Entropy-based Logic Explanations of Neural Networks

Pietro Barbiero[*]
University of Cambridge (UK)
pb737@cam.ac.uk

Gabriele Ciravegna[*]
Università di Firenze (Italy)
Università di Siena (Italy)
gabriele.ciravegna@unifi.it

Francesco Giannini[*]
Università di Siena (Italy)
francesco.giannini@unisi.it

Pietro Lió
University of Cambridge (UK)
pl219@cam.ac.uk

Marco Gori
Università di Siena (Italy)
Universitè Côte d'Azur (France)
marco.gori@unisi.it

Stefano Melacci
Università di Siena (Italy)
mela@diism.unisi.it

Abstract

Explainable artificial intelligence has rapidly emerged since lawmakers have started requiring interpretable models for safety-critical domains. Concept-based neural networks have arisen as explainable-by-design methods as they leverage human-understandable symbols (i.e. concepts) to predict class memberships. However, most of these approaches focus on the identification of the most relevant concepts but do not provide concise, formal explanations of how such concepts are leveraged by the classifier to make predictions. In this paper, we propose a novel end-to-end differentiable approach enabling the extraction of logic explanations from neural networks using the formalism of First-Order Logic. The method relies on an entropy-based criterion which automatically identifies the most relevant concepts. We consider four different case studies to demonstrate that: (i) this entropy-based criterion enables the distillation of concise logic explanations in safety-critical domains from clinical data to computer vision; (ii) the proposed approach outperforms state-of-the-art white-box models in terms of classification accuracy.

1 Introduction

The lack of transparency in the decision process of some machine learning models, such as neural networks, limits their application in many safety-critical domains [1, 2, 3]. For this reason, explainable artificial intelligence (XAI) research has focused either on *explaining* black box decisions [4, 5, 6, 7, 8] or on developing machine learning models *interpretable by design* [9, 10, 11, 12, 13]. However, while interpretable models engender trust in their predictions [14, 15, 16, 17, 18, 19], black box models, such as neural networks, are the ones that provide state-of-the-art task performances [20, 21, 22, 23]. Research to address this imbalance is needed for the deployment of cutting-edge technologies.

Most techniques *explaining* black boxes focus on finding or ranking the most relevant features used by the model to make predictions [24, 25, 26, 27, 28, 29]. Such feature-scoring methods are very efficient and widely used, but they cannot explain how neural networks compose such features to make predictions [30, 31, 32]. In addition, a key issue of most *explaining* methods is that explanations are given in terms of input features (e.g. pixel intensities) that do not correspond to high-level categories that humans can easily understand [31, 33]. To overcome this issue, *concept-based* approaches

[*]Equal contribution

Preprint. Under review.

have become increasingly popular as they provide explanations in terms of human-understandable categories (i.e. the *concepts*) rather than raw features [31, 34, 35, 36, 37, 38]. However, fewer approaches are able to explain how such concepts are leveraged by the classifier and even less provide concise explanations whose validity can be assessed quantitatively [27, 4, 39, 40, 41].

Contributions. In this paper, we first propose an entropy-based layer (Sec. 3.1) that enables the implementation of *concept-based* neural networks, providing First-Order Logic explanations (Fig. 1). We define our approach as *explainable by design*, since the proposed architecture allows neural networks to automatically provide logic explanations of their predictions. Second, we describe how to interpret the predictions of the proposed neural model to distill logic explanations for individual observations and for a whole target class (Sec. 3.3). Finally, we demonstrate how our approach provides high-quality explanations according to six *quantitative* metrics while outperforming state-of-the-art white-box models (described in Sec. 4) in terms of classification accuracy on four case studies (Sec. 5).

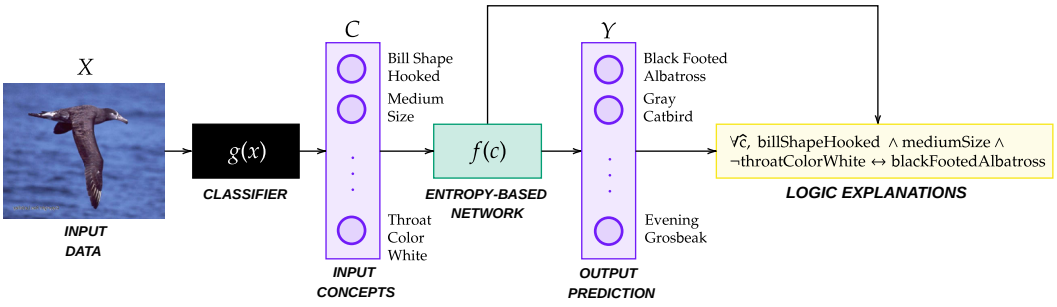

Figure 1: The proposed pipeline on one example from the CUB dataset. The proposed neural network can be regarded as a function $f\colon C \mapsto Y$ which maps concepts onto target classes and provide concise logic explanations (yellow—we dropped the arguments in the logic predicates, for simplicity) of its own decision process. When the features of the input data are non-interpretable (as pixels intensities), a classifier $g\colon X \mapsto C$ maps inputs to concepts.

2 Background

Classification is the problem of identifying a set of categories an observation belongs to. We indicate with $Y \subset \{0,1\}^r$ the space of binary encoded targets in a problem with r categories. Concept-based classifiers f are a family of machine learning models predicting class memberships from the activation scores of k human-understandable categories, $f : C \mapsto Y$, where $C \subset [0,1]^k$ (see Fig. 1). Concept-based classifiers improve human understanding as their input and output spaces consists of interpretable symbols. When observations are represented in terms of non-interpretable input features belonging to $X \subset \mathbb{R}^d$ (such as pixels intensities), a "concept decoder" g is used to map the input into a concept-based space, $g : X \mapsto C$ (see Fig. 1). Otherwise, they are simply rescaled from the unbounded space $\mathbb{R}^d$ into the unit interval $[0,1]^k$, such that input features can be treated as logic predicates.

In the recent literature, the most similar method related to the proposed approach is the ψ network proposed by Ciravegna et al. [7, 40], an end-to-end differentiable concept-based classifier *explaining its own decision process*. The ψ network leverages the intermediate symbolic layer whose output belongs to C to distill First-Order Logic formulas, representing the learned map from C to Y. The model consists of a sequence of fully connected layers with sigmoid activations only. An $L1$-regularization and a strong pruning strategy is applied to each layer of weights in order to allow the computation of logic formulas representing the activation of each node. Such constraints, however, limit the learning capacity of the network and impair the classification accuracy, making standard white-box models, such as decision trees, more attractive.

3 Entropy-based Logic Explanations of Neural Networks

The key contribution of this paper is a novel linear layer enabling entropy-based logic explanations of neural networks (see Fig. 2). The layer input belongs to the concept space C and the outcomes of the layer computations are: (i) the embeddings h^i (as any linear layer), (ii) a truth table $\mathcal{T}^i$ explaining how the network leveraged concepts to make predictions for the i-th target class. Each class of the problem requires an independent entropy-based layer, as emphasized by the superscript i. For ease of reading and without loss of generality, all the following descriptions concern inference for a single observation (corresponding to the concept tuple $c \in C$) and a neural network f^i predicting the class memberships for the i-th class of the problem. For multi-class problems, multiple "heads" of this layer are instantiated, with one "head" per target class (see Sec. 5), and the hidden layers of the class-specific networks could be eventually shared.

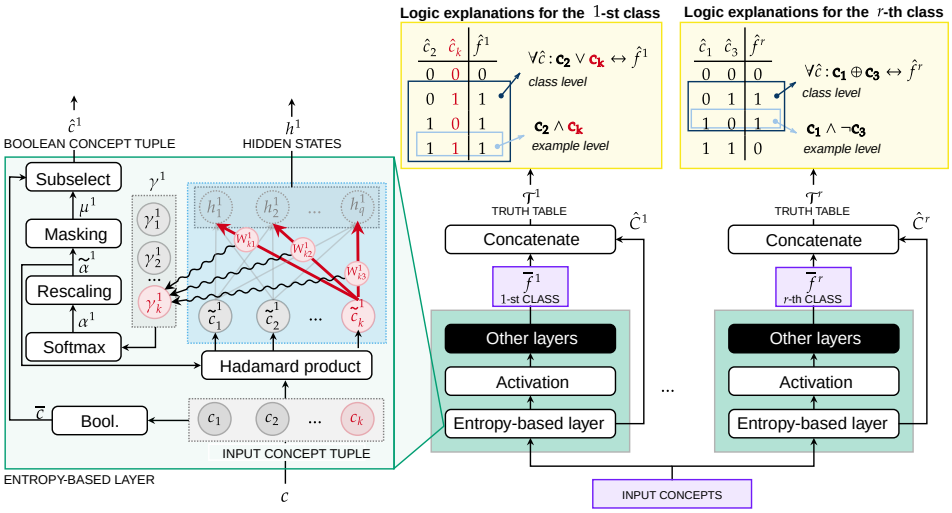

Figure 2: On the right, the proposed neural network learns the function $f : C \mapsto Y$. For each class, the network leverages one "head" of the entropy-based linear layer (green) as first layer. For each target class i, the network provides: the class membership predictions f^i and the truth table $\mathcal{T}^i$ (Eq. 6) to distill FOL explanations (yellows, top). On the left, a detailed view on the entropy-based linear layer for the 1-st class, emphasizing the role of the k-th input concept as example: (i) the scalar γ_k^1 (Eq. 1) is computed from the set of weights connecting the k-th input concept to the output neurons of the entropy-based layer; (ii) the relative importance of each concept is summarized by the categorical distribution α^1 (Eq. 2); (iii) rescaled relevance scores $\tilde{\alpha}^1$ drop irrelevant input concepts out (Eq. 3); (iv) hidden states h^1 (Eq. 4) and Boolean-like concepts $\hat{c}^1$ (Eq. 5) are provided as outputs of the entropy-based layer.

3.1 Entropy-based linear layer

When humans compare a set of hypotheses outlining the same outcomes, they tend to have an implicit bias towards the simplest ones as outlined in philosophy [42, 43, 44, 45], psychology [46, 47], and decision making [48, 49, 50]. The proposed entropy-based approach encodes this inductive bias in an end-to-end differentiable model. The purpose of the entropy-based linear layer is to encourage the neural model to pick a limited subset of input concepts, allowing it to provide concise explanations of its predictions. The learnable parameters of the layer are the usual weight matrix W and bias vector b. In the following, the forward pass is described by the operations going from Eq. 1 to Eq. 4 while the generation of the truth tables from which explanations are extracted is formalized by Eq. 5 and Eq. 6.

The relevance of each input concept can be summarized in a first approximation by a measure that depends on the values of the weights connecting such concept to the upper network. In the case of network f^i (i.e. predicting the i-th class) and of the j-th input concept, we indicate with W_j^i the vector of weights departing from the j-th input (see Fig 2), and we introduce

$$\gamma_j^i = ||W_j^i||_1 \ . \tag{1}$$

The higher γ_j^i, the higher the relevance of the concept j for the network f^i. In the limit case, $\gamma_j^i \to 0$, the model f^i drops the j-th concept out. To select only few relevant concepts for each target class, concepts are set up to compete against each other. To this aim, the relative importance of each concept to the i-th class is summarized in the categorical distribution α^i, composed of coefficients $\alpha_j^i \in [0, 1]$ (with $\sum_j \alpha_j^i = 1$), modeled by the softmax function:

$$\alpha_j^i = \frac{e^{\gamma_j^i/\tau}}{\sum_{l=1}^{k} e^{\gamma_l^i/\tau}} \tag{2}$$

where $\tau \in \mathbb{R}^+$ is a user-defined temperature parameter to tune the intrinsic tendency of the softmax function.[2] high temperature values ($\tau \to \infty$) all concepts have nearly the same relevance. For low temperatures values ($\tau \to 0$), the probability of the most relevant concept tends to ≈ 1, while it becomes ≈ 0 for all other concepts. As the probability distribution α^i highlights the most relevant concepts, this information is directly fed back to the input, weighting concepts by the estimated importance. To avoid numerical cancellation due to values in α^i close to zero, especially when the input dimensionality is large, we replace α^i with its normalized instance $\tilde{\alpha}^i$, still $\in [0, 1]^k$, and each input sample $c \in C$ is modulated by the (normalized) estimated importance,

$$\tilde{c}^i = c \odot \tilde{\alpha}^i \qquad \text{with} \qquad \tilde{\alpha}_j^i = \frac{\alpha_j^i}{\max_u \alpha_u^i}, \tag{3}$$

where $\odot$ denotes the Hadamard (element-wise) product. The highest value in $\tilde{\alpha}^i$ is always 1 (i.e. $\max_j \tilde{\alpha}_j^i = 1$) and it corresponds to the most relevant concept. The embeddings h^i are computed as in any linear layer by means of the affine transformation:

$$h^i = W^i \tilde{c}^i + b^i. \tag{4}$$

Whenever $\tilde{\alpha}_j^i \to 0$, the input $\tilde{c}_j \to 0$. This means that the corresponding concept tends to be dropped out and the network f^i will learn to predict the i-th class without relying on the j-th concept.

