considered instances of LENs. The parsimony criterion is usually combined with a pruning strategy amongst the ones that are defined in the following.

4.3.1 Pruning strategies

The action of parsimony criteria, such as regularizers or human priors, influences the learning process towards specific local minima. Once the model has finally converged to the desired region of the optimization space, the effort can be speed up and finalized by pruning the neural network [84, 85] i.e., removing connections whose likelihood of carrying important information is low. The choice of the connections to be pruned depends on the selected pruning strategy. Such a strategy has a profound impact both on the quality of the explanations but also on the classification performance [86]. Here we present three effective pruning strategies specifically devised for LENs, whose main goal is to keep FOL formulas compact for each LEN's output, namely NODE-LEVEL PRUNING, NETWORK-LEVEL PRUNING, EXAMPLE-LEVEL PRUNING.

NODE-LEVEL PRUNING. The most "fine-grained" pruning strategy considers each neuron of the network independently. This strategy requires the user to define in advance the maximum *fan-in* $\zeta \in \mathbb{Z}^+$ for each neuron of a feed-forward neural network i.e., the number of non-pruned incoming connections each neuron can support. In this case, the pruning strategy removes all the connections associated to the smallest weights entering the neuron, until the target fan-in is matched. We refer to this strategy as *node-level pruning*. In detail, the node-level approach prunes one by one the weights with the smallest absolute value, such that each neuron in the network has a fixed number ζ of incoming non-pruned weights. The parameter ζ determines the computational capabilities of the pruned model, but it is also relevant to handle the complexity of the logic formulas which can be extracted from the network. By reducing the number of edges between neurons, and consequently the number of paths connecting the network inputs to each output neuron, we get smaller truth-tables involving a limited number of different concepts. To get simple explanations out of each neuron, the parameter ζ may range between 2 and 9 [87, 88, 89]). Recent work on explainer networks has shown how node-level pruning strategies may lead to fully explainable models [36]. However, we will show in the experimental section that this pruning strategy strongly reduces the classification performances.

NETWORK-LEVEL PRUNING. In order to overcome the heavy reduction in learning capacity of node-level pruned models, we introduce the so-called *network-level pruning*. This pruning operation aims at reducing the number of concepts involved in FOL explanations by limiting the availability of input concepts. In detail, the $L2$ norm of the connections departing from each input concept is computed. If $w = [w_1, \ldots, w_k]$ is the vector that collects the resulting k norms, then we re-scale it in the interval $[0, 1]$,

$$w' = \frac{w}{\max_j \{w_j : \ j = 1, \ldots, k\}} \tag{14}$$

where the division is intended to be applied in an element-wise fashion. In this way, w' gives a normalized score to rank input concepts by importance. The *network-level strategy* consists in pruning all the input features for which $w'_j < \tau$, where τ is a custom threshold (in our experiments, $\tau = 0.5$). Alternatively, we can retain the ζ most important input concepts discard all the others. This can be achieved by pruning all the connections departing from the least relevant concepts similarly to *node-level pruning*. Anyway, this pruning strategy is far less stringent compared to node-level pruning as it affects only the first layer of the network and it does not prescribe a fixed fan-in for each neuron.

EXAMPLE-LEVEL PRUNING. Example-level pruning is a particular strategy leveraging the Voronoi tessellation generated by neural networks whose activation functions in all hidden layers are Rectified Linear Units (ReLU Networks) [90]. If the LEN is implemented as a ReLU Network, for any input $c \in C$, the Directed Acyclic Graph (DAG) $\mathcal{G}$ describing the structure of the connections in the LEN, can be reduced to $\mathcal{G}_c$, which only keeps the units corresponding to active neurons (the ones for which the ReLU activation is non-zero) and the corresponding arcs (referred to as the "firing path"). Since all neurons operate in "linear regime" (affine functions) in the reduced $\mathcal{G}_c$, as stated in the following, the input-to-output transformation computed by the multi-layer feed-forward ReLU network with structure $\mathcal{G}_c$ is a composition of affine functions over the network hidden layers that, in turn, can be simplified with a single affine function, leading to

$$f(c) = \sigma(\hat{W}^{(c)}c + \hat{b}^{(c)}) \tag{15}$$

being σ the activation of the output layer and $\hat{W}^{(c)}$, $\hat{b}^{(c)}$ a weight matrix and biases (respectively) computed as described in the following.

Theorem 4.1. *Let $\{\xi_1, \ldots, \xi_L\}$ be a collection of affine functions, where for any $k = 1, \ldots, L-1$, we have $\xi_k : X_k \to Y_k$ such that $\xi_k(x) = W_k x + b_k$ and $Y_k \subset X_{k+1}$. If a multi-layer network computes the last layer activations as $\xi(c) = (\xi_L \circ \xi_{L-1} \circ \cdots \circ \xi_2 \circ \xi_1)(c)$, then such transformation is affine and we can re-write it as $\xi(c) = \hat{W}c + \hat{b}$, where*

$$\hat{W} = \prod_{k=1}^{L} W_k, \qquad \hat{b} = \sum_{k=0}^{L-1} b_{L-k} \prod_{h=0}^{k} W_{L-h+1} \tag{16}$$

and $W_{L+1} := I$.

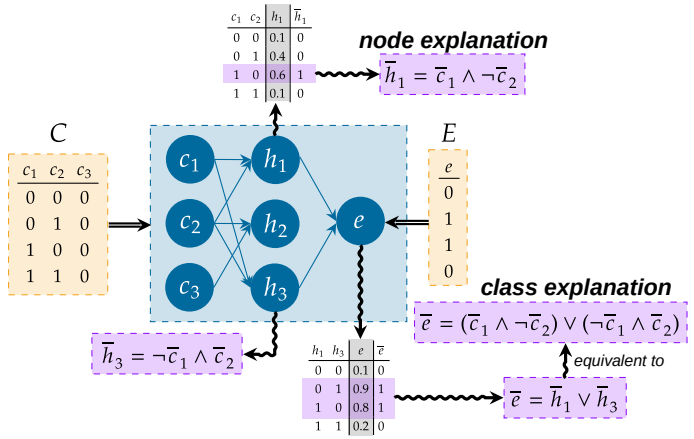
The proof of Theorem 4.1 is straightforward. Such a theorem perfectly applies to the case of ReLU networks once $\mathcal{G}_c$ has been fixed, since they boil down to simple networks with linear activations. The theorem does not introduce any conditions on how input is represented or on the activation functions in the output layer. Given $c \in C$, once $\mathcal{G}_c$ has been determined, the function computed by the (deep) LEN has the following form, $f(c) = \sigma(\xi_L^{(c)} \circ \xi_{L-1}^{(c)} \circ \cdots \circ \xi_2^{(c)} \circ \xi_1^{(c)}(c))$, that reduces to $f(c) = \sigma(\hat{W}^{(c)}c + \hat{b}^{(c)})$, where the superscript (c) is added to the symbols of Theorem 4.1 to highlight the fact they are specifically instantiated in the case of the network with structure $\mathcal{G}_c$. The reduced form of f is much easier than considering the one of the original deep network. However, $\mathcal{G}_c$ might vary for each input c, thus we might get different transformations for different input samples. Indeed, for each sample we can compute the corresponding $\mathcal{G}_c$ and, in turn, $\hat{W}^{(c)}$ and $\hat{b}^{(c)}$, that we can prune as described in the case of network-level pruning, keeping only the connections associated with the most important concepts (for that specific sample).

5 Out-of-the-box LENs

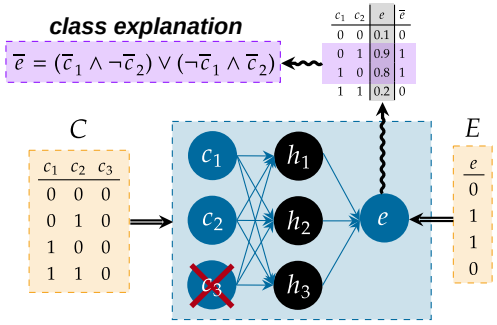
Crafting state-of-the-art fully interpretable models is not an easy task; rather, there is a trade-off between interpretability and performances. The framework introduced in Section 3, with the methods described in Section 4, is designed to provide the building-blocks to create a wide range of models having different interpretability vs. accuracy trade-offs. Here, we showcase three out-of-the-box neural networks implementing different LENs, whose key properties are about different ways of leveraging the parsimony strategies, as visually anticipated in Fig. 3 (right). In Fig. 8 we sketch the so-called ψ, μ, and ReLU out-of-the-box LENs that will be described in the following. Briefly, the ψ network, originally proposed in [36], is a fully interpretable model with limited learning capacity providing mediocre explanations; the μ network is a neural model that can provide high-quality explanations, good learning capacity and modest interpretability; the ReLU network is a model achieving state-of-the-art learning capabilities and good explanations at the cost of very low interpretability. The characteristics of these three LENs are summarized in Table 2.

Table 2: Out-of-the-box LENs and their main properties.

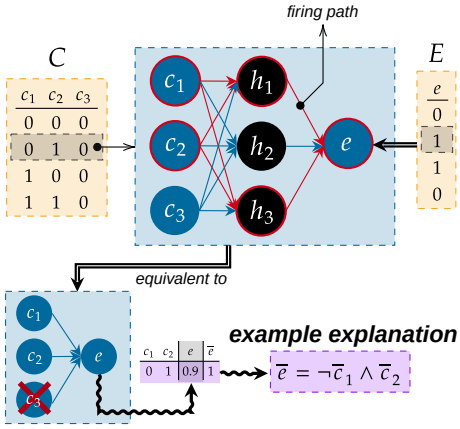
LEN	Pruning	Activation	Learning	Explanation	Interpretability
ψ Net (Fig. 8a)	Node-level	Sigmoid	Low	Low	Very high
μ Net (Fig. 8b)	Network-level	Any	High	Very high	High
ReLU Net (Fig. 8c)	Example-level	ReLU	Very high	High	Low



(a) ψ network (Section 5.1).



(b) μ network (Section 5.2).



(c) ReLU network (Section 5.3).

Figure 8: Out-of-the-box LENs showcased in the case of interpretable classification, with examples of logic rules extracted using the procedure of Section 4.1. (a) ψ networks only admit $[0, 1]$-valued neurons with low fan-in, hence we can associate a Boolean formula to each neuron by analysing its I/O truth-Table (b) μ networks do not have interpretable hidden neurons but the input concepts are drastically pruned which allows to provide simple network-level explanations. (c) ReLU networks supply explanations for each example by means of the equivalent affine transformation. However, nor the nodes, nor the network can be logically interpreted.

