Transparent Classification with Multilayer Logical Perceptrons and Random Binarization

Zhuo Wang,[1] **Wei Zhang,**[2*] **Ning Liu,**[1] **Jianyong Wang**[1*]
[1]Department of Computer Science and Technology, Tsinghua University
[2]School of Computer Science and Technology, Shanghai Key Laboratory of Trustworthy Computing, East China Normal University
wang-z18@mails.tsinghua.edu.cn, {zhangwei.thu2011, victorliucs}@gmail.com, jianyong@tsinghua.edu.cn

Abstract

Models with transparent inner structure and high classification performance are required to reduce potential risk and provide trust for users in domains like health care, finance, security, etc. However, existing models are hard to simultaneously satisfy the above two properties. In this paper, we propose a new hierarchical rule-based model for classification tasks, named Concept Rule Sets (CRS), which has both a strong expressive ability and a transparent inner structure. To address the challenge of efficiently learning the non-differentiable CRS model, we propose a novel neural network architecture, Multilayer Logical Perceptron (MLLP), which is a continuous version of CRS. Using MLLP and the Random Binarization (RB) method we proposed, we can search the discrete solution of CRS in continuous space using gradient descent and ensure the discrete CRS acts almost the same as the corresponding continuous MLLP. Experiments on 12 public data sets show that CRS outperforms the state-of-the-art approaches and the complexity of the learned CRS is close to the simple decision tree.

Introduction

Relying on strong ability of data modeling, machine learning, especially deep learning, becomes the main paradigm for decision-making systems (Goodfellow et al. 2016; Doshi-Velez and Kim 2017). The decision-making systems have widespread usage in important areas such as medicine, finance, politics, as well as law, where people need the explanations why decisions are made to ensure their safety and protect their rights (Goodman and Flaxman 2016; Lipton 2016). As a result, the demand for the transparency of machine learning methods is increasing, which is crucial for earning the trust of users (Doshi-Velez and Kim 2017) and reducing potential risks and bugs (Chu et al. 2018). However, most of the machine learning models can hardly ensure good predictive ability and transparency at the same time, and sacrificing transparency for good performance could result in serious consequences.

One important notion of transparency is that each part of the model, including input, parameter, and calculation, etc., admits an intuitive explanation (Lipton 2016). For example, each node in Decision Tree corresponds to a rule description, e.g., "*if* #citations > 300 *then* a good paper". The main reason for deep neural networks not being transparent models is that the activation value of each neuron is not explainable. Unlike the Boolean state, i.e., True or False, used in the rule-based model, the continuous activation value of deep neural networks is hard to associate with one actual meaning. In addition, ensemble models like Random Forest are not transparent as well for the decision is made by hundreds of models which are hard to explain as a whole.

Linear model and rule-based model are two widely used transparent models, however, they both suffer from low model capability. Linear models cannot fit non-linear data well because of the limitation of their model structures. Rule-based models (e.g., Decision Tree, Rule Set, and Rule List) have strong model expressivity that can fit both linear and non-linear data well. However, a fundamental limitation with these rule-based models is that they find the rules by employing various heuristic methods (Quinlan 1993; Breiman 2017; Cohen 1995) which may not find the globally best solution or a solution with close performance. In addition, the gradient descent method cannot be directly applied to the rule-based model learning for the discrete parameters.

Studies in recent years provide some solutions to improve model transparency in different aspects. Surrogate models (Frosst and Hinton 2017; Ribeiro, Singh, and Guestrin 2016; Selvaraju et al. 2017) try to use a simple model to fit a complex model globally or locally. Then understanding the complex model by interpreting the simple model. Hidden layers investigation (Yosinski et al. 2015; Zhou et al. 2018; Zhang, Wu, and Zhu 2017) aims to visualize and analyze the statuses of hidden neurons or features learned by hidden neurons. All these methods improve the model transparency in specific scenarios. However, there is always a gap between the surrogate model and the complex model it aims to fit, and this inconsistency may have an influence on subsequent model analysis and understanding. The interpretation provided by hidden layer investigation is very intuitive, but it could hardly be quantitatively measured and applied, which limits the scope of application of this method. One fundamental problem of these methods is that they all learn complex models first to obtain high classification performance, then try to interpret these learned complex models which are hard to understand and the interpretations may be incon-

*Corresponding authors.

sistent. Unlike these methods aiming to improve the transparency of complex models with high performance, we can try the opposite way to solve the problem, i.e., improving the performance of transparent models.

As mentioned above, rule-based models have both transparent inner structures and strong model expressivity but suffer from lacking an efficient optimization method because of the discrete parameters. If we can adopt gradient descent to learn rule-based models, we may obtain a model with both transparency and high performance.

In this paper, we propose a new hierarchical rule-based model, **Concept Rule Sets (CRS)** (see Figure 1) and its continuous version, **Multilayer Logical Perceptron (MLLP)**, which is a neural network with logical activation functions and constrained weights. We can use gradient descent to learn the discrete CRS via continuous MLLP. To ensure the discrete CRS and the continuous MLLP have almost the same behavior, we propose a new training method called **Random Binarization (RB)**, which binarizes a randomly selected subset of weights in MLLP during the training process to enable the MLLP to keep consistency with the corresponding CRS.

The main contributions of this paper are as follows:
(i) We propose a new hierarchical rule-based model, Concept Rule Sets, with strong model expressivity and ability to learn transparent data representation for classification tasks. The complete definition of CRS is also given.
(ii) We propose a new neural network, Multilayer Logical Perceptron, and a new training method, Random Binarization to learn CRS efficiently using gradient descent. We also provide solutions to overcome the vanishing gradient problems that occur during training.
(iii) Based on the transparent structure of CRS, we propose two simplification methods for CRS to reduce the model complexity.
(iv) We conduct experiments on 12 public data sets to compare the classification performance and complexity of our model with other representative classification models.

Related Work

There are four classes of methods that are directly related to this work, i.e., rule-based model, surrogate model, hidden layers investigation and binary neural network.

Rule-based Models are considered to be interpretable because of their transparent inner structure. Decision tree, rule list, and rule set are the widely used structure in rule-based models. (Letham et al. 2015; Wang et al. 2017; Yang, Rudin, and Seltzer 2017) leverage Bayesian frameworks to restrict and adjust model structure more reasonably. (Lakkaraju, Bach, and Leskovec 2016) learns interpretable decision sets by using independent if-then rules and a non-monotone submodular objective function. (Angelino et al. 2017) learns certifiably optimal rule lists and leverages algorithmic bounds and efficient data structures to speed up.