In order to get logic explanations, the proposed linear layer generates the truth table $\mathcal{T}^i$ formally representing the behaviour of the neural network in function of Boolean-like representations of the input concepts. In detail, we indicate with $\bar{c}$ the Boolean interpretation of the input tuple $c \in C$, while $\mu^i \in \{0, 1\}^k$ is the binary mask associated to $\tilde{\alpha}^i$. To encode the inductive human bias towards simple explanations [46, 47, 51], the mask μ^i is used to generate the binary concept tuple $\hat{c}^i$, dropping the least relevant concepts out of c,

$$\hat{c}^i = \xi(\bar{c}, \mu^i) \qquad \text{with} \qquad \mu^i = \mathbb{I}_{\tilde{\alpha}^i \geq \tau} \qquad \text{and} \qquad \bar{c} = \mathbb{I}_{c \geq \tau}, \tag{5}$$

where $\mathbb{I}_{z \geq \tau}$ denotes the indicator function that is 1 for all the components of vector z being $\geq \tau$ and 0 otherwise (considering the unbiased case, we set $\tau = 0.5$). The function ξ returns the vector with the components of $\bar{c}$ that correspond to 1's in μ^i (i.e. it sub-selects the data in $\bar{c}$). As a results, $\hat{c}^i$ belongs to a space $\hat{C}^i$ of m_i Boolean features, with $m_i < k$ due to the effects of the subselection procedure.

The truth table $\mathcal{T}^i$ is a particular way of representing the behaviour of network f^i based on the outcomes of processing multiple input samples collected in a generic dataset $\mathcal{C}$. As the truth table involves Boolean data, we denote with $\hat{\mathcal{C}}^i$ the set with the Boolean-like representations of the samples in $\mathcal{C}$ computed by ξ, Eq. 5. We also introduce $\bar{f}^i(c)$ as the Boolean-like representation of the network output, $\bar{f}^i(c) = \mathbb{I}_{f^i(c) \geq \tau}$. From an operational perspective, the contents $\mathbf{T}^i$ of the truth table $\mathcal{T}^i$ are obtained by stacking data of $\hat{\mathcal{C}}^i$ into a 2D matrix $\hat{\mathbf{C}}^i$ (row-wise), and concatenating the result with the column vector $\bar{\mathbf{f}}^i$ whose elements are $\bar{f}^i(c)$, $c \in \mathcal{C}$, that we summarize as

$$\mathbf{T}^i = \left(\hat{\mathbf{C}}^i \,\middle\|\, \bar{\mathbf{f}}^i \right). \tag{6}$$

The truth table $\mathcal{T}^i$ is used to generate logic explanations, as we will explain in Sec. 3.3.

3.2 Loss function

The entropy of the probability distribution α^i (Eq. 2),

$$\mathcal{H}(\alpha^i) = -\sum_{j=1}^{k} \alpha_j^i \log \alpha_j^i \tag{7}$$

[2]For a given set of γ_j^i, when using

is minimized when a single α_j^i is one, thus representing the extreme case in which only one concept matters, while it is maximum when all concepts are equally important. When $\mathcal{H}$ is jointly minimized with the usual loss function for supervised learning $L(f, y)$ (being y the target labels–we used the cross-entropy in our experiments), it allows the model to find a trade off between fitting quality and a parsimonious activation of the concepts, allowing each network f^i to predict i-th class memberships using few relevant concepts only. Overall, the loss function to train the network f is defined as,

$$\mathcal{L}(f, y, \alpha_1, \ldots, \alpha_r) = L(f, y) + \lambda \sum_{i=1}^{r} \mathcal{H}(\alpha^i), \tag{8}$$

where $\lambda > 0$ is the hyperparameter used to balance the relative importance of low-entropy solutions in the loss function. Higher values of λ lead to sparser configuration of α, constraining the network to focus on a smaller set of concepts for each classification task (and vice versa), thus encoding the inductive human bias towards simple explanations [46, 47, 51].

3.3 First-order logic explanations

Any Boolean function can be converted into a logic formula in Disjunctive Normal Form (DNF) by means of its truth-table [52]. We indicate with $\hat{f}^i$ the Boolean function represented by the truth table $\mathcal{T}^i$, $\hat{f}^i : \hat{C}^i \mapsto Y^i$, being Y^i the i-th component of Y. Converting a truth table into a DNF formula provides an effective mechanism to extract logic rules of increasing complexity from individual observations to a whole class of samples. The following rule extraction mechanism is considered for each task i.

FOL extraction. Each row of the truth table $\mathcal{T}^i$ can be partitioned into two parts that are a binary tuple of concept activations, $\hat{q} \in \hat{C}^i$, and the outcome of $\hat{f}^i(\hat{q}) \in \{0, 1\}$. An *example-level* logic formula, consisting in a single minterm, can be trivially extracted from each row for which $\hat{f}^i(\hat{q}) = 1$, by simply connecting with the logic AND $\wedge$ the true concepts and negated instances of the false ones. The logic formula becomes human understandable whenever concepts appearing in such a formula are replaced with human-interpretable strings that represent their name (similar consideration holds for $\hat{f}^i$, in what follows). For example, the following logic formula φ_t^i,

$$\varphi_t^i = \mathbf{c}_1 \wedge \neg \mathbf{c}_2 \wedge \ldots \wedge \mathbf{c}_{m_i}, \tag{9}$$

is the formula extracted from the t-th row of the table where, in the considered example, only the second concept is false, being $\mathbf{c}_z$ the name of the z-th concept. Example-level formulas can be aggregated with the logic OR $\vee$ to provide a *class-level* formula,

$$\bigvee_{t \in S_i} \varphi_t^i, \tag{10}$$

being S_i the set of rows indices of the truth table for which $\hat{f}^i(\hat{q}) = 1$, i.e. it is the support of $\hat{f}^i$. We define with $\phi^i(\hat{c})$ the function that holds true whenever Eq. 10, evaluated on a given Boolean tuple $\hat{c}$, is true. Due to the aforementioned definition of support, we get the following class-level First-Order Logic (FOL) explanation for all the concept tuples,

$$\forall \hat{c} \in \hat{C}^i : \ \phi^i(\hat{c}) \leftrightarrow \hat{f}^i(\hat{c}). \tag{11}$$

We note that in case of non-concept-like input features, we may still derive the FOL formula through the "concept decoder" function g (see Sec. 2),

$$\forall x \in X : \ \phi^i\left(\xi(\overline{g(x)}, \mu^i)\right) \leftrightarrow \hat{f}^i\left(\xi(\overline{g(x)}, \mu^i)\right) \tag{12}$$

An example of the above scheme for both example and class-level explanations is depicted on top-right of Fig. 2.

Remarks. The aggregation of many example-level explanations may increase the length and the complexity of the FOL formula being extracted for a whole class. However, existing techniques as the Quine–McCluskey algorithm can be used to get compact and simplified equivalent FOL expressions [53, 54, 55]. For instance, the explanation (*person* $\wedge$ *nose*) $\vee$ ($\neg$*person* $\wedge$ *nose*) can be formally simplified in *nose*. Moreover, the Boolean interpretation of concept tuples may generate colliding representations for different samples. For instance, the Boolean representation of the

two samples $\{(0.1, 0.7), (0.2, 0.9)\}$ is the tuple $\bar{c} = (0, 1)$ for both of them. This means that their example-level explanations match as well. However, a concept can be eventually split into multiple finer grain concepts to avoid collisions. Finally, we mention that the number of samples for which any example-level formula holds (i.e. the support of the formula) is used as a measure of the explanation importance. In practice, example-level formulas are ranked by support and iteratively aggregated to extract class-level explanations, until the aggregation improves the support of the formula on a validation set.

4 Related work

In order to provide explanations for a given black-box model, most methods focus on identifying or scoring the most relevant input features [24, 25, 26, 27, 56, 28, 29]. Feature scores are usually computed sample by sample (i.e. providing *local explanations*) analyzing the activation patterns in the hidden layers of neural networks [24, 25, 26, 29] or by following a model-agnostic approach [56, 28]. To enhance human understanding of feature scoring methods, concept-based approaches have been effectively employed for identifying common activations patterns in the last nodes of neural networks corresponding to human categories [57, 58] or constraining the network to learn such concepts [38, 37]. Either way, feature-scoring methods are not able to explain *how* neural networks compose features to make predictions [30, 31, 32] and only a few of these approaches have been efficiently extended to provide explanations for a whole class (i.e. providing *global explanations*) [25, 56]. By contrast, a variety of rule-based approaches have been proposed to provide concept-based explanations. Logic rules are used to explain how black boxes predict class memberships for indivudal samples [39, 59], or for a whole class [60, 4, 7, 40]. Distilling explanations from an existing model, however, is not the only way to achieve explainability. Historically, standard machine-learning such as Logistic Regression [61], Generalized Additive Models [62, 63, 64] Decision Trees [9, 65, 66] and Decision Lists [67, 11, 68] were devised to be intrinsically interpretable. However, most of them struggle in solving complex classification problems. Logistic Regression, for instance, in its vanilla definition, can only recognize linear patterns, e.g. it cannot to solve the XOR problem [69]. Further, only Decision Trees and Decision Lists provide explanations in the from of logic rules. Considering decision trees, each path may be seen as a human comprehensible decision rule when the height of the tree is reasonably contained. Another family of concept-based XAI methods is represented by rule-mining algorithms which became popular at the end of the last century [70, 71]. Recent research has led to powerful rule-mining approaches as Bayesian Rule Lists (BRL) [11], where a set of rules is "pre-mined" using the frequent-pattern tree mining algorithm [72] and then the best rule set is identified with Bayesian statistics. In this paper, the proposed approach is compared with methods providing logic-based, global explanations. In particular, we selected one representative approach from different families of methods: Decision Trees[3] (white-box machine learning), BRL[4] (rule mining) and ψ Networks[5] (interpretable neural models).

5 Experiments

The quality of the explanations and the classification performance of the proposed approach are quantitatively assessed and compared to state-of-the-art white-box models. A visual sketch of each classification problem (described in detail in Sec. 5.1 together with all the experimental details) and a selection of the logic formulas found by the proposed approach is reported in Fig. 3. Six quantitative metrics are defined and used to compare the proposed approach with state-of-the-art methods. Sec. 5.2 summarizes the main findings. Further details concerning the experiments are reported in the supplemental material A. A python package and a freely available GitHub repository implementing the proposed approach will be made public upon paper acceptance. In appendix A.1 a snippet of the code extracted from the library is reported.

Quantitative metrics. Measuring the classification quality is of crucial importance for models that are going to be applied in real-world environments. On the other hand, assessing the quality of the explanations is required for their lawful deployment. In contrast with other kind of explanations, logic-based formulas can be evaluated quantitatively. Given a classification problem, first a set of

[3] `https://scikit-learn.org/stable/modules/tree.html`, BSD-3 Clause License.

