Transparent Explainable Logic Layers

Alessio Ragno[a,*], **Marc Plantevit**[b], **Celine Robardet**[c] **and Roberto Capobianco**[d]

[a]Sapienza University of Rome, Rome, Italy
[b]EPITA Research Laboratory (LRE), FR-94276, Le Kremlin-Bicêtre, France
[c]INSA Lyon, CNRS, LIRIS UMR 5205, F-69621 Villeurbanne, France
[d]Sony AI

Abstract. Explainable AI seeks to unveil the intricacies of black box models through post-hoc strategies or self-interpretable models. In this paper, we tackle the problem of building layers that are intrinsically explainable through logic rules. In particular, we address current state-of-the-art methods' lack of fidelity and expressivity by introducing a transparent explainable logic layer (TELL). We propose to constrain a feed-forward layer with positive weights, which, combined with particular activation functions, offer the possibility of a direct translation into logic rules. Additionally, this approach overcomes the limitations of previous models, linked to their applicability to binary data only, by proposing a new way to automatically threshold real values and incorporate the obtained predicates into logic rules. We show that, compared to state-of-the-art, TELL achieves similar classification performances and, at the same time, provides higher explanatory power, measured by the agreement between models' outputs and the activation of the logic explanations. In addition, TELL offers a broader spectrum of applications thanks to the possibility of its use on real data.

1 Introduction

Explainable AI (XAI) is a branch of machine learning and deep learning that focuses on providing explanations for black box models. XAI methods offer explanations in a variety of forms, such as input attributions [19, 21] and counterfactuals [7, 22]. In this work, we focus on providing global explanations in the form of logic rules [8], by transforming the model's reasoning process into a set of logic conditions explaining its global behavior.

Classic tree-based algorithms [17] provide global rule-based explanations. However, their rigid training procedures hinder integration with advanced models like convolutional neural networks and transformers.

Logic-explained networks (LENs) [5], instead, are a family of neural networks that are designed to be interpreted through first-order logic. The idea is that, assuming the input and the output are binary, it is possible to construct truth tables to study these models and extract logical rules describing their decision processes. Nevertheless, due to the post-hoc procedure based on truth tables, the logical explanations are not guaranteed to be aligned with the model.

The intersection of learning systems with logical reasoning is also explored in neuro-symbolic AI (NeSy). In particular, a subfield of NeSy investigates rule induction [13, 28], which is the task of learning a set of independent logical rules in disjunctive normal form (DNF) to perform classification. The works in this area try to build models with logic-inspired architectures by featuring particular layers that are constrained to behave like logical operators. Nonetheless, the number of rules that can be extracted depends on such constraints, which can impact the expressivity of the models.

In the present article, we introduce a novel logic-explainable architecture called transparent explainable logic layer (TELL) to ensure high performances across diverse data types while providing faithful explanations at the same time. In particular, we propose a constrained feed-forward layer with non-negative weights. We show that this formulation allows us to extract explanations that are perfectly aligned with the model through an efficient enumeration strategy.

Alongside tabular data, we also investigate an application to concept-bottleneck [9, 24] and prototype-based models [2, 14, 18, 25]. These architectures represent a common approach to building deep neural networks that are explainable by design. While concept-based models ground the input to binary concepts that are previously labeled, prototype-based models try to extract representative instances (prototypes) in an unsupervised fashion. Both concept- and prototype-based models consist of two parts: a representation learning module and a linear classification layer. Ciravegna et al. [5] show that the latter can be replaced with LENs to provide logic explanations. However, the binary constraint of LENs prevents their application to prototype-based models, which use real values to present the similarity between inputs and prototypes.

To address the limitation of binary input constraints, we also introduce a preprocessing head to our layer, which allows us to make TELL applicable to real-valued data and prototype-based models. Figure 1 provides a practical example of TELL paired with concept- and prototype-based models, showing the explanations for one of the classes of the CUB dataset [23] (yellow-headed blackbird). The explanations consist of logic rules over inequalities on input concept activations or prototype similarities.

Overall, this work's contributions are summarized in the following key points:

- we propose a neural network layer explicitly designed to deliver logic explanations, conducting a comprehensive analysis and benchmarking TELL against state-of-the-art methods in both classification and explanation;
- we expand the domain of applicability of LENs to accommodate non-binary data values, broadening their application scope;
- we provide an open-source implementation, serving as a valuable baseline for future work in the field.[1]

* Corresponding Author. Email: ragno@diag.uniroma1.it.

[1] The code is available at the following link: github.com/KRLGroup/TELL

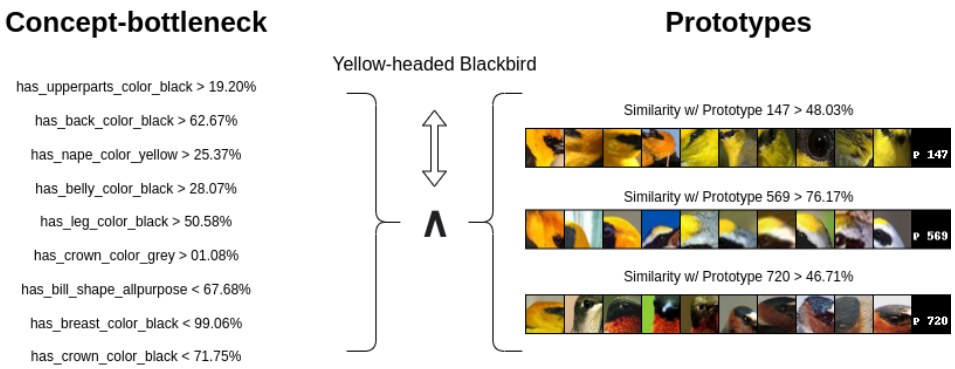

Figure 1: Visualization of learned rules. We show an example of the rule for a class of the CUB dataset learned by our logic layer, starting from the concepts and the prototypes' activations, respectively. On the left, using a concept-bottleneck encoder, the rule is presented as the conjunction of inequalities on the probability that a certain concept is present in the image. On the right, using PIP-Net [14], a prototype-based network, as an encoder, the rule is described as a conjunction of inequalities on the maximum similarity between a set of prototypes and the input image patches.

