Towards Interpretable Deep Neural Networks: An Exact Transformation to Multi-Class Multivariate Decision Trees

Tung D. Nguyen[a,*], Kathryn E. Kasmarik[a], Hussein A. Abbass[a]

[a]*Trusted Autonomy Laboratory, School of Engineering and Information Technology, University of New South Wales - Canberra, Canberra 2600, Australia*

Abstract

Deep neural networks (DNNs) are commonly labelled as black-boxes lacking interpretability; thus, hindering human's understanding of DNNs' behaviors. A need exists to generate a meaningful sequential logic for the production of a specific output. Decision trees exhibit better interpretability and expressive power due to their representation language and the existence of efficient algorithms to generate rules. Growing a decision tree based on the available data could produce larger than necessary trees or trees that do not generalise well. In this paper, we introduce two novel multivariate decision tree (MDT) algorithms for rule extraction from a DNN: an Exact-Convertible Decision Tree (EC-DT) and a Deep C-Net algorithm to transform a neural network with Rectified Linear Unit activation functions into a representative tree which can be used to extract multivariate rules for reasoning. While the EC-DT translates the DNN in a layer-wise manner to represent exactly the decision boundaries implicitly learned by the hidden layers of the network, the Deep C-Net inherits the decompositional approach from EC-DT and combines with a C5 tree learning algorithm to construct the decision rules. The results suggest that while EC-DT is superior in preserving the structure and the accuracy of DNN, C-Net generates the most compact and highly effective trees from DNN. Both proposed MDT algorithms generate rules including combinations of multiple attributes for precise interpretation of decision-making processes.

*Corresponding author

Email address: tung.nguyen@student.adfa.edu.au (Tung D. Nguyen)

Keywords: Explainable Artificial Intelligence, Rule Extraction, Multivariate Decision Tree, Deep Neural Network.

1. Introduction

Symbolic AI technologies adopt interpretable and explainable representation languages with sufficient expressive power for a human to understand the system's behaviours (Shortliffe & Buchanan, 1984; Johnson, 1994; Swartout et al., 1991; Van Lent et al., 2004). Nevertheless, symbolic systems that rely on deductive logic lack the ability to adapt to changes in the environment and context, require excessive knowledge of the problem domain in advance, and normally end up with a hard-to-maintain knowledge base.

Inductive learning affords a machine the ability to form and update its own internal representation using experiences. Data labelled with ground truth, user feedback or self-guidance using predesigned similarity metrics guide the three basic forms of learning: supervised, reinforcement and unsupervised, respectively. Early machine learning and statistical inferencing algorithms required significant feature engineering efforts by the human designer. This effort substantially decreased with the arrival of deep learning algorithms (Wen et al., 2016; Wu et al., 2016; Salakhutdinov et al., 2013). With the possibility for implementing a deep neural network (DNN) on the hardware, and ability to leverage the vectorisation available in graphics processing units (GPUs) in software, DNNs offer significant advantages when it comes to speed of learning and ability to handle big data. Recent advances in black-box models such as DNN have shown significant abilities to learn and even surpass human-level performance in some tasks (Mnih et al., 2015; Churchland et al., 2016; Mnih et al., 2016). One of the major disadvantages normally associated with DNNs is their lack of interpretability; they are labelled as black-boxes. The highly distributed nature of their implicit capture of knowledge and their ability to approximate highly non-linear functions using many local units and different layers of transformations, complicated the interpretability of DNN. In the absence of appropriate representations to interpret what a DNN learns, social integration, human acceptance, verification against requirements or previous knowledge bases, and performance assurance are some of the main challenges to the use of to DNN in practical applications, especially in safety-critical environments. Addressing the drawback of these opaque systems to enable seamless and safe interactions between humans and machines can be attained by developing new technologies for explainable models without sacrificing the performance of the AI models.

Interpretability is required throughout the learning process. For a model that has not learnt the task well-enough, interpretability could shed light on which part of the model is lagging behind. For a model that generalises well, interpretability sheds light on the rationale behind the model's behaviour. Among the techniques to interpret the hidden layers of the neural networks, decision trees (Tsujino & Nishida, 1995; Craven, 1996; Schmitz et al., 1999; Hayashi et al., 2010; Chakraborty et al., 2019) have been used to learn the relationship between the input and output of a learned neural network, which provides a means to extract and represent the implicit knowledge in the network in an interpretable form.

In this paper, we propose two multivariate decision tree algorithms called Deep C-Net and EC-DT algorithms. Deep C-Net learns the relationship between the last hidden layer and the outputs, then infers the input-output relationships by back-projections guided by weights from DNNs. EC-DT constructs the rule sets layer-wise based on the activation of the hidden nodes to find an exact representation of DNNs in tree forms. The performance of the DNN is used as the baseline, together with a direct use of the C5 decision tree algorithms. We use three performance indicators: task performance (accuracy or error of classification), rule compactness (model size), and interpretability (model interpretability).

The contributions of the paper are three-fold. First, the paper introduces novel multivariate tree algorithms as decompositional rule extraction techniques for inspecting the DNNs with a focus on continuous input and Rectified Linear Function (ReLU) activation function. Secondly, a deep investigation is conducted on the relation between rule set compactness and complexity, and the complexity of the data space to address the question of in which situations a simple black-box (pedagogical) rule extraction technique are more favored taking into account the compromise between the generalization capability and transparency. Last but not least, the paper also discusses some methods to transform the decision rules into a more interpretable type of representation that bridges the gap between the mathematical interpretation into common-sense explanation.

The organization of this paper is as follows. Section 2 reviews the literature including previous decision tree approaches and multivariate decision trees (MDTs) used to extract rules from ANNs. Section 3 introduces the EC-DT algorithm, which can convert a DNN to representative rules which preserve 100% performance of the DNN. Section 4 describes the Deep C-Net algorithm to learn the decision rules from deep neural network models with back-projection techniques. Experimental setup and the set of metrics we use to assess the performance and the interpretability of our approach are then presented in Section 5. We then discuss our results and corresponding analysis in Section 6 and conclude the paper

in Section 7.

2. Background

2.1. Artificial Neural Network

An artificial neural network (ANN) is a structure that contains multiple computing units, often arranged in layers, with various connecting configuration among them. Each computing unit is associated with an activation function for linear or non-linear mapping of its input to output. ANN is a function approximator that can transform input data into desired output by the units' properties and the weights associated with the interconnections among them.

Here we introduce some notations of a trained densely feed-forward ANN with K hidden layers:

- I and M are the number of input and output units respectively.
- K is the number of hidden layers.
- $J_1, J_2, ..., J_k, ..., J_K$ are the number of units in each hidden layer.
- $\mathbf{X^t} = (x_1^t, x_2^t, ..., x_I^t)$ is the input fed to the input layer of the network.
- $Y^t \in \mathbf{Y}$ is the output class of the corresponding input.
- $w_{ij_1}, w_{j_1j_2}, ...w_{j_Km}$ are the weights of the links from input unit i to hidden unit j_1 at hidden layer 1, hidden unit j_1 at hidden layer 1 to hidden unit j_2 at hidden layer 2,..., and hidden unit j_K to output unit m respectively.
- $H_{j_1}^1(\mathbf{X^t}) = \sigma_{j_1}^1(\sum_{i=1}^{I} w_{ij_1}x_i^t + \beta_{j_1}^1), j = 1...J_1$ is the output of the hidden unit j_1 at hidden layer 1. $\sigma(.)$ is the activation function.
- $H_{j_k}^k(\mathbf{X^t}) = \sigma_{j_k}^k(\sum_{j_{k-1}=1}^{J_{k-1}} w_{j_{k-1}j_k}H_{j_{k-1}}^{k-1}(\mathbf{X^t}) + \beta_{j_k}^k)$, $j_k = 1...J_k$ is the output of the hidden unit j_k at hidden layer k, where $1 < k < K$.
- $Y_m(\mathbf{X^t}) = \sigma_m(\sum_{j_K=1}^{J_K} w_{j_Km}H_{j_K}^K(\mathbf{X^t}) + \beta_{j_m}^Y)$ is the output of unit m in the output layer of the neural network.
- $\mathbf{O}^t$ is the class output of the neural network corresponding to the pattern of neural network output $\mathbf{Y}(\mathbf{X^t}) = \{Y_1(\mathbf{X^t}, ...Y_K(\mathbf{X^t}\}$.

The ANN can be trained with a back-propagation algorithm, a learning process that modifies the weights of the ANN according to the errors between the outputs of output neurons and the target values, and back-propagated errors to previous layers. The weights of the ANN is refined until a satisfactory level of performance is achieved.

2.2. Explainable AI for Deep Learning Models

ANNs, and DNNs in particular, have demonstrated a significant social impact due to their universal function approximation properties, robustness, very large scale implementation characteristic, generalization abilities, and success in many applications. However, due to their black-box nature, they are not as widely acceptable by humans when compared to classic rule-based systems that rely on symbolic representations (Saad & Wunsch II, 2007; Alexander & Mozer, 1999). For opaque models like DNNs, there is a need to explain their decision-making processes, to be transparent without sacrificing their predictive power.

Currently, the explainable artificial intelligence (xAI) domain calls for solutions to overcome the opaqueness of ANNs to improve their reliability and trustworthiness when they are used in decision support engines and expert systems. Approaches in xAI might alleviate the problems of knowledge extraction in these black-box systems, provide the systems with explanation and reasoning abilities, facilitate the verification and validation of the model, and inspect and diagnose the sources of erroneous interferences (Andrews et al., 1995; Taha & Ghosh, 1999). These abilities could enhance the utility, transparency, and explainability of DNN in safety critical applications (Gunning, 2017; Adadi & Berrada, 2018; Samek, 2019).

Recent literature for explaining deep neural networks focuses on two approaches.

- The first approach relies on two forms of visualization.
  - The first form visualizes correlated information to the predictions. It produces saliency maps from the activation convolutional layers of the Convolutional Neural Networks (CNNs) whose activated areas serve as the regions with highest correlation to the models' decisions or the objects that need to be identified in the input images (Yu et al., 2012; Gan et al., 2015; Zeiler & Fergus, 2014; Selvaraju et al., 2017; Chattopadhay et al., 2018). The advantage of those methods is to provide an understanding of how different convolutional layers' features are formed in response to the input images. Despite the fact that this

technique might be beneficial for domain experts, visualization of a saliency map alone does not provide reasons or sound explanations in a form that could be understood by a wide variety of users.