[4] `https://github.com/tmadl/sklearn-expertsys`, MIT license.

[5] `https://github.com/pietrobarbiero/logic_explainer_networks`, Apache 2.0 License.

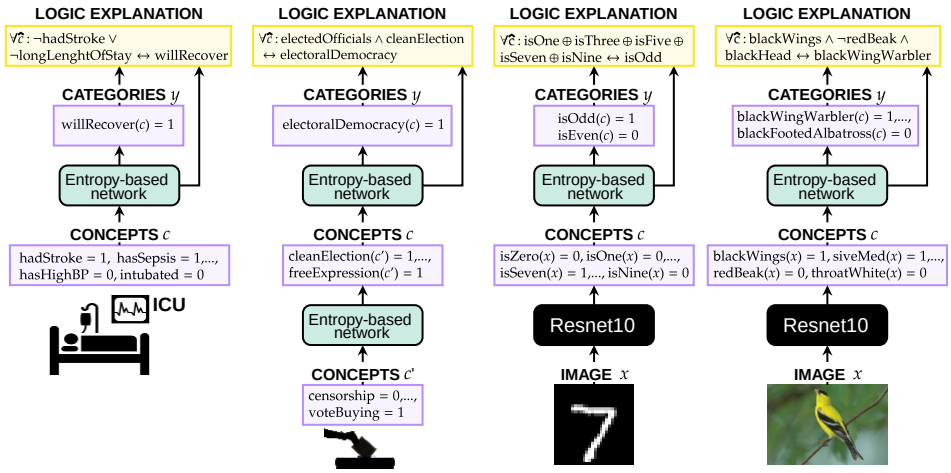

Figure 3: The four case studies show how the proposed Entropy-based networks (green) provide concise logic explanations (yellow—we dropped the arguments in the logic predicates, for simplicity) of their own decision process in different real-world contexts. When input features are non-interpretable, as pixel intensities (MNIST and CUB), a "concept decoder" (ResNet10) is employed to map images into concepts. Entropy-based networks then map concepts into target classes.

rules are extracted from the network for each target category and then each explanation is tested on an unseen set of test samples. The results for each metric are reported in terms of mean and standard error, computed over a 5-fold cross validation [73]. For each experiment and for each classifier six quantitative metrics are measured. (i) The MODEL ACCURACY, as usual, measures how well the classifier identifies the target classes on unseen data (see Table 1). (ii) The EXPLANATION ACCURACY measures how well the extracted logic formulas identifies the target classes (Fig. 4). This metric is obtained as the average of the F1 scores computed for each class explanation. (iii) The COMPLEXITY OF AN EXPLANATION is computed by standardizing the explanations in DNF and then by counting the number of terms of the standardized formula (Fig. 4). The longer the formula, the harder the interpretation for a human being. (iv) The FIDELITY OF AN EXPLANATION measures how well the extracted explanation matches the predictions obtained using the classifier (Table 2). (v) The RULE EXTRACTION TIME measures the time required to obtain an explanation from scratch (see Fig. 5). 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 classifier. (vi) The CONSISTENCY OF AN EXPLANATION measures the similarity of the extracted explanations over different runs (see Table 3). It is computed by counting how many times the same concepts appear in the logic formulas over different iterations.

5.1 Classification tasks and datasets

four classification problems ranging from computer vision to medicine are considered. Computer vision datasets (e.g. CUB) are annotated with high-level concepts (e.g. bird attributes) used to train concept bottleneck pipelines [37]. In the other datasets, the input data is rescaled into a categorical space ($\mathbb{R}^k \to C$) suitable for concept-based networks. All datasets can be downloaded from publicly available resources. Links to all dataset sources are reported in Appendix A.2.

Will we recover from ICU? (MIMIC-II). The Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II, [74, 75]) is a public-access intensive care unit (ICU) database consisting of 32,536 subjects (with 40,426 ICU admissions) admitted to different ICUs. The dataset contains detailed descriptions of a variety of clinical data classes: general, physiological, results of clinical laboratory tests, records of medications, fluid balance, and text reports of imaging studies (e.g. x-ray, CT, MRI, etc). In our experiments, we removed non-anonymous information, text-based features, time series inputs, and observations with missing data. We discretize continuous features into one-hot encoded categories. After such preprocessing step, we obtained an input space C composed of $k = 90$ key features. The task consists in identifying recovering or dying patients after ICU admission.

What kind of democracy are we living in? (V-Dem). Varieties of Democracy (V-Dem, [76, 77]) is a dataset containing a collection of indicators of latent regime characteristics over 202 countries

from 1789 to 2020. The database include $k_1 = 483$ low-level indicators (e.g. media bias, party ban, high-court independence, etc.), $k_2 = 82$ 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 a binary classification problem is considered to identify electoral democracies from non-electoral democracies. We indicate with C_1 and C_2 the spaces associated to the activations of the aforementioned two levels of concepts. Two classifiers f_1 and f_2 are trained to learn the map $C_1 \to C_2 \to Y$. Explanations are given for classifier f_2 in terms of concepts $c_2 \in C_2$.

What does parity mean? (MNIST Even/Odd). The Modified National Institute of Standards and Technology database (MNIST, [78]) contains a large collection of images representing handwritten digits. The input space $X \subset \mathbb{R}^{28\times 28}$ is composed of 28x28 pixel images while the concept space C with $k = 10$ is represented by the label indicator for digits from 0 to 9. The task we consider here is slightly different from the common digit-classification. Assuming $Y \subset \{0, 1\}^2$, we are interested in determining if a digit is either odd or even, and explaining the assignment to one of these classes in terms of the digit labels (concepts in C). The mapping $X \to C$ is provided by a ResNet10 classifier g [79] trained from scratch. while the classifier f is used to learn both the final mapping and the explanation as a function $C \to Y$.

What kind of bird is that? (CUB). The Caltech-UCSD Birds-200-2011 dataset (CUB, [80]) is a fine-grained classification dataset. It includes 11,788 images representing $r = 200$ ($Y = \{0, 1\}^{200}$) different bird species. 312 binary attributes describe visual characteristics (color, pattern, shape) of particular parts (beak, wings, tail, etc.) for each bird image. Attribute annotations, however, is quite noisy. For this reason, attributes are denoised by considering class-level annotations [37][6]. In the end, a total of 108 attributes (i.e. concepts with binary activations belonging to C) have been retained. The mapping $X \to C$ from images to attribute concepts is performed again with a ResNet10 model g trained from scratch while the classifier f learns the final function $C \to Y$.

5.2 Results and discussion

Experiments show how entropy-based networks outperform state-of-the-art white box models such as BRL and decision trees[7] and interpretable neural models such as ψ networks on challenging classification tasks (Table 1). At the same time, the logic explanations provided by entropy-based networks are better than ψ networks and almost as accurate as the rules found by decision trees and BRL, while being far more concise, as demonstrated in Fig. 4. Furthermore, the time required to train entropy-based networks is only slightly higher with respect to Decision Trees but is lower that ψ Networks and BRL by one to three orders of magnitude (Fig. 5), making it feasible for explaining also complex tasks. The fidelity (Table 2)[8] of the formulas extracted by the entropy-based network is always higher than 90% with the only exception of MIMIC. This means that almost any prediction made using the logic explanation matches the corresponding prediction made by the model, making the proposed approach very close to a white box model. The combination of these results empirically shows that our method represents a viable solution for the lawful deployment of *explainable* cutting-edge models.

Table 1: Classification accuracy (%) of the compared models.

	Entropy net	Tree	BRL	ψ net
MIMIC-II	$\mathbf{79.05 \pm 1.35}$	77.53 ± 1.45	76.40 ± 1.22	77.19 ± 1.64
V-Dem	$\mathbf{94.51 \pm 0.48}$	85.61 ± 0.57	91.23 ± 0.75	89.77 ± 2.07
MNIST	$\mathbf{99.81 \pm 0.02}$	99.75 ± 0.01	99.80 ± 0.02	99.79 ± 0.03
CUB	$\mathbf{92.95 \pm 0.20}$	81.62 ± 1.17	90.79 ± 0.34	91.92 ± 0.27

The reason why the proposed approach consistently outperform ψ networks across all the key metrics (i.e. classification accuracy, explanation accuracy, and fidelity) can be explained observing how entropy-based networks are far less constrained than ψ networks, both in the architecture

[6] 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 after this refinement.

[7] The height of the tree is limited to obtain rules of comparable lengths. See supplementary materials A.2.

[8] We did not compute the fidelity of decision trees and BRL as they are trivially rule-based models.

(our approach does not apply weight pruning) and in the loss function (our approach applies a regularization on the distributions α^i and not on all weight matrices). Likewise, the main reason why the proposed approach provides a higher classification accuracy with respect to BRL and decision trees may lie in the smoothness of the decision functions of neural networks which tend to generalize better than rule-based methods, as already observed by Tavares et al. [81]. For each dataset, we report in the supplemental material (Appendix A.3) a few examples of logic explanations extracted by each method, as well as in Fig. 3. We mention that the proposed approach is the only matching the logically correct ground-truth explanation for the MNIST even/odd experiment, i.e. $\forall x, \mathrm{isOdd(x)} \leftrightarrow \mathrm{isOne(x)} \oplus \mathrm{isThree(x)} \oplus \mathrm{isFive(x)} \oplus \mathrm{isSeven(x)} \oplus \mathrm{isNine(x)}$ and $\forall x, \mathrm{isEven(x)} \leftrightarrow \mathrm{isZero(x)} \oplus \mathrm{isTwo(x)} \oplus \mathrm{isfour(x)} \oplus \mathrm{isSix(x)} \oplus \mathrm{isEight(x)}$, being $\oplus$ the exclusive OR. In terms of formula consistency, we observe how BRL is the most consistent rule extractor, closely followed by the proposed approach (Table 3).

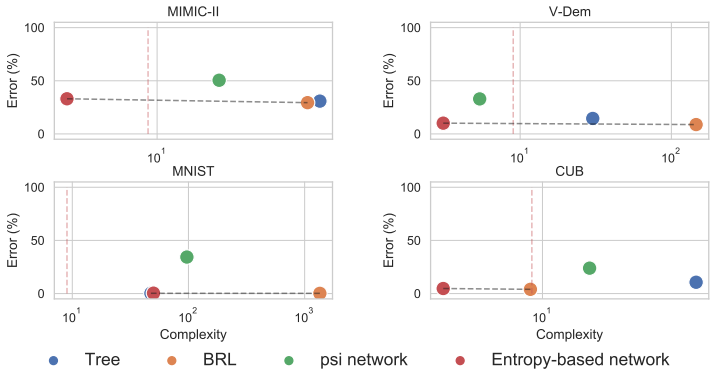

Figure 4: Non-dominated solutions [82] (dotted black line) in terms of average classification test error and their average complexity of the explanations. The vertical dotted red line marks the maximum explanation complexity laypeople can handle (i.e. complexity ≈ 9, see [46, 47, 51]). When humans compare a set of hypotheses outlining the same outcomes, they tend to have an implicit bias towards the simplest ones, making explanations from entropy-based networks the best choice.

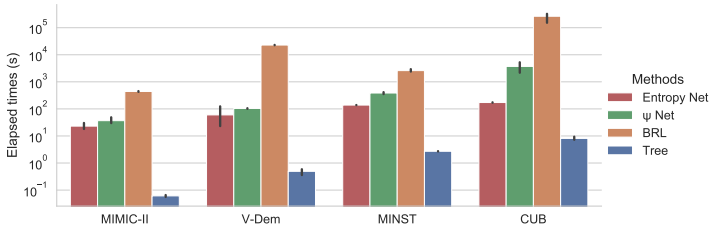

Figure 5: Time required to train models and to extract the explanations. Our model compares favorably with the competitors, with the exception of Decision Trees. BRL is by one to three order of magnitude slower than our approach. Error bars show the 95% confidence interval of the mean.