We structure our work as follows: Section 2 reviews relevant literature closely related to our research; Section 3 outlines the proposed approach, including mathematical proofs and implementation strategies; Section 4 presents a series of experiments conducted to validate our proposal; in Section 5 we report two ablation studies to evaluate the efficacy of some implementation choices; and finally, Section 6 concludes the paper by summarizing the results and discussing the contributions, limitations, and potential future extensions of our work.

2 Related Work

In this section, we review the relevant literature pertaining to our work. We begin by introducing XAI, with a particular emphasis on methods capable of providing logical explanations. Within the field of XAI, we also discuss concept-based and prototype-based models, as outlined by Ciravegna et al. [5], which offer practical applications for our models. Finally, we explore NeSy, an emerging field that combines symbolic reasoning with neural networks. This field encompasses a broad scope of application, and from recent surveys [13, 28], we find knowledge induction as particularly aligned with our aim as it involves learning logical rules for classification tasks.

Explainable AI XAI constitutes a wide field, with extensive literature proposing diverse ways of explaining neural networks. Some methods [12, 19, 21], for instance, try to identify relevant portions of the inputs for the decision process, while others [7, 22] focus on providing data samples that can help understand the model's behavior. This study specifically concentrates on deriving explanations in the form of logic formulas that describe the decision processes of neural networks [8].

Our work is closely connected to that of Ciravegna et al. [5], who put forth a family of neural models called logic-explained networks (LENs), tailored for categorical learning, seamlessly integrating aspects from both deep learning and logic. A LEN is a function $f : C_i \rightarrow C_o$ that maps an input concept space $C_i = [0, 1]^I$ to an output concept space $C_o = [0, 1]^O$, where I and O denote the input and output dimensions. Using this form, Ciravegna et al. [5] propose to provide explanations in the form of first-order logic rules directly distilled from f by building a truth table on inputs and outputs.

To enhance the interpretability of extracted logic rules, LENs incorporate regularization techniques to filter out non-relevant input features, allowing the construction of logic rules dependent on a concise set of features. For instance, ψ networks [3, 4] employ a sigmoid activation function coupled with L1-regularization, which shrinks less important weight values toward zero. Additionally, they implement a pruning strategy that forces all neurons to utilize the same number of inputs. In contrast, μ networks [5] eliminate the sigmoid activation function and conduct pruning at the network level, filtering out less crucial inputs. ReLU networks [5], instead, employ rectified linear unit activation function across all layers and leverage L1-regularization on network weights, with pruning applied exclusively during explanation extraction. Ciravegna et al. [5] justify this approach for enhancing the classification performance of ψ networks without altering the network architecture, at the expense of obtaining less faithful explanations. Finally, entropy-based LENs [1] present a wholly distinct approach by replacing L1-regularization with an entropy-based layer, which learns a mask over inputs for each class.

Although other machine learning models allow extracting logic rules as explanations, such as DT [17] and Bayesian Rule Lists (BRL) [27], they cannot be trained through backpropagation. This aspect makes them unsuited for cases with complex data types that involve neural networks. This is not the case for LEN, which can instead be directly applied to neural networks. A notable application domain for LENs involves self-explainable neural networks, a category of neural models designed for a more interpretable inference process than black boxes. Concept- and prototype-based methods are popular self-explainable models that achieve interpretability in different ways. Concept-based methods seek to streamline the understanding of neural networks by bifurcating them into two distinct classifiers: a concept encoder responsible for predicting a set of labels describing the input and an interpretable classifier, often a linear layer, which takes the concept probabilities and predicts the class. Concept-bottleneck models [9, 24] can enable interventions at the concept level through human-model interaction, making them particularly appealing for sensitive applications. However, obtaining datasets labeled with concepts is not always feasible, prompting alternative solutions. In such cases, prototype-based methods emerge as viable alternatives. Prototypes represent inputs or portions thereof that function as references to make predictions. When the model classifies an input, the inference involves comparing its features with those of learned prototypes, offering insights into the decision-making process. Most prototype-based models like ProtoPNet [2], TesNet [25], and PIP-Net [14] typically employ a two-part architecture comprising a prototype extractor and an interpretable linear layer. In this context, prototype activations do not generally exist in binary space, posing a challenge for the application of LENs.

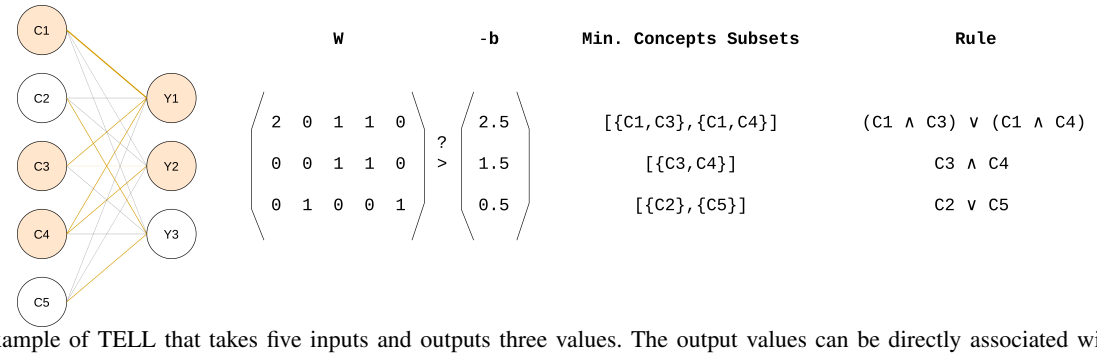

Figure 2: Example of TELL that takes five inputs and outputs three values. The output values can be directly associated with logic rules: the subsets of inputs such that the sum of the weights is bigger than the opposite of the bias. The network links are colored in orange (the thickness is proportional to the respective weight values). In the example, we show the case of an input where inputs 1, 3, and 4 are activated (orange-colored input neurons). Consequently, outputs 1 and 2 are activated by the layer (orange-colored output neurons).