However, most existing rule-based models need frequent itemsets mining and/or long-time searching, which limits their applications. Moreover, it is hard for these interpretable rule-based models to get comparable performance with complex models like Random Forest.

Surrogate Models use simple models to fit or approximate complex models (e.g., deep neural networks) globally or locally, and explain the complex model by interpreting the simple model.

(Hinton, Vinyals, and Dean 2015) proposed a distillation method that trains a relatively small neural network to predict the output of a large network, regarding as learning knowledge of the large network. To get a more transparent model by distillation methods, (Frosst and Hinton 2017) took place the small neural network with a decision tree.

(Ribeiro, Singh, and Guestrin 2016) proposed LIME to interpret any classifier by using a transparent model to fit the classifier locally. (Chu et al. 2018) developed OpenBox to transform a piecewise linear neural network into a set of linear classifiers which help interpret the network.

(Zhou et al. 2016) presented a method that maps the predicted class score back to the previous convolutional layer to generate the class activation maps (CAMs). (Selvaraju et al. 2017) proposed a Grad-CAM method that generalizes CAM by using the First-order Taylor-series approximation to approximate the part after the last convolutional layer.

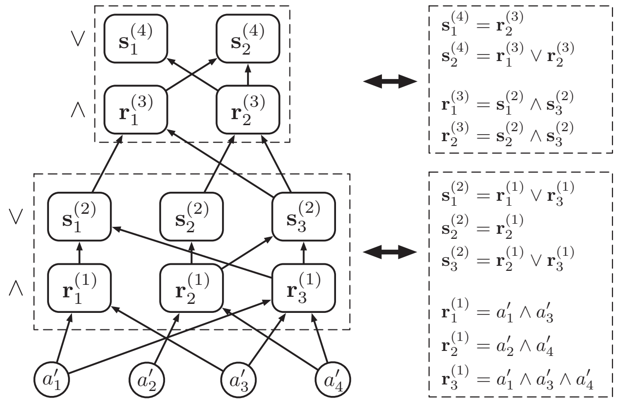
The surrogate approaches can use complex models to get higher accuracy and use simple approximate models to get interpretation. However, inconsistencies always exist between an actual model and its surrogate model (Kim and Doshi-Velez 2017), and there is no guarantee for the authenticity of the interpretation from the surrogate model because of these inconsistencies.

Hidden Layers Investigations visualize and analyze statuses of hidden neurons or their learned features.

(Yosinski et al. 2015) introduced two tools to visualize the activations and features at each layer of a trained convnet. (Mahendran and Vedaldi 2015) contributed a general framework to reconstruct the image by inverting representations to analyze the visual information contained in representations. (Zhou et al. 2018) proposed a Network Dissection method that interprets networks by providing meaningful labels to their individual units. (Zhang, Wu, and Zhu 2017) proposed an interpretable CNN that each filter in a high convolutional layer is assigned with an object part automatically during the learning process. Based on the interpretable CNN, (Zhang et al. 2018) further proposed to use a decision tree to mine the decision mode memorized in fully-connected layers.

The interpretation provided by hidden layer investigation is intuitive with visualization, but it could hardly be quantitatively measured and applied. The main difference between hidden layers investigation and our method is hidden layers investigation tries to interpret learned data representation while our method aims to learn a specific and interpretable data representation.

Binary Neural Networks train a DNN with binary weights during the forward and backward propagations (Courbariaux, Bengio, and David 2015; Courbariaux et al. 2016). Although their weights are binary, they are still much more complex than the rule-based models. And it is harder to understand the operations of binary neural networks than logical operations. Moreover, the model capability of the binary neural network is also restricted.



$\vee$
$\wedge$
$\vee$
$\wedge$
$\mathbf{s}_1^{(4)}$
$\mathbf{s}_2^{(4)}$
$\mathbf{r}_1^{(3)}$
$\mathbf{r}_2^{(3)}$
$\mathbf{s}_1^{(2)}$
$\mathbf{s}_2^{(2)}$
$\mathbf{s}_3^{(2)}$
$\mathbf{r}_1^{(1)}$
$\mathbf{r}_2^{(1)}$
$\mathbf{r}_3^{(1)}$
a'_1
a'_2
a'_3
a'_4
$\mathbf{s}_1^{(4)} = \mathbf{r}_2^{(3)}$
$\mathbf{s}_2^{(4)} = \mathbf{r}_1^{(3)} \vee \mathbf{r}_2^{(3)}$
$\mathbf{r}_1^{(3)} = \mathbf{s}_1^{(2)} \wedge \mathbf{s}_3^{(2)}$
$\mathbf{r}_2^{(3)} = \mathbf{s}_2^{(2)} \wedge \mathbf{s}_3^{(2)}$
$\mathbf{s}_1^{(2)} = \mathbf{r}_1^{(1)} \vee \mathbf{r}_3^{(1)}$
$\mathbf{s}_2^{(2)} = \mathbf{r}_2^{(1)}$
$\mathbf{s}_3^{(2)} = \mathbf{r}_2^{(1)} \vee \mathbf{r}_3^{(1)}$
$\mathbf{r}_1^{(1)} = a'_1 \wedge a'_3$
$\mathbf{r}_2^{(1)} = a'_2 \wedge a'_4$
$\mathbf{r}_3^{(1)} = a'_1 \wedge a'_3 \wedge a'_4$

Figure 1: A Concept Rule Sets example for intuitive understanding. The left side and the right side are two forms of CRS. One dashed box corresponds to one level in CRS. Arrows of the edges indicate the flow of data.

Concept Rule Sets

Notation Description

Let $\mathcal{D} = \{(X_1, \mathbf{y}_1), \ldots, (X_N, \mathbf{y}_N)\}$ denote the training data set with N instances, where X_i is the observed feature vector of the i-th instance with the j-th entry as $X_{i,j}$, and $\mathbf{y}_i$ is the corresponding class label, $i \in \{1, \ldots, N\}$. All feature values can be discrete or continuous, and all the classes are discrete values. After the discretization and binarization of all the data features and class labels, $\mathcal{D}$ is converted into $\mathcal{D}' = \{(X'_1, Y'_1), \ldots, (X'_N, Y'_N)\}$, where $X'_i \in \{0,1\}^J$ is a binary feature vector with the size of J and $Y'_i \in \{0,1\}^C$ is a one-hot class label vector whose size is equal to C, the number of class labels. Let A' denote the set of binary features, and $a'_j \in A'$ is the j-th binary feature. Throughout this paper, we use 1 (True) and 0 (False) to represent the two states of a Boolean variable. Therefore, each dimension of the binary feature vector and one-hot class label vector can be considered as a Boolean variable.