5.1 ψ Network

A ψ network, originally proposed in [36, 37], is the first model in the LENs family. A ψ network is based on three design principles: (i) all activation functions for all neurons (including hidden units) should be $[0, 1]$-valued differentiable functions, like e.g. sigmoids; (ii) a strong $L1$-regularization is used in all the layers to shrink the less important weight values towards zero; (iii) a node-level pruning strategy is considered, where each neuron of the network must have the same number of incoming non-pruned weights (suggested values are between 2 and 9). This number is directly proportional to the number of terms involved in the explanations. Pruning occurs during the training by progressively zeroing the least important weights in input to each neurons. In line with previous works [36, 37], the rule generation mechanism involves the extraction of rules from each neuron of the network (also the hidden ones). The node-level pruning favours simple rules on each neuron and having $[0, 1]$-valued activation functions favours the process of booleanization necessary to extract node-level logic formulas. Rules are then combined layer-by-layer to get the final explanations of the network. The number of hidden layers in the network needs to be small in order to avoid the creation of formulas that are too complex for explanation purposes. In our implementation, pruning is performed after half of the training epochs has been completed, so that the pruned network can refine the value of the remaining weights during the last epochs.

We can cast the ψ network as a member of the LENs family. The computational pipeline presented in the original work corresponds to a Concept-Bottleneck pipeline. However, from the perspective of the rule-extraction mechanism, the ψ network can be seen as a Cascading LEN where each layer corresponds to a new level of LEN, each of them generating rules. However, the network is trained as a whole, since only the first layer gets human-understandable concepts. Neuron-specific explanations are then aggregated in order generate final explanations involving input concepts only. For instance, by recalling Example 1, a ψ network with a single hidden layer with two hidden nodes may learn the logic formula:

$$\varphi_1 = (\neg\bar{c}_1 \wedge \bar{c}_2) \vee (\bar{c}_1 \wedge \neg\bar{c}_2) \tag{17}$$

with hidden nodes learning the formulas $\bar{h}_1 = \neg\bar{c}_1 \wedge \bar{c}_2$ and $\bar{h}_2 = \bar{c}_1 \wedge \neg\bar{c}_2$ respectively, and the output node f_1 learning $\varphi_1 = \bar{h}_1 \vee \bar{h}_2$.

We remark that the ψ network is specifically designed to be a fully interpretable neural network. As shown in Fig. 8a, this network provides a high level of interpretability as each neuron can be explained. On the contrary, the strong regularization and the restrictive pruning strategy lead to poor classification accuracy making ψ networks hard to train and not suitable for solving complex categorical learning problems. Besides, the provided explanations may be disappointing, due to the limited learning capabilities of the model.

5.2 μ Network

A μ network is a LEN based on a multi-layer perceptron without the strong constraints on neurons' fan-in of the ψ net. In particular, a μ network is based on two design principles: (i) a strong L1-regularization is used in the weights connecting the input to the first hidden layer of the network; (ii) a network-level pruning strategy, which prunes input neurons only. After the pruning step, the μ network can be fine-tuned to better adapt it to the new configuration (we pruned the network after half of the training epochs has completed). No assumptions need to be made nor on hidden layers nor on activation functions. However, by applying a network-level pruning strategy, the same set of retained input concepts is used for the whole network. On one hand, this allows a simple logic interpretation of the network and, therefore, to provide concise explanations. On the other hand, this may represent a severe limitation for multi-class problems as each class may rely on its own input concepts, but these may be very different among the classes. However, μ networks can be efficiently adapted to multi-class experiments (as the ones of Section 6) by splitting the multi-class problem into a set of binary classification problems (one per class), i.e. using a set of light binary μ networks.

Still considering Example 1, μ networks with a different number of hidden layers may easily learn the same logic formulas. The network-level pruning strategy in this specific example should not prune any of the input features as both c_1 and c_2 are relevant for the target class. However, if we assume the presence of additional redundant features, they would be likely discarded in favor of c_1 and c_2, as shown in Fig. 8b. Assuming a successful training, the support set O_1 will be composed by the second and third examples of the yellow-box in Fig. 8b, that will yield the following example-level explanations respectively

$$\neg c_1 \wedge c_2 \quad \text{and} \quad c_1 \wedge \neg c_2$$

The two explanations can be considered altogether (Eq. 2) to explain the whole class 1 (class-level explanation) by:

$$\varphi_1 = \bigvee_{c \in O_1} \varphi_{1,c} = (\neg\bar{c}_1 \wedge \bar{c}_2) \vee (\bar{c}_1 \wedge \neg\bar{c}_2)$$

Thanks to the permissive pruning strategy, the learning capabilities of the network almost match an unconstrained network. As a consequence, μ networks are suitable for solving and explaining more complex categorical learning problems. As we will show in Section 6, the quality of the explanations provided by the μ networks are among the highest of the proposed models. At last, a mild-level of interpretability is guaranteed as μ nets can be logically interpreted as a whole, but hidden neurons are not as interpretable as in ψ networks.

5.3 ReLU network

The ReLU network is another member of the LENs family providing a different accuracy vs. interpretability trade-off. This model is based on three design principles: (i) all activation functions for all *hidden* neurons are rectified linear units; (ii) a mild L1-regularization is applied to all the weights associated to each layer of the network; (iii) an example-level pruning strategy is used, i.e., a specialized pruning strategy for each sample. Principle (iii) can be applied due to the presence of rectified linear unit activation functions. The restriction to ReLU activations is not as limiting as it sounds, since it is among the most widely used and efficient activation functions employed in deep learning [91, 92]. What makes this LEN significantly different from both ψ and μ nets, 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 the single-affine instance of $f(c)$ of Eq. 15, whose values are collected in $\hat{W}^{(c)}$ 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 c, and they may not always lead to optimal explanations. To this purpose we impose a mild L1-regularization to encourage the network to rely on as fewest weights as possible. This, in turn, forces the network to employ the same connection patterns to classify similar samples and, therefore, to provide the same explanation.

Recalling again Example 1, a ReLU network with hidden layers can learn the correct logic formula by aggregating different example-level explanations, as we did in the case of the μ network. For example, in Fig. 8c we show the case of the processing the second sample from the yellow table, that provides a portion of the explanation of the XOR function. However, when we restrict the connections to the arcs of $\mathcal{G}_c$ for a certain sample c, we actually discard several weights of the original ReLU network, i.e. the ones of all the connections that are not needed for classifying the considered c correctly. This implies that the single-affine instance of $f(c)$ of Eq. 15, being it function of $\mathcal{G}_c$, has a very localized dependence on the space region to which c belongs. Since Eq. 15 is the form of the LEN from which we extract the example-level explanation, there is the serious risk of obtaining an explanation that does not carry much information from the original structure $\mathcal{G}$ and that does not globally applies to the whole class. As a matter of fact, there might be strong differences on the example level explanations of samples, even when belonging to the same class. This issue might be mitigated by the L1 regularization, where stronger regularization indirectly favours the network to employ a common affine instance of $f(c)$ for similar inputs.

Summing up, this LEN has the capacity to provide the best performances in terms of classification accuracy thanks to the example-level pruning strategy that do not alter the original network. However, this comes at the cost of poor model interpretability and mild explanation capacity.

6 Benchmarking out-of-the-box LENs

In this section, we quantitatively assess the quality of the explanations and the performance of LENs, compared to state-of-the-art white-box models. In order to have a fair comparison, we selected Decision Trees and BRL as they are prominent examples of approaches providing both logical and global explanations. Even if several other well-known competitors exist, they mostly focus on local explanations (Anchors), or even on not logical ones (LIME). For the sake of completeness, we provide more comments about the possibility of a comparison with Anchors in Section 6.3. In the experimental analysis, we consider several tasks, covering multiple combinations of computational pipelines (Section 3.1) and objectives (Section 3.2). The summary of each task is reported in Table 3. A brief description of each task, the corresponding dataset and all the related experimental details are exposed in Section 6.1. In Section 6.2 six quantitative metrics are defined and used to compare LENs with the considered state-of-the-art methods.

Table 3: Summary of the experiments.

Dataset	Description	Pipeline	Objective
MIMIC-II (Fig. 4, top)	Predict patient survival from clinical data	E2E	Interpretable classification
MNIST E/O (Fig. 5, top)	Predict parity from digit images	CB	Interpretable classification
CUB (Fig. 2)	Predict bird species from bird images	CB	Interpretable classification
V-Dem (Fig. 6)	Predict electoral democracy from social indexes	Cascading	Interpretable classification
MIMIC-II (EBB)	Predict patient survival from clinical data	E2E	Explaining black-box (Fig. 4, bottom)
MNIST E/O (ICLU) (Fig. 5, bottom)	Cluster digit properties from digit images	CB	Interpretable clustering

The Python code and the scripts used for the experiments, including parameter values and documentation, is freely available under Apache 2.0 Public License from a GitHub repository[6]. The code is based on our "Logic Explained Networks" library [93], designed to make out-of-the-box LENs accessible to researchers and neophytes by means of intuitive APIs requiring only a few lines of code to train and get explanations from a LEN, as we sketch in the code example of Listing 1.

[6] https://github.com/pietrobarbiero/logic_explained_networks

```
import lens

# import train, validation and test data loaders
[...]

# instantiate a "psi network"
model = lens.models.PsiNN(n_classes=n_classes,
                          n_features=n_features,
                          hidden_neurons=[200],
                          l1_weight=0.001, fan_in=10)

# fit the model
model.fit(train_data, val_data, epochs=100, l_r=0.0001)

# get predictions on test samples
outputs, labels = model.predict(test_data)

# get first-order logic explanations for the 1st class
target_class = 1
formula = model.get_global_explanation(x_val, y_val,
                                       target_class)

# compute explanation accuracy
accuracy = lens.logic.test_explanation(formula, target_class,
                                       x_test, y_test)
```

Listing 1: Example on how to use the "Logic Explained Networks" library.

Further details about low-level APIs can be found in Appendix A. The code for the experiments is implemented in Python 3, relying upon open-source libraries [94, 95]. All the experiments have been run on the same machine, that is based on an Intel® Core™ i7-10750H 6-Core Processor at 2.60 GHz, equipped with 16 GiB RAM and an NVIDIA GeForce RTX 2060 GPU.