Neuro-symbolic AI While we focus on designing a model whose reasoning process can be transformed into a set of rules, NeSy, on the contrary, embeds symbolic reasoning into learning systems. This field is composed of a spectrum of architectures and techniques that aim at solving several tasks that span from constraining reasoning with domain knowledge to knowledge graph completion and generative tasks [13, 28]. However, our work only intersects with a small subfield of NeSy [13, 28], knowledge induction, that aims at identifying logic rules in data that can be used for classification. DR-Net [16] falls in our category and is an architecture composed of two layers. The former is a linear layer constrained to extract conjunctions between features that form rules. These rules are then passed to an OR layer, aggregating them to form a final DNF condition. To design this architecture, the authors exploit the assumption of having binary inputs. This allows them to mimic the AND and the OR operators through the dot product operation. This structure has a limitation: the number of literals in the DNF corresponds to the hidden size of the network. This means that, to shape the layer perfectly, the user should be provided with previous domain knowledge. Relational Rule Network (R2N) [10] is another module similar to DR-Net, with a literal learning layer that comes before the AND and OR layers, and it is used to transform the input features into literals used in the final logic rule. Rule-based Representation Learner (RRL) [26] expands the structure of the network by combining a stack of logical layers followed by a linear layer. The logic layers are particular architectures composed of two submodules: the first is responsible for conjunction, the latter for disjunction. The particularity of RRL is that the model comes in two coexisting forms: discrete and continuous. The former is used for inference, while the latter is used during training. For this reason, the training involves a particular procedure called gradient grafting [26]. RRL changes the rule presentation differently from the other methods cited so far. In fact, instead of returning a unique logic rule in the DNF form, in RRL, several rules are returned paired with a score that indicates their importance.

All these methods focus on rule induction without emphasizing explanations quality other than rule complexity. In the experimental section, we will compare our proposal with DR-Net, the only method disposing of official code that produces rules in the same form as the other explainable methods in analysis.

3 Transparent Explainable Logic Layers

Our objective is to develop a model that can be easily interpreted by inspecting its weights to extract logic rules. Differently from LENs, where models consist of regularized multi-layer perceptron explained through post-hoc truth tables, we aim to design a specific layer whose weights are converted into rules. In this section, we show that we can achieve such an aim by adding non-negativity constraints on the weights of a feed-forward layer. We start by defining a layer in the binary space and show how we can use backtracking to efficiently enumerate logic rules from the weights. Finally, we extend our proposal to real numbers by adding a preprocessing head that learns thresholds over the feature values. Let us consider the following architecture:

$$y = f(X) = \sigma(XW_+^T + b). \tag{1}$$

Here, $X \in C_i$ represents binary input features, $y \in C_o$ denotes predicted output probabilities, $W_+ \in \mathbb{R}_{\geq 0}^{O \times I}$, $b \in \mathbb{R}^O$ are the weights and bias, respectively, and σ is the sigmoid activation function:

$$\sigma(z) = \frac{1}{1 + e^{-\tau z}}, \tag{2}$$

where $\tau \in \mathbb{R}_{>0}$ is a temperature hyperparameter that skews the logistic function to make it more steep. To determine the output value, we binarize y using a threshold of 0.5. The k-th output value is defined as:

$$y_k^{\text{bin}} = \begin{cases} 1 & \text{if } y_k > 0.5 \\ 0 & \text{otherwise} \end{cases} \qquad k \in [1, .., O]. \tag{3}$$

Despite our proposal and LENs sharing the same input and output spaces, the main difference lies in the explanation procedure. Indeed, we demonstrate that we can transform our layer into a logic rule without the need to build truth tables.

Proposition 1. *Let f be a model in the form of Equation 1, and y_k^{bin} be the binarized form of the k-th output of f. We can express the values of y_k^{bin} as a logic rule in DNF of the inputs. The literals of the DNF are all the minimum subsets of inputs where the corresponding weights in the k-th row of W_+ sum up to a value greater than b_k, the k-th element of b.*

Proof. Since the output is binarized at a threshold of 0.5 and $\sigma(z) > 0.5 \iff z > 0$, then we have:

$$y_k^{\text{bin}} = 1 \iff XW_{+k}^T + b_k > 0. \tag{4}$$

Given that the input X is binary and belongs to the input space $C_i = [0, 1]^I$, we have:

$$XW_{+k}^T = \sum_{i=1}^{I} X_i W_{+ki} = \sum_{i=1}^{I} \mathbb{1}_{i \in \mathcal{X}} W_{+ki}, \tag{5}$$

where $\mathcal{X} = \{i \in [1, ..., I] : X_i = 1\}$. Utilizing Equation 4 and Equation 5, to find the explanation for y_k^{bin}, we identify all combina-

tions of inputs such that $y_k^{\text{bin}} = 1$. Therefore, we define the explanation $\mathcal{E}_k$ of y_k^{bin} as:

$$\mathcal{E}_k = \{\epsilon \in \mathbb{P}(\{1, ..., I\}) : \sum_{i=1}^{I} \mathbb{1}_{i \in \epsilon} W_{+ki} > -b_k\}, \tag{6}$$

where $\mathbb{P}([1, ..., I])$ is the power set of the indices between 1 and I. The explanation $\mathcal{E}_k$ can be finally expressed in a DNF logic formula as a disjunction of the conjunction of the sets of $\mathcal{E}_k$:

$$\text{logic}(\mathcal{E}_k) = \bigvee_{\epsilon \in \mathcal{E}_k} \left(\bigwedge_{e \in \epsilon} e \right). \tag{7}$$

□

With Proposition 1, we can now convert a layer in the form of Equation 1 into a logic formula by enumerating the minimal subsets of features whose corresponding weights sum up to a value greater than the negative of the bias. We represent an example of this procedure in Figure 2: we show a layer with 5 inputs and 3 outputs. The layer is pictured as a set of weights and biases (the integer nature of the values is solely due to readiness purposes, the same reasoning applies for real weight and bias values), and we show how we extract minimal subsets and, finally, how they are converted into rules. This result recalls the concept of abductive explanations [6], which aim at identifying the smallest subsets of features responsible for a prediction. The main difference is that the rule we obtain does not explain a single prediction but the whole layer.