Definition and Application of CRS

Before giving the definition of Concept Rule Sets, we first formulate the concepts of rule and rule set respectively. A rule $\mathbf{r}_i$ is a conjunction of one or more Boolean variables.

$$\mathbf{r}_i = \mathbf{b}_{i_1} \wedge \mathbf{b}_{i_2} \wedge \cdots \wedge \mathbf{b}_{i_p}$$

where $\mathbf{b}_{i_k}$ ($k \in \{1, \ldots, p\}$) is the Boolean variable in rule $\mathbf{r}_i$ and p is the length of the rule $\mathbf{r}_i$. A rule set $\mathbf{s}_j$ is a disjunction of one or more rules, i.e., Disjunctive Normal Form (DNF).

$$\mathbf{s}_j = \mathbf{r}_{j_1} \vee \mathbf{r}_{j_2} \vee \cdots \vee \mathbf{r}_{j_q}$$

where $\mathbf{r}_{j_k}$ ($k \in \{1, \ldots, q\}$) is the rule in rule set $\mathbf{s}_j$ and q is the number of rules in $\mathbf{s}_j$.

A Concept Rule Sets (CRS), denoted by $\mathcal{F}$, is a multi-level structure in which each level contains a conjunction layer followed by a disjunction layer. For a CRS with L levels, we denote its l-th layer ($l \in \{1, 3, \ldots, 2L-1\}$) by $\mathcal{R}^{(l)}$ for being a conjunction layer and denote the l-th layer ($l \in \{2, 4, \ldots, 2L\}$) by $\mathcal{S}^{(l)}$ for being a disjunction layer. For ease of expression, we represent the input layer, i.e., 0-th layer, by $\mathcal{S}^{(0)}$. Each layer in CRS contains a specific number of nodes and the edges connected with its previous layer, except $\mathcal{S}^{(0)}$. Let $\mathbf{n}_l$ denote the number of nodes in the l-th layer, and $W^{(l)}$ denote an $\mathbf{n}_l$-by-$\mathbf{n}_{l-1}$ adjacency matrix of the l-th layer and the $(l-1)$-th layer, where $W^{(l)}_{i,j} \in \{0,1\}$. $W^{(l)}_{i,j} = 1$ indicates there exists an edge connecting the i-th node in l-th layer to the j-th node in $(l-1)$-th layer, otherwise $W^{(l)}_{i,j} = 0$. Similar to neural networks, we regard these adjacency matrices as the weight matrices of CRS.

We denote the i-th node in layer $\mathcal{R}^{(l)}$ ($l \in \{1, 3, \ldots, 2L-1\}$) by $\mathbf{r}_i^{(l)}$, and the i-th node in layer $\mathcal{S}^{(l)}$ ($l \in \{0, 2, \ldots, 2L\}$) by $\mathbf{s}_i^{(l)}$. Specifically speaking, node $\mathbf{r}_i^{(l)}$ corresponds to one rule, in which the Boolean variables are nodes in the previous layer connected with $\mathbf{r}_i^{(l)}$, while node $\mathbf{s}_i^{(l)}$ corresponds to one rule set, in which the rules are nodes in previous layer connected with $\mathbf{s}_i^{(l)}$. Formally, the two types of nodes are defined as follows:

$$\mathbf{r}_i^{(l)} = \bigwedge_{W^{(l)}_{i,j}=1} \mathbf{s}_j^{(l-1)}, \qquad \mathbf{s}_i^{(l+1)} = \bigvee_{W^{(l+1)}_{i,j}=1} \mathbf{r}_j^{(l)}. \tag{1}$$

A concrete example of CRS is shown in Figure 1.

For any given instance, after setting the values of $\mathcal{S}^{(0)}$ according to its binary feature vector, we can compute the values of all the nodes in CRS layer by layer. If we set the number of nodes in the last layer, i.e., $\mathbf{n}_{2L}$, to C, CRS can work as a classifier $\mathcal{F} : \{0,1\}^J \to \{0,1\}^C$, which outputs the values of nodes in the last layer $\mathcal{S}^{(2L)}$, and $\mathbf{s}_i^{(2L)} = 1$ indicates that the CRS classifies the input instance as the i-th class label. If more than one dimension of $\mathbf{s}^{(2L)}$ has value equals to 1, we choose the first one as the result. Meanwhile, a crucial byproduct of CRS is the learned layer-wise representation of each instance, similar to the mechanism of deep neural networks (Goodfellow et al. 2016) but more transparent. Let $\mathbf{h}^{(l)}$ denote the data representation learnt by the l-th layer in CRS, which is a binary vector with $\mathbf{n}_l$ dimensions.

$$\mathbf{h}^{(l)} = \begin{cases} [\mathbf{r}_1^{(l)}, \mathbf{r}_2^{(l)}, \ldots, \mathbf{r}_{\mathbf{n}_l}^{(l)}]^\top & l \in \{1, 3, \ldots, 2L-1\} \\ [\mathbf{s}_1^{(l)}, \mathbf{s}_2^{(l)}, \ldots, \mathbf{s}_{\mathbf{n}_l}^{(l)}]^\top & l \in \{2, 4, \ldots, 2L\} \end{cases}$$

The value of $\mathbf{h}_i^{(l)}$ is equal to the value of the i-th node in the l-th layer which corresponds to a rule or a rule set. We can understand each dimension of $\mathbf{h}^{(l)}$ by analyzing the corresponding rule or rule set. It is much easier than analyzing the real-valued weights and activation values of hidden neurons in deep neural networks.