Table 2: Out-of-distribution fidelity (%)

	Entropy net	ψ net
MIMIC-II	$\mathbf{79.11 \pm 2.02}$	51.63 ± 6.67
V-Dem	$\mathbf{90.90 \pm 1.23}$	69.67 ± 10.43
MNIST	$\mathbf{99.63 \pm 0.00}$	65.68 ± 5.05
CUB	$\mathbf{99.86 \pm 0.01}$	77.34 ± 0.52

Table 3: Consistency (%)

Entropy net	Tree	BRL	ψ net
28.75	**40.49**	30.48	27.62
46.25	72.00	**73.33**	38.00
100.00	41.67	**100.00**	96.00
35.52	21.47	**42.86**	41.43

6 Conclusions

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 by making them explainable-by-design, thanks to an entropy-based approach that yields FOL-based explanations. However, the

extraction of a FOL explanation requires symbolic input and output spaces. In some contexts, such as computer vision, the use of concept-based approaches may require additional annotations and attribute labels to get a consistent symbolic layer of concepts. However, recent works on automatic concept extraction may alleviate the related costs, leading to more cost-effective concept annotations [34, 58]. As the proposed approach provides logic explanations for how a model arrives at a decision, it can be effectively used to reverse engineer algorithms, processes, to find vulnerabilities, or to improve system design powered by end-to-end differentiable black boxes. From a scientific perspective, formal knowledge distillation from state-of-the-art networks may enable scientific discoveries or falsification of existing theories.

References

[1] EUGDPR. Gdpr. general data protection regulation., 2017.

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

[3] Public Authorities Law. Code of federal regulations. *Wash.: Gov. print. off*, 10.

[4] Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen. Deepred – rule extraction from deep neural networks. In Toon Calders, Michelangelo Ceci, and Donato Malerba, editors, *Discovery Science*, pages 457–473, Cham, 2016. Springer International Publishing.

[5] Gabriëlle Ras, Marcel van Gerven, and Pim Haselager. Explanation methods in deep learning: Users, values, concerns and challenges. In *Explainable and Interpretable Models in Computer Vision and Machine Learning*, pages 19–36. Springer, 2018.

[6] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32:9240, 2019.

[7] Gabriele Ciravegna, Francesco Giannini, Marco Gori, Marco Maggini, and Stefano 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}*, pages 2234–2240. International Joint Conferences on Artificial Intelligence Organization, 2020.

[8] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020.

[9] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[10] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[11] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Annals of Applied Statistics*, 9(3):1350–1371, 2015.

[12] Miles D Cranmer, Rui Xu, Peter Battaglia, and Shirley Ho. Learning symbolic physics with graph networks. *arXiv preprint arXiv:1909.05862*, 2019.

[13] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[14] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[15] Finale Doshi-Velez and Been Kim. Considerations for evaluation and generalization in interpretable machine learning. In *Explainable and interpretable models in computer vision and machine learning*, pages 3–17. Springer, 2018.

[16] Muhammad Aurangzeb Ahmad, Carly Eckert, and Ankur Teredesai. Interpretable machine learning in healthcare. In *Proceedings of the 2018 ACM international conference on bioinformatics, computational biology, and health informatics*, pages 559–560, 2018.

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

[18] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Toward interpretable machine learning: Transparent deep neural networks and beyond. *arXiv preprint arXiv:2003.07631*, 2020.

[19] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *arXiv preprint arXiv:2103.11251*, 2021.

[20] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[21] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[22] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[23] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

[24] Dumitru Erhan, Aaron Courville, and Yoshua Bengio. Understanding representations learned in deep architectures. *Department dInformatique et Recherche Operationnelle, University of Montreal, QC, Canada, Tech. Rep*, 1355(1), 2010.

[25] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[26] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[27] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

[28] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[29] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[30] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[31] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, et al. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). In *International conference on machine learning*, pages 2668–2677. PMLR, 2018.

[32] David Alvarez-Melis and Tommi S Jaakkola. Towards robust interpretability with self-explaining neural networks. *arXiv preprint arXiv:1806.07538*, 2018.

[33] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[34] Amirata Ghorbani, James Wexler, James Zou, and Been Kim. Towards automatic concept-based explanations. *arXiv preprint arXiv:1902.03129*, 2019.

[35] Yash Goyal, Amir Feder, Uri Shalit, and Been Kim. Explaining classifiers with causal concept effect (cace). *arXiv preprint arXiv:1907.07165*, 2019.

[36] Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. *Advances in Neural Information Processing Systems*, 33, 2020.

[37] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International Conference on Machine Learning*, pages 5338–5348. PMLR, 2020.

[38] Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.

[39] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018.

[40] Gabriele Ciravegna, Francesco Giannini, Stefano Melacci, Marco Maggini, and Marco Gori. A constraint-based approach to learning and explanation. In *AAAI*, pages 3658–3665, 2020.

[41] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[42] Aristotle. Posterior analytics, 350 B.C.

[43] Roald Hoffmann, Vladimir I Minkin, and Barry K Carpenter. Ockham's razor and chemistry. *Bulletin de la Société chimique de France*, 2(133):117–130, 1996.

[44] Andrei N Soklakov. Occam's razor as a formal basis for a physical theory. *Foundations of Physics Letters*, 15(2):107–135, 2002.

[45] Samuel Rathmanner and Marcus Hutter. A philosophical treatise of universal induction. *Entropy*, 13(6):1076–1136, 2011.

[46] George Armitage Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63:81–97, 1956.

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

[48] Herbert A Simon. Rational choice and the structure of the environment. *Psychological review*, 63(2):129, 1956.

[49] Herbert A Simon. *Models of man; social and rational.* New York: John Wiley and Sons, Inc., 1957.

[50] Herbert A Simon. Rational decision making in business organizations. *The American economic review*, 69(4):493–513, 1979.

[51] Wei Ji Ma, Masud Husain, and Paul M Bays. Changing concepts of working memory. *Nature neuroscience*, 17(3):347, 2014.

[52] Elliott Mendelson. *Introduction to mathematical logic*. CRC press, 2009.

[53] Hugh McColl. The calculus of equivalent statements (third paper). *Proceedings of the London Mathematical Society*, 1(1):16–28, 1878.

[54] Willard V Quine. The problem of simplifying truth functions. *The American mathematical monthly*, 59(8):521–531, 1952.

[55] Edward J McCluskey. Minimization of boolean functions. *The Bell System Technical Journal*, 35(6):1417–1444, 1956.

[56] Marco Tulio Ribeiro, Sameer Singh, and Carlos 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.

[57] Been Kim, Justin Gilmer, Martin Wattenberg, and Fernanda Viégas. Tcav: Relative concept importance testing with linear concept activation vectors, 2018.

[58] Dmitry Kazhdan, Botty Dimanov, Mateja Jamnik, Pietro Liò, and Adrian Weller. Now you see me (cme): Concept-based model extraction. *arXiv preprint arXiv:2010.13233*, 2020.

[59] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[60] Makoto Sato and Hiroshi Tsukimoto. Rule extraction from neural networks via decision tree induction. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 3, pages 1870–1875. IEEE, 2001.

[61] Richard D McKelvey and William Zavoina. A statistical model for the analysis of ordinal level dependent variables. *Journal of Mathematical Sociology*, 4(1):103–120, 1975.

[62] Trevor Hastie and Robert Tibshirani. Generalized additive models: some applications. *Journal of the American Statistical Association*, 82(398):371–386, 1987.

[63] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158, 2012.

[64] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie 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*, pages 1721–1730, 2015.

[65] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[66] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[67] Ronald L Rivest. Learning decision lists. *Machine learning*, 2(3):229–246, 1987.

[68] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. Learning certifiably optimal rule lists for categorical data, 2018.

[69] Marvin Minsky and Seymour A Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 2017.

[70] Robert C Holte. Very simple classification rules perform well on most commonly used datasets. *Machine learning*, 11(1):63–90, 1993.

[71] William W Cohen. Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier, 1995.

[72] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. *ACM sigmod record*, 29(2):1–12, 2000.

[73] Martin Krzywinski and Naomi Altman. Error bars: the meaning of error bars is often misinterpreted, as is the statistical significance of their overlap. *Nature methods*, 10(10):921–923, 2013.

[74] Mohammed Saeed, Mauricio Villarroel, Andrew T Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H Kyaw, Benjamin Moody, and Roger 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.

[75] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[76] Daniel Pemstein, Kyle L Marquardt, Eitan Tzelgov, Yi-ting Wang, Joshua Krusell, and Farhad Miri. The v-dem measurement model: latent variable analysis for cross-national and cross-temporal expert-coded data. *V-Dem Working Paper*, 21, 2018.

[77] Michael Coppedge, John Gerring, Carl Henrik Knutsen, Staffan I Lindberg, Jan Teorell, David Altman, Michael Bernhard, Agnes Cornell, M Steven Fish, Lisa Gastaldi, et al. V-dem codebook v11, 2021.

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

[79] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[80] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[81] Anderson R Tavares, Pedro Avelar, João M Flach, Marcio Nicolau, Luis C Lamb, and Moshe Vardi. Understanding boolean function learnability on deep neural networks. *arXiv preprint arXiv:2009.05908*, 2020.

[82] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[83] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[84] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[85] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Section 3.1 for contributions.
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] All authors have read the ethics guidelines. References have been anonymized when appropriate.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] In bar plots we reported error bars showing the 95% confidence interval of the mean. In tables we reported the standard error of the mean.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Section 5.1.
   (b) Did you mention the license of the assets? [Yes] See footnotes in Section 4.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We are only using already public data, providing full references to their source. In the case of MIMIC-II dataset we submitted an application to the Physionet project to download data `https://archive.physionet.org/physiobank/database/mimic2cdb/`.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Section 5.1. In the case of MIMIC-II dataset we removed all non-anonymous information.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

A.1 Software

In order to make the proposed approach accessible to the whole community, we released of a Python package [9] with an extensive documentation on methods and unit tests. 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.

The code library is designed with intuitive APIs requiring only a few lines of code to train and get explanations from the neural network as shown in the following code snippet 1.

```
import torch_explain as te
from torch_explain.logic import test_explanation
from torch_explain.logic.nn import explain_class

# XOR problem with additional features
x0 = torch.zeros((4, 100))
x = torch.tensor([
    [0, 0, 0],
    [0, 1, 0],
    [1, 0, 0],
    [1, 1, 0],
], dtype=torch.float)
x = torch.cat([x, x0], dim=1)
y = torch.tensor([0, 1, 1, 0], dtype=torch.long)

# network architecture
layers = [
    te.nn.EntropyLogicLayer(x.shape[1], 10, n_classes=2),
    torch.nn.LeakyReLU(),
    te.nn.LinearIndependent(10, 10, n_classes=2),
    torch.nn.LeakyReLU(),
    te.nn.LinearIndependent(10, 1, n_classes=2, top=True)
]
model = torch.nn.Sequential(*layers)

# train loop
optimizer = torch.optim.AdamW(model.parameters(), lr=0.01)
loss_form = torch.nn.CrossEntropyLoss()
model.train()
for epoch in range(1001):
    optimizer.zero_grad()
    y_pred = model(x)
    loss = loss_form(y_pred, y) + \
           0.00001 * te.nn.functional.entropy_logic_loss(model)
    loss.backward()
    optimizer.step()

# logic explanations
y1h = one_hot(y)
_, class_explanations, _ = explain_class(model, x, y1h, x, y1h)
```

Listing 1: Example on how to use the APIs to implement the proposed approach.

A.2 Experimental details

Batch gradient-descent and the Adam optimizer with decoupled weight decay [83] and learning rate set to 10^{-2} are used for the optimization of all neural models' parameters (Entropy-based Network and ψ Network). An early stopping strategy is also applied: the model with the highest accuracy on the validation set is saved and restored before evaluating the test set.