As this formulation only applies rules on positive input values, we can account for the possibility of negative literals by duplicating the input space as follows:

$$\tilde{I} = 2I, \text{ and } \quad \tilde{X} = \begin{pmatrix} X & 1 - X \end{pmatrix} \in [0, 1]^{\tilde{I}}. \tag{8}$$

In the remainder of this section, we present how to practically implement a layer under the premises of Proposition 1, the training procedure we adopt, and an efficient way of extracting explanations. Finally, we address the case of real-valued inputs by introducing a preprocessing function that learns thresholds over the input data.

Layer definition The model specified in Equation 1 relies on utilizing a W_+ matrix with non-negative values. To enforce non-negativity, the layer parameters can be transformed using a function $t : \mathbb{R} \to \mathbb{R}_{\geq 0}$. A straightforward approach might involve applying the Rectified Linear Unit (ReLU) function on the weights, which truncates negative values to 0. While simple, using the ReLU operator may pose learning challenges, as negative parameters would remain unaltered during gradient descent due to zero gradient. Alternatively, two other valid solutions can be derived using the logistic (Equation 9) or the exponential (Equation 10) functions:

$$W_+ = \sigma(W), \quad W \in \mathbb{R}^{O \times I}; \tag{9}$$

$$W_+ = \exp(W), \quad W \in \mathbb{R}^{O \times I}. \tag{10}$$

In this study, we concentrate on an implementation that combines both the logistic and the exponential functions (Equation 11). In this configuration, the logistic operator serves as a gate, selecting crucial inputs, while the exponential function scales the weights' values, which would otherwise lie in the range $[0, 1]$.

$$W_+ = \sigma(W_1) \odot \exp(W_2), \quad W_1 \in \mathbb{R}^{O \times I}, W_2 \in \mathbb{R}^{O}. \tag{11}$$

Here, $\odot$ represents the Hadamard element-wise product with broadcasting, hence $W_+ \in \mathbb{R}^{O \times I}$.

Training Procedure Given the utilization of the sigmoid activation function, we employ a binary cross-entropy (BCE) loss function to train our models. Furthermore, to promote sparsity and encourage the model to utilize the minimum required inputs, we incorporate L1-regularization. The comprehensive learning objective involves minimizing the following loss:

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + \lambda |W_+|. \tag{12}$$

In the context of the implementation outlined in Equation 11, applying L1 regularization becomes straightforward, specifically on the gating operator. For scenarios involving multi-class classification tasks, the use of the sigmoid activation function proves suboptimal as it does not yield a probability distribution across all classes. Conversely, switching to softmax compromises the interpretability sought for our model, as it goes out of the premises in Proposition 1. Therefore, we opt to train each class using a one-vs-the-rest (OvR) strategy, maintaining the sigmoid and BCE as activation and loss functions, respectively. To address non-normalized outputs, an additional orthogonality loss is introduced, encouraging output probabilities to resemble one-hot vectors:

$$\mathcal{L}_O = ||yy^T - I_O|| \tag{13}$$

Here I_O represents the identity matrix of size $O \times O$, and $|| \cdot ||$ denotes the Frobenius matrix norm. Consequently, the aggregate loss becomes:

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + \lambda(|\sigma(W_1)| + \mathcal{L}_O). \tag{14}$$

Extracting explanations As illustrated in Proposition 1, extracting explanations for the k-th output involves identifying the minimal subsets of elements in W_{+k} such that their sum exceeds $-b_k$. To expedite this process and generate a minimal DNF, we employ a recursive search in the combination space using backtracking and returning all relevant subsets. To enhance efficiency and achieve a minimal DNF, we conduct the search using sorted indices based on the values in W_{+k}. Pseudocode for this extraction process is provided in Algorithm 1 and Algorithm 2. Subsequently, the explanations are derived using Equation 7. The overall complexity of the procedure is exponential in the worst-case scenario as it enumerates all the pos-

Algorithm 1 Subset Extraction Algorithm

Input: Matrix W of size $O \times I$, Vector b of size O
Output: Vector S of size O, containing the list of subsets.
for $k = 1$ **to** O **do**
  $W_k \leftarrow k$-th row of W
  $b_k \leftarrow k$-th element of b
  $\mathcal{I}_k \leftarrow$ argsort(W_k)
  $S_k \leftarrow []$
  Call FINDSUBSETS($W_k, b_k, \{\}, 0, \mathcal{I}_k, S_k$)
  S[k] = S_k
end for

Algorithm 2 FindSubsets Function

Function FINDSUBSETS($W_k, b_k, s_k, sum, \mathcal{I}, S_k$)
Input: Vector W_k of size I, float b_k, Set s_k, float sum, List $\mathcal{I}$, List S_k
Output: In place modification of S_k.
if $sum > -b_k$ **then**
  Add s_k to S_k
else
  for i in $\mathcal{I}$ **do**
    FINDSUBSETS($W_k, b_k, s_k \cup \{W_k[i]\}, sum + W_k[i], \mathcal{I} - \{i\}$, S_k)
  end for
end if
End Function

sible combinations, and the time required depends on the number of output rules. However, we empirically observe that thanks to backtracking and regularization, we are able to extract explanations in a feasible amount of time. Additionally, pruning techniques are applied to reduce the search space dimension to address this challenge. Pruning can take various forms, such as limiting the number of inputs to a predefined value or, as implemented here, filtering out values of W_{+k} up to a specified percentile. This technique proves beneficial, given that the W_{+ki} value correlates with the number of literals where i appears. Discarding low-valued weights not only facilitates explanation extraction, but also contributes to creating more general rules, thus mitigating overfitting concerns. We apply pruning after a specific number of epochs, and after that, we continue training only the non-pruned weights. The same procedure applies to multiple stacked layers. The explanations of single layers can be merged by substituting the input of a layer with the output of the previous one.