  - In the second form, the functionalities of the deep networks are modified with new structures to generate the description. An explanation structure, such as a Recurrent Neural Network (RNN) for language generation, is combined with the deep network to translate the features of the operating network into understandable visual explanation.

    Hendricks et al. (2016) advance a visual explanation model from captioning models to take into account both image-relevant and class-relevant properties. A fine-grained CNN is used to identify components in a given images and link to appropriate linguistic terms while multiple Long-Short Term Memory (LSTM) layers generate a sequence of words that formulates all details into an explanation. The explanations provided by this type of models are much more intuitive due to the existence of both visual marking and explanatory language. Both variations of visual explanation techniques are model-specific and application-specific.

    These techniques are currently most suited for machine vision applications such as image classification and object detection.

- The second approach extracts the rules governing the mapping between the inputs and outputs of the models without modifying the original operation of the networks (Ribeiro et al., 2016). These rule extraction algorithms are flexible and some are general and can apply to any black-box model. They generate rules that can be translated easily into a human-understandable language. Extracting knowledge from neural networks has been practiced in many applications such as classifying products in industries (Amin, 2013) as well as business analysis (Hayashi et al., 2010). We will expand on these algorithms below.

2.3. Rule Extraction from Artificial Neural Networks

The usefulness of a rule extraction algorithm depends on multiple criteria. Taha & Ghosh (1999) listed different aspects that need to be considered when designing a rule extraction system from ANNs including:

- **Level of detail**: The presentation of information in the explanation based on hypotheses of the system.

- **Comprehensiveness:** The fidelity of rules to represent the knowledge within the black-box system.
- **Comprehensibility:** The property of rule set to identify knowledge of a model's processes. It can be represented by the number of extracted rules and premises in each rule expression.
- **Transparency:** The ability of the rules to be easily inspected in order to inform explanations or decisions.
- **Generalization:** The performance of the extracted rule set on new, unprecedentedly observed data.
- **Mobility:** The ability to apply the rule extraction algorithm to different network architectures.
- **Adaptability:** The ability to modify the set of extracted rules when the networks are updated after a further training session.
- **Theory refinement:** The ability to overcome restrictions due to missing data or inaccurate domain knowledge.
- **Robustness:** The insensitivity of the extracted rules to the noise in the training data.
- **Computational complexity:** Demand of computational resources for extracting the set of rules and the execution of inference relative to the size and number of attributes in the dataset.
- **Scalability**: The ability of the rule extraction algorithm to scale corresponding to the change of problem complexity and network structure.

These criteria serve as a guideline to design evaluation metrics of rule extraction models. However, some of these criteria are in conflict with each other and require a trade-off. A very comprehensible model such as a nonlinear mapping may not be transparent enough to a group of users. In this paper, the criteria of *comprehensiveness*, *comprehensibility*, *transparency*, *generalization*, and *adaptability* with different rule extraction techniques are investigated.

Various categories of rule extraction algorithms are reported in the literature. One of the earliest classification frameworks was based on different features that rule extraction methods exhibit including: (1) expressive power of the extracted

rules, (2) transparency of the rule extraction method, (3) usage of specialized training scheme, (4) quality of the extracted rules, and (5) computational complexity of the techniques (Andrews et al., 1995). However, this categorization system seems complex due to the overlap between its elements, such as the strong dependency between expressive power and quality of extracted rules. Another taxonomy, called *Input-Network-Training-Output-Extraction-Knowledge*, divided clearly the techniques based on modules of the classification frameworks (Gupta et al., 1999). Following this type of taxonomy, it is simpler for system designers to select or design the rule extraction algorithms with clear requirements for each component.

Some other taxonomies include *Fuzzy Rule Extraction* such as NeuroFuzzy networks and *Recurrent Network Rule Extraction* algorithms (Taha & Ghosh, 1999; Saad & Wunsch II, 2007) which are specialized for those types of networks and are not the focus of this paper.

In this paper, we will follow Hruschka & Ebecken (2006)'s classification due to its popularity and wide acceptance. They divide rule extraction techniques into three categories: **pedagogical (black-box rule extraction)**, **decompositional (link rule extraction)**, and **eclectic (hybrid)** techniques.

Pedagogical approaches are data-driven and only find the direct mapping between the inputs and outputs of the ANN using some machine learning techniques. This set of methods does not reach inside the black-box to find the real links within the neural networks. Quinlan's C4.5 (Quinlan, 1987) is one of the most popular algorithms for building a tree representation that utilizes a discrimination process over different data attributes to maximize the information gain ratio. The C4.5 decision tree is commonly used for extracting rules from neural networks. Decision Detection by Rule Extraction (DEDEC) is another rule extraction technique that ranks the input attributes of the input data relative to the outputs of the ANN (Tickle et al., 1996). These rankings are then used to cluster the input space and produce a set of binary rules describing the relationships between data attributes in each cluster and the ANN's outputs. Other notable examples of black-box rule extraction approaches are BRAINNE (Sestito, 1992), Rule-extraction-as-learning (Craven & Shavlik, 1994), TREPAN (Craven & Shavlik, 1996) and BIO-RE (Taha & Ghosh, 1999). The strengths of these approaches are they offer a fast, simple rule set with high transparency and scalability. However, the extracted rules might not be comprehensive and or able to generalize to the test data in various domains.

Decompositional rule extraction techniques consider the links between layers of an ANN (the weights and activation at each hidden and output nodes) to com-

pose the rules. These approaches generally describe more accurately the input-output relationship than *pedagogical* approaches. Most techniques in this class consist of two main stages including searching for the weighted sum of the input links for the activation of each hidden node and then producing a rule with inputs as premises. Typical examples of this class of techniques are SUBSET (Towell & Shavlik, 1993), KT (Fu, 1994), NeuroRule (Setiono & Liu, 1996), NeuroLinear (Setiono & Liu, 1997), rule extraction by successive regularization (Ishikawa, 2000), and Greedy Rule Generation (Odajima et al., 2008). Towell & Shavlik (1993) developed a well-known rule-extraction algorithm called MofN that can address the limits of SUBSET in terms of binary inputs, scalability, and repetition of extracted rules and achieve higher fidelity compared to some other black-box and link rule extraction methods. RuleNet (McMillan et al., 1991) and RULEX (Andrews & Geva, 2002) are decompositional techniques that were specialized for ANNs with localized hidden units. While RULEX extracts rules from a Constrained Error Back-Propagation (CEBP) network whose hidden nodes are localized in a bounded area of the training samples, RuleNet extracts binary rules from a mixture of experts trained on a localized ANN. The extracted rules using these approaches are much more comprehensive, but complicated if the number of attributes or input nodes of the ANN is large. The decompositional approaches face some challenges in the transparency, computational complexity, and scalability when applying for large networks.

In this paper, we compare the generalization capability, comprehensiveness, and transparency of three rule extraction methods. C5.0 decision trees (Pedagogical) is selected as a baseline method. We proposed an algorithm to exactly convert a neural network into a decision tree (Decompositional) and a modified C-Net algorithm, an *eclectic/hybrid* method combining the strengths of both methods above.

2.4. Multi-variate Decision Trees

Decision Trees are interpretable representations able to approximate the underlying functions that the ANNs represent. Using Decision Trees (Boz, 2002; Johansson & Niklasson, 2009) to approximate the input-output relationship of a neural network is a popular practice because of the ease when converting a decision tree to a set of simple rules.

Univariate decision trees are limited to produce a set of rules each composed of multiple constraints considering a single data attribute at a time. They rely on axis-parallel decision hyperplanes to approximate the decision boundary, which

results in a huge model when the decision boundary is oblique and/or highly non-linear (Brodley & Utgoff, 1995). These drawbacks limit their capability to produce succinct explanations.

A multivariate decision tree can generate a rule expression in terms of a combination of multiple data attributes as inputs. OC1 (Murthy et al., 1994) is one of the earliest to build a multivariate decision tree. OC1 searches for optimal set of weights on all data attributes and tries different weighted sums of input attributes to find the best local decision boundaries. Sok et al. (2016) modify a univariate alternating decision tree algorithm into a multivariate one by proposing three approaches to weight multiple attributes and replace the base of univariate conditions by combinations of multivariate ones. The resultant trees demonstrate significantly better accuracy while maintaining acceptable comprehensiveness in comparison with their univariate counterparts and ensembles of univariate decision trees. A PCA-partitioned multivariate decision trees algorithm is proposed to solve the multi-label classification problems in large scale datasets (Wang et al., 2018). The multivariate trees, constructed by using the maximum eigenvalue of PCA to choose the weights of each variable in combination, produce a high accuracy.

In the next sections, we describe our proposed multivariate decision trees built with the decompositional rule extraction and hybrid techniques to address the problems of binary and multi-class classification problems with a focus on continuous input data.

3. Conversion of a Deep FeedForward Network to a Multivariate Decision Tree Using EC-DT Algorithm

In this section, we propose a multivariate tree algorithm that transforms the DNN structure into an equivalent decision tree. The algorithm is designed to be complete and sound; that is, it preserves 100% performance of DNNs by maintaining the generalizability of the network while providing an interpretable representation. This method is a *decompositional extraction* approach.