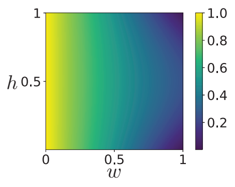
The CRS model enjoys two major intrinsic advantages. One is that the model has strong expressivity because one level in CRS is equal to the widely used rule sets model which can fit both the linear and nonlinear data appropriately. The multi-level structure enhances the expressivity of CRS further. The other advantage is that the CRS model has a transparent inner structure and is able to learn transparent data representations, as aforementioned.

h	w	F_c
0	0	1
0	1	0
1	0	1
1	1	1

(a)

h	w	F_d
0	0	0
0	1	0
1	0	0
1	1	1

(b)



(c)

Figure 2: (a) Truth table of $F_c(\cdot)$. (b) Truth table of $F_d(\cdot)$. (c) Heatmap of $F_c(h, w) \cdot (1 - F_d(h, w))$.

However, how to search the appropriate weight matrices in CRS to obtain high classification performance and low model complexity remains a challenging problem. We will discuss our proposed solution in the next section.

Multilayer Logical Perceptron

In this section, we introduce how to use Multilayer Logical Perceptron and Random Binarization method to learn CRS and then simplify the learned CRS.

Data Discretization and Binarization

For the reason that CRS only takes binary vector input and rules containing continuous features are hard to understand, we apply the recursive minimal entropy partitioning algorithm (Dougherty, Kohavi, and Sahami 1995) to discretize feature values. This algorithm partitions one feature recursively by searching the partition boundary which minimizes the class information entropy of candidate partitions. Minimal Description Length Principle is used to determine the stopping criteria. After data discretization, we use one-hot encoding to covert all the discrete features into binary features.

Network Structure and Training Method

For the similarity between the structure of CRS and multilayer perceptron (MLP), a straightforward idea is to use the gradient descent method to train CRS, like the way of training MLP. However, we cannot compute the gradient of CRS due to the discrete weights. To overcome this problem, we propose a new neural network called Multilayer Logical Perceptron (MLLP) and a tailored training method, which can search the discrete solution of CRS in a continuous space using gradient descent.

MLLP and its corresponding training method aim at tackling the following three challenges:

- Logical Activation Function. Commonly used activation functions cannot simulate the conjunction and disjunction operations.
- Vanishing Gradient Problem. The gradient could be extremely small due to the conjunction and disjunction operations.
- Discrete CRS Extraction. There is no guarantee the solutions found in continuous space could still work after converting them into discrete values.

Overall Structure MLLP has $2L + 1$ layers, same as the CRS model. And one neuron in MLLP corresponds to one node in CRS. Therefore, the number of neurons in l-th layer of MLLP is also $\mathbf{n}_l$. We denote the neuron corresponding to CRS node $\mathbf{r}_i^{(l)}$ by $\hat{\mathbf{r}}_i^{(l)}$ ($l \in \{1, 3, \dots, 2L-1\}$), and $\mathbf{s}_i^{(l)}$ by $\hat{\mathbf{s}}_i^{(l)}$ ($l \in \{2, 4, \dots, 2L\}$). All the layers in MLLP are fully connected layers. We denote the weights of the l-th layer of MLLP by $\hat{W}^{(l)}$, and $\hat{W}_{i,j}^{(l)} \in [0, 1]$.

Logical Activation Function In order to ensure the neurons in MLLP have the same behaviors as the corresponding nodes in CRS, activation functions that can simulate conjunction and disjunction operations are required. We adopt the conjunction function and disjunction function proposed in (Payani and Fekri 2019). Specifically, given the two n-dimensional vectors $\mathbf{h}$ and W_i, the conjunction function *Conj*$(\cdot)$ and disjunction function *Disj*$(\cdot)$ are given by:

$$\mathit{Conj}(\mathbf{h}, W_i) = \prod_{j=1}^{n} F_c(\mathbf{h}_j, W_{i,j}), \tag{2}$$

$$\mathit{Disj}(\mathbf{h}, W_i) = 1 - \prod_{j=1}^{n} (1 - F_d(\mathbf{h}_j, W_{i,j})), \tag{3}$$

where $F_c(h, w) = 1 - w(1 - h)$ and $F_d(h, w) = h \cdot w$.

In Equations 2 and 3, if $\mathbf{h}$ and W_i are both binary vectors, then $\mathit{Conj}(\mathbf{h}, W_i) = \bigwedge_{W_{i,j}=1} \mathbf{h}_j$ and $\mathit{Disj}(\mathbf{h}, W_i) = \bigvee_{W_{i,j}=1} \mathbf{h}_j$. The truth tables, shown in Figure 2, indicate that $F_c(h, w)$ and $F_d(h, w)$ work as selectors, and they only select $\mathbf{h}_j$ to participate the operation when $W_{i,j} = 1$. In other words, they replace $\mathbf{h}_j$ with the value that cannot affect the final results when $W_{i,j} = 0$.

We apply the conjunction and disjunction functions to the neurons in MLLP as follows:

$$\hat{\mathbf{r}}_i^{(l)} = \mathit{Conj}(\hat{\mathbf{s}}^{(l-1)}, \hat{W}_i^{(l)}),\ l \in \{1, 3, \dots, 2L-1\}, \tag{4}$$

$$\hat{\mathbf{s}}_i^{(l)} = \mathit{Disj}(\hat{\mathbf{r}}^{(l-1)}, \hat{W}_i^{(l)}),\ l \in \{2, 4, \dots, 2L\}. \tag{5}$$

In order to maintain the characteristics of conjunction and disjunction functions, we need to constrain all the weights of MLLP in the range $[0, 1]$. A common approach is to replace $\hat{W}_{i,j}^{(l)}$ with $sigmoid(\hat{W}_{i,j}^{(l)})$ or $\frac{1}{2}(tanh(\hat{W}_{i,j}^{(l)}) + 1)$. However, these constraint functions keep the weights from being exact 0 or 1, and cause the vanishing gradient problem, for which we will discuss the reason later. To overcome this problem, we propose to use *Clip* function to clip the weights after updating them with gradients, given as:

$$Clip(\hat{W}_{i,j}^{(l)}) = Max(0, Min(1, \hat{W}_{i,j}^{(l)})). \tag{6}$$

So far, MLLP has the ability to act exactly the same as the corresponding CRS when their weights are equal to the same discrete values, and MLLP is still differentiable.