Extension to real-valued inputs In many real-world scenarios, input values typically extend beyond the binary space. Consequently, we propose a method for applying this approach without necessitating manually set thresholds for the inputs. To achieve this, we introduce a preprocessing head in the layer, defined as follows:

$$\hat{X} = \sigma(X \odot \exp(W_i) + b_i), \qquad W_i \in \mathbb{R}^I, b_i \in \mathbb{R}^I \tag{15}$$

Applying a similar rationale as in Proposition 1, we can demonstrate that Equation 15 learns a threshold on X, generating values $\hat{X} \in [0,1]^I$ where:

$$\hat{X} > 0.5 \iff X > -\frac{b_i}{\exp(W_i)}. \tag{16}$$

The output of σ is continuous between 0 and 1. Due to this characteristic, achieving a perfect binarization of the input data is not feasible, therefore falling out of the premises of Proposition 1. To address this challenge, we propose incorporating a loss function that encourages the outputs of the preprocessing head to be closer to 0 or 1, thus making them more akin to binary data. To accomplish this, we employ the following entropy loss on $\hat{X}$:

$$\mathcal{L}_E = -\hat{X} log(\hat{X}) - (1-\hat{X}) log(1-\hat{X}). \tag{17}$$

4 Experiments

We start the experimental section by presenting a motivating experiment for our proposal through a synthetic dataset to show the capacity of the models in the exam to identify logic rules. We continue validating and comparing our approach with state-of-the-art methods by analyzing the classification and explanation performances on four real-world classification datasets taken from [5], which offer a spectrum of input data type including binary-valued features and image concept activations. Successively, we compare the performances of TELL with those of LENs on a classification task using prototypes extracted by a self-explainable neural network, PIP-Net. For all the experiments, we utilize the same dataset preparation and training procedures from [5], and we replicate each experiment 15 times for statistical significance. In all the experiments, we compare the models in terms of accuracy to check their ability to perform the classification task. At the same time, we aim to have models that produce explanations aligned with their behavior. To this aim, we use the fidelity score, defined as the F1-score between the models' outputs and the activation of the logic explanations. We detail all the hyperparameters and implementation information in the supplementary material.

Learning Logic Rules In this section, we analyze a learning scenario with real-valued data. We design a dataset with 5 features uniformly distributed in the range [0,1]. We then define the target class using the following rule:

$$y = 1 \iff (x_1 > 0.5 \wedge x_2 > 0.2) \vee (x_1 > 0.5 \wedge x_3 > 0.7) \tag{18}$$

We treat the problem as multi-class classification. Therefore, we set the outputs of the models to 2. Table 1 reports the performances of TELL, LENs, and DR-Net on this dataset. We observe that all the models manage to produce high-quality predictions, almost perfectly solving the task. This means that all the models are capable of learning the rule that determines the correct class. Focusing on the explanations, instead, we observe that using the rules to perform the classification results in an accuracy decay for all the models, except for TELL. This is because, among the methods, TELL is the only which is directly convertible into a logic rule, thanks to Proposition 1. The same result is observable when analyzing the fidelity of the models. Finally, we also report the rule explaining the best model over the 15 runs: TELL is the only model that correctly identifies the classification logic. This experiment showcases the need to define a model that is capable of learning over logic rules and, at the same time, provides explanations that are faithful with respect to the model's behavior. In the next section, instead, we analyze complex cases with real-world datasets.

Classification Performances This section compares our model's performance against other LEN models, neuro-symbolic and interpretable machine learning algorithms, including DR-Net, BRL and DTs. For these experiments, we use four datasets from [5], using the same preprocessing and training procedure:

Table 2: Datasets properties. For each dataset, we report the number of features (input concepts), the number of classes, the feature space, and how features are extracted. Finally, we also report the performance of the feature extraction modules for MNIST E/O and CUB, which are extracted using a CNN classifier.

	MIMIC-II	V-DEM	MNIST E/O	CUB
# Features	90	14	10	216
Feature Space	Binary	[0,1]	[0,1]	[0,1]
# Classes	2	2	2	200
Input type	Raw Features	Encoded Concepts	Encoded Image Concepts	Encoded Image Concepts
Feature Extraction	-	Same Model	CNN	CNN
Concept Encoder Accuracy	-	-	83.51%	99.57%

Table 1: Logic induction experiment. We test the ability of the models to learn a logic rule with real-valued data. We build a dataset where the class is determined by the following rule: $(x_1 > 0.5 \wedge x_2 > 0.2) \vee (x_1 > 0.5 \wedge x_3 > 0.7)$. We report mean values with standard deviations over 15 runs, and we highlight in bold the highest mean values and the ones within the standard error of the highest one.

Model	Accuracy (%)	Rules Accuracy (%)	Fidelity (%)	Complexity	Best Model's Rule
TELL	96.84 ± 0.07	$\mathbf{95.70 \pm 0.38}$	$\mathbf{97.79 \pm 0.39}$	3.40 ± 0.17	$(x_1 > 0.48 \wedge x_2 > 0.20) \vee (x_1 > 0.48 \wedge x_3 > 0.70)$
ENTROPY	94.68 ± 0.33	59.48 ± 0.56	74.62 ± 8.25	1.00 ± 0.00	$x_1 > 0.50$
ψ	89.98 ± 0.36	65.45 ± 0.60	64.74 ± 3.26	1.73 ± 0.13	$x_1 > 0.50$
RELU	$\mathbf{99.12 \pm 1.00}$	93.12 ± 0.14	92.92 ± 0.29	1.00 ± 0.00	$x_1 > 0.50$
μ	$\mathbf{98.38 \pm 0.41}$	92.49 ± 0.23	92.11 ± 0.34	2.93 ± 0.48	$x_1 > 0.50$
DR-NET	96.72 ± 0.37	92.70 ± 0.35	90.85 ± 0.53	133.13 ± 2.07	Explanation too long

Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II) [20], a dataset containing clinical data of patients admitted in the intensive care unit (ICU). The task is to classify between recovering and dying patients, after 28 days from ICU admission. The input features are already binary, hence they are directly used to train the models.

Varieties of democracy (V-DEM) [15] is a dataset containing a collection of indicators and indices describing the regimes of 202 countries from 1789 to 2020. In this case, we organize the models in two submodules: (i) a model is used to predict 82 indices (e.g., freedom of expression, freedom of association, equality before the law, etc) using 483 indicators of latent regime characteristics (e.g., media bias, party ban, high-court independence, initiatives permitted, etc); (ii) another model uses the prediction of the first one to predict five indices describing democracy principles (electoral, liberal, participatory, deliberative, and egalitarian). In the experiments, we measure the quality of the rules extracted by the second model.

MNIST E/O [11], is a dataset that classifies digits in the even and odd classes. For this dataset, we utilize a convolutional neural network (CNN) as concept-bottleneck, that is trained to classify the 10 digits. Finally, the logic classifiers are trained on the output probabilities of the CNN.