In our study, the activation functions of hidden layers are Rectified Linear Function (ReLU). The ReLU function has the form:

$$\sigma(f) = \begin{cases} 0, & \text{for } f \leq 0 \\ f, & \text{for } f > 0 \end{cases} \tag{1}$$

For ReLU, a node is considered activated if its output is greater than 0. We might construct a decision tree to represent the constraint by which the activation

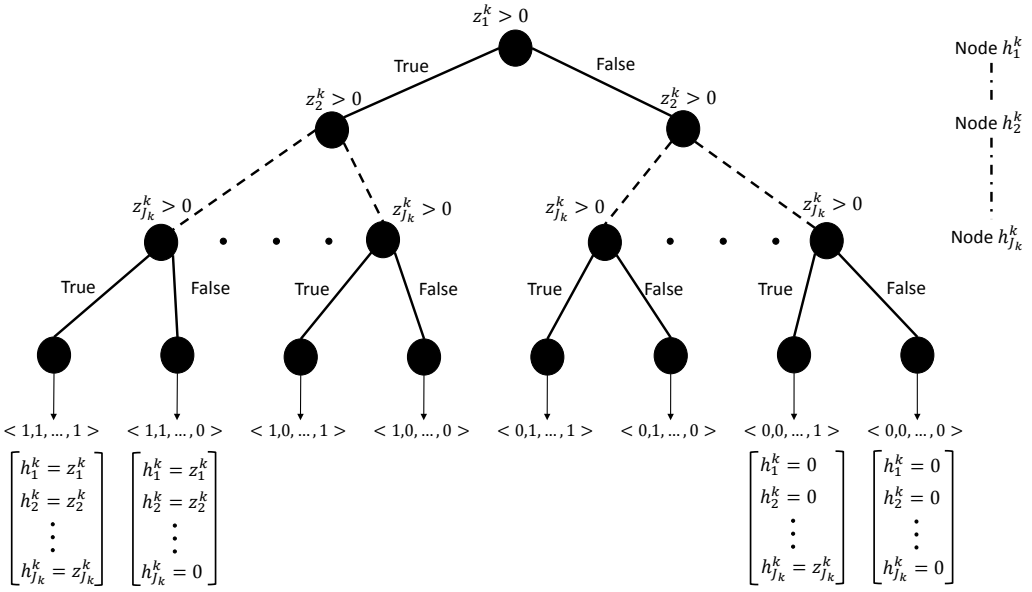


Figure 1: Binary EC-DT representing the output of a hidden layer with ReLU activation function.

of each node in the hidden layers is decided. Denote z_{j_k} as the value of hidden node j_k in hidden layer k before applying the activation function, that is

$$z_{j_k}^k = \sum_{j_{k-1}=1}^{J_{k-1}} w_{j_{k-1}j_k} H_{j_{k-1}}^{k-1} + \beta_{j_k}^k \tag{2}$$

For nodes in the first hidden layer, we have alternatively

$$z_{j_1}^1 = \sum_{i=1}^{I} w_{ij_1} x_i + \beta_{j_1}^1 \tag{3}$$

Therefore, we can represent the overall neural network by a binary tree with $2^{\sum_{l=1} KJ_l}$ leaves for a deep network with K hidden layers. The size of the tree grows exponentially as the network size grows; a drawback we will fix below.

Due to the nature of ReLU activation as described in Equation 1, the output of this activation function is either 0 or a weighted sum of the input to the neural node, which we called excitepd or activated node. Thus, each hidden layer of the neural network can be transformed into a binary decision tree with each tree stage deciding the activation of the corresponding node in the hidden layer based on the

constraint of whether or not the value before the activation function is greater than 0.

Given a hidden layer of J_K nodes with ReLU activation, the direct translation of this layer to a binary tree can be illustrated in Figure 1. The tree can be considered a multi-target tree with the output being a binary vector of the active nodes (value of 1) and disabled node (value of 0). In the case that a node is activated (satisfying constraint for TRUE branch), the real output of the node after activation is the same as its value prior activation: $h^k_{J_k} = z^k_{J_k}$. That is, we can alternatively replace this activation array by a vector of regression values for each node based on weights.

Algorithm 1 illustrates the steps to build an EC-DT from a Deep Feedforward Network with ReLU activation function. Each leaf node in the produced tree has a list of true or false branches that starts from the root node all the way to the leaf. As a result, we can produce a list of DNN layers' activations $S = \{S^1, S^2, ...S^K\}$, where S^k is an array representing the activations of hidden nodes in layer k of the DNN. The set of rules can be extracted from the EC-DT tree by converting every leaf node in the tree into constraints given list of hidden nodes'activations, the weights matrices, and biases matrices from DNN (see Algorithm 2). The values of weights and biases of disabled nodes do not contribute to the constraints of the next neural layer. Equivalently, we set the weights of connections from the disable nodes of current layer to zero and then compute the updated weights and biases matrices representing the linear combination between the input variables and the outputs value of next neural layer.

We present a multiplexer problem with binary input as an example to illustrate how our proposed EC-DT algorithm can convert a neural network into a decision tree. Figure 2 introduces a gate that takes two binary inputs X_1 and X_2 and produces a function $Y = XOR(X_1, X2)$. Assume that we have a neural network with weights as shown in Figure 2 and zero biases to represent this function. The EC-DT algorithm converts the neural network into a binary tree. Each layer in the tree includes nodes which represent the constraint for the test of whether a corresponding unit in the DNN is activated or not. The paths from root to leaf represent combinations of activation of hidden units. This results in four cases (with one impossible case) of output value Y. The constraints in each layer and the consequences at the leaves form the rule set of the neural network.

Algorithm 1: EC-DT Tree Generation Algorithm

Input : Number of nodes in hidden layers $\{J_1, J_2, ..., J_K\}$
Output: Decision tree's set of nodes $\mathcal{T}$
Initialize: $\mathcal{T} \leftarrow \varnothing$, current set of parent nodes $\mathcal{P} \leftarrow \varnothing$, current set of child nodes $\mathcal{C} \leftarrow \varnothing$, node tuple $q^{ID_\tau}(id, hidden\_layer, hidden\_node\_id, parent\_id, branch, leaf)$
Set tree node id $ID_\tau = 1$
Set hidden layer $k = 1$.
Set hidden node id $ID_k = 1$.
Create root node $q^r(ID_\tau, k, ID_k, None, None, leaf = False)$.
Add q^r to $\mathcal{T}$, add $q^r.id$ to $\mathcal{P}$.
$ID_\tau \leftarrow 2$; $ID_k \leftarrow 2$.
while $k \leq K$ **do**
  while $ID_k \leq J_k$ **do**
    Set $\mathcal{E}$ the number of elements in $\mathcal{P}$.
    for $i = 1$ **to** $\mathcal{E}$ **do**
      if $k = K$ *and* $ID_k = J_K$ **then**
        // leaves nodes
        Create node ; // true branch
        $q_1^{ID_\tau}(ID_\tau, k, ID_k, ID_{p_i \in \mathcal{P}}, 1, True)$
        Create node ; // false branch
        $q_0^{ID_\tau}(ID_\tau, k, ID_k, ID_{p_i \in \mathcal{P}}, 0, True)$
        Find set of branches $S_1^{ID_\tau}$ and $S_0^{ID_\tau}$ leading to $q_1^{ID_\tau}$ and $q_0^{ID_\tau}$ by tracing back to root.
        $q_1^{ID_\tau}.value \leftarrow S_1^{ID_\tau}$; $q_0^{ID_\tau}.value \leftarrow S_0^{ID_\tau}$
        $ID_\tau \leftarrow ID_\tau + 1$
      else
        Create node ; // true branch
        $q_1^{ID_\tau}(ID_\tau, k, ID_k, ID_{p_i \in \mathcal{P}}, 1, False)$
        Create node ; // false branch
        $q_0^{ID_\tau}(ID_\tau, k, ID_k, ID_{p_i \in \mathcal{P}}, 0, False)$
        $ID_\tau \leftarrow ID_\tau + 1$
      end
      Add nodes to $\mathcal{T}$.
      Add nodes IDs to $\mathcal{C}$.
    end
    $\mathcal{P} \leftarrow \mathcal{C}$; $\mathcal{C} \leftarrow \varnothing$.
    $ID_k \leftarrow ID_k + 1$.
  end
  $k \leftarrow k + 1$.
end
return $\mathcal{T}$.

Algorithm 2: EC-DT Rule Extraction Algorithm (for a leaf)

Input : A leaf node with its list of branches corresponding to activations of DNN'hidden layers $S = \{S^1, S^2, ...S^K\}$, number of nodes in hidden layers $\{J_1, J_2, ..., J_K\}$, a set of weights matrices of DNN $\mathcal{W} = \{\mathcal{W}^{I1}, \mathcal{W}^{12}, ..., \mathcal{W}^{(K-1)K}, \mathcal{W}^{KY}\}$, and a set of biases matrices of DNN $\mathcal{B} = \{\mathcal{B}^1, \mathcal{B}^2, ..., \mathcal{B}^K, \mathcal{B}^Y\}$

Output: A rule/set of constraints and consequences $\mathcal{R}$

Set $\mathcal{R} \leftarrow \varnothing$.

for $k = 1$ **to** K **do**

  for $s = 1$ **to** J_k **do**

    if $S^k(s) = 0$ **then**

      Convert matrix form $X\mathcal{W}^{Ik}_{*,s} + \mathcal{B}^{Ik}(s) > 0$ into linear inequation form.

      Elements in s^{th} row of $\mathcal{W}^{k(k+1)}$ are set to 0.

    else

      Convert matrix form $X\mathcal{W}^{Ik}_{*,s} + \mathcal{B}^{Ik}(s) \leq 0$ into linear inequation form.

    end

    Add inequation to $\mathcal{R}$ as constraint.

  end

  if $k = K$ **then**

    Compute $\mathcal{W}^{IY} = \mathcal{W}^{IK}\mathcal{W}^{KY}$.

    Compute $\mathcal{B}^{IY} = \mathcal{B}^{IK}\mathcal{W}^{KY} + \mathcal{B}^{Y}$.

  else

    Compute $\mathcal{W}^{I(k+1)} = \mathcal{W}^{Ik}\mathcal{W}^{(k(k+1)}$.

    Compute $\mathcal{B}^{I(k+1)} = \mathcal{B}^{Ik}\mathcal{W}^{k(k+1)} + \mathcal{B}^{k+1}$.

  end

end

Convert matrix form $\mathcal{Y} = X\mathcal{W}^{IY} + \mathcal{B}^{IY}$ to equations.

Add equations to $\mathcal{R}$ as consequences.

return $\mathcal{R}$.