Loss Function The MLLP is denoted by $\hat{\mathcal{F}}$ and parameterized by $\hat{\mathcal{W}}$ including all the weights in $\hat{\mathcal{F}}$. The loss function for training is given by:

$$Loss = \frac{1}{N} \sum_{i=1}^{N} \mathit{MSE}(Y'_i, \hat{\mathcal{F}}(X'_i, \hat{\mathcal{W}})) + \lambda \Omega(\hat{\mathcal{W}}), \tag{7}$$

where *MSE*$(\cdot)$ is Mean Squared Error (MSE) and $\Omega(\hat{\mathcal{W}})$ is the L2 regularization. The MSE criterion aims to minimize the gaps between the continuous output vector and one-hot label vector in each dimension, which could benefit the discrete CRS extraction. The L2 regularization encourages the MLLP to search for a CRS model with shorter rules.

Vanishing Gradient Problem MLLP suffers from vanishing gradient problem during training and fails to converge. There are two main reasons for this problem:

One reason is the constraint functions mentioned above. Let $\sigma(\cdot)$ denote these constraint functions. The derivative of $\sigma(\hat{W}^{(l)}_{i,j})$ is close to 0 when $\sigma(\hat{W}^{(l)}_{i,j})$ is close to 0 or 1. However, the weights of CRS that MLLP aims to learn are 0 or 1, which means derivatives close to 0 are inevitable. The *Clip* function we propose to use has no influence on the gradient and only clips the weights after weight updating. Therefore, the *Clip* function would not cause the vanishing gradient problem. We have found that using the *Clip* function instead of constraint functions is indeed beneficial for model convergence in practice.

The other reason for the vanishing gradient problem can be found by analyzing the partial derivative of each neuron w.r.t. its directly connected weights as follows:

$$\frac{\partial \hat{\mathbf{r}}^{(l)}_i}{\partial \hat{W}^{(l)}_{i,j}} = (\hat{\mathbf{s}}^{(l-1)}_j - 1) \cdot \prod_{k \neq j} F_c(\hat{\mathbf{s}}^{(l-1)}_k, \hat{W}^{(l)}_{i,k}), \tag{8}$$

$$\frac{\partial \hat{\mathbf{s}}^{(l)}_i}{\partial \hat{W}^{(l)}_{i,j}} = \hat{\mathbf{r}}^{(l-1)}_j \cdot \prod_{k \neq j} (1 - F_d(\hat{\mathbf{r}}^{(l-1)}_k, \hat{W}^{(l)}_{i,k})). \tag{9}$$

Due to the values of inputs and weights are in the range $[0, 1]$, the values of $F_c(\cdot)$ and $F_d(\cdot)$ in Equations 8 and 9 are in the range $[0, 1]$ as well. If $\mathbf{n}_l$ is large and most of the values of $F_c(\cdot)$ or $(1 - F_d(\cdot))$ are not close to 1, then the derivative is close to 0 because of the multiplications. As such, if we randomly initialize weights in range $[0, 1]$, the derivatives could be extremely small, especially when the size of model is very large. The heatmap of values about $F_c(h, w) \cdot (1 - F_d(h, w))$ is shown in Figure 2c. We can observe that if w is close to 0, then both $F_c(h, w)$ and $(1 - F_d(h, w))$ are colse to 1. Therefore, a simple but efficient solution is to initialize the weights with small values close to 0. Actually, we randomly initialize the weights with the distribution *Uniform*$(0, 0.1)$.

Discrete CRS Extraction After traning the MLLP, we need to extract the discrete CRS from it. A straightforward approach is to binarize all the weights using a threshold. The function for binarization could be:

$$Binarize(w, \mathcal{T}) = \mathbb{I}(w > \mathcal{T}), \tag{10}$$

where $\mathbb{I}(\cdot)$ is the 0-1 indicator function and $\mathcal{T} \in (0, 1)$ is the threshold for binarization. We usually set $\mathcal{T}$ to 0.5.

However, the behavior of extracted CRS is very different from MLLP, and the extracted CRS can hardly be used for classification, especially when MLLP is deep. This situation is also shown in the experimental results of Table 1. The reason for this problem is that the neurons do not play the role of conjunction and disjunction operators as we expected due to the real-valued weights.

To tackle this problem, we propose a new training method called Random Binarization (RB). When using the RB method during training, we randomly select a subset of weights in MLLP and binarize these weights. We fix these binary weights and only update other weights at each step. After several steps, we set these selected weights to their original values before binarization. Then we select a new subset of weights and repeat the above procedure. Let $M^{(l)}$ denote the mask matrix of $\hat{W}^{(l)}$, where $M^{(l)}_{i,j} \in \{0, 1\}$ and $M^{(l)}_{i,j} = \mathbb{I}(p < \mathcal{P})$ with $p \sim$ *Uniform*$(0, 1)$ and $\mathcal{P}$ as the rate of binarization. Let $\tilde{W}^{(l)}_{i,j}$ denote the weight after random binarization, which is given as:

$$\tilde{W}^{(l)}_{i,j} = \begin{cases} \hat{W}^{(l)}_{i,j} & M^{(l)}_{i,j} = 0, \\ Binarize(\hat{W}^{(l)}_{i,j}, \mathcal{T}) & M^{(l)}_{i,j} = 1. \end{cases} \tag{11}$$

$\hat{W}^{(l)}_i$ in Equation 4 and 5 is now replaced by $\tilde{W}^{(l)}_i$. To fix the values of selected weights, we compute the partial derivative of $\tilde{W}^{(l)}_{i,j}$ w.r.t. $\hat{W}^{(l)}_{i,j}$ by $\frac{\partial \tilde{W}^{(l)}_{i,j}}{\partial \hat{W}^{(l)}_{i,j}} = 1 - M^{(l)}_{i,j}$, and update the weights based on the following formula,

$$\frac{\partial Loss}{\partial \hat{W}^{(l)}_{i,j}} = \frac{\partial Loss}{\partial \tilde{W}^{(l)}_{i,j}} \cdot \frac{\partial \tilde{W}^{(l)}_{i,j}}{\partial \hat{W}^{(l)}_{i,j}} = \frac{\partial Loss}{\partial \tilde{W}^{(l)}_{i,j}} \cdot (1 - M^{(l)}_{i,j}). \tag{12}$$

We use $W^{(l)}_{i,j} = Binarize(\hat{W}^{(l)}_{i,j}, \mathcal{T})$ to extract a discrete CRS from the MLLP trained by the RB method. Thus the behaviors of CRS are almost the same as MLLP, which has a good classification performance.

Moreover, the RB method also works as the Dropout regularization (Srivastava et al. 2014) which significantly reduces overfitting. The reason is that those weights binarized to 0 could be considered to be removed from the model, which is similar to randomly dropping neurons (along with their connections) as the Dropout regularization does.