6.1 Dataset and classification task details

We considered five categorical learning problems ranging from computer vision to medicine and democracy. Some datasets (e.g., CUB) were already annotated with high-level concepts (e.g., bird attributes) which can be leveraged to train a concept bottleneck pipeline. For datasets without annotations for high-level concepts, we transformed the input space into a predicate space (i.e., $\mathbb{R}^9 \rightarrow [0,1]^d$) to make it suitable for training LENs. According to the considered data type, we will evaluate and discuss the usage of different LEN pipelines and objectives, as already anticipated in Table 3. For the sake of consistency, in all the experiments based on supervised learning criteria (Section 4.2) LENs were trained by optimizing Eq. 6, therefore extracting IFF rules, that better cope with the ground truth of the datasets. However, as already discussed in Section 4.2, also Eq. 4 and Eq. 5 could be considered, yielding different target rule types. In the following we describe the details of each task.

MIMIC-II - E2E, INTERPRETABLE CLASSIFICATION. The Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II, [96, 97]) is a public-access intensive care unit (ICU) database consisting of 32,536 subjects (with 40,426 ICU admissions) admitted to ICUs at a single tertiary care hospital. The dataset contains detailed description of a variety of clinical data classes: general, physiological, results of clinical laboratory tests, records of medications, fluid balance, and free text notes and reports of imaging studies (e.g., x-ray, CT, MRI, etc). In our experiments, we removed the text-based input features and we discretized blood pressure (BP) into five different categories (one-hot encoded): very low BP, low BP, normal BP, high BP, and very high BP. After such preprocessing step, we obtained an input space X composed of 90 key features. The task consists in training a classifier function to identify recovering or dying patients, after 28 days from ICU admission ($Y = [0,1]^2$). In particular, we considered the case of *interpretable classification* where an End-to-End LEN $C \rightarrow E$ is employed to directly carry out the classification task, i.e., with $C = X$ and $E = Y$.

MIMIC-II (EBB) - E2E, EXPLAINING A BLACK-BOX MODEL. On the same dataset, a second task is set up. A black-box model g is employed to solve the previously described classification task. The end-to-end LEN f instead is trained to mimic the behaviour of the black-box g, using the learning criterion of Eq. 11.

MNIST E/O - CB, INTERPRETABLE CLASSIFICATION. The Modified National Institute of Standards and Technology database (MNIST, [98]) contains a large collection of images representing handwritten digits. The input space $X \subset \mathbb{R}^{28\times 28}$ is composed of 28×28 pixel images of digits from 0 to 9. However, the task we aim to solve is slightly different from the common digit-classification. We are interested in determining if a digit is either odd or even, and explain the assignment to one of these classes in terms of the specific digit categories from 0 to 9. In the notation of this paper, we can consider that each image comes with 12 attributes, where the first 10 ones are binary attributes about

the specific digit type, i.e. from 0 to 9, while the last 2 ones are binary labels that encode whether the digit is even or odd ($Y = [0,1]^{12}$). We consider a concept-bottleneck pipeline that consists of a concept space C that is about the attributes of the specific digit type. We focus on the objective of *interpretable classification* of the odd/even classes, so that the mapping $X \to C$, with $C = Y^{(1:10)}$ is learned by a ResNet10 classifier g [99] trained from scratch, while a LEN f is used to learn both the mapping and the explanation as a function $C \to E$, with $E = Y^{(11:12)}$.

MNIST E/O (ICLU) - CB, INTERPRETABLE CLUSTERING. The same dataset has been used with the objective of *interpretable clustering*. In this case, we considered a concept space $C = Y = [0,1]^{12}$ which comprise both the digits and the even/odd labels. The mapping $X \to C$ is learned again by a Resnet10 model g trained as a multi-label classifier, while the LEN f performs clustering from $C \to E$, where $E = [0,1]^2$ space, in order two extract two clusters that are expected to group data due to two properties of the concept space that are not-defined in advance.

CUB - CB, INTERPRETABLE CLASSIFICATION. The Caltech-UCSD Birds-200-2011 dataset (CUB, [100]) is a fine-grained classification dataset. It includes 11,788 images representing 200 different bird species. Moreover, 312 lower-level binary attributes have been also attached to each image representing visual characteristics (color, pattern, shape) of particular parts (beak, wings, tail, etc.). Attributes annotations, however, are quite noisy. Similarly to [28], we denoised attributes by considering class-level annotations, i.e., a certain attribute is set as present only if it is also present in at least 50% of the images of the same class. Furthermore, we only considered attributes present in at least 10 classes (species) after this refinement. In the end, a total of 108 lower-level attributes have been retained and paired with the higher-level attributes about the 200 species, so that each image is represented with binary targets in $Y = [0,1]^{200+108=308}$ (where the first 108 targets are for the lower-level attributes). In a concept-bottleneck pipeline with the LEN objective of *interpretable classification*, the mapping $X \to C$ from images to lower-level concepts ($C = Y^{(1:108)}$) is performed again with a ResNet10 model g trained from scratch while a LEN f learns the final function that classifies the bird specie, $C \to E$, with $E = Y^{(109:308)}$.

V-DEM - CASCADING, INTERPRETABLE CLASSIFICATION. Varieties of Democracy (V-Dem, [101, 102]) is a dataset containing a collection of indicators of latent regime characteristics over 202 countries from 1789 to 2020. The database includes 483 low-level indicators (e.g., media bias, party ban, high-court independence, initiatives permitted, etc), 82 mid-level indices (e.g., freedom of expression, freedom of association, equality before the law, etc), and 5 high-level indices of democracy principles (i.e., electoral, liberal, participatory, deliberative, and egalitarian). In the experiments, we considered each example to be paired with low/mid level indices and the information on electoral/non-electoral democracies taken from high level indices. Keeping the same order of the previous description, each sample is then paired with binary targets in $Y = [0,1]^{483+82+2=567}$. We considered the *interpretable classification* objective in the problem of classifying electoral/non-electoral democracies in a cascading LEN with two LEN components, where $C_1 = Y^{(1:483)}$, $E_1 = C_2 = Y^{(484:566)}$, and $E_2 = Y^{(566:567)}$. In detail, cascading LENs are trained to learn the map $C_1 \to E_1$ with $E_1 = C_2$ and $C_2 \to E_2$, with $E_2 = Y^{(566:567)}$. We measured the quality of the rules extracted by the second LEN.

6.2 Metrics

Seven metrics are used to compare the performance of LENs with respect to state-of-the-art approaches. While measuring classification metrics is necessary for models to be viable in practice to perform interpretable classification tasks, assessing the quality of the explanations is required to justify their use for explainability. In contrast with other kinds of explanations, logic-based formulas can be evaluated quantitatively. To sum up, given a classification problem, we first train a LEN and evaluate its accuracy on the test set (*model accuracy*). The training procedure designed for the LENs allows the extraction of meaningful logic rules that can be used by their own both as classifiers and for explanation purpose. Hence, after the training stage we extract a set of rules from the model and test/evaluate each explanation on the same test data (*explanation accuracy*). The results for each metric are reported in terms of mean and standard deviation, computed over a K-fold stratified cross-validation, with $K = 10$ [103]. Each training partition was further divided into non-overlapping training and validation sets, and the optimal values of the hyper-parameters were found by maximizing the average validation accuracy. Only in the CUB experiments we used $K = 5$ due to timing issues related to BRL (each fold required about 3 hours)—competitors were described in Section 2. In particular, for each experiment we consider the following metrics.

- *Model accuracy*: it measures how well the LEN f_i or the competitor classifier correctly identifies the target classes, in the case of interpretable classification $Acc(\bar{f}_i, \bar{y}_i)$[7]. When the LEN explains the predictions of a black-box classifier g_i, this metric represents the accuracy of the model in mimicking the black-box classifier $Acc(\bar{f}_i, \bar{g}_i)$ (Table 4).

[7]Where the function $Acc(\hat{y}, y)$ denotes the accuracy of the predictions $\hat{y}$ with respet to y.

- *Explanation accuracy*: it measures how well the extracted logic formula φ_i correctly identifies the target class $Acc(\varphi_i, \bar{y}_i)$ (Table 5).
- *Complexity of an explanation*: it measures how hard would it be for a human being to understand the logic formula φ_i (Table 6). This is simulated by standardizing the explanations in disjunctive normal form $\tilde{\varphi}$ and then by counting the average number of terms of the standardized formula $|\{\ell : \ \ell \text{ is a literal of } \varphi_i\}|$.
- *Fidelity of an explanation*: it measures how well the predictions obtained by applying the extracted explanations φ_i match the predictions obtained by using the classifier (Table 7). When the LEN f_i is the classifier itself (i.e., interpretable classification), this metric represents the match between the extracted explanation and the LEN prediction $Acc(\varphi_i, \bar{f}_i)$. Instead, when the LEN is explaining the predictions of a black-box classifier g_i, this metric represents the agreement between the extracted explanation and the prediction of black-box classifier $Acc(\varphi_i, \bar{g}_i)$.
- *Rule extraction time*: it measures the time required to obtain an explanation from scratch (Fig. 9). It is computed as the sum of the time required to train the model and the time required to extract the formula from a trained model. This is justified by the fact that for some models, like BRL, training and rule extraction consist of just one simultaneous process.
- *Consistency of an explanation*: it measures the similarity of the extracted explanations over different runs (Table 8). It is computed by counting how many times the same concepts appear in the logic formulas over different folds of a K-fold cross-validation.

Table 4: Model accuracy (%). The two best results are in bold.

	Tree	BRL	ψ net	ReLU net	μ net	BB net
MIMIC-II	77.53 ± 1.45	76.40 ± 1.22	77.19 ± 1.09	$\mathbf{80.11 \pm 1.87}$	$\mathbf{80.00 \pm 0.95}$	77.81 ± 2.45
vDem	85.61 ± 0.57	91.23 ± 0.75	89.78 ± 1.64	$\mathbf{92.08 \pm 0.37}$	90.40 ± 0.51	$\mathbf{94.53 \pm 1.17}$
MNIST E/O	99.75 ± 0.01	99.80 ± 0.02	99.80 ± 0.03	$\mathbf{99.88 \pm 0.02}$	$\mathbf{99.83 \pm 0.01}$	99.72 ± 0.03
CUB	81.62 ± 1.17	90.79 ± 0.34	91.92 ± 0.27	$\mathbf{92.29 \pm 0.40}$	92.21 ± 0.33	$\mathbf{93.10 \pm 0.51}$
MIMIC-II (EBB)	77.53 ± 1.45	77.87 ± 1.24	76.74 ± 1.52	$\mathbf{80.00 \pm 0.95}$	$\mathbf{79.44 \pm 0.97}$	(EBB)

Table 5: Explanation accuracy (%). The two best results are in bold (in case of ties in the average values, we highlighted all the involved models).