4. C-Net: An approach to learn multivariate decision trees from a neural network

Decision trees have been used with neural networks to build an interpretable model that explains the decision-making process of neural-networks. The C-Net algorithm, proposed by Abbass et al. (2001), is one of those early algorithms which uses a univariate decision tree (UDT) to generate a multivariate decision tree (MDT) from neural networks. In this paper, we propose a modification of

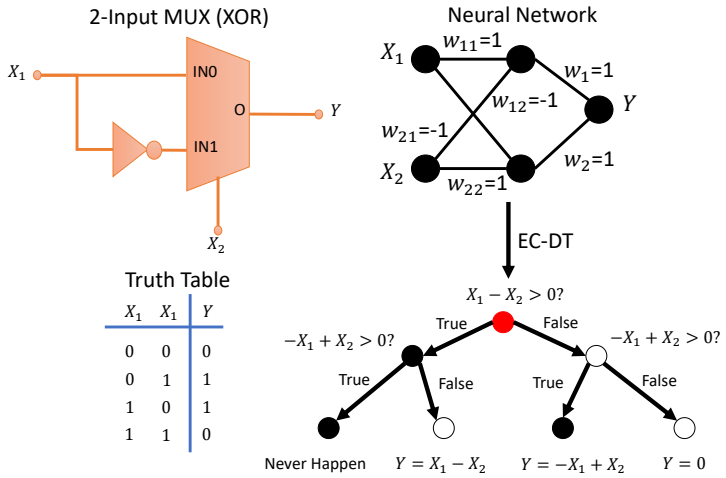


Figure 2: A XOR gate example with binary inputs that can be represented by a neural network which is converted into an EC-DT.

the algorithm into a deep version of C-Net to extract the rules from DNNs. We describe the original algorithm and the modification below, where the C-Net Algorithm is adopted to the specific use of the ReLU activation functions in the hidden layers.

After the ANN is trained, new data set are introduced and the outputs of the last hidden layer are computed. We may split the new data set into training and testing sets for the purpose of training the decision tree. Therefore from a set of training and test data, denoted as $\langle \mathbf{X}_{training}, \mathbf{O}_{training} \rangle$ and $\langle \mathbf{X}_{testing}, \mathbf{O}_{testing} \rangle$ respectively, we can compute the mapping between the last hidden output layer and the output, denoted as $\langle \mathbf{H}^K_{training}, \mathbf{O}_{training} \rangle$ and $\langle \mathbf{H}^K_{testing}, \mathbf{O}_{testing} \rangle$.

We use the data representing the relationship between the last hidden layer and the output of the network to train a C5 decision tree whose algorithm adapts an entropic information gain ratio for branch-splitting criterion and is demonstrated to be more accurate, and less memory intensive (Quinlan, 2004).

4.1. Back Projection of C-Net

After the DNN is trained, we use the output of the last hidden layer $\mathbf{H}^K(\mathbf{X^t})$ and the class prediction $\mathbf{O}^t$ to be the input and output of the UDT layers, respectively. Figure 3 illustrates the C-Net architecture.

Commonly, a DT can be represented by a set of polyhedrons expressed in the form of linear constraints. Every constraint as learned by the DT has the form of $H_{j_K}(\mathbf{X^t})$ *op* C_{j_K}, in which *op* represents the binary operators $\{\leq, <, =, >, \geq\}$,

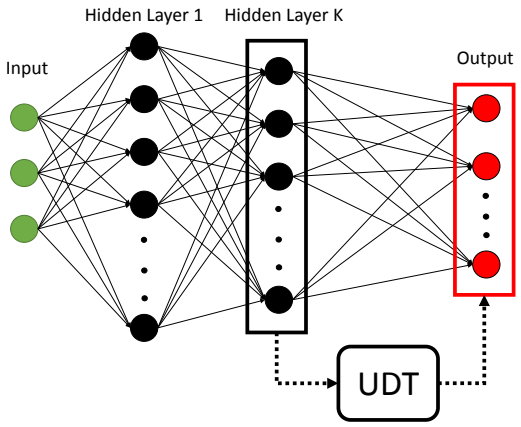


Figure 3: The C-Net framework where C5 algorithm extracts UDT rules between the last hidden layer and the output.

and C_{j_K} is the numeric threshold of such a constraint on input $H_{j_K}(\mathbf{X^t})$. It is needed to back-project from the output of the neural network to the input of the neural network to obtain a multivariate form of the expression. C-Net adopts the inverse of the used activation function, that is

$$\Big(\sum_{j_{K-1}=1}^{J_{K-1}} w_{j_{K-1}j_K} H_{j_{K-1}}^{K-1}(\mathbf{X^t}) + \beta_{j_K}^K\Big) \quad op. \quad \sigma_K^{-1}(C_{j_K}) \tag{4}$$

In our DNN, the activation functions of hidden layers are ReLU. We chose ReLU due to their linear simplicity and popularity within DNN. Hence the inverse of this function is

$$\sigma^{-1}(f) = \begin{cases} f, & \text{for } f > 0 \\ \eta, & \text{for } f \leq 0 \end{cases} \tag{5}$$

where η is an arbitrary negative number. ReLU is partially invertible; hence, it is a challenge to find the mathematical expression relating the output back to the input. Some of the nodes might not be activated resulting in a value of zero, which then might not appear in the mathematical combination expression for nodes of the next level. The expression is dynamic at the output depending on the value of inputs.

The input-output mapping is represented by the following expression in case of two hidden layers activated by ReLU, in which the outputs of the second hidden

layer are fed as inputs to the UDT:

$$\sigma(\sum_{j_1=1}^{J_1} w_{j_1 j_2}\sigma(\sum_{i=1}^{I} w_{ij_1}x_i + \beta_{j_1}^1) + \beta_{j_2}^2) \quad op. \quad C_{j_2} \tag{6}$$

A node is considered active if its output is greater than 0. We could rewrite the non-linear constraint into one linear constraint on the activation function level, and a second linear constraint on an index of the former. We note that a constraint on the index is mathematically too complex, but we will compensate that by using this constraint as a filter.

$$\sum_{j_1}\sum_{i=1}^{I} w_{j_1 j_2} w_{ij_1} x_i + w_{j_1 j_2}\beta_{j_1}^1 + \beta_{j_2}^2 \quad op. \quad C_{j_2} \tag{7}$$

rewritten as

$$\sum_{j_1}\sum_{i=1}^{I} w_{j_1 j_2} w_{ij_1} x_i \quad op. \quad C_{j_2} - (w_{j_1 j_2}\beta_{j_1}^1) + \beta_{j_2}^2 \tag{8}$$

with additional constraints:
$\forall j_1$ satisfying $\sum_{i=1}^{I} w_{ij_1}x_i + \beta_{j_1}^1 > 0$

The multivariate decision tree C-Net is a rewrite of the DNN into interpretable logic using nodes/conditions and branches/information-flows. Each rule induced by each leaf of the resultant C-Net tree can be traversed back to weights of the network to deduce the input-target relationships as represented by the network and its weights. The C-Net algorithm takes advantage of the EC-DT tree generation algorithm, but with leaves produced earlier at layer $K-1$ and target values of those leaves are linear combinations of input X as hidden nodes' outputs at layer K. The procedures of C-Net algorithm can be described as follows:

- **Step 1:** Feed inputs to DNN and compute the values at final hidden layer's nodes $H_{j_K}^K$.
- **Step 2:** Use $H_{j_K}^K$ values and labels of instances to train a UDT (C5 decision trees) and extract rules from UDT tree.
- **Step 3:** Build a decision tree using algorithm S1 (Supplementary Document, Section S.1) until hidden layer $K-1$.

- **Step 4:** Use Algorithm 3 to extract final set of rules for every leaf.

Due to the use of both a data-driven method to build UDT tree and a decompositional method to build the decision tree that representing the constraints between inputs and the last hidden layers, C-Net algorithm belongs to *eclectic/hybrid* approach. An example of how the C-Net algorithm works can be found in Section S.1 in Supplementary document.

5. Experiments

5.1. Synthetic Dataset

We start by constructing a deep feedforward network for binary classification with two input attributes. An artificial dataset is generated with polynomial relationships between the attributes. The polynomial relationship varies in complexity from one dataset to another to control the level of non-linearity in the decision boundary. A UDT may need to grow arbitrarily large before it is able to approximate a highly nonlinear function properly. We will discuss the generalization and misclassification tolerance of decision trees and our method in Section 6. The classification problem is for the DNN to estimate the classes with the data distributed as

$$y = \begin{cases} 1, & \text{for } x_1^2 + x_2^2 \geq 5 \\ 0, & \text{for } x_1^2 + x_2^2 < 5 \end{cases} \tag{9}$$

The artificial data attributes, x_1 and x_2, are sampled uniformly within the interval of $[0, \sqrt{6}]$. Data are also sampled subject to the constraint $4 \leq x_1^2 + x_2^2 \leq 6$ so that the samples concentrate around the decision boundary. As the level of non-linearity increases, the concentration of the data increases the probability of an error due to a coarse approximation of the decision boundary.

In this problem (P2 problem), 10 data sets are randomly sampled each with 5000 data points. Each set is then split into training and test subsets according to the ratio 80:20. Every training subset is shuffled 10 times. Each shuffled subset is used to train a DNN. A trained model is tested with the corresponding test subset. In total, the number of trials are $10 \times 10 = 100$ models.

The DNNs use two hidden layers with ReLU activation functions and one output unit with a sigmoid function. The number of nodes for each hidden layers are 3, 4 or 5. The network is trained with a learning rate of 0.001, 500 epochs, and mini-batch size of 128. After a network is fully trained, decision trees using C5, C-Net and EC-DT are generated to represent the network. The trees are then pruned

Algorithm 3: C-Net Rule Extraction Algorithm (for a leaf)

Input : A leaf node with its list of branches corresponding to activations of DNN'hidden layers $S = \{S^1, S^2, ...S^{(K-1)}\}$, number of nodes in hidden layers $\{J_1, J_2, ..., J_{(K-1)}\}$, a set of weights matrices of DNN $\mathcal{W} = \{\mathcal{W}^{I1}, \mathcal{W}^{12}, ..., \mathcal{W}^{(K-1)K}\}$, a set of biases matrices of DNN $\mathcal{B} = \{\mathcal{B}^1, \mathcal{B}^2, ..., \mathcal{B}^K\}$, and a set of constraints from UDT's leaf in form: $\mathbf{H^K_{j_K}}$ **op** $\mathbf{C_{j_K}}$ $\forall j_K \in \Upsilon$; $\Upsilon \subseteq \{1, 2, ..., J_K\}$

Output: A rule/set of constraints and consequences $\mathcal{R}$

Set $\mathcal{R} \leftarrow \varnothing$.

for $k = 1$ **to** $K - 1$ **do**

  for $s = 1$ **to** J_k **do**

    if $S^k(s) = 0$ **then**

      Convert matrix form $X\mathcal{W}^{Ik}_{*,s} + \mathcal{B}^{Ik}(s) > 0$ into linear inequation form.