CRS Simplification

Benefiting from the transparent inner structure of CRS, we simplify CRS by analyzing each node. Specifically, we propose two methods to simplify the CRS model extracted from MLLP:

Dead Nodes Detection We name a node v in CRS as "dead node" if there exists no path from the top layer to the bottom layer that contains node v or inputting all the training data into CRS cannot activate node v. We can delete these dead nodes without affecting the performance of CRS.

A node v in CRS is activated when the value of v is 1 and inactivated when the value is 0. We are able to know whether a node is activated clearly, which is very difficult to distinguish the boundaries in common neural networks because of real-valued activation values.

Redundant Rules Elimination The redundant rules in CRS are useless, resulting the model to be more complex.

Table 1: 5-fold cross validated F1 score (Macro) of each comparing algorithm on 12 UCI data sets.

Dataset	**CRS**	MLLP($\mathcal{P}$=0)	CRS($\mathcal{P}$=0)	C4.5	CART	SBRL	LR	SVM	PLNN(MLP)	GBDT	RF(e=10)	RF(e=100)
adult	**80.95**	74.59	51.39	75.40	74.77	79.88	78.43	63.63	73.55	80.36	77.48	78.83
bank-marketing	73.34	63.38	46.88	71.24	70.21	72.67	69.81	66.78	72.40	**75.28**	69.89	72.01
banknote	94.93	93.29	88.68	98.45	97.85	94.44	98.82	**100.00**	**100.00**	99.48	99.11	99.19
blogger	**85.33**	**85.33**	20.00	75.90	78.27	67.64	55.55	62.11	56.24	67.58	77.33	85.17
chess	80.21	**80.49**	71.56	79.90	79.15	26.44	33.06	36.83	77.85	71.41	66.38	74.25
connect-4	**65.88**	56.35	26.71	61.66	61.24	48.54	49.87	50.17	64.55	64.45	61.95	62.72
letRecog	84.96	81.26	40.32	88.20	87.62	64.32	72.05	74.90	92.34	**96.51**	93.61	96.15
magic04	80.87	82.25	39.24	80.31	80.05	82.52	75.72	75.64	83.07	**86.67**	84.90	86.48
mushroom	**100.00**	**100.00**	34.40	**100.00**	99.98	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**	**100.00**
nursery	**99.69**	98.02	12.69	95.55	95.47	71.32	64.64	82.48	79.71	87.32	88.43	90.31
tic-tac-toe	**99.77**	**99.77**	38.06	91.70	94.21	98.39	98.12	98.07	98.26	99.19	94.85	98.37
wine	97.78	98.10	97.78	95.48	94.39	95.84	95.16	96.05	76.07	**98.44**	96.90	98.31
Average	**86.98**	84.40	47.31	84.48	84.44	75.17	74.27	75.56	81.17	85.56	84.24	86.82

Table 2: Data sets properties.

Dataset	#instances	#classes	#original features	#binary features
adult	32561	2	14	155
bank-marketing	45211	2	16	88
banknote	1372	2	4	17
blogger	100	2	5	15
chess	28056	18	6	40
connect-4	67557	3	42	126
letRecog	20000	26	16	155
magic04	19020	2	10	79
mushroom	8124	2	22	117
nursery	12960	5	8	27
tic-tac-toe	958	2	9	27
wine	178	3	13	37

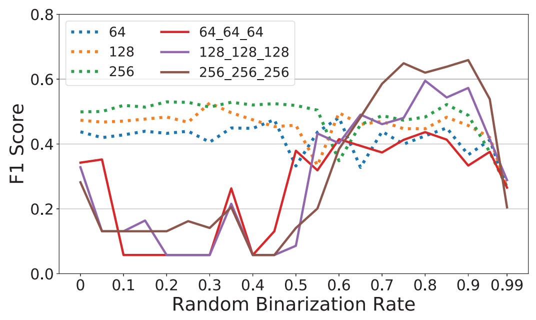


Figure 3: The impact of different binarization rates on CRS with different structures trained on Connect-4 data set.

Two typical examples of redundant rules are shown in Figure 1. $\mathbf{s}_1^{(2)} = \mathbf{r}_1^{(1)} \vee \mathbf{r}_3^{(1)}$ can be simplified as $\mathbf{s}_1^{(2)} = \mathbf{r}_1^{(1)}$ for the weights of $\mathbf{r}_1^{(1)}$ is the subset of the weights of $\mathbf{r}_3^{(1)}$. Similarly, $\mathbf{r}_2^{(3)} = \mathbf{s}_2^{(2)} \wedge \mathbf{s}_3^{(2)}$ can be simplified as $\mathbf{r}_2^{(3)} = \mathbf{s}_2^{(2)}$. We define the subset check function of the weights by

$$Subset(W_i^{(l)}, W_j^{(l)}) = \mathbb{I}(\forall W_{i,k}^{(l)} = 1, W_{j,k}^{(l)} = 1). \quad (13)$$

For $l \in \{2, 3, \ldots, 2L\}$, the set of redundant weights that can be simplified is:

$$\{W_{i,j}^{(l)} \mid \exists k, Subset(W_k^{(l-1)}, W_j^{(l-1)}) \wedge W_{i,k}^{(l)} = 1\}. \quad (14)$$

Experiments

In this section, we conduct experiments to evaluate the proposed method and answer the following questions:

1. How is the classification performance of CRS compared to other state-of-the-art models?
2. How does the binarization rate $\mathcal{P}$ affect CRS?
3. How is the model complexity of CRS?

Dataset Description

We took 12 datasets from the UCI machine learning repository (Dua and Graff 2017), all of which are often used to test classification performance and model transparency (Letham et al. 2015; Wang et al. 2017; Yang, Rudin, and Seltzer 2017; Hühn and Hüllermeier 2009). Table 2 summarizes the statistics of these 12 datasets. Together they show the data diversity, ranging from 100 to 67557 instances, from 2 to 26 classes, and from 4 to 42 original features.

Experimental Settings

Evaluation Protocols Considering that some of the data sets are imbalanced, i.e., the number of different classes are quite different, we adopt the F1 score (Macro) as the classification performance metric. To evaluate the classification performance of our model and baselines more fairly, 5-fold cross-validation is adopted to have a lower bias on experimental results. Additionally, 80% of the training set is used for training and 20% for validation when parameters tuning is required. The total length of all rules is a commonly used metric for model complexity of rule-based models. However, in some of the rule-based models, there are lots of reused structures, e.g., one branch in Decision Tree can correspond to several rules, and the total length of all rules may be not fair for these models. Considering the fact that edges in Decision Tree and CRS determine the final model structure, we use the total number of edges in the model to measure model complexity.