	Tree	BRL	ψ net	ReLU net	μ net
MIMIC-II	69.15 ± 2.24	$\mathbf{70.59 \pm 2.17}$	49.51 ± 3.92	70.28 ± 1.67	$\mathbf{71.84 \pm 1.82}$
vDem	85.45 ± 0.58	$\mathbf{91.21 \pm 0.75}$	67.08 ± 9.68	$\mathbf{90.21 \pm 0.55}$	88.18 ± 1.07
MNIST E/O	$\mathbf{99.74 \pm 0.01}$	$\mathbf{99.79 \pm 0.02}$	65.64 ± 5.05	$\mathbf{99.74 \pm 0.02}$	$\mathbf{99.74 \pm 0.02}$
CUB	89.36 ± 0.92	$\mathbf{96.02 \pm 0.17}$	76.10 ± 0.56	87.96 ± 2.81	$\mathbf{93.69 \pm 0.27}$
MIMIC-II (EBB)	69.15 ± 2.24	$\mathbf{71.68 \pm 2.21}$	51.71 ± 4.78	70.53 ± 1.56	$\mathbf{71.84 \pm 1.82}$

Table 6: Complexity of explanations. The two best results are in bold (the lower, the better).

	Tree	BRL	ψ net	ReLU net	μ net
MIMIC-II	66.60 ± 1.45	57.70 ± 35.58	$\mathbf{20.60 \pm 5.36}$	39.50 ± 11.62	$\mathbf{15.80 \pm 1.37}$
vDem	30.20 ± 1.20	145.70 ± 57.93	$\mathbf{5.40 \pm 2.70}$	18.40 ± 2.17	$\mathbf{9.10 \pm 0.94}$
MNIST E/O	$\mathbf{47.50 \pm 0.72}$	1352.30 ± 292.62	96.90 ± 10.01	$\mathbf{73.30 \pm 5.77}$	80.50 ± 4.85
CUB	45.92 ± 1.16	$\mathbf{8.87 \pm 0.11}$	15.96 ± 0.96	60.57 ± 6.95	$\mathbf{14.65 \pm 0.16}$
MIMIC-II (EBB)	66.60 ± 1.45	40.50 ± 32.46	$\mathbf{24.30 \pm 5.40}$	61.30 ± 13.61	$\mathbf{15.80 \pm 1.37}$

Table 7: Fidelity of explanations (%). Tree and BRL trivially get 100%. We highlighted in bold the best LEN model.

	Tree	BRL	ψ net	ReLU net	μ net
MIMIC-II	100.00 ± 0.00	100.00 ± 0.00	51.63 ± 6.68	75.62 ± 2.07	$\mathbf{88.37 \pm 2.73}$
vDem	100.00 ± 0.00	100.00 ± 0.00	69.67 ± 10.43	$\mathbf{96.36 \pm 0.64}$	94.73 ± 2.16
MNIST E/O	100.00 ± 0.00	100.00 ± 0.00	65.68 ± 5.05	$\mathbf{99.85 \pm 0.02}$	99.83 ± 0.02
CUB	100.00 ± 0.00	100.00 ± 0.00	77.34 ± 0.52	89.28 ± 2.90	$\mathbf{95.21 \pm 0.25}$
MIMIC-II (EBB)	100.00 ± 0.00	100.00 ± 0.00	55.72 ± 8.55	74.14 ± 2.32	$\mathbf{88.37 \pm 2.73}$

Table 8: Rule consistency (%). The two best results are in bold.

	Tree	BRL	ψ net	ReLU net	μ net
MIMIC-II	40.49	30.48	27.62	**53.75**	**71.43**
vDem	**72.00**	**73.33**	38.00	64.62	41.67
MNIST E/O	41.67	**100.00**	96.00	**100.00**	**100.00**
CUB	21.47	42.86	41.43	**44.17**	**76.92**
MIMIC-II (EBB)	40.49	40.00	25.71	**58.67**	**71.43**

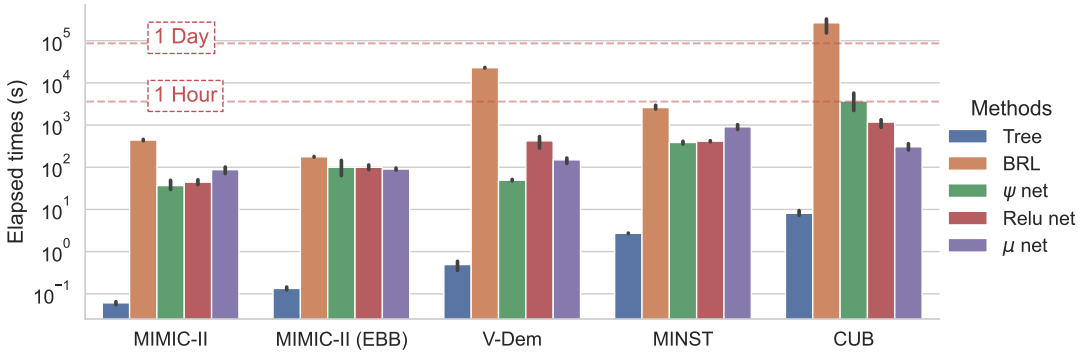


Figure 9: Rule extraction time (seconds). Time required to train models and to extract the explanations. Error bars show the 95% mean confidence interval. The result bars refer left-to-right to Tree, BRL, ψ net, Relu net and μ net.

6.3 Results and discussion

Our results are organized in order to compare the behavior of the models in all the considered tasks jointly, with the exception of interpretable clustering, that will be discussed in a separate manner. Experiments of Table 4 show how all the considered out-of-the-box LENs generalize better than decision trees on complex Boolean functions (e.g., CUB) as expected [38], and they usually outperform BRL as well. LENs based on ReLU networks are definitely the ones with better classification accuracy, confirming the intuitions reported in Section 5.3. μ nets, however, are quite close in terms of classification accuracy. It is interesting to compare these results with the one obtained when training a classic neural network (i.e., without the LEN design constraints, without pruning, without the LEN training losses) to directly predict the final class labels of the considered task. In other words, such a Black-Box neural network (BB net) does not have the burden of predicting intermediate concepts, and it is fully focused on maximizing the quality of the predictions that are then evaluated in Table 4.[8] Anyhow, we can appreciate that the drop of performance in LEN (when present) is relatively low. In the case of MIMIC-II, LENs perform similarly or even better than BB net, mostly due to the regularization introduced in the training procedure, which resulted to have a positive effect in this data. These results must be paired with the ones of Table 5, since having high classification performance and very low explanation quality would represent a major issue. For most experiments the formulas extracted from LENs are either

[8]It is a fully connected multi-Layer perceptron in the case of MIMIC-II and v-Dem, and a CNN in computer vision tasks, with a similar number of units of the LEN models (ReLU activations–the best architecture was selected by cross-validation).

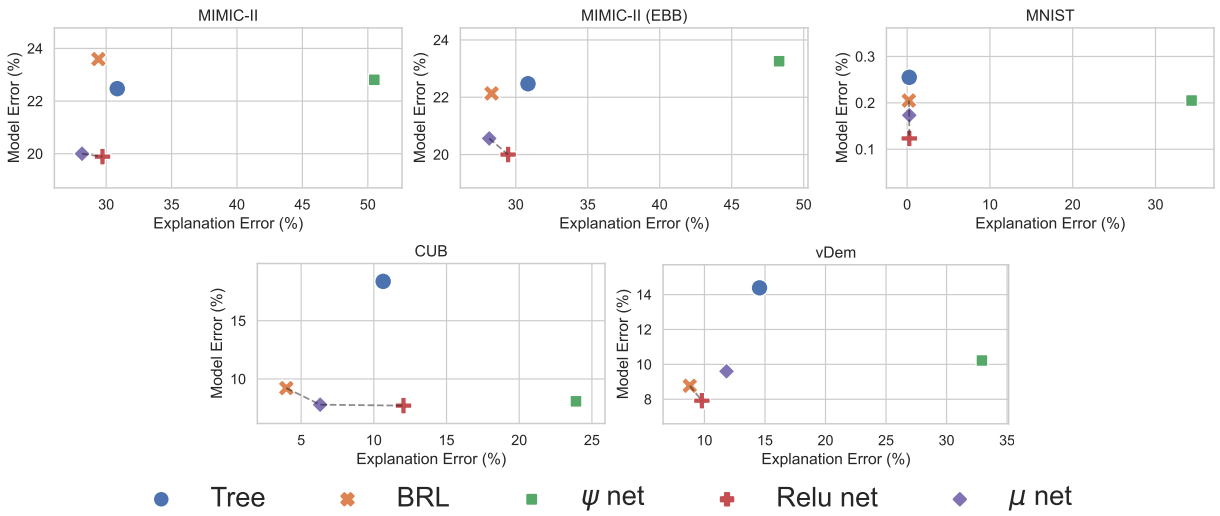


Figure 10: Pareto frontiers (dotted black line) in terms of average model test error and average explanation test error.

better or almost as accurate as the formulas found by decision trees or mined by BRL, even if the top performance are reached by BRL. What makes LENs formulas preferable with respect to BRL is the significantly reduced complexity of the considered explanations, as shown in Table 6. Notice how less complex rules implies more readable formulas, that is a crucial feature in the context of Explainable AI. More specifically, the complexity of LENs explanations is usually lower than the complexity of the rules extracted both from a decision tree[9] or mined by BRL. This is mostly due to the use of ad hoc pruning strategies to keep low the explanations' complexity of LENs. Indeed, by selecting a limited number of concepts and by filtering out low-frequency example-level explanations, LENs can actually produce readable rules whose size is bounded, while keeping competitive classification performances. Comparing the results of the different out-of-the-box LENs, we observe that ψ networks yield moderately complex explanations, sometimes with limited accuracy, while ReLUs, due to the variability of example-level rules, lead to more complex explanations. Overall, the case of μ networks is the most attractive one, confirming the quality of their parsimony/pruning criterion.