      Elements in s^{th} row of $\mathcal{W}^{k(k+1)}$ are set to 0.

    else

      Convert matrix form $X\mathcal{W}^{Ik}_{*,s} + \mathcal{B}^{Ik}(s) \leq 0$ into linear inequation form.

    end

    Add inequation to $\mathcal{R}$ as constraint.

  end

  if $k = K - 1$ **then**

    Compute $\mathcal{W}^{IK} = \mathcal{W}^{I(K-1)}\mathcal{W}^{(K-1)K}$.

    Compute $\mathcal{B}^{IK} = \mathcal{B}^{I(K-1)}\mathcal{W}^{(K-1)K} + \mathcal{B}^K$.

  else

    Compute $\mathcal{W}^{I(k+1)} = \mathcal{W}^{Ik}\mathcal{W}^{(k(k+1)}$.

    Compute $\mathcal{B}^{I(k+1)} = \mathcal{B}^{Ik}\mathcal{W}^{k(k+1)} + \mathcal{B}^{k+1}$.

  end

end

Convert matrix form $H^K = X\mathcal{W}^{IK} + \mathcal{B}^{IK}$ to linear equations.

for $j_K \in \Upsilon$ **do**

  Add the following constraint to $\mathcal{R}$:

  $w_{1(j_K)}x_1 + w_{2(j_K)}x_2 + ... + w_{M(j_K)}x_M \;\; op. \;\; (C_{j_K} - \mathcal{B}^{IK}_{j_K})$.

end

Add UDT rule's consequence to $\mathcal{R}$ as consequence.

return $\mathcal{R}$.

with a pessimistic pruning algorithm identical to the one used for the C5 algorithm described in Quinlan's work (Quinlan, 1998). The decision tree algorithm is also

performed directly on the data set without the use of DNN to serve as a baseline method.

5.2. Benchmarking Datasets: UCI Data

We further apply our proposed methodology on benchmark datasets obtained from the UCI repositories (Dua & Graff, 2017). Some properties of every data set used in this study can be found in Table S.4 (Supplementary Document, Section S.2). In each problem, one DNN is built with 2 hidden layers each with 5 hidden nodes. The number of outputs of the DNN for binary classification is 1 with sigmoid activation functions while the number for multi-class classification is the number of output classes with softmax activation functions. The training and testing schemes are similar to the process used for the P2 problem. Decision trees trained directly on data sets are also used in comparison with DNN and DNN with rule extraction methods.

5.3. Performance Metrics

To assess the performance of our decision tree approach and the baseline C-Net and C5 algorithms, we introduce three metrics:

- *Accuracy*: In those experiments, we first measure the accuracy of the neural network for classification on test data. The extracted decision trees are also tested for their prediction accuracy on the same test set. This metric reflects the accuracy of the DT algorithm when approximating the function learned by the DNN.

- *Compactness* refers to the capability of the algorithm to represent information with the smallest model size. For decision trees, this could be measured by the number of leaves or size of the tree. In this paper, we also examine the average number of constraints within one leaf. One tree may be constructed with a low number of leaf nodes. However, the final complexity of rule interpretation also depends on the number of constraints under each leaf.

- We also investigated the decision boundaries generated by the decision trees. This could be implemented by first converting the trees into sets of rules, and second visualizing the hyper-planes corresponding to the constraints in rules on the data space.

Table 1: Accuracy of DNN with 2 hidden layers each of 3,4 and 5 nodes, C5, C-Net, and EC-DT models for P2 problem.

DNN hidden layers	DNN Accuracy ($\mu \pm \sigma\%$)	C5 ($\mu \pm \sigma\%$)	C-Net ($\mu \pm \sigma\%$)	EC-DT ($\mu \pm \sigma\%$)
3-3	78.62±18.69	78.12±18.45	78.55±18.72	78.62±18.69
4-4	87.63±13.99	87.13±13.95	87.56±14.06	87.63±13.99
5-5	93.81±5.45	91.32±10.17	92.02±10.32	93.81±5.45

6. Results and Discussion

6.1. Accuracy

The predictive accuracy of the 100 base deep neural networks for P2 problem are shown in Table 1. The performance of the DNN are improved with a larger network. The mean accuracy of DNNs of two hidden layers each with 5 nodes is better than the accuracy of decision trees ($91.94 \pm 0.87\%$) learned on the same data. Among all rule extraction methods for DNN, the EC-DT can maintain exactly the same performance of DNNs as expected. For the Pruned C5 and C-Net trees, we can observe the decrease of accuracy compared to the original performance of DNNs, though by an insignificant value from approximately 0.1% to 1.8%. C-Net in all cases preserves the performance of DNN better when compared to the baseline C5 extraction method. It is important to emphasize that all transformations are deterministic, thus, variations of performance from the original DNN are not due to any stochastic variations.

The classification accuracy for UCI datasets follow a similar trend where EC-DT captures the accuracy of DNNs perfectly. Table 2 summarizes the mean and standard deviations of the predictive accuracy of the decision trees extracted from 100 different trained DNN of two hidden layers each with 5 nodes. In a majority of data sets, the null hypothesis of the significant difference between the accuracy of EC-DT and the other methods is rejected with significance level of 0.05 or 0.01 (ANOVA and two-sample t-test). For the C-Net algorithm, the accuracy of the generated multivariate trees are higher compared to the performance of C5 trees in seven datasets (*banknote*, *wdbc*, *balance*, *new-thyroid*, *wine*, *wifi*, and *satimage*) by 1-5% in average, and are equivalent to EC-DT in soem cases (*banknote* and *wifi*). The performance of C-Net is equivalent to C5 on 7 other datasets (*skin*, *occupancy*, *climate*, *ionosphere*, *wall-following-2*, *segment*, and *ecoli*). The uses

of multivariate constraints between attributes of the input data can better form the necessary decision hyperplanes for classifying the output. Another interesting observation is that an increase in the number of attributes and the number of samples of datasets contributes to the deterioration of C5 and C-Net's accuracy.

Table 2: Accuracy of pruned trees generated from C5, C-Net and EC-DT algorithms for UCI benchmarks.

Data set	**DT (C5) Accuracy** ($\mu \pm \sigma\%$)	**DNN Accuracy** ($\mu \pm \sigma\%$)	**Rule Extraction Algorithm for DNN**		
			C5 ($\mu \pm \sigma\%$)	**C-Net** ($\mu \pm \sigma\%$)	**EC-DT** ($\mu \pm \sigma\%$)
skin	**99.87±0.02***	94.43±8.93	86.77±21.80	86.79±21.81	94.43±8.93
banknote	98.43±0.82	**99.84±0.49**	98.16±0.82	**99.74±0.56**	**99.84±0.49**
occupancy	**99.02±0.18***	98.86±0.15	98.85±0.15	98.86±0.16	98.86±0.15
wdbc	93.22±2.44	**94.57±1.88***	92.42±2.46	93.61±2.18	**94.57±1.88***
climate	88.78±8.29	**91.01±3.02**†	84.55±14.75	84.59±15.29	**91.01±3.02**†
ionosphere	90.80±3.78	**93.11±3.38**†	91.64±6.64	89.10±6.68	**93.11±3.38**†
balance	79.28±3.18	**95.12±3.22***	77.95±3.33	92.30±3.59	**95.12±3.22***
new-thyroid	92.47±3.82	**97.12±2.98**†	90.53±7.69	95.42±7.41	**97.12±2.98**†
wine	92.28±3.74	**97.13±2.29***	92.78±5.30	95.25±3.59	**97.13±2.29***
wall-following-2	**99.99±0.04***	97.82±2.84	95.78±9.06	95.30±8.92	97.82±2.84
wall-following-4	**99.99±0.05***	97.59±1.90	97.30±6.06	96.45±5.94	97.59±1.90
wifi	97.15±0.77	**97.61±1.17**	96.99±0.88	**97.38±1.21**	**97.61±1.17**
dermatology	**95.72±2.36**	**95.34±3.70**	**95.05±3.40**	93.85±3.54	**95.34±3.70**
satimage	86.31±1.21	**89.21±1.01***	85.05±1.15	86.64±1.05	**89.21±1.01***
segment	**96.50±0.96**	**95.44±1.35**	94.65±1.60	94.10±1.60	**95.44±1.35**
ecoli	80.19±4.20	**89.66±4.85***	80.55±5.06	80.39±5.53	**89.66±4.85***

Figures in **bold** are the best among methods.
† significantly better than its counterparts at significant level of 0.05.
* significantly better than its counterparts at significant level of 0.01.

6.2. Compactness

The analysis of the sizes of the decision trees extracted from the neural networks provides us the information on the compactness achieved by the proposed methods compared to the basic C5 rule extraction algorithm. As mention earlier, the number of leaves in a tree is one measure of compactness; however, the interpretability of rules converted from the decision trees is also influenced by the

number of constraints that forms each rule. A higher number of constraints in each rule can make the interpretation of it becoming more complex, and vice versa.

Table 3: Means and standard deviations of sizes of C5, C-Net, and EC-DT trees to represent DNN behaviour for P2 problem.

Data	C5 (pruned)		C-Net (pruned)		EC-DT (pruned)	
	# leaves (L)	#constraints per leaf (C)	# leaves (L)	#constraints per leaf (C)	# leaves (L)	#constraints per leaf (C)
P2 (3-3)	32.08±17.65	**1.74±0.39***	**6.09±6.60**	3.74±1.47	**6.36±4.75**	6.97±0.11
P2 (4-4)	40.56±14.83	**1.88±0.23***	**9.17±6.91**	5.39±1.18	**9.70±6.10**	9.00±0.02
P2 (5-5)	38.73±11.87	**1.91±0.16***	**11.12±6.00**†	6.69±1.05	13.40±6.35	11.00±0.01

Values in **bold** are the best among the three extraction methods.
† significantly better than its counterparts at significant level of 0.05.
* significantly better than its counterparts at significant level of 0.01.