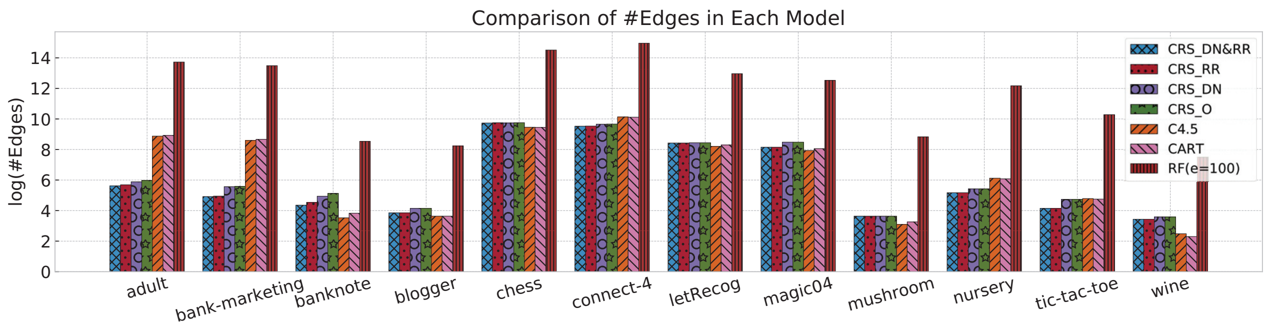


Figure 4: The number of edges in models trained on 12 UCI data sets. The logarithm is adopted for a better viewing experience.

Parameter Settings. We set the number of logical layers in CRS, i.e., $2L$, to 4. The number of nodes in each middle layer ranges from 32 to 256 depending on the number of binary features of the data set. The batch size is set to 128, and is trained for 400 epochs. We initialize the learning rate to 5×10^{-3} and decay it by a factor of 0.75 every 100 epochs. The weight decay is set to 10^{-8}. When using the RB method, we change the selected subset of weights after every epoch and tune the rate of binarization $\mathcal{P}$ using validation sets.

Experimental Results

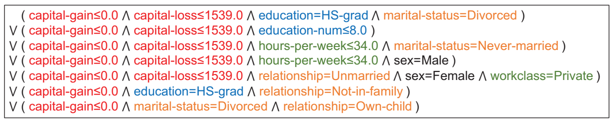
Classification Performance We first compare the classification F1 score (Macro) of CRS, MLLP trained without RB method (MLLP($\mathcal{P}$=0)), CRS extracted from MLLP trained without RB method (CRS($\mathcal{P}$=0)), C4.5 (Quinlan 1993), CART (Breiman 2017), Scalable Bayesian Rule Lists (SBRL) (Yang, Rudin, and Seltzer 2017), Logistic Regression (LR) (Kleinbaum et al. 2002), Piecewise Linear Neural Network (PLNN) (Chu et al. 2018), Support Vector Machines (SVM) (Scholkopf and Smola 2001) with linear or RBF kernel, Gradient Boosting Decision Tree (GBDT) (Ke et al. 2017), and Random Forest (Breiman 2001) with 10 estimators (RF(e=10)) and 100 estimators (RF(e=100)). C4.5, CART and SBRL are all rule-based models, and LR is a linear model. These four models are considered as transparent models and often used as surrogate models. PLNN is a Multilayer Perceptron (MLP) that adopts piecewise linear activation functions, e.g., ReLU (Nair and Hinton 2010). PLNN, SVM, GBDT, and Random Forest are considered as complex models.

The results are presented in Table 1. We can observe that the average F1 score of CRS on all the data sets is higher than those of other models and CRS outperforms other models on most of the data sets. Only one complex model, i.e., RF(e=100), has a comparable result. However, the Random Forest needs 100 estimators to obtain this result, which is hard to be considered as a transparent model. It should be noted that CRS performs not well on the banknote and letRecog data set. The reason is the recursive minimal entropy partitioning algorithm we applied to discretize continuous feature values brings bias to the data sets. Other models also do not perform well if we train them on these biased data sets. The requirement of data discretization before training is the point we need to improve in future work.

To verify the effect of the RB method, we compare CRS with MLLP trained without the RB method, i.e., MLLP($\mathcal{P}$=0), and the CRS extracted from it, i.e., CRS($\mathcal{P}$=0). We can see that CRS($\mathcal{P}$=0) performs poorly and acts quite differently from its corresponding MLLP, which demonstrates the necessity of the RB method. Moreover, we can see that CRS with RB method outperforms the MLLP trained without the RB method, the reason is the RB method works as the Dropout regularization which significantly alleviates overfitting.

Impact of Binarization Rate and CRS Structure To show the impact of varied random binarization rates $\mathcal{P}$ on CRS with different structures, we train three 3-layer CRS and three 5-layer CRS on the UCI Connect-4 data set using varied binarization rates for illustration. The results are shown in Figure 3, and legend labels show the number of nodes in each middle layer, e.g., 64_64_64 represents three middle layers and each middle layer has 64 nodes. We can see that without RB method ($\mathcal{P}$=0), deep CRS performs poorly compared to the shallow CRS, and neither shallow CRS nor deep CRS performs well. However, if we set $\mathcal{P}$ to an appropriate value, e.g., ranging from 0.7 to 0.9, the deep CRS can outperform shallow CRS and get a high F1 score. The RB method has no significant influence on the shallow CRS but is very important for the deep CRS. The RB method enables us to train a deeper CRS for higher classification performance. If we set $\mathcal{P}$ too small, the RB method cannot ensure MLLP to keep consistency with CRS. If we set $\mathcal{P}$ too close to 1.0, CRS can hardly be trained well for most of the weights are fixed. Moreover, for CRS with the same depth, the wider one performs better.