Caltech-UCSD Birds-200-2011 (CUB) [23] is a dataset of 11,788 images representing 200 bird species. Each image comes with 312 lower-lever binary attributes that have been manually labeled. These attributes represent visual characteristics (color, pattern, shape) of specific parts of the birds (beak, wings, tail, etc.). Similarly to the MNIST E/O dataset, we employ a CNN to predict such attributes, and we use these predictions as input features for our model.

MIMIC-II represents a baseline dataset with already binary features. Despite the apparent ease, MIMIC contains an elevate number of features which could hinder the rule identification process. In V-DEM, the models are first used to extract concepts and then to perform classification, showcasing the capability of models to perform in multi-layer settings. Finally, MNIST E/O and CUB offer to experiment with image data types. In both cases the input is real and represents the output probability of the concept encoder. We summarize the main properties of the datasets in Table 2, including the input and output dimensionalities and whether inputs are obtained through a concept-extraction neural network. In this case, we also indicate the concept accuracy of the encoders. For the case of V-DEM, as the encoder is different for each method, we report the encoder accuracy scores in the appendix.

Table 3 reports the accuracy scores together with the model's fidelity scores. The fidelity is defined as the F1-score between logic rule activations and the model's predictions. Generally, neural models exhibit superior accuracy compared to BRL and DT, although they come with a marginal decrease in explanation fidelity. Regarding classification performance, we find μ to be the model with the highest accuracy scores. We address this result to the fact that this model is less constrained than the others. At the same time, such "freedom" carries a cost regarding alignment between the logic rules and the real model behavior. Indeed, we observe that our approach achieves the highest fidelity scores across all datasets among the neural models. Unlike LENs, our approach TELL employs a OvR training schema utilizing binary cross-entropy instead of the conventional categorical cross-entropy combined with softmax. Consequently, the individual outputs of our model are not normalized to form a probability distribution, contributing to a less-than-perfect fidelity score. However, our model benefits from Proposition 1, resulting in transparent predictions. If fidelity is calculated directly on the individual outputs, a perfect 100% score is obtained for all datasets. Regarding DR-Net, being only designed for binary classification, it is only applicable to MIMIC-II and MNIST E/O, as in V-DEM, we use a two-module architecture with the first module having multiple outputs, and in CUB, there are 200 classes.

Overall, our model's constrained logic decision process allows for a direct translation into logic rules. These rules are more faithful, derived directly from the weights and not through a post-hoc truth-table procedure. While other LEN models may achieve superior classification performance in some cases due to their less constrained internal architecture, they do not provide equally robust explanations.

Prototype-based classification In this section, we present an application of TELL and LENs to a prototype-based self-explainable neural network, PIP-Net [14]. For this experiment, we leverage the pre-trained weights accessible on the official repository of PIP-Net [14][2] to generate prototypical representations of CUB images. This process yields a dataset with 768 features in the $[0, 1]$ space, each representing the activation level of a specific prototype. We use PIP-Net because it uses prototype activations in the $[0, 1]$ range compared to other prototype-based architectures. In other cases, LENs could not be applicable. We then train the models on this dataset and present the outcomes in Table 6. Our observations reveal that LENs achieve comparable accuracy with respect to the baseline model. More specifically, μ LEN surpasses the baseline model's performance. Simultaneously, our proposed approach outperforms all models in generating accurate and faithful rules. We address this result because, unlike LENs, our model can automatically learn a threshold over the features. Additionally, we include the accuracy obtained by applying individual logic rules for predicting classes using an OvR evaluation. Remarkably, our model outperforms even the best LEN model in this evaluation, showcasing the effectiveness of our proposed methodology.

[2] https://github.com/M-Nauta/PIPNet

Table 3: Classification performances. We compare the accuracy and fidelity scores of the models. We replicate each experiment 15 times for statistical validity and report mean and standard error. We highlight in bold the highest mean values and the ones within the standard error of the highest one.

	Model	Binary Datasets				Concept-Bottleneck			
		MIMIC-II		V-DEM		MNIST E/O		CUB	
		Accuracy (%)	Fidelity (%)	Accuracy (%)	Fidelity (%)	Accuracy (%)	Fidelity (%)	Accuracy (%)	Fidelity
Explainable NN	TELL (ours)	**79.28 ± 0.80**	**96.15 ± 0.62**	**92.50 ± 0.43**	**97.56 ± 0.45**	99.85 ± 0.01	**99.98 ± 0.00**	92.39 ± 0.19	**97.99 ± 0.09**
	ψ	76.57 ± 0.69	58.38 ± 3.60	91.11 ± 0.29	60.30 ± 5.10	99.84 ± 0.01	69.62 ± 2.84	92.22 ± 0.19	75.35 ± 0.92
	RELU	**79.06 ± 0.87**	75.30 ± 2.57	**92.85 ± 0.42**	85.46 ± 3.77	99.85 ± 0.01	93.72 ± 3.88	92.43 ± 0.20	97.45 ± 0.07
	μ	**79.16 ± 0.82**	91.15 ± 1.12	91.82 ± 0.42	**97.49 ± 0.48**	**99.90 ± 0.01**	99.90 ± 0.01	**92.65 ± 0.18**	94.80 ± 0.21
	ENTROPY	**78.89 ± 0.72**	71.47 ± 2.49	90.28 ± 0.60	87.44 ± 5.40	99.84 ± 0.01	41.03 ± 8.24	**92.57 ± 0.19**	97.56 ± 0.08
	DR-NET	74.16 ± 0.60	75.36 ± 1.72	N/A	N/A	85.69 ± 0.06	85.52 ± 0.07	N/A	N/A
ML	DT	76.52 ± 0.75	100.00 ± 0.00	84.91 ± 0.67	100.00 ± 0.00	99.89 ± 0.01	100.00 ± 0.00	80.59 ± 0.79	100.00 ± 0.00
	BRL	77.20 ± 1.10	100.00 ± 0.00	77.20 ± 1.10	100.00 ± 0.00	99.84 ± 0.01	100.00 ± 0.00	91.15 ± 0.45	100.00 ± 0.00

5 Ablation Study

In this section, we include two ablation studies to analyze the role of the proposed activation and loss functions in influencing both classification performances and fidelity of the rules.