For P2, it can be demonstrated from the data in Table 3 that the number of rules or leaves extracted by C-Net or EC-DT are significantly lower than the number of rules extracted by C5 algorithm. C-Net among all methods achieves the lowest number of leaves on average with a significant difference with significance level of 0.05 in the P2 problem with 10 hidden layer nodes (5 in each hidden layer). The number of leaves in C5 method is 3-5 times higher than the C-Net, which implies that a much greater number of decision hyperplanes is used for the classification problem.

The constraints that one can find on average at each leaf of the C5 trees, on the other hand, is much lower than the others. With less than 2 constraints per leaf, the null hypothesis of the difference between the number of constraints per leaf in C5 and C-Net is significant (ANOVA and two-sample t-test in both one and two tails with significant level of 0.01). The reason for this comes from the requirement for a number of node activation rules at the first hidden layer of the neural network so that the C-Net can represent the input-output constraints by propagating back to the conditions of the input layer from the output layer. Due to this requirement, given a deeper neural network, the C-Net will cost more constraints, as a trade-off for more accuracy. For the P2 problem, however, the total number of constraints per tree generated from 100 DNNs are lower than C5 which indicates less complex trees (see Table S.5, Section S.3). It is also interesting to note that the mean total number of constraints for EC-DT in one tree is lower than the one in C5, but for an increase in the number of nodes in the network which offers higher accuracy,

the number of constraints per leaf increases linearly.

Table 4: Means and standard deviations of sizes of C5, C-Net, and EC-DT trees to represent DNN behaviour for UCI problems.

Data set	C5 (pruned)		C-Net (pruned)		EC-DT (pruned)	
	# leaves (L)	#constraints per leaf (C)	# leaves (L)	#constraints per leaf (C)	# leaves (L)	#constraints per leaf (C)
skin	48.25±30.77	**2.59±0.98***	**7.65±8.49***	5.30±2.59	28.93±20.48	11.00±0.02
banknote	12.07±2.36	**2.33±0.10***	**3.91±1.33***	6.46±0.27	45.72±9.71	10.99±0.01
occupancy	2.52±1.35	**1.18±0.41***	**2.05±0.41***	6.01±0.09	13.50±7.36	11.00±0.02
wdbc	8.90±1.58	**2.11±0.25***	**3.08±1.15***	6.28±0.28	11.39±4.52	11.00±0.01
climate	6.92±3.19	**1.95±0.76***	**2.87±1.35***	5.46±1.97	16.96±11.85	11.00±0.01
ionosphere	7.59±1.58	**1.65±0.48***	**2.92±1.28***	6.13±0.79	28.01±10.16	11.00±0.01
balance	18.98±4.37	**2.85±0.27***	**9.19±4.06***	7.26±0.41	20.39±11.09	10.00±0.01
new-thyroid	4.73±0.97	**1.55±0.36***	**2.97±0.36***	6.27±0.77	9.53±3.53	10.00±0.02
wine	4.51±0.50	**1.72±0.22***	**3.00±0.00***	6.48±0.17	27.39±4.84	9.99±0.02
wall-following-2	**9.92±3.99***	**2.08±0.56***	18.87±9.20	7.23±0.98	12.15±4.61	9.00±0.02
wall-following-4	**11.38±4.23***	**2.35±0.45***	21.71±10.39	7.63±0.45	16.50±6.49	9.00±0.01
wifi	13.96±3.22	**2.83±0.30***	**8.09±1.83***	7.15±0.24	14.50±5.56	9.00±0.01
dermatology	8.06±0.87	**2.68±0.20***	**6.29±0.55***	7.36±0.81	27.96±8.19	8.99±0.02
satimage	70.83±6.76	**4.96±0.21***	**38.62±6.93***	8.53±0.21	80.86±18.63	8.99±0.01
segment	**24.20±3.42**	**3.46±0.33***	**23.60±4.47**	8.29±0.30	37.85±11.71	8.99±0.01
ecoli	10.67±2.09	**2.80±0.33***	**8.88±1.96***	7.40±0.30	13.84±6.48	9.00±0.01

Figures in **bold** are the best among methods.
† significantly better than its counterparts at significant level of 0.05.
* significantly better its counterparts at significant level of 0.01.

C-Net also achieves the lowest number of leaves (or rules) in most UCI problems (Table 4). The null hypotheses means that differences between the number of leaves generated with C-Net and the ones generated from other methods are rejected at significance level 0.01 for most datasets. The differences between C-Net tree sizes and C5 tree sizes, according to the figures, varies dramatically from around 1 (e.g.*occupancy* and *wine*) up to 40 (e.g. *skin*) depending on the complexity and nonlinearity of the problem. In cases with such large differences, the accuracy of the C-Net are also equivalent or better than the accuracy of C5. This is due to the capability of C-Net to generate multivariate trees for better generalization of the problem which is not easily achieved with a UDT such as C5.

Similar trends of low numbers of constraints per leaf for a C5 tree can be

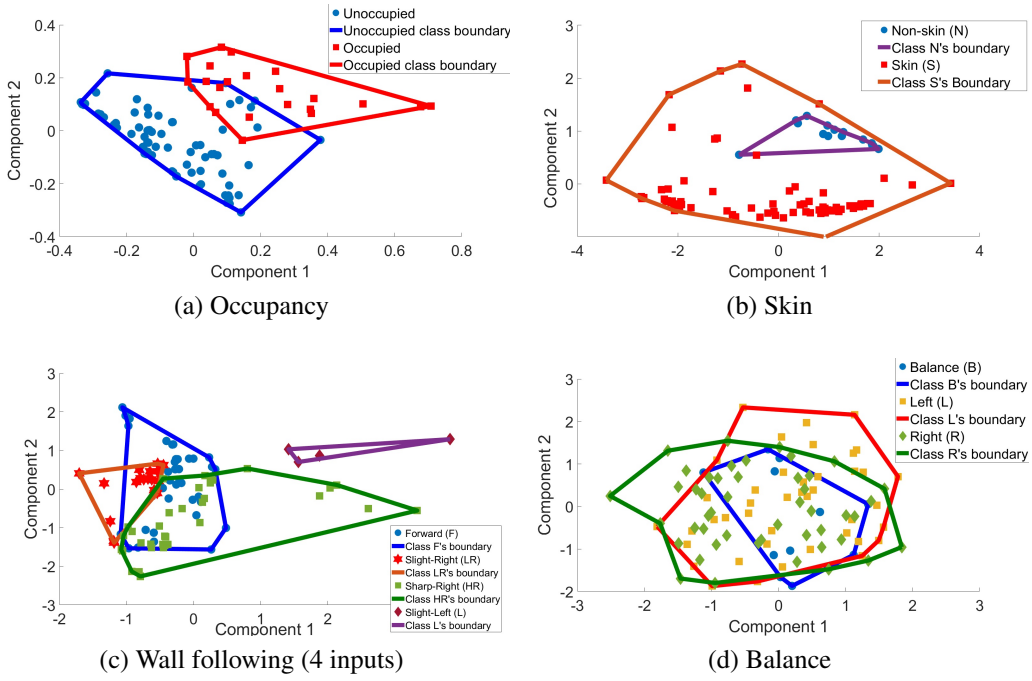
illustrated in the same Table where in most cases the figures are less than 3 constraints per leaf. The low number of constraints per leaf reflects the simplicity of using only some of the most relevant attributes for rule extraction. The use of less attributes cannot achieve much generalization power given the axis-parallel nature of C5 hyperplanes. However, for some problems such as *occupancy*, *wall-following-2*, *wall-following-4*, *dermatology*, *segment*, and *ecoli*, it is noticeable that even with a much lower number of constraints per leaf leading to the lower number of total constraints per tree on average, the C5 algorithm still shows more effectiveness compared to C-Net (though both are not comparable to EC-DT in terms of generalization power). EC-DT generates a comparable size of trees to the C5 ones, but with a huge number of constraints per tree leaf.

Table S.6 (Supplementary document, Section S.4) demonstrates the number of leaves, the number of constraints per leaf and the total constraints per tree of decision trees trained directly from data sets without the use of DNN (called direct DTs). When using direct DTs on data sets, we can observe the similar trend of the generated tree size compared to DNN-C5 algorithm. For problems with highly complex decision boundary, the performance of direct DTs are likely lower than performance of C-Net algorithm. In most cases the number of leaves and the number of constraints of trees generated by direct DTs are significantly larger than the figures of C-Net.

6.3. Decision Boundary Complexity

While C-Net provides the most compact trees and lower size of rule sets, C5 provides more interpretability with the lowest number of constraints for each rule. That raises a question of when to use a simple algorithm such as C5 instead of a more complex and comprehensible model like C-Net or EC-DT. In other words, how to decide the best compromise between simplicity versus accuracy. The analysis of the decision boundary shed some light on this issue.

For analyzing the complexity of the data spaces and class distributions, we apply Principle Component Analysis (PCA) transformation on each data set and visualize the two largest components with the largest variance. Figure 4 illustrates the distribution of classes according to two chosen principle components in two binary classification problem (*occupancy* and *skin*) and classification problem with more than two target labels (*wall-following-4* and *balance*). It can be seen that the data classes in the problem of *occupancy* and *wall-following-4* are more linearly separable than the other problems where the class coverage strongly overlaps with one another. Due to this linear-separability, it is feasible to use a simple tree with low number of leaves and constraints under each leaf to represent