[9] https://pypi.org/project/torch_explain/

With regard to the Entropy-based Network, Tab. 4 reports the hyperparameters employed to train the network in all experiments. A grid search cross-validation strategy has been employed on the validation set to select hyperparameter values. The objective was to maximize at the same time both model and explanation accuracy. λ represents the trade-off parameter in Eq. 8 while τ is the temperature of Eq. 2.

Table 4: Hyper parameters of entropy-based networks.

	λ	τ	max epochs	hidden neurons
MIMIC-II	10^{-3}	0.7	200	20
V-Dem	10^{-5}	5	200	20, 20
MNIST	10^{-7}	5	200	10
CUB	10^{-4}	0.7	500	10

Concerning the ψ network in all experiments one network per class has been trained. They are composed of two hidden layer of 10 and 5 hidden neurons respectively. As indicated in the original paper, an l_1 weight regularization has been applied to all layers of the network. As in this work, the contribute in the overall loss of the l_1 regularization is weighted by an hyperparameter $\lambda = 10^-4$. The maximum number of non-zero input weight (fan-in) is set to 3 in in MIMIC and V-Dem while for MNIST and CUB200 it is set to 4. In Ciravegna et al. [7], ψ networks were devised to provide explanations of existing models; in this paper, however, we have shown how they can directly solve classification problems.

Decision Trees have been limited in their maximum height in all experiments to maintain the complexity of the rules at a comparable level w.r.t the other methods. More precisely the maximum height has been set to 5 in all binary classification tasks (MIMIC-II, V-Dem, MNIST) while we allowed a maximum height of 30 in the CUB experiment due to the high number of classes to predict (200).

BRL algorithms requires to first run the FP-growth algorithm [72] (an enhanced version of Apriori) to mine a first set of frequent rules. The hyperparameter used by FP-growth are: the minimum support in percentage of training samples for each rule (set to 10%), the minimum and the maximum number of features considered by each rule (respectively set to 1 and 2). Regarding the Bayesian selection of the best rules, the number of Markov chain Monte Carlo used for inference is set to 3, while 50000 iterations maximum are allowed. At last the expected length and width of the extracted rule list is set respectively to 3 and 1. These are the default values indicated in the BRL repository. Due to the computational complexity and the high number of hyperparameters, they have not been cross validated.

The code for the experiments is implemented in Python 3, relying upon open-source libraries [84, 85]. All the experiments have been run on the same machine: Intel® Core™ i7-10750H 6-Core Processor at 2.60 GHz equipped with 16 GiB RAM and NVIDIA GeForce RTX 2060 GPU.

All datasets employed are freely available (only MIMIC-II requires an online registration) and can be downloaded from the following links:
MIMIC: `https://archive.physionet.org/mimic2`.
V-Dem: `https://www.v-dem.net/en/data/data/v-dem-dataset-v111`.
MNIST: `http://yann.lecun.com/exdb/mnist`.
CUB: `http://www.vision.caltech.edu/visipedia/CUB-200-2011.html`.

A.3 Extra Results

In the following we report again in tabular form the results concerning the explanation accuracy and the complexity of the rules (Fig. 4) and the extraction time (Fig. 5). In Table 8, instead, a selection of the rule extracted by each method in all experiments is shown. For all methods we report only the explanations of the first class for the first split of the Cross-validation. At last, for the Entropy-based method only, Tables 9, 10, 11, 12 resume the explanations of all classes in all experiments.

Table 5: Explanation's accuracy (%) computed as the average of the F1 scores computed for each class.

	Entropy net	Tree	BRL	ψ net
MIMIC-II	66.93 ± 2.14	69.15 ± 2.24	$\mathbf{70.59 \pm 2.17}$	49.51 ± 3.91
V-Dem	89.88 ± 0.50	85.45 ± 0.58	$\mathbf{91.21 \pm 0.75}$	67.08 ± 9.68
MNIST	99.62 ± 0.00	99.74 ± 0.01	$\mathbf{99.79 \pm 0.02}$	65.64 ± 5.05
CUB	95.24 ± 0.05	89.36 ± 0.92	$\mathbf{96.02 \pm 0.17}$	76.10 ± 0.56

Table 6: Complexity computed as the number of terms in each minterm of the DNF rules.

	Entropy net	Tree	BRL	ψ net
MIMIC-II	$\mathbf{3.50 \pm 0.88}$	66.60 ± 1.45	57.70 ± 35.58	20.6 ± 5.36
V-Dem	$\mathbf{3.10 \pm 0.51}$	30.20 ± 1.20	145.70 ± 57.93	5.40 ± 2.70
MNIST	50.00 ± 0.00	47.50 ± 0.72	1352.30 ± 292.62	96.90 ± 10.01
CUB	$\mathbf{3.74 \pm 0.03}$	45.92 ± 1.16	8.87 ± 0.11	15.96 ± 0.96

Table 7: Rule extraction time (s) calculated as the time required to train the models and to extract the corresponding rules.

	Entropy net	Tree	BRL	ψ net
MIMIC-II	23.08 ± 3.53	$\mathbf{0.06 \pm 0.00}$	440.24 ± 9.75	36.68 ± 6.10
V-Dem	59.90 ± 31.18	$\mathbf{0.49 \pm 0.07}$	22843.21 ± 194.49	103.78 ± 1.65
MNIST	138.32 ± 0.63	$\mathbf{2.72 \pm 0.02}$	2594.79 ± 177.34	385.57 ± 17.81
CUB	171.87 ± 1.95	$\mathbf{8.10 \pm 0.65}$	264678.29 ± 56521.40	3707.29 ± 1006.54

Table 8: Comparison of the formulas obtained in the first run of each experiment for all methods. Only the formula explaining the first class has been reported. Ellipses are used to truncate overly long formulas.

Dataset	Method	Formulas
MIMIC-II	Entropy	non_recover $\leftrightarrow \neg$liver_flg $\wedge \neg$stroke_flg $\wedge \neg$mal_flg
	DTree	non_recover $\leftrightarrow$ (age_high < 0.5 $\wedge$ mal_flg < 0.5 $\wedge$ stroke_flg < 0.5 $\wedge$ age_normal < 0.5 $\wedge$ iv_day_1_normal < 0.5) $\vee$ (age_high < 0.5 $\wedge$ mal_flg < 0.5 $\wedge$ stroke_flg < 0.5 $\wedge$ age_normal < 0.5 $\wedge$ iv_day_1_normal > 0.5) $\vee$ (age_high < 0.5 $\wedge$ ...
	BRL	non_recover $\leftrightarrow$ (age_low $\wedge$ sofa_first_low $\wedge \neg$(mal_flg $\wedge \neg$weight_first_normal)) $\vee$ (age_high $\wedge \neg$service_num_normal $\wedge \neg$(age_low $\wedge$ sofa_first_low) $\wedge \neg$(chf_flg $\wedge \neg$day_icu_intime_num_high) $\wedge \neg$(mal_flg $\wedge \neg$weight_first_normal) $\wedge \neg$(stroke_flg $\wedge$ ...
	ψ Net	non_recover $\leftrightarrow$ (iv_day_1_normal $\wedge \neg$age_high $\wedge \neg$hour_icu_intime_normal $\wedge \neg$sofa_first_normal) $\vee$ (mal_flg $\wedge \neg$age_high $\wedge \neg$hour_icu_intime_normal $\wedge \neg$sofa_first_normal) $\vee$ ...
V-Dem	Entropy	non_electoral_democracy $\leftrightarrow \neg$v2xel_frefair $\vee \neg$v2x_elecoff $\vee \neg$v2x_cspart $\vee \neg$v2xeg_eqaccess $\vee \neg$v2xeg_eqdr
	DTree	non_electoral_democracy $\leftrightarrow$ (v2xel_frefair < 0.5 $\wedge$ v2xdl_delib < 0.5 $\wedge$ v2x_frassoc_thick < 0.5) $\vee$ (v2xel_frefair < 0.5 $\wedge$ v2xdl_delib < 0.5 $\wedge$ v2x_frassoc_thick > 0.5 $\wedge$ v2x_freexp_altinf < 0.5 $\wedge$ v2xeg_eqprotec < 0.5) $\vee$ ...
	BRL	non_electoral_democracy $\leftrightarrow \neg$v2x_cspart $\vee \neg$v2x_elecoff $\vee \neg$v2x_frassoc_thick $\vee \neg$v2x_freexp_altinf $\vee \neg$v2xcl_rol $\vee$ ($\neg$v2x_mpi $\wedge \neg$v2xel_frefair)
	ψ Net	non_electoral_democracy $\leftrightarrow \neg$v2xeg_eqaccess $\vee$ (v2x_egal $\wedge \neg$v2x_frassoc_thick) $\vee$ (v2xeg_eqdr $\wedge \neg$v2x_egal) $\vee$ (v2xel_frefair $\wedge \neg$v2x_frassoc_thick) $\vee$ ($\neg$v2x_cspart $\wedge \neg$v2x_suffr) $\vee$ ($\neg$v2x_frassoc_thick $\wedge \neg$v2x_suffr) $\vee$ ...
MNIST	Entropy	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 $\wedge$ ...
	DTree	even $\leftrightarrow$ (one < 0.54 $\wedge$ nine < $1.97{\cdot}10^{-}5$ $\wedge$ three < 0.00 $\wedge$ five < 0.09 $\wedge$ seven < 0.20) $\vee$ (one < 0.54 $\wedge$ nine < $1.97{\cdot}10^{-}5$ $\wedge$ three < 0.00 $\wedge$ five > 0.09 $\wedge$ two > 0.97) $\vee$ (one < 0.54 $\wedge$ nine < $1.97{\cdot}10^{-}5$ $\wedge$ three > 0.00 $\wedge$ two < 0.99 $\wedge$ eight > 1.00) $\vee$ ...
	BRL	even $\leftrightarrow$ (two $\wedge \neg$one $\wedge \neg$seven $\wedge \neg$three $\wedge \neg$(seven $\wedge \neg$two)) $\vee$ (four $\wedge \neg$five $\wedge \neg$nine $\wedge \neg$seven $\wedge \neg$three $\wedge \neg$(seven $\wedge \neg$two) $\wedge \neg$(two $\wedge \neg$one)) $\vee$ (four $\wedge \neg$five $\wedge \neg$seven $\wedge \neg$three $\wedge \neg$(four $\wedge \neg$nine) $\wedge \neg$(seven $\wedge$ ...
	ψ Net	even $\leftrightarrow$ (four $\wedge$ nine $\wedge$ six $\wedge$ three $\wedge$ zero $\wedge \neg$eight $\wedge \neg$one $\wedge \neg$seven) $\vee$ (four $\wedge$ nine $\wedge$ six $\wedge$ two $\wedge$ zero $\wedge \neg$eight $\wedge \neg$one $\wedge \neg$seven) $\vee$ (eight $\wedge$ six $\wedge \neg$four $\wedge \neg$nine $\wedge \neg$seven $\wedge \neg$three $\wedge \neg$two) $\vee$ (eight $\wedge$ six $\wedge$...
CUB	Entropy	black_footed_albatross $\leftrightarrow$ has_bill_length_about_the_same_as_head $\wedge$ has_wing_pattern_solid $\wedge \neg$has_upper_tail_color_grey $\wedge \neg$has_belly_color_white $\wedge \neg$has_wing_shape_roundedwings $\wedge \neg$has_bill_color_black
	DTree	black_footed_albatross $\leftrightarrow$ (has_back_pattern_striped < 0.46 $\wedge$ has_back_color_buff < 0.69 $\wedge$ has_upper_tail_color_white < 0.59 $\wedge$ has_under_tail_color_buff < 0.82 $\wedge$ has_shape_perchinglike < 0.66 $\wedge$ ...
	BRL	black_footed_albatross $\leftrightarrow$ (has_back_pattern_striped $\wedge$ has_belly_color_black $\wedge$ has_bill_shape_hooked_seabird $\wedge \neg$has_belly_color_white) $\vee$ (has_back_pattern_striped $\wedge$ ...
	ψ Net	black_footed_albatross $\leftrightarrow$ (has_bill_shape_hooked_seabird $\wedge \neg$has_breast_color_white $\wedge \neg$has_size_small_5__9_in $\wedge \neg$has_wing_color_grey) $\vee$ (has_bill_shape_hooked_seabird $\wedge$ ...