Activation Functions Here, we investigate variations among the solutions proposed in Equations 9, 10 and 11, for implementing our proposed approach. Specifically, we present the outcomes of these diverse implementations on the datasets in Table 4. Generally, the form using both exponential and logistic activations consistently outperforms alternative solutions in both classification accuracy and explanation fidelity. However, it is noteworthy that in some instances, the two alternative methods achieve higher fidelity values at the expense of a significant decrease in accuracy. Additionally, the results validate our initial hypothesis, identifying weights activated through the logistic function as gates for features and those activated via the exponential function as scalers. For the model relying solely on the logistic function, having weights confined to the $[0, 1]$ interval proves suboptimal when handling non-binary input data with real-valued inputs. This is evident from the notable drops in accuracy and fidelity, particularly observed in MNIST E/O, CUB, and V-DEM datasets. Conversely, employing only the exponential activation function results in a system that struggles to filter out non-important features. This is highlighted in CUB and V-DEM, where the pruning of features significantly impacts the learning process, yielding models incapable of making accurate predictions.

Loss Functions Here, we analyze the influence of the loss functions $\mathcal{L}_O$ (Equation 13) and $\mathcal{L}_E$ (Equation 17) on the accuracy and fidelity of TELL. We replicate the experiment in Section 4 using combinations of the losses. Table 5 reports the models' accuracy, the fidelity, the extracted rules' accuracy, complexity, and the best model's learned rule. We observe that the combination of the two losses provides the best results in terms of accuracy and fidelity. This is due to the fact that $\mathcal{L}_O$ guides training to encourage the model to return "one hot" vectors in a multi-class setting. Therefore, when evaluating the fidelity as the F1-score between the models' outputs and rules activations, the models trained with this loss return better results. Similarly, as $\mathcal{L}_E$ influences the threshold of real values by encouraging the preprocessing head to return values that are closer to 0 or 1, it impacts classification and explanations.

6 Conclusions

In this study, we addressed the challenge of training a classifier capable of offering explanations through logic rules. In particular, we introduced a novel layer that is explicitly designed to incorporate logic constraints. In contrast to previous methods, which typically employ post-hoc strategies for rule extraction, our formulation ensures that rules can be discerned by examining the model weights. Our results demonstrated that our proposed approach achieves comparable classification performance compared to state-of-the-art methods and enhances the model's explanation capabilities. Simultaneously, we expanded the applicability of LENs beyond concept-based models to encompass other self-explanatory networks, allowing its application also to prototype-based models without any structural modification. Limitations of our approach lie in the explanation extraction procedure, which, in the case of highly complex datasets, could result in multiple rules and long explanation times. Future work could instead focus on the application of TELL's reasoning to more complex layers, such as attention and convolutional layers.

Acknowledgements

This work has been partially supported by the Italian PNRR MUR project "PE0000013-FAIR" and benefited from France state aid managed by the National Research Agency under France 2030 with the reference "ANR-22-PEAE-0008".

Table 6: Prototype-based classification. We report the performances of TELL and LENs trained on the prototypes learned by a PIP-Net Model. We use the pre-trained model from [14] and extract the learned prototypes. We report the accuracy of the models, the fidelity of the extracted rules, the accuracy we would obtain applying the single rules, and the explanation time. We replicate each experiment 15 times for statistical validity and report mean and standard error. We highlight in bold the highest mean values and the ones within the std of the highest one.

Model	Accuracy (%)	Rules Accuracy (%)	Fidelity (%)	Expl. Time (s)
PIP-Net	84.78	-	-	-
TELL (ours)	82.26 ± 0.10	$\mathbf{90.56 \pm 0.08}$	$\mathbf{95.22 \pm 0.09}$	59.09 ± 0.23
ψ	83.04 ± 0.07	82.53 ± 0.44	86.17 ± 0.52	15.24 ± 5.76
RELU	83.98 ± 0.07	63.07 ± 2.62	63.77 ± 2.78	152.91 ± 21.72
μ	$\mathbf{84.84 \pm 0.03}$	74.39 ± 0.07	75.58 ± 0.08	15.47 ± 5.91
ENTROPY	82.81 ± 0.19	89.45 ± 0.07	92.06 ± 0.05	36.23 ± 1.66

Table 4: Activation function ablation study. We compare three different variations of our proposed approach. The models are obtained by changing the activation function of the weights of the layers: TELL is our proposed solution that uses two weight matrices activated with the sigmoid and the exponential function, respectively; TELL_σ employs a single weight matrix that uses the logistic activation function; TELL_e uses a single weight matrix that is activated with the exponential function. We calculate the accuracy and fidelity scores of the models' predictions. We replicate each experiment 15 times for statistical validity and report mean and standard error. We highlight in bold the highest mean values and the ones within the standard error of the highest one.

Model	MIMIC-II		V-DEM		MNIST E/O		CUB	
	Accuracy (%)	Fidelity (%)	Accuracy (%)	Fidelity (%)	Accuracy (%)	Fidelity (%)	Accuracy (%)	Fidelity (%)
TELL	$\mathbf{79.28 \pm 0.80}$	96.15 ± 0.62	$\mathbf{92.50 \pm 0.43}$	$\mathbf{97.56 \pm 0.45}$	$\mathbf{99.85 \pm 0.01}$	$\mathbf{99.98 \pm 0.00}$	$\mathbf{92.39 \pm 0.19}$	97.99 ± 0.09
TELL_σ	$\mathbf{79.05 \pm 0.34}$	94.36 ± 1.06	92.03 ± 0.48	96.75 ± 0.49	$\mathbf{99.85 \pm 0.01}$	83.35 ± 5.24	92.18 ± 0.18	92.34 ± 0.24
TELL_e	72.02 ± 1.00	$\mathbf{98.33 \pm 0.01}$	53.15 ± 0.50	34.11 ± 1.07	$\mathbf{99.84 \pm 0.01}$	96.81 ± 2.01	0.51 ± 0.00	$\mathbf{99.09 \pm 0.07}$

Table 5: Loss function ablation study. We evaluate the efficiency of the loss functions $\mathcal{L}_O$ (Equation 13) and $\mathcal{L}_E$ (Equation 17). We calculate the accuracy and fidelity scores of the models' predictions on the dataset from Section 4. We replicate each experiment 15 times for statistical validity and report mean and standard error. We highlight in bold the highest mean values and the ones within the standard error of the highest one.