(a) Occupancy (b) Skin

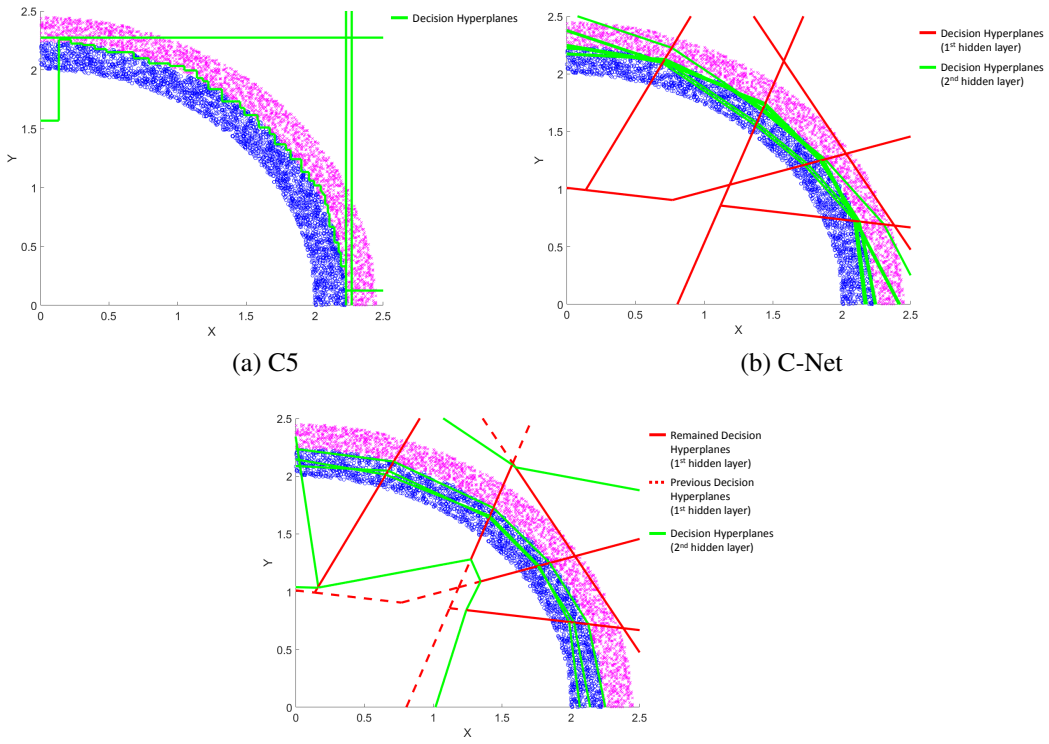
(c) Wall following (4 inputs) (d) Balance

Figure 4: Two PCA components with highest variances of different datasets.

the classification rules and hence achieve a higher fidelity with the performance of neural networks. The visualization of PCA components of different classes in remaining data sets can be found in Section S.5 (Supplementary document).

According to the figures in Table 2 and Table 4, the accuracy of linearly-separable problems such as *occupancy* and *wall-following-4* are not significantly different or at least as close to the EC-DT as the C-Net algorithm, while the number of leaves and the number of constraints under each leaf on average is equivalent or much lower than the ones of C-Net. In these cases, the use of C5 for extracting rules from the neural networks is more appropriate as it achieves acceptable performance with better simplicity. On the other hand, for more complex problems with non-linear separable properties, it is less accurate when using C5 to extract the rules. In the *skin* problem, to achieve around 86% of accuracy, the C5 trees have to use up to nearly 50 leaves each with more than 2.5 rules on average. As simple as around 19 rules with 3 constraints in one rule but the C5 cannot achieve even 80% of accuracy on average while C-Net reaches more than 92% accuracy with half the number of rules but with more than 7 constraints each.

Therefore, it is favorable to use C5 with simple rule sets for linearly-separable datasets with low dimensions while using C-Net or EC-DT if one favors higher accuracy and understanding of correlation among a large number of attributes of input space.



(a) C5 (b) C-Net

(c) EC-DT

Figure 5: Decision hyperplanes extracted from pruned C5, C-Net, and EC-DT converted from a DNN model for P2 problem with two hidden layers of 5 nodes

The synthetic problem P2 is representative for the set of highly nonlinear problems. To address this non-linearity, the simple C5 rule extraction algorithm has to generate a huge number of axis-parallel hyperplanes joining together to create a highly complex decision boundary as illustrated in Figure 5a. On the contrary, C-Net extracts the rules from DNN by considering the constraints for hidden nodes activation. Therefore, the set of rules from the first hidden layer tries to cluster the data space into sub-regions demonstrated by the area surrounded by the red lines (Figure 5b), while the second layer forms the constraints that are hyperplanes which at this stage directly separate two classes. In EC-DT, the tree representa-

tion of the true process of the DNN, the rules extracted from the first hidden layer resemblance the same clustering method as implemented by C-Net. However, the set of rules extracted from the second hidden layer takes two roles at the same time where some nodes attempt to further shrink different data sub-regions and the others directly get involved in creating decision hyper-planes between two classes (see Figure 5c). The C-Net, due to the employment of the similar rule structure at early layers of the EC-DT, exhibits a closer data processing to the DNN.

6.4. Interpretation of Rules

In this paper, we present the figures on tree size as a compactness metrics to analyze the global interpretability of the methodologies in the previous subsection. In this section, the instance-based interpretation of a rule is considered so that we can have a comprehensive view of how the explanation of a neural network as a black-box model can be generated.

Given an instance in a dataset, an explanation can be generated from constraints which are included from an activated leaf that the instance falls under. We provide the rule of one leaf in a tree so that we can evaluate the complexity and correctness of the constraints. We analyze the decision boundaries for the *skin segmentation* dataset, which are a collection of samples extracted randomly from RGB images in FERET and PAL databases. These databases contain a variety of images of people with different characteristics such as age, race, and genders. Given an area in an RGB image, with three attributes of Green (G), Red (R), and Blue (B) values ranging between 0 and 255, one rule to identify that this area is human skin from a C5 tree can be found below:

IF:

- $B > 92$
- $G \leq 157$
- $R > 231$

THEN: This is *skin*

In case of C-Net the constructed explanation for one leaf can be displayed as below:

IF:

- $(-0.47 * B) + (1.51 * G) + (0.04 * R) > -5.23$
- $(0.28 * B) + (0.35 * G) + (-0.47 * R) > -4.00$

- $(0.69 * B) + (-0.49 * G) + (0.24 * R) > -5.68$
- $(0.30 * B) + (-0.66 * G) + (0.32 * R) > 4.22$
- $(0.53 * B) + (-0.18 * G) + (-0.29 * R) > -10.14$
- $(2.75 * B) + (-1.82 * G) + (-0.81 * R) > -5.37$

THEN: This is *skin*

Meanwhile, EC-DT shows a more complex explanation in exchange for the highest accuracy with the highest number of constraints:

IF:

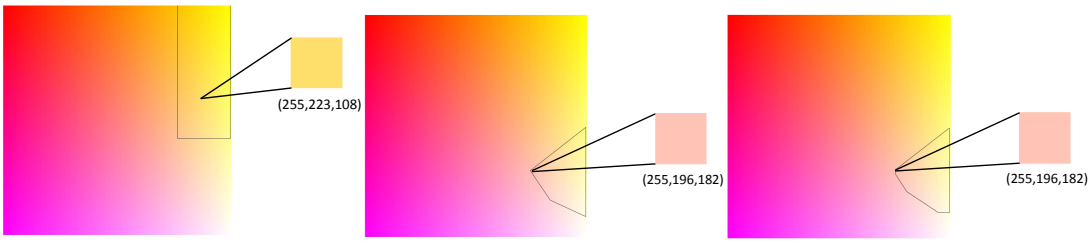
- $(-0.47 * B) + (1.51 * G) + (0.04 * R) > -5.23$
- $(0.28 * B) + (0.35 * G) + (-0.47 * R) > -4.00$
- $(0.69 * B) + (-0.49 * G) + (0.24 * R) > -5.68$
- $(0.30 * B) + (-0.66 * G) + (0.32 * R) > 4.22$
- $(0.53 * B) + (-0.18 * G) + (-0.29 * R) > -10.14$
- $(-0.32 * B) + (0.50 * G) + (-0.10 * R) > 11.20$
- $(-0.38 * B) + (0.56 * G) + (-0.49 * R) \leq -1.52$
- $(-0.64 * B) + (0.27 * G) + (0.29 * R) > 7.15$
- $(-0.03 * B) + (4.52 * G) + (-4.14 * R) \leq -36.45$
- $(2.75 * B) + (-1.82 * G) + (-0.81 * R) \leq -7.76$
- $(3.32 * B) + (-1.80 * G) + (-1.27 * R) + (46.98457) > 0$

THEN: This is *skin*

In the cases of C-Net and EC-DT, the rules involve a combination between all attributes. The attributes with higher weights have more influence on the final decisions than those with lower weights. The positive/negative signs of the weights emphasize the contribution of the attributes towards positive or negative classes in the problem.

Despite the fact that the exhibition of rules extracted from C-Net and EC-DT trees provides a more complete explanation of the decision, the complexity from the number and structure of the constraints within the rules are extremely higher than the C5 constraints. The rule as a result becomes less interpretable. Nevertheless, a way forward on this issue might come from an additional technique to transform the mode of explanation depending on problem.

In the *skin segmentation* problem, one might find it more appropriate to transform the rule constraints into visualization of the RGB colour maps and a sample



(a) C5 (b) C-Net (c) EC-DT

Figure 6: The RGB colormaps with bounded regions representing corresponding rules from different tree models.

of data needs to be explained can be also projected in the map. For example, given the red value $R = 255$, we can construct the colour map for the rules above as illustrated in Figure 6.

With this type of visualization, it is also interesting to note that the decision regions created by rules from C5 are axis-parallel rectangles which is simple, but less precise than the polygons and the shrunk polygons created by C-Net and EC-DT respectively.

In previous literature regarding skin segmentation from images, a branch in computer vision, many studies (Kovac et al., 2003; Vezhnevets et al., 2003) have introduced the explicitly defined skin region approach, which defines fixed conditions for RGB or YUV color ranges, to discriminate between skin and non-skin areas. While this method is simple, fast, and highly interpretable, it faces a significant challenge in achieving a competitive level of accuracy. Nevertheless, the method is still reliable to use as an initial screening method for skin detection (Li et al., 2010). Many machine learning methods, including neural networks, have been used to enhance the detection rate in this problem. However, their interpretability is lower than that of the simple explicitly defined skin region approach. The translation of the rules extracted from a DNN into a visualizable representation might improve the transparency of the decision making processes. Our transformation of rules into colormaps shows similar utility to the visualization of color ranges in the literature (Naji et al., 2012).

7. Conclusion

In this paper, we propose two novel multivariate decision tree frameworks which can generate interpretable rules for explaining the operation of deep neural networks. The first framework is a modification of C-Net algorithm into a Deep

C-Net which can learn the relationship between the last layer of a DNN and the output to back-project and extract the multivariate constraints on input as a set of highly accurate rules. The second is an algorithm called EC-DT that can directly translate the DNN layer-wise and buliding the set of rules with 100% fidelity to the DNN.