Table 9: Formulas extracted from the MIMIC-II dataset.

Formulas
non_recover $\leftrightarrow \neg$liver_flg $\wedge \neg$stroke_flg $\wedge \neg$mal_flg
recover $\leftrightarrow$ mal_flg $\vee$ (age_HIGH $\wedge \neg$iv_day_1_NORMAL)

Table 10: Formulas extracted from the V-Dem dataset.

Formulas
non_electoral_democracy $\leftrightarrow \neg$v2xel_frefair $\vee \neg$v2x_elecoff $\vee \neg$v2x_cspart $\vee \neg$v2xeg_eqaccess $\vee \neg$v2xeg_eqdr
electoral_democracy $\leftrightarrow$ v2xel_frefair $\wedge$ v2x_elecoff $\wedge$ v2x_cspart

Table 11: Formulas extracted from the MNIST dataset.

Formulas
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)
odd $\leftrightarrow$ (one $\wedge \neg$zero $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (three $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (five $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (seven $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$eight $\wedge \neg$nine) $\vee$ (nine $\wedge \neg$zero $\wedge \neg$one $\wedge \neg$two $\wedge \neg$three $\wedge \neg$four $\wedge \neg$five $\wedge \neg$six $\wedge \neg$seven $\wedge \neg$eight)

Table 12: Formulas extracted from the CUB dataset.

Formulas
Black_footed_Albatross ↔ has_bill_length_about_the_same_as_head ∧ has_wing_pattern_solid ∧ ¬has_upper_tail_color_grey ∧ ¬has_belly_color_white ∧ ¬has_wing_shape_roundedwings ∧ ¬has_bill_color_black
Laysan_Albatross ↔ has_crown_color_white ∧ has_wing_pattern_solid ∧ ¬has_under_tail_color_white
Sooty_Albatross ↔ has_upper_tail_color_grey ∧ has_size_medium_9__16_in ∧ has_bill_color_black ∧ ¬has_belly_color_white
Groove_billed_Ani ↔ has_breast_color_black ∧ has_leg_color_black ∧ ¬has_bill_shape_allpurpose ∧ ¬has_bill_length_about_the_same_as_head ∧ ¬has_wing_shape_roundedwings
Crested_Auklet ↔ has_nape_color_black ∧ ¬has_eye_color_black ∧ ¬has_belly_color_white
Least_Auklet ↔ has_breast_color_black ∧ has_breast_color_white ∧ ¬has_nape_color_white ∧ ¬has_size_small_5__9_in
Parakeet_Auklet ↔ has_size_medium_9__16_in ∧ has_primary_color_white ∧ has_leg_color_grey
Rhinoceros_Auklet ↔ has_size_medium_9__16_in ∧ has_leg_color_buff
Brewer_Blackbird ↔ has_breast_color_black ∧ has_wing_shape_roundedwings ∧ ¬has_bill_length_about_the_same_as_head ∧ ¬has_shape_perchinglike
Red_winged_Blackbird ↔ has_belly_color_black ∧ has_wing_pattern_multicolored ∧ ¬has_wing_color_white
Rusty_Blackbird ↔ has_back_color_brown ∧ has_belly_color_black ∧ ¬has_crown_color_brown
Yellow_headed_Blackbird ↔ has_forehead_color_yellow ∧ has_primary_color_black
Bobolink ↔ has_belly_color_black ∧ ¬has_upper_tail_color_grey ∧ ¬has_upper_tail_color_black
Indigo_Bunting ↔ has_forehead_color_blue ∧ has_back_pattern_solid ∧ has_wing_pattern_multicolored
Lazuli_Bunting ↔ has_leg_color_black ∧ has_bill_color_grey ∧ ¬has_under_tail_color_white
Painted_Bunting ↔ has_nape_color_blue ∧ has_leg_color_grey ∧ has_bill_color_grey
Cardinal ↔ has_forehead_color_red ∧ has_wing_shape_roundedwings ∧ has_wing_pattern_multicolored ∧ ¬has_nape_color_black
Spotted_Catbird ↔ has_leg_color_grey ∧ ¬has_breast_pattern_solid ∧ ¬has_breast_color_black ∧ ¬has_belly_color_white ∧ ¬has_crown_color_black
Gray_Catbird ↔ has_under_tail_color_grey ∧ has_belly_color_grey ∧ has_crown_color_black ∧ ¬has_primary_color_black
Yellow_breasted_Chat ↔ has_primary_color_yellow ∧ has_bill_color_black ∧ ¬has_back_color_grey ∧ ¬has_throat_color_grey ∧ ¬has_throat_color_black ∧ ¬has_nape_color_yellow ∧ ¬has_belly_color_white
Eastern_Towhee ↔ has_breast_color_black ∧ has_nape_color_black ∧ ¬has_belly_color_black ∧ ¬has_tail_pattern_multicolored ∧ ¬has_primary_color_white
Chuck_will_Widow ↔ has_under_tail_color_brown ∧ has_belly_color_buff ∧ has_crown_color_brown ∧ ¬has_bill_shape_allpurpose
Brandt_Cormorant ↔ has_bill_shape_hooked_seabird ∧ has_breast_color_black ∧ ¬has_wing_shape_roundedwings
Red_faced_Cormorant ↔ has_belly_color_black ∧ ¬has_size_small_5__9_in ∧ ¬has_bill_color_black

Table 12 continued from previous page

Formulas
Pelagic_Cormorant ↔ has_size_medium_9__16_in ∧ has_leg_color_black ∧ ¬has_bill_shape_hooked_seabird ∧ ¬has_tail_shape_notched_tail ∧ ¬has_belly_color_white ∧ ¬has_wing_shape_roundedwings
Bronzed_Cowbird ↔ has_belly_color_black ∧ has_shape_perchinglike ∧ has_wing_pattern_solid ∧ ¬has_bill_shape_allpurpose ∧ ¬has_underparts_color_yellow ∧ ¬has_bill_length_about_the_same_as_head
Shiny_Cowbird ↔ has_belly_color_black ∧ has_shape_perchinglike ∧ has_wing_pattern_solid ∧ ¬has_wing_shape_roundedwings
Brown_Creeper ↔ has_nape_color_buff ∧ ¬has_shape_perchinglike
American_Crow ↔ has_belly_color_black ∧ has_shape_perchinglike ∧ ¬has_breast_color_buff ∧ ¬has_bill_length_shorter_than_head
Fish_Crow ↔ has_bill_shape_allpurpose ∧ has_bill_length_about_the_same_as_head ∧ ¬has_under_tail_color_grey ∧ ¬has_belly_color_white ∧ ¬has_shape_perchinglike
Black_billed_Cuckoo ↔ has_leg_color_grey ∧ has_crown_color_brown
Mangrove_Cuckoo ↔ has_belly_color_buff ∧ has_leg_color_grey ∧ ¬has_back_color_black
Yellow_billed_Cuckoo ↔ has_shape_perchinglike ∧ has_tail_pattern_solid ∧ has_primary_color_white ∧ ¬has_bill_color_black
Gray_crowned_Rosy_Finch ↔ has_under_tail_color_black ∧ has_crown_color_grey ∧ has_wing_pattern_striped
Purple_Finch ↔ has_forehead_color_red ∧ ¬has_wing_shape_roundedwings ∧ ¬has_belly_pattern_solid ∧ ¬has_bill_color_black
Northern_Flicker ↔ has_belly_color_black ∧ has_leg_color_grey ∧ ¬has_nape_color_black
Acadian_Flycatcher ↔ has_breast_color_white ∧ has_leg_color_black ∧ ¬has_under_tail_color_white ∧ ¬has_bill_color_black
Great_Crested_Flycatcher ↔ has_tail_pattern_solid ∧ has_primary_color_grey ∧ has_wing_pattern_striped
Least_Flycatcher ↔ has_tail_shape_notched_tail ∧ has_tail_pattern_solid ∧ ¬has_bill_shape_cone ∧ ¬has_underparts_color_black ∧ ¬has_back_color_brown ∧ ¬has_breast_color_yellow ∧ ¬has_throat_color_black ∧ ¬has_bill_length_about_the_same_as_head ∧ ¬has_primary_color_buff ∧ ¬has_leg_color_black
Olive_sided_Flycatcher ↔ has_belly_color_grey ∧ has_belly_color_white
Scissor_tailed_Flycatcher ↔ has_forehead_color_white ∧ ¬has_under_tail_color_white ∧ ¬has_shape_perchinglike ∧ ¬has_tail_pattern_solid
Vermilion_Flycatcher ↔ has_upper_tail_color_black ∧ has_wing_shape_roundedwings ∧ has_leg_color_black ∧ ¬has_belly_color_white ∧ ¬has_back_pattern_striped ∧ ¬has_primary_color_black
Yellow_bellied_Flycatcher ↔ has_tail_shape_notched_tail ∧ has_wing_pattern_multicolored ∧ ¬has_wing_shape_roundedwings ∧ ¬has_primary_color_yellow ∧ ¬has_bill_color_black
Frigatebird ↔ has_underparts_color_black ∧ has_underparts_color_white ∧ has_head_pattern_plain ∧ ¬has_shape_perchinglike
Northern_Fulmar ↔ has_under_tail_color_white ∧ has_crown_color_white ∧ ¬has_upper_tail_color_white
Gadwall ↔ has_under_tail_color_black ∧ has_size_medium_9__16_in ∧ has_bill_color_black ∧ ¬has_leg_color_grey ∧ ¬has_crown_color_black