Moving to more fine-grained details, Table 7 shows how white-box models, like decision trees and BRL, outperform LENs in terms of fidelity. This is due to the fact that such models make predictions based on the explanation directly. Therefore fidelity is (trivially) always 100%. However, we see how the fidelity of the formulas extracted by the μ and the ReLU network is often higher than 90%. This means that almost any prediction has been correctly explained, making these networks very close to white boxes. On the contrary, the fidelity of the formulas extracted by the ψ network is usually lower than the one of μ net formulas. This is mostly due to the fact that ψ networks extract a formula for every neuron of the net, and compound them layer-by-layer up to the output neurons. As a result, even though the behaviour of each neuron of the ψ network can be interpreted, the overall logic formula may be a less precise explanation of the model predictions, since each neuron-related explanation might introduce errors that propagate when composing them. Therefore, as anticipated in the previous section, ψ network provides higher interpretability, but lower explanation performances. Fig. 9 compares the rule extraction times. BRL is the slowest rule extractor over all the experiments, and LENs are faster by one to three orders of magnitude. In two cases, BRL takes about 1 hour to extract an explanation and over 1 day in the case of CUB, making it unsuitable for supporting decision-making. Decision trees are the fastest model overall. In terms of consistency, LENs seem to provide more stable results, but overall there is not a dominant method (see Table 8). We can see, instead, how this result is clearly impacted by the considered dataset/task. Our intuition is that those datasets that are more coherently represented by the data in the different folds are expected to lead to more consistent behaviors.

Fig. 10 shows a combined view of two of the main metrics considered so far, reporting the Pareto frontiers [104] for each experiment in terms of the explanation and model error (100 minus the explanation/model accuracy). The figure provides a broad perspective showing how the LEN framework is flexible enough to instantiate different models having different compromises between explanation and classification accuracy. The limits of the ψ network are overcome by the two other out-of-the-box LENs we investigated. In all considered tasks, the μ and ReLU LENs are always on the Pareto frontier, therefore providing, on average, the better accuracy vs. interpretability trade-offs.

[9]Decision trees have been limited to a maximum of 5 decision levels in order to extract rules of comparable length with the other methods.

Table 9: Comparison of μ nets with a local post-hoc explainer, MIMIC-II data. Such an explainer is the outcome of having tweaked Anchors to yield global explanations, using the same aggregation criterion of LENs. Results clearly show that a simple adaptation of a local model to yield global explanation is not enough, and specific attention is needed during the training stage of the explainer, as in the case of LENs.

Method	Exp. Acc. (%)	Complexity	Fidelity (%)	Extr. Time (s)	Consistency (%)
μ net	71.84 ± 1.82	15.80 ± 1.37	88.37 ± 2.73	44.22 ± 3.38	71.43
Global Anchors	43.01 ± 2.39	19.00 ± 2.95	41.95 ± 2.77	21137.43 ± 5957.47	39.09

Table 10: MNIST E/O (ICLU). All the metrics are reported.

Method	Model Acc. (%)	Exp. Acc. (%)	Complexity	Extr. Time (sec)	Consistency (%)
ψ net	$\mathbf{99.90 \pm 0.01}$	$\mathbf{99.90 \pm 0.01}$	$\mathbf{2.00 \pm 0.00}$	$\mathbf{0.33 \pm 0.15}$	**100.00**
μ net	94.91 ± 4.99	95.41 ± 4.50	2.30 ± 0.64	2.36 ± 0.75	35.00
ReLU net	96.87 ± 3.04	95.97 ± 3.94	2.15 ± 0.39	4.15 ± 1.62	45.00

Our experimental analysis quantitatively demonstrates the quality of LENs and of the FOL rules they extract. In order to provide the reader also a qualitative view on the type of rules that get extracted by the compared models, Table 11 reports a representative subset of the formulas over the different tasks. Comparing the LEN models, we can appreciate the compactness of rules coming from the μ and ψ networks. All the LEN-rules are way more compact than the ones from Trees, and usually also the ones from BRL.

We also compared the proposed approach with a well-known post-hoc explainer, Anchors [49]. However, since Anchors extracts local explanations, we created a global version (Global Anchors) which provides global explanations by aggregating the example-level explanations following the same approach described in Section 4.1 for LENs. Table 9 shows the results obtained with the μ net, compared with Global Anchors. Due to the computational complexity of Global Anchors, we restricted the comparison to the MIMIC-II dataset. On all the considered metrics, the explanations provided by Global Anchors are significantly worse that the ones of the μ net, particularly when focusing on accuracy and fidelity. We believe that these results are due to the locality of the explanations provided by Anchors, that aim at explaining a single prediction and are not well-suited to be directly promoted to global explanation with a simple aggregation procedure (of course, there could be other more specific/advanced ad-hoc procedures). This consideration is further is supported by the lower consistency of the explanations, remarking the high variance of the features involved in the final explanation. Overall, this result remarks the importance of intervening during the training procedure (as done in the LENs) to better shed lights on the overall/global behaviour of a neural network.

A separate investigation is dedicated to the interpretable clustering objective in MNIST E/O (ICLU). In this scenario, competitors are not involved as they only operate in a supervised manner, while LENs can also handle the unsupervised discovery of FOL formulas. LENs are instructed to learn two clusters considering the data represented in the concept space where the digit identity and the property of being even/odd are encoded. In this setting, we are interested in evaluating whether LENs can discover that data can be grouped into two balanced clusters of even and odd digits, and to provide explanations of them (such as, *odd and not even* and *even and not odd*). We remark that this task does not use any supervisions, and we recall that LENs do not know that odd/even are mutually exclusive properties, but they are just two input concepts out of 12. In Table 10, all the previously introduced metrics are reported for this task, where the model accuracy is computed considering how the system develops clusters that match to the even/odd classes. We can see that all methods are capable of reaching very high level of cluster accuracy, therefore correctly identifying the even and odd groups. By inspecting the extracted rules, we observed that while all the LENs correctly consider odd and even as predicates that participate to the FOL rules, only the ψ network consistently explains the two clusters in terms of only such important labels (even and odd), as we can see from the complexity of the rules (that is higher than 2 for the μ network and ReLU net) and from the consistency of terms used in the explanation (100% for the ψ network). Other examples and applications of interpretable clustering have been performed in previous works employing the ψ network only [36, 37].

Table 11: A selection of some of the best rules extracted for each experiment. Each rule involves concept names that are taken from the respective dataset. We dropped the argument (c) from each predicate to make the notation more compact, and each rule is intended to hold $\forall c \in C$. Rule from decision trees are not shown since they involve a very large number of terms.