Model	Accuracy (%)	Rules Accuracy (%)	Fidelity (%)	Complexity	Best Model's Rule
TELL	$\mathbf{96.84 \pm 0.07}$	$\mathbf{95.70 \pm 0.38}$	$\mathbf{97.79 \pm 0.39}$	3.40 ± 0.17	$(x_1 > 0.48 \wedge x_2 > 0.20) \vee (x_1 > 0.48 \wedge x_3 > 0.70)$
TELL (no $\mathcal{L}_O$)	96.38 ± 0.07	91.88 ± 0.49	93.86 ± 2.40	3.00 ± 0.21	$(x_1 > 0.49 \wedge x_2 > 0.21) \vee (x_1 > 0.49 \wedge x_3 > 0.72)$
TELL (no $\mathcal{L}_E$)	93.44 ± 0.07	93.01 ± 1.73	95.06 ± 1.36	3.43 ± 0.43	$(x_1 > 0.48 \wedge x_2 > 0.21) \vee (x_1 > 0.48 \wedge x_3 > 0.70)$
TELL (no $\mathcal{L}_E$ no $\mathcal{L}_O$)	94.70 ± 0.07	93.42 ± 0.94	95.96 ± 1.15	3.47 ± 0.44	$(x_1 > 0.48 \wedge x_2 > 0.21) \vee (x_1 > 0.48 \wedge x_3 > 0.71)$

References

[1] P. Barbiero, G. Ciravegna, F. Giannini, P. Lió, M. Gori, and S. Melacci. Entropy-based logic explanations of neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6046–6054, Jun. 2022. doi: 10.1609/aaai.v36i6.20551. URL https://ojs.aaai.org/index.php/AAAI/article/view/20551.

[2] C. Chen, O. Li, D. Tao, A. Barnett, C. Rudin, and J. K. Su. This looks like that: deep learning for interpretable image recognition. *Advances in neural information processing systems*, 32, 2019.

[3] G. Ciravegna, F. Giannini, M. Gori, M. Maggini, and S. Melacci. Human-driven fol explanations of deep learning. In C. Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 2234–2240. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/309. URL https://doi.org/10.24963/ijcai.2020/309. Main track.

[4] G. Ciravegna, F. Giannini, S. Melacci, M. Maggini, and M. Gori. A constraint-based approach to learning and explanation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3658–3665, Apr. 2020. doi: 10.1609/aaai.v34i04.5774. URL https://ojs.aaai.org/index.php/AAAI/article/view/5774.

[5] G. Ciravegna, P. Barbiero, F. Giannini, M. Gori, P. Liò, M. Maggini, and S. Melacci. Logic explained networks. *Artificial Intelligence*, 314: 103822, 2023. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2022.103822. URL https://www.sciencedirect.com/science/article/pii/S000437022200162X.

[6] M. C. Cooper and J. Marques-Silva. Tractability of explaining classifier decisions. *Artificial Intelligence*, 316:103841, Mar. 2023. ISSN 0004-3702. doi: 10.1016/j.artint.2022.103841. URL http://dx.doi.org/10.1016/J.ARTINT.2022.103841.

[7] S. Dandl, C. Molnar, M. Binder, and B. Bischl. Multi-objective counterfactual explanations. In *International Conference on Parallel Problem Solving from Nature*, pages 448–469. Springer, 2020.

[8] A. Darwiche. Logic for explainable ai. In *2023 38th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–11. IEEE, 2023.

[9] P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang. Concept bottleneck models. In H. D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 5338–5348. PMLR, 13–18 Jul 2020.

[10] R. Kusters, Y. Kim, M. Collery, C. d. S. Marie, and S. Gupta. Differentiable rule induction with learned relational features. *arXiv preprint arXiv:2201.06515*, 2022.

[11] Y. LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[12] S. M. Lundberg and S.-I. Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[13] G. Marra, S. Dumančić, R. Manhaeve, and L. De Raedt. From statistical relational to neurosymbolic artificial intelligence: A survey. *Artificial Intelligence*, 328:104062, 2024. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2023.104062. URL https://www.sciencedirect.com/science/article/pii/S0004370223002084.

[14] M. Nauta, J. Schlötterer, M. van Keulen, and C. Seifert. Pip-net: Patch-based intuitive prototypes for interpretable image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2744–2753, 2023.

[15] D. Pemstein, K. L. Marquardt, E. Tzelgov, Y.-t. Wang, J. Krusell, and 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.

[16] L. Qiao, W. Wang, and B. Lin. Learning accurate and interpretable decision rule sets from neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4303–4311, May 2021. ISSN 2159-5399. doi: 10.1609/aaai.v35i5.16555. URL http://dx.doi.org/10.1609/aaai.v35i5.16555.

[17] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

[18] A. Ragno, B. La Rosa, and R. Capobianco. Prototype-based interpretable graph neural networks. *IEEE Transactions on Artificial Intelligence*, 2022.

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

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

[21] M. Sundararajan, A. Taly, and Q. Yan. Axiomatic attribution for deep networks. In *International conference on machine learning*, pages 3319–3328. PMLR, 2017.

[22] S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2017.

[23] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[24] B. Wang, L. Li, Y. Nakashima, and H. Nagahara. Learning bottleneck concepts in image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10962–10971, 2023.

[25] J. Wang, H. Liu, and L. Jing. Transparent embedding space for interpretable image recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[26] Z. Wang, W. Zhang, N. Liu, and J. Wang. Scalable rule-based representation learning for interpretable classification. *Advances in Neural Information Processing Systems*, 34:30479–30491, 2021.

[27] H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *International conference on machine learning*, pages 3921–3930. PMLR, 2017.

[28] D. Yu, B. Yang, D. Liu, H. Wang, and S. Pan. A survey on neural-symbolic learning systems. *Neural Networks*, 166:105–126, 2023. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2023.06.028. URL https://www.sciencedirect.com/science/article/pii/S0893608023003398.