Model Complexity Considering that model complexity affects the model transparency, we compare the model complexity of CRS with the decision tree, i.e., C4.5 and CART, and Random Forest, RF(e=100). We also compare the model complexity of CRS after different simplification methods. The metric of model complexity is introduced in the Evaluation Protocols section. The results are shown in Figure 4. CRS_O represents the original extracted CRS without any simplification, CRS_DN represents CRS using the dead nodes detection method, CRS_RR represents CRS using the



```
  ( capital-gain≤0.0 ∧ capital-loss≤1539.0 ∧ education=HS-grad ∧ marital-status=Divorced )
∨ ( capital-gain≤0.0 ∧ capital-loss≤1539.0 ∧ education-num≤8.0 )
∨ ( capital-gain≤0.0 ∧ capital-loss≤1539.0 ∧ hours-per-week≤34.0 ∧ marital-status=Never-married )
∨ ( capital-gain≤0.0 ∧ capital-loss≤1539.0 ∧ hours-per-week≤34.0 ∧ sex=Male )
∨ ( capital-gain≤0.0 ∧ capital-loss≤1539.0 ∧ relationship=Unmarried ∧ sex=Female ∧ workclass=Private )
∨ ( capital-gain≤0.0 ∧ education=HS-grad ∧ relationship=Not-in-family )
∨ ( capital-gain≤0.0 ∧ marital-status=Divorced ∧ relationship=Own-child )
```

Figure 5: One rule set in CRS trained on the Adult data set.

redundant rules elimination method, CRS_DN&RR represents CRS using both simplification methods. We can observe that the simplification methods we proposed can reduce the model complexity of CRS on most of the data sets. By comparing CRS_DN&RR with C4.5 and CART, we can see that CRS trained on adult and bank-marketing data sets have a lower model complexity than C4.5 and CART. On other data sets, there is no significant difference, which verifies the model complexity of CRS is close to decision tree while CRS has a better classification performance. Moreover, Random Forest needs 100 estimators to obtain high classification performance, which leads to extremely high model complexity.

Case Study on Inner Structure of CRS To show the inner structure of CRS, we select one rule set from the CRS trained on the Adult data set as an example. The rule set shown in Figure 5 corresponds to one CRS node used for predicting whether the income of someone is below $50K/yr based on census data. Different types of features are marked in different colors, e.g., red for features related to capital. We can clearly see that lacking capital behavior, not being married, short work hours per week and low educational background may lead to low income. Each node in CRS corresponds to one rule or rule set like this example, and we can analyze these nodes to interpret the behavior of the whole model structure.

Conclusion and Future Work

We propose a new hierarchical rule-based model called Concept Rule Sets and its continuous version, Multilayer Logical Perceptron. We can use gradient descent to learn the discrete CRS efficiently via the continuous MLLP and Random Binarization method we proposed. Our experimental results show that CRS enjoys both high classification performance and low model complexity. As future work, we plan to eliminate the need for data discretization before training and explore the application of our method for unstructured data.

Acknowledgments

The authors would like to thank the anonymous area chair and (senior) PCs for their valuable comments and suggestions. This work is supported by National Natural Science Foundation of China under Grant No. 61532010, 61521002, and 61702190, and Beijing Academy of Artificial Intelligence (BAAI).

References

Angelino, E.; Larus-Stone, N.; Alabi, D.; Seltzer, M.; and Rudin, C. 2017. Learning certifiably optimal rule lists. In *SIGKDD*, 35–44. ACM.

Breiman, L. 2001. Random forests. *Machine learning* 45(1):5–32.

Breiman, L. 2017. *Classification and regression trees*. Routledge.

Chu, L.; Hu, X.; Hu, J.; Wang, L.; and Pei, J. 2018. Exact and consistent interpretation for piecewise linear neural networks: A closed form solution. In *SIGKDD*, 1244–1253. ACM.

Cohen, W. W. 1995. Fast effective rule induction. In *MLP*. Elsevier. 115–123.

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *NeurIPS*, 3123–3131.

Courbariaux, M.; Hubara, I.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*.

Doshi-Velez, F., and Kim, B. 2017. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*.

Dougherty, J.; Kohavi, R.; and Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. In *MLP*. Elsevier. 194–202.

Dua, D., and Graff, C. 2017. UCI machine learning repository.

Frosst, N., and Hinton, G. 2017. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*.

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Goodman, B., and Flaxman, S. 2016. European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hühn, J., and Hüllermeier, E. 2009. Furia: an algorithm for unordered fuzzy rule induction. *DMKD* 19(3):293–319.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 3146–3154.

Kim, B., and Doshi-Velez, F. 2017. Interpretable machine learning: the fuss, the concrete and the questions. *ICML*.

Kleinbaum, D. G.; Dietz, K.; Gail, M.; Klein, M.; and Klein, M. 2002. *Logistic regression*. Springer.

Lakkaraju, H.; Bach, S. H.; and Leskovec, J. 2016. Interpretable decision sets: A joint framework for description and prediction. In *SIGKDD*, 1675–1684. ACM.

Letham, B.; Rudin, C.; McCormick, T. H.; Madigan, D.; et al. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics* 9(3):1350–1371.

Lipton, Z. C. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Mahendran, A., and Vedaldi, A. 2015. Understanding deep image representations by inverting them. In *CVPR*, 5188–5196.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814.

Payani, A., and Fekri, F. 2019. Learning algorithms via neural logic networks. *arXiv preprint arXiv:1904.01554*.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *SIGKDD*, 1135–1144. ACM.

Scholkopf, B., and Smola, A. J. 2001. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D.; et al. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 618–626.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Wang, T.; Rudin, C.; Doshi-Velez, F.; Liu, Y.; Klampfl, E.; and MacNeille, P. 2017. A bayesian framework for learning rule sets for interpretable classification. *JMLR* 18(1):2357–2393.

Yang, H.; Rudin, C.; and Seltzer, M. 2017. Scalable bayesian rule lists. In *ICML*, 3921–3930. JMLR. org.

Yosinski, J.; Clune, J.; Nguyen, A.; Fuchs, T.; and Lipson, H. 2015. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*.

Zhang, Q.; Yang, Y.; Wu, Y. N.; and Zhu, S.-C. 2018. Interpreting cnns via decision trees. *arXiv preprint arXiv:1802.00121*.

Zhang, Q.; Wu, Y. N.; and Zhu, S.-C. 2017. Interpretable convolutional neural networks. *arXiv preprint arXiv:1710.00935* 2(3):5.

Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A. 2016. Learning deep features for discriminative localization. In *CVPR*, 2921–2929.

Zhou, B.; Bau, D.; Oliva, A.; and Torralba, A. 2018. Interpreting deep visual representations via network dissection. *TPAMI*.