	Model	Sample Rule
MIMIC-II	μ net	Death $\leftrightarrow$ stroke $\wedge$ age_HIGH $\wedge$ $\neg$atrial_fibrillation
	ReLU net	Death $\leftrightarrow$ stroke $\wedge$ age_HIGH $\wedge$ $\neg$atrial_fibrillation $\wedge$ $\neg$sapsi_first_LOW $\wedge$ $\neg$sapsi_first_HIGH
	ψ net	Death $\leftrightarrow$ stroke $\vee$ bun_first_NORMAL $\vee$ sapsi_first_HIGH
	Tree	Death $\leftrightarrow$ *formula is too long
	BRL	Death $\leftrightarrow$ (stroke $\wedge$ resp $\wedge$ $\neg$age_LOW $\wedge$ $\neg$sapsi_first_LOW) $\vee$ (stroke $\wedge$ age_HIGH $\wedge$ $\neg$age_LOW $\wedge$ $\neg$sapsi_first_LOW)
MIMIC-II (ÆBB)	μ net	Death $\leftrightarrow$ atrial_fibrillation $\wedge$ stroke $\wedge$ $\neg$sapsi_first_HIGH $\wedge$ $\neg$weight_first_NORMAL
	ReLU net	Death $\leftrightarrow$ stroke $\wedge$ $\neg$sapsi_first_LOW $\wedge$ $\neg$sapsi_first_HIGH $\wedge$ $\neg$sodium_first_LOW
	ψ net	Death $\leftrightarrow$ stroke $\wedge$ age_HIGH
	Tree	Death $\leftrightarrow$ *formula is too long
	BRL	Death $\leftrightarrow$ stroke $\wedge$ age_HIGH $\wedge$ weight_first_LOW $\wedge$ $\neg$sapsi_first_LOW) $\vee$ (stroke $\wedge$ platelet_first_HIGH $\wedge$ weight_first_LOW $\wedge$ $\neg$sapsi_first_LOW
MNIST E/O	μ net	Even $\leftrightarrow$ $\neg$One $\wedge$ $\neg$Three $\wedge$ $\neg$Five $\wedge$ $\neg$Seven $\wedge$ $\neg$Nine
	ReLU net	Even $\leftrightarrow$ (Zero $\wedge$ $\neg$One $\wedge$ $\neg$Two $\wedge$ $\neg$Three $\wedge$ $\neg$Four $\wedge$ $\neg$Five $\wedge$ $\neg$Six $\wedge$ $\neg$Seven $\wedge$ $\neg$Eight $\wedge$ $\neg$Nine) $\vee$ (Two $\wedge$ $\neg$Zero $\wedge$ $\neg$One $\wedge$ $\neg$Three $\wedge$ $\neg$Four $\wedge$ $\neg$Five $\wedge$ $\neg$Six $\wedge$ $\neg$Seven $\wedge$ $\neg$Eight $\wedge$ $\neg$Nine) $\vee$ (Four $\wedge$ $\neg$Zero $\wedge$ $\neg$One $\wedge$ $\neg$Two $\wedge$ $\neg$Three $\wedge$ $\neg$Five $\wedge$ $\neg$Six $\wedge$ $\neg$Seven $\wedge$ $\neg$Eight $\wedge$ $\neg$Nine) $\vee$ (Six $\wedge$ $\neg$Zero $\wedge$ $\neg$One $\wedge$ $\neg$Two $\wedge$ $\neg$Three $\wedge$ $\neg$Four $\wedge$ $\neg$Five $\wedge$ $\neg$Seven $\wedge$ $\neg$Eight $\wedge$ $\neg$Nine) $\vee$ (Eight $\wedge$ $\neg$Zero $\wedge$ $\neg$One $\wedge$ $\neg$Two $\wedge$ $\neg$Three $\wedge$ $\neg$Four $\wedge$ $\neg$Five $\wedge$ $\neg$Six $\wedge$ $\neg$Seven $\wedge$ $\neg$Nine)
	ψ net	Even $\leftrightarrow$ (Six $\wedge$ Zero $\wedge$ $\neg$One $\wedge$ $\neg$Seven) $\vee$ (Six $\wedge$ Zero $\wedge$ $\neg$One $\wedge$ $\neg$Three) $\vee$ (Six $\wedge$ Zero $\wedge$ $\neg$Seven $\wedge$ $\neg$Three) $\vee$ (Six $\wedge$ $\neg$One $\wedge$ $\neg$Seven $\wedge$ $\neg$Three) $\vee$ (Zero $\wedge$ $\neg$One $\wedge$ $\neg$Seven $\wedge$ $\neg$Three) $\vee$ (Six $\wedge$ Zero $\wedge$ $\neg$One $\wedge$ $\neg$Seven $\wedge$ $\neg$Three)
	Tree	Even $\leftrightarrow$ *formula is too long
	BRL	Even $\leftrightarrow$ (Six $\wedge$ $\neg$Three $\wedge$ $\neg$(Five $\wedge$ $\neg$Two)) $\vee$ (Eight $\wedge$ $\neg$Nine $\wedge$ $\neg$Six $\wedge$ $\neg$Three $\wedge$ $\neg$(Five $\wedge$ $\neg$Two)) $\vee$ (Four $\wedge$ $\neg$Nine $\wedge$ $\neg$Six $\wedge$ $\neg$Three $\wedge$ $\neg$(Eight $\wedge$ $\neg$Nine) $\wedge$ $\neg$(Five $\wedge$ $\neg$Two) $\wedge$ $\neg$(Seven $\wedge$ $\neg$Two)) $\vee$ (Two $\wedge$ $\neg$One $\wedge$ $\neg$Six $\wedge$ $\neg$Three $\wedge$ $\neg$(Eight $\wedge$ $\neg$Nine) $\wedge$ $\neg$(Five $\wedge$ $\neg$Two) $\wedge$ $\neg$(Four $\wedge$ $\neg$Nine) $\wedge$ $\neg$(Seven $\wedge$ $\neg$Nine) $\wedge$ $\neg$(Seven $\wedge$ $\neg$Two)) $\vee$ (Four $\wedge$ $\neg$Six $\wedge$ $\neg$Three $\wedge$ $\neg$(Eight $\wedge$ $\neg$Nine) $\wedge$ $\neg$(Five $\wedge$ $\neg$Two) $\wedge$ $\neg$(Four $\wedge$ $\neg$Nine) $\wedge$ $\neg$(Seven $\wedge$ $\neg$Nine) $\wedge$ $\neg$(Seven $\wedge$ $\neg$Two) $\wedge$ $\neg$(Two $\wedge$ $\neg$One)) $\vee$ (Zero $\wedge$ $\neg$Four $\wedge$ $\neg$Nine $\wedge$ $\neg$Six $\wedge$ $\neg$Three $\wedge$ $\neg$(Eight $\wedge$ $\neg$Nine) $\wedge$ $\neg$(Five $\wedge$ $\neg$Two) $\wedge$ $\neg$(Four $\wedge$ $\neg$Nine) $\wedge$ $\neg$(Nine $\wedge$ $\neg$Zero) $\wedge$ $\neg$(One $\wedge$ $\neg$Two) $\wedge$ $\neg$(Seven $\wedge$ $\neg$Nine) $\wedge$ $\neg$(Seven $\wedge$ $\neg$Two) $\wedge$ $\neg$(Two $\wedge$ $\neg$One))
CUB	μ net	Black_foot_albatross $\leftrightarrow$ bill_shape_hooked_seabird $\wedge$ size_medium $\wedge$ wing_pattern_solid $\wedge$ $\neg$wing color_black $\wedge$ $\neg$underparts color_white $\wedge$ $\neg$upper_tail_color_grey $\wedge$ $\neg$breast_color_white $\wedge$ $\neg$throat_color_white $\wedge$ $\neg$tail_pattern_solid $\wedge$ $\neg$crown_color_white
	ReLU net	Black_foot_albatross $\leftrightarrow$ size_medium $\wedge$ $\neg$bill_shape_allpurpose $\wedge$ $\neg$upperparts_color_black $\wedge$ $\neg$head_pattern_plain $\wedge$ $\neg$under_tail_color_black $\wedge$ $\neg$nape_color_buff $\wedge$ $\neg$wing_shape_roundedwings $\wedge$ $\neg$shape_perchinglike $\wedge$ $\neg$leg_color_grey $\wedge$ $\neg$leg_color_black $\wedge$ $\neg$bill_color_grey $\wedge$ $\neg$bill_color_black $\wedge$ $\neg$wing_pattern_multicolored
	ψ net	Black_foot_albatross $\leftrightarrow$ (bill_shape_hooked seabird $\wedge$ tail_pattern_solid $\wedge$ $\neg$underparts_color_white $\wedge$ ($\neg$breast color white $\vee$ $\neg$ wing_color_grey))
	Tree	Black_foot_albatross $\leftrightarrow$ *formula is too long
	BRL	Black_foot_albatross $\leftrightarrow$ (bill_shape_hooked_seabird $\wedge$ forehead color blue $\wedge$ $\neg$bill_color_black $\wedge$ $\neg$nape_color_white) $\vee$ (bill_shape_hooked_seabird $\wedge$ $\neg$bill_color_black $\wedge$ $\neg$nape_color_white $\wedge$ $\neg$tail_pattern_solid)
V-Dem	μ net	Elect_Dem $\leftrightarrow$ v2x_freexp_altinf $\wedge$ v2x_frassoc_thick $\wedge$ v2xel_frefair $\wedge$ v2x_elecoff $\wedge$ v2xcl_rol
	ReLU net	Elect_Dem $\leftrightarrow$ v2x_freexp_altinf $\wedge$ v2xel_frefair $\wedge$ v2x_elecoff $\wedge$ v2xcl_rol
	ψ net	Elect_Dem $\leftrightarrow$ v2x_frassoc_thick $\wedge$ v2xeg_eqaccess
	Tree	Elect_Dem $\leftrightarrow$ *formula is too long
	BRL	Elect_Dem $\leftrightarrow$ v2x_freexp_altinf $\wedge$ v2x_frassoc_thick $\wedge$ v2xel_frefair $\wedge$ v2x_elecoff $\wedge$ v2xcl_rol

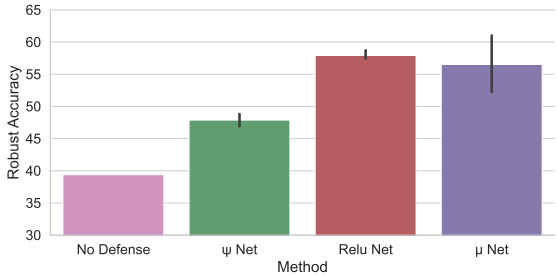


Figure 11: Quality of the explanations extracted by the LENs in terms of robust accuracy when used to defend a model from adversarial examples. In pink, the accuracy of the model when facing a white box attack without any defense. Error bars show the 95% confidence interval of the mean.

6.4 LENs as Adversarial Defense

In nowadays machine learning literature, there is a serious concerns about the vulnerability of several machine learning models, such as neural networks, to the so called *adversarial examples*. These are input samples maliciously altered with a slight perturbation that may lead to wrong predictions, even if they do not look to be visually altered for a human (in the case of images). This kind of issue has received a lot of attention, and several work has been done to develop some techniques, *adversarial defenses*, to prevent an AI model from fraudulent *adversarial attacks* [105, 106]. Recently, it has been shown how explicit domain knowledge, expressed by a set of FOL formulas, can be used to discriminate if a certain sample has to be considered as adversarial [107], especially in case of multi-label classification. However this approach requires that the logic rules are already available for the considered task, so that a domain expert is expected to collect them.

Interestingly, LENs can be applied to a set of clean data to automatically learn a set of FOL rules that may capture the important properties of the target domain. Then, these rules can be used to detect adversarial samples as in the framework introduced in [107], since they are exactly of the same type of the LEN ones.

Without any attempts of being exhaustive on this topic, we briefly explored whether what we stated in the previous sentence would work in practice. We considered the bird-identification dataset (CUB 200), facing the classification problem in its original multi-label fashion, where a convolutional neural network g (ResNet18) is trained from scratch to classify both the 200 classes and the ones of the additional bird attributes. The FOL formulas automatically extracted by the LENs in the already discussed experimental experience are directly plugged in the rejection mechanism of [107], thus making g equipped with a rejection function. We fixed the adversarial perturbation upper-bound to $\epsilon = 0.5$, and we generated perturbed data attacking the 200 classes with the state-of-the-art attack APGD-CE [108], measuring the adversarial-aware accuracy described in [107], that here we refer to as robust accuracy. The experimental results for the three out-of-the-box LENs introduced in Section 5 are shown in Fig. 11. Interestingly, the accuracy of the attacked g classifier increases in a significant manner using the rules extracted by LENs, confirming the validity of the proposed approach. The ReLU net leads to more detailed rules that better defend g, even if they introduce a larger computational burden in the defense mechanism, since they are usually more complex that the ones of the μ net.

7 Conclusions and Future Work

In this paper we presented a family of neural networks called Logic Explained Networks (LENs). We presented an in-depth study on the idea of using a neural model either to provide explanations for black-box or to solve a classification problem in an interpretable manner. Explanations are provided by First-Order Logic formulas, whose type is strongly interconnected with the learning criterion that drives LENs' training. Our framework covers a large set of computational pipelines and different user objectives. We investigated and experimented the case of three out-of-the-box LEN models, showing that they represent a balanced trade-off among a set of important properties, comparing favourably with state-of-the art white-box classifiers.

The extraction of a first-order logic explanation requires symbolic input and output spaces. This constraint is the main limitation of our framework, as it narrows the range of applications down to symbolic input/output problems. In some contexts, such as computer vision, the use of LENs may require additional annotations and attribute labels to get a consistent symbolic layer of concepts. However, recent work may partially solve this issue leading to more cost-effective concept annotations [32, 109]. Another area to focus on might be the improvement of out-of-the-box LENs. The efficiency and the classification performances of fully interpretable LENs, i.e. ψ network, is still quite limited

due to the extreme pruning strategy. Even more flexible approaches, like μ networks, are not as efficient as standard neural models when tackling multi-class or multi-label classification problems, as they require a bigger architecture to provide a disentangled explanation for each class. In our vision this framework would thrive and express its full potential by interacting with the external world and especially with humans in a dynamic way. In this case, logic formulas might be rewritten as sentences to allow for a more natural interaction with end users. Moreover, a dynamic interaction may call for an extended expressivity leading to higher-order or temporal logic explanations. Finally, in some contexts different neural architectures as graph neural networks might be worth exploring as they may be more suitable to solve the classification problem.