Table 12 continued from previous page

Formulas
American_Goldfinch ↔ has_under_tail_color_black ∧ has_back_pattern_solid ∧ has_wing_pattern_multicolored ∧ ¬has_belly_color_white ∧ ¬has_bill_color_black
European_Goldfinch ↔ has_leg_color_buff ∧ has_wing_pattern_multicolored ∧ ¬has_tail_pattern_solid
Boat_tailed_Grackle ↔ has_throat_color_black ∧ has_wing_shape_roundedwings ∧ ¬has_bill_length_shorter_than_head ∧ ¬has_size_small_5__9_in ∧ ¬has_size_medium_9__16_in
Eared_Grebe ↔ has_belly_color_grey ∧ has_primary_color_black ∧ ¬has_tail_pattern_solid
Horned_Grebe ↔ has_primary_color_black ∧ has_bill_color_black ∧ ¬has_nape_color_black ∧ ¬has_size_small_5__9_in ∧ ¬has_belly_pattern_solid
Pied_billed_Grebe ↔ has_under_tail_color_brown ∧ has_size_medium_9__16_in
Western_Grebe ↔ has_size_medium_9__16_in ∧ has_primary_color_white ∧ ¬has_throat_color_black ∧ ¬has_under_tail_color_white
Blue_Grosbeak ↔ has_under_tail_color_black ∧ has_bill_color_grey ∧ ¬has_tail_pattern_solid ∧ ¬has_crown_color_black
Evening_Grosbeak ↔ has_nape_color_brown ∧ has_tail_pattern_solid ∧ ¬has_nape_color_buff ∧ ¬has_back_pattern_solid
Pine_Grosbeak ↔ has_under_tail_color_grey ∧ has_leg_color_black ∧ has_wing_pattern_multicolored ∧ ¬has_back_pattern_solid
Rose_breasted_Grosbeak ↔ has_bill_shape_cone ∧ has_wing_shape_roundedwings ∧ has_primary_color_white ∧ ¬has_nape_color_buff
Pigeon_Guillemot ↔ has_underparts_color_black ∧ has_size_medium_9__16_in ∧ ¬has_leg_color_black
California_Gull ↔ has_under_tail_color_black ∧ has_wing_pattern_solid ∧ ¬has_back_pattern_solid
Glaucous_winged_Gull ↔ has_upper_tail_color_white ∧ has_under_tail_color_grey
Heermann_Gull ↔ has_nape_color_grey ∧ has_crown_color_white ∧ ¬has_shape_perchinglike
Herring_Gull ↔ has_size_medium_9__16_in ∧ has_primary_color_grey ∧ has_wing_pattern_solid ∧ ¬has_upper_tail_color_grey ∧ ¬has_upper_tail_color_black
Ivory_Gull ↔ has_leg_color_black ∧ has_bill_color_grey ∧ ¬has_shape_perchinglike
Ring_billed_Gull ↔ has_under_tail_color_white ∧ has_bill_color_black ∧ ¬has_head_pattern_plain ∧ ¬has_forehead_color_black ∧ ¬has_shape_perchinglike ∧ ¬has_wing_pattern_striped
Slaty_backed_Gull ↔ has_upperparts_color_black ∧ has_forehead_color_white ∧ has_size_medium_9__16_in ∧ ¬has_upper_tail_color_grey
Western_Gull ↔ has_crown_color_white ∧ ¬has_shape_perchinglike ∧ ¬has_back_pattern_solid
Anna_Hummingbird ↔ has_size_very_small_3__5_in ∧ ¬has_breast_color_white ∧ ¬has_wing_shape_roundedwings
Ruby_throated_Hummingbird ↔ has_belly_color_white ∧ has_leg_color_black ∧ ¬has_wing_shape_roundedwings ∧ ¬has_size_small_5__9_in ∧ ¬has_back_pattern_solid
Rufous_Hummingbird ↔ has_size_very_small_3__5_in ∧ has_wing_pattern_multicolored ∧ ¬has_shape_perchinglike
Green_Violetear ↔ has_nape_color_blue ∧ ¬has_bill_length_shorter_than_head

Table 12 continued from previous page

Formulas
Long_tailed_Jaeger ↔ (has_wing_color_grey ∧ has_under_tail_color_black ∧ ¬has_back_color_grey ∧ ¬has_bill_length_shorter_than_head) ∨ (has_under_tail_color_black ∧ ¬has_wing_color_black ∧ ¬has_back_color_grey ∧ ¬has_size_small_5__9_in ∧ ¬has_primary_color_brown)
Pomarine_Jaeger ↔ has_size_medium_9__16_in ∧ has_leg_color_black ∧ has_crown_color_black ∧ ¬has_breast_color_black ∧ ¬has_under_tail_color_white
Blue_Jay ↔ has_forehead_color_blue ∧ has_under_tail_color_black ∧ has_leg_color_black
Florida_Jay ↔ has_breast_pattern_multicolored ∧ has_back_pattern_multicolored
Green_Jay ↔ has_under_tail_color_yellow ∧ has_leg_color_grey ∧ ¬has_nape_color_grey ∧ ¬has_crown_color_black
Dark_eyed_Junco ↔ has_underparts_color_white ∧ has_throat_color_grey
Tropical_Kingbird ↔ has_forehead_color_grey ∧ has_primary_color_yellow ∧ has_bill_color_black ∧ ¬has_back_pattern_multicolored
Gray_Kingbird ↔ has_forehead_color_grey ∧ ¬has_bill_length_shorter_than_head ∧ ¬has_under_tail_color_black
Belted_Kingfisher ↔ has_breast_pattern_multicolored ∧ has_wing_shape_roundedwings ∧ ¬has_back_color_black ∧ ¬has_bill_length_shorter_than_head
Green_Kingfisher ↔ has_throat_color_white ∧ has_tail_pattern_solid ∧ ¬has_breast_color_white ∧ ¬has_belly_pattern_solid
Pied_Kingfisher ↔ has_breast_color_black ∧ has_wing_shape_roundedwings ∧ has_leg_color_black ∧ ¬has_wing_pattern_solid ∧ ¬has_wing_pattern_striped ∧ ¬has_wing_pattern_multicolored
Ringed_Kingfisher ↔ has_size_small_5__9_in ∧ has_primary_color_grey ∧ ¬has_nape_color_grey ∧ ¬has_wing_shape_roundedwings ∧ ¬has_wing_pattern_multicolored
White_breasted_Kingfisher ↔ has_crown_color_brown ∧ has_wing_pattern_multicolored
Red_legged_Kittiwake ↔ has_wing_color_white ∧ has_bill_length_shorter_than_head ∧ ¬has_tail_shape_notched_tail ∧ ¬has_forehead_color_blue ∧ ¬has_forehead_color_grey ∧ ¬has_nape_color_brown ∧ ¬has_back_pattern_striped ∧ ¬has_tail_pattern_multicolored ∧ ¬has_crown_color_black
Horned_Lark ↔ has_primary_color_buff ∧ ¬has_under_tail_color_black ∧ ¬has_wing_shape_roundedwings ∧ ¬has_back_pattern_solid ∧ ¬has_wing_pattern_striped
Pacific_Loon ↔ has_size_medium_9__16_in ∧ has_leg_color_grey ∧ ¬has_belly_pattern_solid
Mallard ↔ has_breast_color_brown ∧ has_wing_pattern_multicolored ∧ ¬has_forehead_color_yellow
Western_Meadowlark ↔ has_belly_color_yellow ∧ has_leg_color_buff ∧ has_bill_color_grey
Hooded_Merganser ↔ has_tail_pattern_solid ∧ has_bill_color_black ∧ ¬has_eye_color_black
Red_breasted_Merganser ↔ has_forehead_color_black ∧ ¬has_belly_color_white ∧ ¬has_belly_pattern_solid ∧ ¬has_wing_pattern_striped
Mockingbird ↔ has_forehead_color_grey ∧ has_wing_shape_roundedwings ∧ ¬has_upperparts_color_grey
Nighthawk ↔ has_breast_color_brown ∧ ¬has_underparts_color_brown ∧ ¬has_belly_pattern_solid
Clark_Nutcracker ↔ has_forehead_color_grey ∧ has_leg_color_grey ∧ has_wing_pattern_multicolored ∧ ¬has_primary_color_yellow
White_breasted_Nuthatch ↔ has_back_pattern_multicolored ∧ has_tail_pattern_multicolored ∧ ¬has_nape_color_white ∧ ¬has_belly_color_yellow

Table 12 continued from previous page

Formulas
Baltimore_Oriole ↔ has_breast_color_yellow ∧ has_under_tail_color_yellow ∧ ¬has_wing_shape_roundedwings
Hooded_Oriole ↔ has_breast_color_yellow ∧ has_back_pattern_solid ∧ has_tail_pattern_solid ∧ has_wing_pattern_multicolored
Orchard_Oriole ↔ has_leg_color_grey ∧ has_crown_color_black ∧ has_wing_pattern_multicolored ∧ ¬has_under_tail_color_yellow ∧ ¬has_belly_color_white
Scott_Oriole ↔ has_under_tail_color_yellow ∧ has_wing_pattern_multicolored ∧ ¬has_back_pattern_solid ∧ ¬has_back_pattern_multicolored
Ovenbird ↔ has_breast_color_black ∧ has_throat_color_white ∧ has_wing_pattern_solid ∧ ¬has_leg_color_grey
Brown_Pelican ↔ has_wing_pattern_solid ∧ ¬has_breast_pattern_solid ∧ ¬has_back_pattern_solid ∧ ¬has_primary_color_yellow
White_Pelican ↔ has_crown_color_white ∧ ¬has_head_pattern_plain ∧ ¬has_under_tail_color_black ∧ ¬has_size_medium_9__16_in ∧ ¬has_shape_perchinglike
Western_Wood_Pewee ↔ has_tail_pattern_solid ∧ has_bill_color_black ∧ has_crown_color_grey ∧ ¬has_under_tail_color_grey ∧ ¬has_wing_shape_roundedwings
Sayornis ↔ has_upper_tail_color_brown ∧ has_head_pattern_plain
American_Pipit ↔ has_nape_color_buff ∧ has_wing_shape_roundedwings ∧ ¬has_belly_pattern_solid ∧ ¬has_primary_color_brown
Whip_poor_Will ↔ has_wing_shape_roundedwings ∧ ¬has_belly_color_white ∧ ¬has_shape_perchinglike ∧ ¬has_leg_color_black ∧ ¬has_crown_color_brown ∧ ¬has_wing_pattern_solid
Horned_Puffin ↔ has_throat_color_black ∧ has_eye_color_black ∧ ¬has_breast_color_black ∧ ¬has_wing_shape_roundedwings ∧ ¬has_shape_perchinglike
Common_Raven ↔ has_wing_shape_roundedwings ∧ has_size_medium_9__16_in ∧ ¬has_bill_shape_hooked_seabird ∧ ¬has_shape_perchinglike
White_necked_Raven ↔ has_nape_color_white ∧ ¬has_throat_color_white ∧ ¬has_size_small_5__9_in
American_Redstart ↔ has_underparts_color_black ∧ has_wing_pattern_multicolored ∧ ¬has_belly_color_black ∧ ¬has_leg_color_grey ∧ ¬has_crown_color_grey
Geococcyx ↔ has_nape_color_brown ∧ has_leg_color_grey ∧ ¬has_primary_color_white
Loggerhead_Shrike ↔ has_nape_color_grey ∧ has_tail_pattern_multicolored ∧ ¬has_tail_shape_notched_tail ∧ ¬has_breast_color_yellow ∧ ¬has_bill_length_about_the_same_as_head
Great_Grey_Shrike ↔ has_forehead_color_grey ∧ has_wing_shape_roundedwings ∧ has_wing_pattern_multicolored ∧ ¬has_upperparts_color_white ∧ ¬has_back_pattern_multicolored
Baird_Sparrow ↔ has_back_color_brown ∧ has_tail_shape_notched_tail ∧ ¬has_wing_shape_roundedwings
Black_throated_Sparrow ↔ has_forehead_color_grey ∧ has_belly_color_white ∧ ¬has_throat_color_white ∧ ¬has_wing_pattern_multicolored
Brewer_Sparrow ↔ has_wing_shape_roundedwings ∧ has_back_pattern_striped ∧ has_primary_color_buff ∧ ¬has_under_tail_color_brown ∧ ¬has_size_very_small_3__5_in ∧ ¬has_primary_color_brown ∧ ¬has_crown_color_black
Chipping_Sparrow ↔ has_nape_color_grey ∧ has_back_pattern_striped ∧ ¬has_upper_tail_color_buff