EC-DT offers the best accuracy when it preserves perfectly the performance of the DNN, but with the cost of a larger number of rules and number of constraints in each rule. It is understandable as the high number of rules are to capture the generalization that the DNN offers. Compared to traditional approach of generating trees with a baseline C5 algorithm, Deep C-Net in general can maintain better the accuracy of the DNN while achieving most compact trees implying a smaller number of rules in use.

However, the use of simple versus complex models results in the trade-off between the simplicity and interpretability against the accuracy and precision. To decide on which model to use, one should consider the complexity of the problem space. For linear-separable classification problems, a simple C5 can achieve similar results to DNN with a very low and simple set of rules. For highly nonlinear problems where a large number of attributes are involved, EC-DT and Deep C-Net exhibit significantly higher accuracy than a simple C5. In general, in the situation where the priority is accuracy, the EC-DT can be used, while in cases where the balance between accuracy and interpretability is required, Deep C-Net is favored.

The weakness of the more complex models such as Deep C-Net and EC-DT comes from the large number of multivariate constraints for each rule, where the form of C5 constraints is very simple. The plain display of the mathematical conditions as an explanation might lower the transparency. Therefore, a suitable transformation of the representation of rules to some explanation mode that reduces the number of dimensions can be employed to overcome the issue. The visualization of decision hyperplanes that is introduced in this paper in a specific problem of *skin segmentation* is an example for an effective explanation interface for instance-based interpretation.

In future work, the interpretability of rules extracted from our proposed EC-DT and C-Net algorithms will be investigated on more problems. The suitability of the extracted knowledge may contribute to new pieces of knowledge to different human experts, which would call for a subject-matter expert-evaluation of the extracted knowledge.

References

References

Abbass, H. A., Towsey, M., & Finn, G. (2001). C-Net: A method for generating non-deterministic and dynamic multivariate decision trees. *Knowledge and Information Systems*, *3*, 184–197.

Adadi, A., & Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, *6*, 52138–52160.

Alexander, J. A., & Mozer, M. (1999). Template-based procedures for neural network interpretation. *Neural Networks*, *12*, 479–498.

Amin, A. (2013). A novel classification model for cotton yarn quality based on trained neural network using genetic algorithm. *Knowledge-Based Systems*, *39*, 124–132.

Andrews, R., Diederich, J., & Tickle, A. B. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, *8*, 373–389.

Andrews, R., & Geva, S. (2002). Rule extraction from local cluster neural nets. *Neurocomputing*, *47*, 1–20.

Boz, O. (2002). Extracting decision trees from trained neural networks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 456–461). ACM.

Brodley, C. E., & Utgoff, P. E. (1995). Multivariate decision trees. *Machine learning*, *19*, 45–77.

Chakraborty, M., Biswas, S. K., & Purkayastha, B. (2019). Rule extraction from neural network using input data ranges recursively. *New Generation Computing*, *37*, 67–96.

Chattopadhay, A., Sarkar, A., Howlader, P., & Balasubramanian, V. N. (2018). Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (pp. 839–847). doi:10.1109/WACV.2018.00097.

Churchland, P. S., Sejnowski, T. J., & Poggio, T. A. (2016). *The computational brain*. MIT press.

Craven, M., & Shavlik, J. W. (1996). Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems* (pp. 24–30).

Craven, M. W. (1996). *Extracting comprehensible models from trained neural networks*. Technical Report University of Wisconsin-Madison Department of Computer Sciences.

Craven, M. W., & Shavlik, J. W. (1994). Using sampling and queries to extract rules from trained neural networks. In *Machine learning proceedings 1994* (pp. 37–45). Elsevier.

Dua, D., & Graff, C. (2017). UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

Fu, L. (1994). Rule generation from neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, *24*, 1114–1124.

Gan, C., Wang, N., Yang, Y., Yeung, D.-Y., & Hauptmann, A. G. (2015). Devnet: A deep event network for multimedia event detection and evidence recounting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2568–2577).

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

Gupta, A., Park, S., & Lam, S. M. (1999). Generalized analytic rule extraction for feedforward neural networks. *IEEE transactions on knowledge and data engineering*, *11*, 985–991.

Hayashi, Y., Hsieh, M.-H., & Setiono, R. (2010). Understanding consumer heterogeneity: A business intelligence application of neural networks. *Knowledge-Based Systems*, *23*, 856–863.

Hendricks, L. A., Akata, Z., Rohrbach, M., Donahue, J., Schiele, B., & Darrell, T. (2016). Generating visual explanations. In *European Conference on Computer Vision* (pp. 3–19). Springer.

Hruschka, E. R., & Ebecken, N. F. (2006). Extracting rules from multilayer perceptrons in classification problems: A clustering-based approach. *Neurocomputing*, *70*, 384–397.

Ishikawa, M. (2000). Rule extraction by successive regularization. *Neural Networks*, *13*, 1171–1183.

Johansson, U., & Niklasson, L. (2009). Evolving decision trees using oracle guides. In *2009 IEEE Symposium on Computational Intelligence and Data Mining* (pp. 238–244). IEEE.

Johnson, W. L. (1994). Agents that learn to explain themselves. In *AAAI* (pp. 1257–1263).

Kovac, J., Peer, P., & Solina, F. (2003). *Human skin color clustering for face detection* volume 2. IEEE.

Li, Z., Xue, L., & Tan, F. (2010). Face detection in complex background based on skin color features and improved adaboost algorithms. In *2010 IEEE International Conference on Progress in Informatics and Computing* (pp. 723–727). IEEE volume 2.

McMillan, C., Mozer, M. C., & Smolensky, P. (1991). The connectionist scientist game: rule extraction and refinement in a neural network. In *Proceedings of the 13th Annual Conference of the Cognitive Science Society* (pp. 424–430).

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G. et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*, 529.

Murthy, S. K., Kasif, S., & Salzberg, S. (1994). A system for induction of oblique decision trees. *Journal of artificial intelligence research*, *2*, 1–32.

Naji, S. A., Zainuddin, R., & Jalab, H. A. (2012). Skin segmentation based on multi pixel color clustering models. *Digital Signal Processing*, *22*, 933–940.

Odajima, K., Hayashi, Y., Tianxia, G., & Setiono, R. (2008). Greedy rule generation from discrete data and its use in neural network rule extraction. *Neural Networks*, *21*, 1020–1028.

Quinlan, J. R. (1987). Generating production rules from decision trees. In *ijcai* (pp. 304–307). Citeseer volume 87.

Quinlan, J. R. (2004). Data mining tools see5 and c5. 0. *http://www. rulequest. com/see5-info. html*, .

Quinlan, R. (1998). C5. 0: An informal tutorial.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). 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* (pp. 1135–1144). ACM.

Saad, E. W., & Wunsch II, D. C. (2007). Neural network explanation using inversion. *Neural networks*, *20*, 78–93.

Salakhutdinov, R., Tenenbaum, J. B., & Torralba, A. (2013). Learning with hierarchical-deep models. *IEEE transactions on pattern analysis and machine intelligence*, *35*, 1958–1971.

Samek, W. (2019). *Explainable AI: Interpreting, explaining and visualizing deep learning* volume 11700. Springer Nature.

Schmitz, G. P. J., Aldrich, C., & Gouws, F. S. (1999). Ann-dt: an algorithm for extraction of decision trees from artificial neural networks. *IEEE Transactions on Neural Networks*, *10*, 1392–1401. doi:10.1109/72.809084.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618–626).

Sestito, S. (1992). Automated knowledge acquisition of rules with continuously valued attributes. In *Proceedings of the 12th international conference on expert systems and their applications, 1992*.

Setiono, R., & Liu, H. (1996). Symbolic representation of neural networks. *Computer*, *29*, 71–77.

Setiono, R., & Liu, H. (1997). Neurolinear: From neural networks to oblique decision rules. *Neurocomputing*, *17*, 1–24.

Shortliffe, E. H., & Buchanan, B. G. (1984). A model of inexact reasoning in medicine. *Rule-based expert systems*, (pp. 233–262).

Sok, H. K., Ooi, M. P.-L., Kuang, Y. C., & Demidenko, S. (2016). Multivariate alternating decision trees. *Pattern Recognition*, *50*, 195–209.

Swartout, W., Paris, C., & Moore, J. (1991). Explanations in knowledge systems: Design for explainable expert systems. *IEEE Expert*, *6*, 58–64.

Taha, I. A., & Ghosh, J. (1999). Symbolic interpretation of artificial neural networks. *IEEE Transactions on knowledge and data engineering*, *11*, 448–463.

Tickle, A. B., Orlowski, M., & Diederich, J. (1996). *DEDEC: A methodology for extracting rules from trained artificial neural networks*. Neurocomputing Research Centre, Queensland University of Technology.

Towell, G. G., & Shavlik, J. W. (1993). Extracting refined rules from knowledge-based neural networks. *Machine learning*, *13*, 71–101.

Tsujino, K., & Nishida, S. (1995). Implementation and refinement of decision trees using neural networks for hybrid knowledge acquisition. *Artificial Intelligence in Engineering*, *9*, 265–276.

Van Lent, M., Fisher, W., & Mancuso, M. (2004). An explainable artificial intelligence system for small-unit tactical behavior. In *Proceedings of the National Conference on Artificial Intelligence* (pp. 900–907). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Vezhnevets, V., Sazonov, V., & Andreeva, A. (2003). A survey on pixel-based skin color detection techniques. In *Proc. Graphicon* (pp. 85–92). Moscow, Russia volume 3.

Wang, F., Wang, Q., Nie, F., Yu, W., & Wang, R. (2018). Efficient tree classifiers for large scale datasets. *Neurocomputing*, *284*, 70–79.

Wen, Y., Zhang, K., Li, Z., & Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision* (pp. 499–515). Springer.

Wu, S., Chen, Y.-C., Li, X., Wu, A.-C., You, J.-J., & Zheng, W.-S. (2016). An enhanced deep feature representation for person re-identification. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on* (pp. 1–8). IEEE.

Yu, Q., Liu, J., Cheng, H., Divakaran, A., & Sawhney, H. (2012). Multimedia event recounting with concept based representation. In *Proceedings of the 20th ACM international conference on Multimedia* (pp. 1073–1076). ACM.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision* (pp. 818–833). Springer.