Current legislation in US and Europe binds AI to provide explanations especially when the economical, ethical, or financial impact is significant [2, 3]. This work contributes to a lawful and safer adoption of some of the most powerful AI technologies allowing deep neural networks to have a greater impact on society. The formal language of logic provides clear and synthetic explanations, suitable for laypeople, managers, and in general for decision makers outside the AI research field. The experiments show how this framework can be used to aid bias identification and to make black-boxes more robust to adversarial attacks. As out-of-the-box LENs are easy to use even for neophytes and can be effectively used to understand the behavior of an existing algorithm, our approach might be used to reverse engineer competitor products, to find vulnerabilities, or to improve system design. From a scientific perspective, formal knowledge distillation from state-of-the-art networks may enable scientific discoveries or confirmation of existing theories.

Acknowledgments

We thank Ben Day, Dobrik Georgiev, Dmitry Kazhdan, and Alberto Tonda for useful feedback and suggestions.

This work was partially supported by the GO-DS21 project funded by the European Union's Horizon 2020 research and innovation programme under grant agreement No 848077 and 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). This work was also 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. Furthermore, this work was partially supported from 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.

References

[1] A. Chander, R. Srinivasan, S. Chelian, J. Wang, K. Uchino, Working with beliefs: Ai transparency in the enterprise, in: IUI Workshops, 2018. 1

[2] EUGDPR, Gdpr. general data protection regulation. (2017). 1, 29

[3] P. A. Law, Code of federal regulations, Wash.: Gov. print. off (10). 1, 29

[4] M. Goddard, The eu general data protection regulation (gdpr): European regulation that has a global impact, International Journal of Market Research 59 (6) (2017) 703–705. 1

[5] D. Gunning, Explainable artificial intelligence (xai), Defense Advanced Research Projects Agency (DARPA), nd Web 2 (2) (2017). 1

[6] A. Das, P. Rad, Opportunities and challenges in explainable artificial intelligence (xai): A survey, arXiv preprint arXiv:2006.11371 (2020). 1, 2

[7] M. Brundage, S. Avin, J. Wang, H. Belfield, G. Krueger, G. Hadfield, H. Khlaaf, J. Yang, H. Toner, R. Fong, et al., Toward trustworthy ai development: mechanisms for supporting verifiable claims, arXiv preprint arXiv:2004.07213 (2020). 1

[8] D. V. Carvalho, E. M. Pereira, J. S. Cardoso, Machine learning interpretability: A survey on methods and metrics, Electronics 8 (8) (2019) 832. 1, 2, 4

[9] C. Rudin, Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead, Nature Machine Intelligence 1 (5) (2019) 206–215. 1, 2

[10] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, L. Kagal, Explaining explanations: An overview of interpretability of machine learning, in: 2018 IEEE 5th International Conference on data science and advanced analytics (DSAA), IEEE, 2018, pp. 80–89. 1

[11] Z. C. Lipton, The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery., Queue 16 (3) (2018) 31–57. 1, 2

[12] R. Marcinkevičs, J. E. Vogt, Interpretability and explainability: A machine learning zoo mini-tour, arXiv preprint arXiv:2012.01805 (2020). 1, 4

[13] F. Doshi-Velez, B. Kim, Towards a rigorous science of interpretable machine learning, arXiv preprint arXiv:1702.08608 (2017). 2

[14] F. Doshi-Velez, B. Kim, Considerations for evaluation and generalization in interpretable machine learning, in: Explainable and interpretable models in computer vision and machine learning, Springer, 2018, pp. 3–17. 2

[15] M. A. Ahmad, C. Eckert, A. Teredesai, Interpretable machine learning in healthcare, in: Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics, 2018, pp. 559–560. 2

[16] W. Samek, G. Montavon, S. Lapuschkin, C. J. Anders, K.-R. Müller, Toward interpretable machine learning: Transparent deep neural networks and beyond, arXiv preprint arXiv:2003.07631 (2020). 2

[17] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, C. Zhong, Interpretable machine learning: Fundamental principles and 10 grand challenges, arXiv preprint arXiv:2103.11251 (2021). 2

[18] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen, Classification and regression trees, CRC press, 1984. 2, 5, 6

[19] M. Schmidt, H. Lipson, Distilling free-form natural laws from experimental data, science 324 (5923) (2009) 81–85. 2

[20] B. Letham, C. Rudin, T. H. McCormick, D. Madigan, et al., Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model, Annals of Applied Statistics 9 (3) (2015) 1350–1371. 2, 5, 6, 7

[21] M. D. Cranmer, R. Xu, P. Battaglia, S. Ho, Learning symbolic physics with graph networks, arXiv preprint arXiv:1909.05862 (2019). 2

[22] C. Molnar, Interpretable machine learning, Lulu. com, 2020. 2, 4, 5

[23] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al., Relational inductive biases, deep learning, and graph networks, arXiv preprint arXiv:1806.01261 (2018). 2

[24] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018). 2

[25] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., An image is worth 16x16 words: Transformers for image recognition at scale, arXiv preprint arXiv:2010.11929 (2020). 2

[26] Q. Xie, M.-T. Luong, E. Hovy, Q. V. Le, Self-training with noisy student improves imagenet classification, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10687–10698. 2

[27] R. Srinivasan, A. Chander, Explanation perspectives from the cognitive sciences—a survey, in: 29th International Joint Conference on Artificial Intelligence, 2020, pp. 4812–4818. 2

[28] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, P. Liang, Concept bottleneck models, in: International Conference on Machine Learning, PMLR, 2020, pp. 5338–5348. 2, 3, 22

[29] T. Miller, Explanation in artificial intelligence: Insights from the social sciences, Artificial intelligence 267 (2019) 1–38. 2

[30] H. A. Simon, Rational decision making in business organizations, The American economic review 69 (4) (1979) 493–513. 2

[31] B. Kim, M. Wattenberg, J. Gilmer, C. Cai, J. Wexler, F. Viegas, et al., Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav), in: International conference on machine learning, PMLR, 2018, pp. 2668–2677. 2, 7

[32] A. Ghorbani, J. Wexler, J. Y. Zou, B. Kim, Towards automatic concept-based explanations, Advances in Neural Information Processing Systems 32 (2019). 2, 28

[33] H. McColl, The calculus of equivalent statements (third paper), Proceedings of the London Mathematical Society 1 (1) (1878) 16–28. 2

[34] W. V. Quine, The problem of simplifying truth functions, The American mathematical monthly 59 (8) (1952) 521–531. 2

[35] E. J. McCluskey, Minimization of boolean functions, The Bell System Technical Journal 35 (6) (1956) 1417–1444. 2

[36] G. Ciravegna, F. Giannini, S. Melacci, M. Maggini, M. Gori, A constraint-based approach to learning and explanation., in: AAAI, 2020, pp. 3658–3665. 3, 4, 14, 15, 16, 17, 18, 26

[37] G. Ciravegna, F. Giannini, M. Gori, M. Maggini, S. Melacci, Human-driven fol explanations of deep learning, in: Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence {IJCAI-PRICAI-20}, International Joint Conferences on Artificial Intelligence Organization, 2020, pp. 2234–2240. 3, 4, 11, 14, 15, 16, 18, 26

[38] A. R. Tavares, P. Avelar, J. M. Flach, M. Nicolau, L. C. Lamb, M. Vardi, Understanding boolean function learnability on deep neural networks, arXiv preprint arXiv:2009.05908 (2020). 4, 5, 24

[39] A. Adadi, M. Berrada, Peeking inside the black-box: a survey on explainable artificial intelligence (xai), IEEE access 6 (2018) 52138–52160. 4

[40] D. Erhan, A. Courville, Y. Bengio, Understanding representations learned in deep architectures, Department dInformatique et Recherche Operationnelle, University of Montreal, QC, Canada, Tech. Rep 1355 (1) (2010). 5

[41] K. Simonyan, A. Vedaldi, A. Zisserman, Deep inside convolutional networks: Visualising image classification models and saliency maps, arXiv preprint arXiv:1312.6034 (2013). 5

[42] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: European conference on computer vision, Springer, 2014, pp. 818–833. 5

[43] M. T. Ribeiro, S. Singh, C. Guestrin, Model-agnostic interpretability of machine learning, arXiv preprint arXiv:1606.05386 (2016). 5

[44] M. T. Ribeiro, S. Singh, C. Guestrin, " why should i trust you?" explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 2016, pp. 1135–1144. 5

[45] S. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, arXiv preprint arXiv:1705.07874 (2017). 5

[46] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, Grad-cam: Visual explanations from deep networks via gradient-based localization, in: Proceedings of the IEEE international conference on computer vision, 2017, pp. 618–626. 5

[47] E. Angelino, N. Larus-Stone, D. Alabi, M. Seltzer, C. Rudin, Learning certifiably optimal rule lists for categorical data (2018). arXiv:1704.01701. 5

[48] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, F. Giannotti, Local rule-based explanations of black box decision systems, arXiv preprint arXiv:1805.10820 (2018). 5

[49] M. T. Ribeiro, S. Singh, C. Guestrin, Anchors: High-precision model-agnostic explanations, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018. 6, 26

[50] S. Waeldchen, J. Macdonald, S. Hauch, G. Kutyniok, The computational complexity of understanding binary classifier decisions, Journal of Artificial Intelligence Research 70 (2021) 351–387. 6

[51] A. Shih, A. Choi, A. Darwiche, A symbolic approach to explaining bayesian network classifiers, in: Proceedings of the 27th International Joint Conference on Artificial Intelligence, 2018, pp. 5103–5111. 6

[52] Y. Izza, A. Ignatiev, N. Narodytska, M. C. Cooper, J. Marques-Silva, Efficient explanations with relevant sets, arXiv preprint arXiv:2106.00546 (2021). 6

[53] M. Sato, H. Tsukimoto, Rule extraction from neural networks via decision tree induction, in: IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222), Vol. 3, IEEE, 2001, pp. 1870–1875. 6

[54] J. R. Zilke, E. Loza Mencía, F. Janssen, Deepred – rule extraction from deep neural networks, in: T. Calders, M. Ceci, D. Malerba (Eds.), Discovery Science, Springer International Publishing, Cham, 2016, pp. 457–473. 6

[55] T. Hastie, R. Tibshirani, Generalized additive models: some applications, Journal of the American Statistical Association 82 (398) (1987) 371–386. 6