Table 12 continued from previous page

Formulas
Clay_colored_Sparrow ↔ has_throat_color_white ∧ has_forehead_color_brown ∧ has_primary_color_buff ∧ ¬has_nape_color_brown
House_Sparrow ↔ has_back_pattern_striped ∧ has_bill_color_black ∧ ¬has_breast_color_yellow ∧ ¬has_forehead_color_black ∧ ¬has_leg_color_grey
Field_Sparrow ↔ has_belly_color_buff ∧ has_wing_pattern_striped ∧ ¬has_leg_color_buff
Fox_Sparrow ↔ has_breast_pattern_striped ∧ ¬has_back_pattern_solid ∧ ¬has_wing_pattern_striped
Grasshopper_Sparrow ↔ has_under_tail_color_buff ∧ has_belly_color_buff ∧ has_leg_color_buff
Harris_Sparrow ↔ has_nape_color_buff ∧ has_primary_color_white
Henslow_Sparrow ↔ has_breast_color_black ∧ has_leg_color_buff ∧ ¬has_primary_color_yellow
Le_Conte_Sparrow ↔ has_wing_shape_roundedwings ∧ has_back_pattern_striped ∧ ¬has_back_color_brown ∧ ¬has_bill_color_black
Lincoln_Sparrow ↔ has_size_very_small_3__5_in ∧ has_wing_pattern_striped ∧ ¬has_belly_pattern_solid ∧ ¬has_crown_color_white
Nelson_Sharp_tailed_Sparrow ↔ has_back_pattern_striped ∧ ¬has_nape_color_buff ∧ ¬has_size_small_5__9_in ∧ ¬has_crown_color_black
Savannah_Sparrow ↔ has_back_pattern_striped ∧ ¬has_back_color_buff ∧ ¬has_under_tail_color_black ∧ ¬has_belly_pattern_solid ∧ ¬has_leg_color_black
Seaside_Sparrow ↔ has_shape_perchinglike ∧ has_tail_pattern_solid ∧ ¬has_belly_pattern_solid ∧ ¬has_bill_color_black ∧ ¬has_wing_pattern_solid
Song_Sparrow ↔ has_nape_color_buff ∧ has_back_pattern_striped ∧ ¬has_forehead_color_black ∧ ¬has_primary_color_buff
Tree_Sparrow ↔ has_tail_shape_notched_tail ∧ has_belly_color_white ∧ has_back_pattern_striped ∧ ¬has_back_color_buff ∧ ¬has_under_tail_color_brown
Vesper_Sparrow ↔ has_breast_color_white ∧ has_back_pattern_striped ∧ has_leg_color_buff ∧ ¬has_under_tail_color_buff
White_crowned_Sparrow ↔ has_forehead_color_black ∧ has_nape_color_grey ∧ ¬has_leg_color_buff
White_throated_Sparrow ↔ has_forehead_color_yellow ∧ has_primary_color_brown
Cape_Glossy_Starling ↔ has_nape_color_blue ∧ has_wing_pattern_solid
Bank_Swallow ↔ has_bill_shape_cone ∧ has_breast_color_white ∧ has_bill_color_black ∧ ¬has_forehead_color_blue ∧ ¬has_forehead_color_black ∧ ¬has_wing_pattern_solid
Barn_Swallow ↔ has_back_pattern_solid ∧ has_primary_color_black ∧ has_bill_color_black ∧ ¬has_belly_color_black ∧ ¬has_shape_perchinglike ∧ ¬has_leg_color_black
Cliff_Swallow ↔ has_belly_color_buff ∧ has_tail_pattern_solid ∧ ¬has_back_pattern_solid
Tree_Swallow ↔ has_primary_color_white ∧ has_bill_color_black ∧ ¬has_upperparts_color_black ∧ ¬has_size_medium_9__16_in ∧ ¬has_primary_color_brown
Scarlet_Tanager ↔ has_upperparts_color_black ∧ has_forehead_color_red ∧ ¬has_bill_length_about_the_same_as_head ∧ ¬has_under_tail_color_white
Summer_Tanager ↔ has_tail_shape_notched_tail ∧ has_leg_color_grey ∧ ¬has_throat_color_white ∧ ¬has_forehead_color_black ∧ ¬has_primary_color_grey
Artic_Tern ↔ has_head_pattern_capped ∧ has_nape_color_black ∧ ¬has_bill_shape_dagger ∧ ¬has_upper_tail_color_black

Table 12 continued from previous page

Formulas
Black_Tern ↔ has_belly_color_black ∧ ¬has_under_tail_color_black ∧ ¬has_wing_shape_roundedwings
Caspian_Tern ↔ has_head_pattern_capped ∧ has_size_medium_9__16_in ∧ has_wing_pattern_solid ∧ ¬has_nape_color_black
Common_Tern ↔ has_wing_color_grey ∧ has_back_color_white ∧ ¬has_forehead_color_white
Elegant_Tern ↔ has_forehead_color_white ∧ has_size_medium_9__16_in ∧ ¬has_head_pattern_plain
Forsters_Tern ↔ has_head_pattern_capped ∧ has_nape_color_black ∧ has_bill_color_black ∧ ¬has_upper_tail_color_black
Least_Tern ↔ has_forehead_color_white ∧ has_crown_color_black ∧ has_wing_pattern_solid
Green_tailed_Towhee ↔ has_wing_shape_roundedwings ∧ has_bill_color_grey ∧ ¬has_throat_color_yellow ∧ ¬has_nape_color_blue ∧ ¬has_nape_color_brown
Brown_Thrasher ↔ has_nape_color_brown ∧ ¬has_forehead_color_brown ∧ ¬has_belly_color_yellow ∧ ¬has_leg_color_grey ∧ ¬has_wing_pattern_striped
Sage_Thrasher ↔ has_wing_pattern_striped ∧ ¬has_eye_color_black
Black_capped_Vireo ↔ has_nape_color_black ∧ has_size_very_small_3__5_in ∧ has_leg_color_grey
Blue_headed_Vireo ↔ has_primary_color_grey ∧ has_leg_color_grey ∧ has_wing_pattern_striped
Philadelphia_Vireo ↔ has_nape_color_grey ∧ has_size_very_small_3__5_in ∧ has_bill_color_grey
Red_eyed_Vireo ↔ has_upperparts_color_buff ∧ has_forehead_color_grey
Warbling_Vireo ↔ has_nape_color_grey ∧ has_size_very_small_3__5_in ∧ has_primary_color_buff ∧ ¬has_under_tail_color_buff
White_eyed_Vireo ↔ has_tail_shape_notched_tail ∧ has_tail_pattern_multicolored ∧ ¬has_upperparts_color_black
Yellow_throated_Vireo ↔ has_nape_color_yellow ∧ has_belly_color_white
Bay_breasted_Warbler ↔ has_wing_shape_roundedwings ∧ has_back_pattern_striped ∧ has_leg_color_grey
Black_and_white_Warbler ↔ has_size_very_small_3__5_in ∧ has_wing_pattern_striped ∧ ¬has_primary_color_buff
Black_throated_Blue_Warbler ↔ has_primary_color_black ∧ has_wing_pattern_multicolored ∧ ¬has_breast_color_yellow ∧ ¬has_under_tail_color_black
Blue_winged_Warbler ↔ has_back_color_grey ∧ ¬has_nape_color_grey ∧ ¬has_back_pattern_striped ∧ ¬has_primary_color_grey ∧ ¬has_crown_color_black
Canada_Warbler ↔ has_under_tail_color_grey ∧ has_nape_color_grey ∧ has_belly_color_yellow ∧ ¬has_back_pattern_multicolored ∧ ¬has_primary_color_grey
Cape_May_Warbler ↔ has_back_pattern_striped ∧ has_bill_color_black ∧ ¬has_belly_color_white ∧ ¬has_belly_pattern_solid
Cerulean_Warbler ↔ has_nape_color_blue ∧ has_size_very_small_3__5_in ∧ has_shape_perchinglike
Chestnut_sided_Warbler ↔ has_underparts_color_white ∧ has_upper_tail_color_grey ∧ ¬has_wing_color_white ∧ ¬has_bill_length_about_the_same_as_head ∧ ¬has_belly_color_grey ∧ ¬has_back_pattern_solid ∧ ¬has_primary_color_yellow
Golden_winged_Warbler ↔ has_forehead_color_yellow ∧ has_wing_pattern_multicolored ∧ ¬has_primary_color_yellow

Table 12 continued from previous page

Formulas
Hooded_Warbler ↔ has_forehead_color_yellow ∧ has_primary_color_yellow ∧ has_leg_color_buff ∧ has_crown_color_black
Kentucky_Warbler ↔ has_size_small_5__9_in ∧ has_primary_color_yellow ∧ has_leg_color_buff ∧ has_crown_color_black
Magnolia_Warbler ↔ has_forehead_color_grey ∧ has_primary_color_black
Mourning_Warbler ↔ has_forehead_color_grey ∧ has_leg_color_buff
Myrtle_Warbler ↔ has_leg_color_black ∧ has_wing_pattern_striped ∧ ¬has_tail_pattern_solid
Nashville_Warbler ↔ has_under_tail_color_yellow ∧ has_back_pattern_multicolored ∧ ¬has_crown_color_black
Orange_crowned_Warbler ↔ has_bill_shape_allpurpose ∧ has_tail_pattern_multicolored ∧ ¬has_breast_color_yellow ∧ ¬has_primary_color_white ∧ ¬has_crown_color_grey ∧ ¬has_crown_color_black
Palm_Warbler ↔ has_primary_color_yellow ∧ has_wing_pattern_striped ∧ ¬has_belly_color_yellow
Pine_Warbler ↔ has_forehead_color_yellow ∧ has_under_tail_color_grey ∧ has_bill_color_grey
Prairie_Warbler ↔ has_size_very_small_3__5_in ∧ has_crown_color_yellow
Prothonotary_Warbler ↔ has_under_tail_color_black ∧ has_tail_pattern_solid ∧ has_leg_color_grey ∧ ¬has_upperparts_color_black
Swainson_Warbler ↔ has_tail_shape_notched_tail ∧ has_bill_length_about_the_same_as_head ∧ ¬has_nape_color_grey ∧ ¬has_primary_color_black
Tennessee_Warbler ↔ has_upper_tail_color_grey ∧ has_primary_color_yellow ∧ ¬has_breast_color_yellow
Wilson_Warbler ↔ has_under_tail_color_yellow ∧ has_crown_color_black ∧ ¬has_leg_color_grey
Worm_eating_Warbler ↔ has_crown_color_yellow ∧ ¬has_primary_color_yellow ∧ ¬has_bill_color_black
Yellow_Warbler ↔ has_under_tail_color_yellow ∧ has_wing_pattern_striped
Northern_Waterthrush ↔ has_size_small_5__9_in ∧ has_tail_pattern_solid ∧ has_leg_color_buff ∧ ¬has_breast_color_yellow ∧ ¬has_primary_color_grey
Louisiana_Waterthrush ↔ has_breast_pattern_striped ∧ has_wing_pattern_solid ∧ ¬has_nape_color_brown
Bohemian_Waxwing ↔ has_upper_tail_color_grey ∧ has_wing_pattern_multicolored ∧ ¬has_under_tail_color_grey
Cedar_Waxwing ↔ has_nape_color_buff ∧ has_wing_pattern_multicolored
American_Three_toed_Woodpecker ↔ has_under_tail_color_white ∧ has_tail_pattern_solid ∧ has_leg_color_grey
Pileated_Woodpecker ↔ has_nape_color_white ∧ has_leg_color_grey ∧ ¬has_primary_color_white
Red_bellied_Woodpecker ↔ has_forehead_color_red ∧ has_wing_pattern_striped
Red_cockaded_Woodpecker ↔ has_head_pattern_capped ∧ has_belly_color_black
Red_headed_Woodpecker ↔ has_forehead_color_red ∧ has_back_pattern_solid ∧ has_wing_pattern_multicolored
Downy_Woodpecker ↔ has_under_tail_color_white ∧ has_back_pattern_multicolored ∧ ¬has_wing_pattern_multicolored
Bewick_Wren ↔ has_under_tail_color_brown ∧ has_under_tail_color_black

Table 12 continued from previous page

Formulas
Cactus_Wren ↔ has_nape_color_white ∧ ¬has_belly_color_white ∧ ¬has_back_pattern_solid
Carolina_Wren ↔ has_breast_color_buff ∧ has_bill_color_grey ∧ ¬has_leg_color_grey
House_Wren ↔ has_breast_color_buff ∧ ¬has_forehead_color_black ∧ ¬has_wing_shape_roundedwings ∧ ¬has_leg_color_black ∧ ¬has_wing_pattern_solid
Marsh_Wren ↔ has_nape_color_buff ∧ has_belly_color_white ∧ has_belly_color_buff
Rock_Wren ↔ has_under_tail_color_buff ∧ has_size_very_small_3__5_in ∧ ¬has_crown_color_brown
Winter_Wren ↔ has_breast_pattern_solid ∧ has_breast_color_buff ∧ ¬has_belly_pattern_solid
Common_Yellowthroat ↔ has_forehead_color_black ∧ has_under_tail_color_yellow ∧ ¬has_crown_color_black