[56] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, N. Elhadad, Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission, in: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015, pp. 1721–1730. 6

[57] R. Agarwal, N. Frosst, X. Zhang, R. Caruana, G. E. Hinton, Neural additive models: Interpretable machine learning with neural nets, arXiv preprint arXiv:2004.13912 (2020). 6

[58] J. R. Quinlan, Simplifying decision trees, International journal of man-machine studies 27 (3) (1987) 221–234. 6

[59] W. W. Cohen, Fast effective rule induction, in: Machine learning proceedings 1995, Elsevier, 1995, pp. 115–123. 7

[60] L. G. Valiant, A theory of the learnable, in: Proceedings of the sixteenth annual ACM symposium on Theory of computing, ACM, 1984, pp. 436–445. 7

[61] C. Bessiere, F. Koriche, N. Lazaar, B. O'Sullivan, Constraint acquisition, Artificial Intelligence 244 (2017) 315–342. 7

[62] S. Kolb, S. Teso, A. Passerini, L. De Raedt, Learning smt (lra) constraints using smt solvers, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence,{IJCAI-18}, Vol. 104, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 2067–2073. 7

[63] J. R. Quinlan, Learning logical definitions from relations, Machine learning 5 (3) (1990) 239–266. 7

[64] S. Muggleton, Inductive logic programming, New generation computing 8 (4) (1991) 295–318. 7

[65] L. De Raedt, A. Dries, T. Guns, C. Bessiere, Learning constraint satisfaction problems: An ilp perspective, in: Data Mining and Constraint Programming, Springer, 2016, pp. 96–112. 7

[66] M. Richardson, P. Domingos, Markov logic networks, Machine learning 62 (1) (2006) 107–136. 7

[67] S. H. Bach, M. Broecheler, B. Huang, L. Getoor, Hinge-loss markov random fields and probabilistic soft logic, arXiv preprint arXiv:1505.04406 (2015). 7

[68] P. Domingos, D. Lowd, Markov logic: An interface layer for artificial intelligence, Synthesis lectures on artificial intelligence and machine learning 3 (1) (2009) 1–155. 7

[69] P. Campigotto, R. Battiti, A. Passerini, Learning modulo theories for preference elicitation in hybrid domains, arXiv preprint arXiv:1508.04261 (2015). 7

[70] F. Yang, Z. Yang, W. W. Cohen, Differentiable learning of logical rules for knowledge base reasoning, in: Advances in Neural Information Processing Systems, 2017, pp. 2319–2328. 7

[71] L. De Raedt, A. Dries, I. Thon, G. Van den Broeck, M. Verbeke, Inducing probabilistic relational rules from probabilistic examples, in: Twenty-fourth international joint conference on artificial intelligence, 2015. 7

[72] E. Mendelson, Introduction to mathematical logic, CRC press, 2009. 12

[73] S. J. Russell, P. Norvig, Artificial intelligence: a modern approach, Malaysia; Pearson Education Limited,, 2016. 13

[74] H. A. Simon, Rational choice and the structure of the environment., Psychological review 63 (2) (1956) 129. 14

[75] G. Gnecco, M. Gori, S. Melacci, M. Sanguineti, Foundations of support constraint machines, Neural computation 27 (2) (2015) 388–480. 14, 15

[76] M. Gori, S. Melacci, Constraint verification with kernel machines, IEEE Transactions on Neural Networks and Learning Systems 24 (5) (2013) 825–831. doi:10.1109/TNNLS.2013.2241787. 15

[77] S. Melacci, M. Gori, Unsupervised learning by minimal entropy encoding, IEEE transactions on neural networks and learning systems 23 (12) (2012) 1849–1861. 15

[78] Aristotle, Posterior analytics (n.d.). 16

[79] D. J. MacKay, D. J. Mac Kay, Information theory, inference and learning algorithms, Cambridge university press, 2003. 16

[80] A. G. Wilson, The case for bayesian deep learning, arXiv preprint arXiv:2001.10995 (2020). 16

[81] J. Kukačka, V. Golkov, D. Cremers, Regularization for deep learning: A taxonomy, arXiv preprint arXiv:1710.10686 (2017). 16

[82] F. Santosa, W. W. Symes, Linear inversion of band-limited reflection seismograms, SIAM Journal on Scientific and Statistical Computing 7 (4) (1986) 1307–1330. 16

[83] R. Tibshirani, Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society: Series B (Methodological) 58 (1) (1996) 267–288. 16

[84] Y. LeCun, J. S. Denker, S. A. Solla, R. E. Howard, L. D. Jackel, Optimal brain damage., in: NIPs, Vol. 2, Citeseer, 1989, pp. 598–605. 16

[85] B. Hassibi, D. G. Stork, Second order derivatives for network pruning: Optimal brain surgeon, Morgan Kaufmann, 1993. 16

[86] J. Frankle, M. Carbin, The lottery ticket hypothesis: Finding sparse, trainable neural networks, arXiv preprint arXiv:1803.03635 (2018). 16

[87] G. A. Miller, The magical number seven, plus or minus two: Some limits on our capacity for processing information, Psychological review 63 (1956) 81–97. 16

[88] N. Cowan, The magical number 4 in short-term memory: A reconsideration of mental storage capacity, Behavioral and brain sciences 24 (1) (2001) 87–114. 16

[89] W. J. Ma, M. Husain, P. M. Bays, Changing concepts of working memory, Nature neuroscience 17 (3) (2014) 347. 16

[90] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, H. S. Seung, Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit, Nature 405 (6789) (2000) 947–951. 17

[91] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: Proceedings of the fourteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2011, pp. 315–323. 19

[92] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, arXiv preprint arXiv:1710.05941 (2017). 19

[93] P. Barbiero, G. Ciravegna, D. Georgiev, F. Giannini, Pytorch, explain! a python library for logic explained networks, arXiv preprint arXiv:2105.11697 (2021). 20, 34

[94] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., Pytorch: An imperative style, high-performance deep learning library, arXiv preprint arXiv:1912.01703 (2019). 21

[95] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al., Scikit-learn: Machine learning in python, the Journal of machine Learning research 12 (2011) 2825–2830. 21

[96] M. Saeed, M. Villarroel, A. T. Reisner, G. Clifford, L.-W. Lehman, G. Moody, T. Heldt, T. H. Kyaw, B. Moody, R. G. Mark, Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database, Critical care medicine 39 (5) (2011) 952. 21

[97] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, H. E. Stanley, Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals, circulation 101 (23) (2000) e215–e220. 21

[98] Y. LeCun, The mnist database of handwritten digits, http://yann. lecun. com/exdb/mnist/ (1998). 21

[99] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778. 22

[100] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The caltech-ucsd birds-200-2011 dataset (2011). 22

[101] D. Pemstein, K. L. Marquardt, E. Tzelgov, Y.-t. Wang, J. Krusell, F. Miri, The v-dem measurement model: latent variable analysis for cross-national and cross-temporal expert-coded data, V-Dem Working Paper 21 (2018). 22

[102] M. Coppedge, J. Gerring, C. H. Knutsen, S. I. Lindberg, J. Teorell, D. Altman, M. Bernhard, A. Cornell, M. S. Fish, L. Gastaldi, et al., V-dem codebook v11 (2021). 22

[103] J. Lever, M. Krzywinski, N. Altman, Points of significance: model selection and overfitting, Nature methods 13 (9) (2016) 703–705. 22

[104] R. T. Marler, J. S. Arora, Survey of multi-objective optimization methods for engineering, Structural and multidisciplinary optimization 26 (6) (2004) 369–395. 25

[105] D. J. Miller, Z. Xiang, G. Kesidis, Adversarial learning targeting deep neural network classification: A comprehensive review of defenses against attacks, Proceedings of the IEEE 108 (3) (2020) 402–433. 28

[106] M. Ozdag, Adversarial attacks and defenses against deep neural networks: a survey, Procedia Computer Science 140 (2018) 152–161. 28

[107] S. Melacci, G. Ciravegna, A. Sotgiu, A. Demontis, B. Biggio, M. Gori, F. Roli, Domain knowledge alleviates adversarial attacks in multi-label classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence early access (2021) 1–1. doi:10.1109/TPAMI.2021.3137564. 28

[108] F. Croce, M. Hein, Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks (2020). arXiv:2003.01690. 28

[109] D. Kazhdan, B. Dimanov, M. Jamnik, P. Liò, A. Weller, Now you see me (cme): Concept-based model extraction, arXiv preprint arXiv:2010.13233 (2020). 28

A Python APIs for LENs

In order to make LENs paradigms accessible to the whole community, we released "PyTorch, Explain!" [93], a Python package[10] with an extensive documentation on methods and low-level APIs. Low levels APIs allow the design of custom LENs as illustrated in the example of Listing 2.

```python
import torch
from torch.nn.functional import one_hot
import torch_explain as te
from torch_explain.nn.functional import l1_loss
from torch_explain.logic.nn import psi
from torch_explain.logic.metrics import test_explanation, complexity

# train data
x_train = torch.tensor([
    [0, 0],
    [0, 1],
    [1, 0],
    [1, 1],
], dtype=torch.float)
y_train = torch.tensor([0, 1, 1, 0], dtype=torch.float).unsqueeze(1)

# instantiate a "psi network"
layers = [
    torch.nn.Linear(x_train.shape[1], 10),
    torch.nn.Sigmoid(),
    torch.nn.Linear(10, 5),
    torch.nn.Sigmoid(),
    torch.nn.Linear(5, 1),
    torch.nn.Sigmoid(),
]
model = torch.nn.Sequential(*layers)

# fit (and prune) the model
optimizer = torch.optim.AdamW(model.parameters(), lr=0.01)
loss_form = torch.nn.BCELoss()
model.train()
for epoch in range(6001):
    optimizer.zero_grad()
    y_pred = model(x_train)
    loss = loss_form(y_pred, y_train) + 0.000001 * l1_loss(model)
    loss.backward()
    optimizer.step()

    model = prune_equal_fanin(model, epoch, prune_epoch=1000, k=2)

# get first-order logic explanations for a specific target class
y1h = one_hot(y_train.squeeze().long())
explanation = psi.explain_class(model, x_train)

# compute explanation accuracy and complexity
accuracy, preds = test_explanation(explanation, x_train, y1h, target_class=1)
explanation_complexity = complexity(explanation)
```

Listing 2: Example on how to use the "PyTorch Explain!" library to solve the XOR problem.

[10] https://pypi.org/project/torch-explain/.