IEEE Open Journal of the
Computer Society

Received 2 May 2023; accepted 30 May 2023. Date of publication 5 June 2023;
date of current version 16 June 2023. The review of this article was arranged by Associate Editor Milos Stojmenovic.

Digital Object Identifier 10.1109/OJCS.2023.3282948

Disjunctive Threshold Networks for Tabular Data Classification

WEIJIA WANG, LITAO QIAO, AND BILL LIN

ECE Department, University of California, San Diego, La Jolla, CA 92093-0407 USA

CORRESPONDING AUTHOR: WEIJIA WANG (e-mail: wweijia@ eng.ucsd.edu).

ABSTRACT While neural networks have been achieving increasingly significant excitement in solving classification tasks such as natural language processing, their lack of interpretability becomes a great challenge for neural networks to be deployed in certain high-stakes human-centered applications. To address this issue, we propose a new approach for generating interpretable predictions by inferring a simple three-layer neural network with threshold activations, so that it can benefit from effective neural network training algorithms and at the same time, produce human-understandable explanations for the results. In particular, the hidden layer neurons in the proposed model are trained with floating point weights and binary output activations. The output neuron is also trainable as a threshold logic function that implements a disjunctive operation, forming the logical-OR of the first-level threshold logic functions. This neural network can be trained using state-of-the-art training methods to achieve high prediction accuracy. An important feature of the proposed architecture is that only a simple greedy algorithm is required to provide an explanation with the prediction that is human-understandable. In comparison with other explainable decision models, our proposed approach achieves more accurate predictions on a broad set of tabular data classification datasets.

INDEX TERMS Interpretable artificial intelligence, decision rule learning, tabular data classification, neural networks.

I. INTRODUCTION

Machine learning is finding its way to impact every sector of our society, including healthcare, transportation, finance, retail, and criminal justice. In high-stakes human-centered applications like medical-diagnosis and criminal justice, where decisions can have serious consequences on human lives, the critical importance of interpretability to explain predictions or decisions is well-recognized in the machine learning community [1].

A. RELATED WORK

One popular approach to interpretable models is the use of decision rule sets [2], [3], [4], [5], [6], which are inherently interpretable because the rules are expressed in simple if-then sentences that correspond to logical combinations of input conditions that must be satisfied for a classification. While decision rule sets are natural classifiers, for which the performance is generally measured by the overall classification accuracy, coverage and rule precision are also commonly considered important metrics of decision rules. In particular, [6] proposes to impose an additional constraint on precision to improve the performance of the rule sets. Besides decision rule sets, decision lists [7], [8] and decision trees [9] are also interpretable rule-based models. Not only do these decision models provide predictions, but the corresponding matching rules also serve as human-understandable explanations. Gradient boosting decision trees [10], [11] and random forests [12] have also been successfully used in learning problems involving tabular data. Although these methods provide superior predictive performance in comparison with design rule learning, they are generally considered to be lacking in interpretability, which may limit their adoption in certain application domains.

Neural networks have also been recently proposed for tabular data classification [13], [14], [15], [16]. The work in [13] proposes a new neural network module for tabular datasets, which achieves effective performance in tabular classification and regression problems by explicitly grouping the correlative input features and generating higher-level features for semantics abstraction. However, this approach concentrates on predictive performance, it is still a black-box model that is not explainable to humans. The work in [15], [16] introduces additional inductive bias to over-parameterized neural networks by designing specific neural network structures to emulate the axis-aligned splits of decision trees that have made the ensembles of trees so successful for tabular datasets. Although both works leverage feature selection techniques as part of their structure design, which can be extracted to interpret the feature attributions to the prediction or classification, this level of interpretability is very limited compared to rule-based sentences that can be easily understood by humans.

In contrast, the recent work in [14] proposes a specific neural network architecture to encode an underlying disjunctive normal form representation that can be mapped to a decision rule set. To achieve this one-to-one correspondence, the hidden layer neurons in the proposed model are restricted in a manner so that they directly map to conjunctions (logical-ANDs) of input features. These conjunctions correspond to interpretable decision rules. The output neuron implements a disjunctive (logical-OR) operation that aggregates the interpretable decision rules in the hidden layer into a decision rule set. The proposed solution has the same advantage as the class of decision rule learning and tree approaches [2], [3], [4], [5], [9] in that it can also provide meaningful explanations, but it is able to do so with superior predictive performance. However, the approach in [14] imposes restrictions on the hidden layer neurons in a way that limits the search space.

There is also a body of work [17], [18], [19], [20], [21], [22] on compiling models into tractable forms. The tractable form can then be analyzed to produce explanations. In contrast, our approach derives human understandable explanations directly from our proposed model using a fast and simple algorithm.

B. OUR CONTRIBUTION

We propose to address the tabular data classification problem with a new neural network model called DT-Net (Disjunctive Threshold Network). The hidden layer neurons in the proposed model are trained with floating point weights and binary output activations. These neurons can be interpreted as threshold logic functions, which provides considerably greater flexibility than the DR-net [14] approach that restricts hidden layer neurons to implement conjunction (AND) operations. In particular, [14] incorporates stochastic gradient descent [23] with the straight-through estimator [24] and state-of-the-art regularization techniques proposed in [25], [26] to achieve high predictive performance and interpretability. Unlike traditional black-box approaches like gradient boosting trees, random forests, and conventional neural networks, DT-Net can also provide rule-like explanations that are comprehensible to humans. However, unlike prior work on decision rule learning [4], [5], [14], our approach does not require the explicit construction of a decision rule set. This means that our disjunctive network of threshold functions can implicitly encode a potentially complicated set of rules to achieve high predictive performance, but yet the derived explanations can nonetheless be simple.

The remainder of the article is organized as follows: Section II describes our proposed DT-Net architecture. Section III describes how explanations can be efficiently derived from a DT-Net inference. Section IV describes how sparsity-inducing regularization can help to simplify explanations. Our proposed approach is extensively evaluated in Section V. Finally, concluding remarks are given in Section VI.

II. DISJUNCTIVE THRESHOLD NETWORK

We introduce in this section the Disjunctive Threshold Neural Network architecture, or DT-Net for short. It is aimed at tabular classification problems in which the ability to *explain decisions* is essential, in addition to making accurate predictions. DT-Net is a simple three-layer neural network architecture comprising n input units, k hidden layer units, and a single output unit. A toy example of the proposed architecture is shown in Fig. 1, which we use to explain the main points of our work.

Input layer: Each of the n units at the input layer passes its corresponding assigned binarized value to each neuron in the hidden layer. Generally, tabular datasets can have input attributes that are binary, categorical, or numerical. To handle categorical and numerical attributes, well established and studied pre-processing procedures in the machine learning literature can be used to encode them into binarized input vectors. In particular, standard one-hot encoding can be used to transform categorical attributes into binary vectors, and standard quantile discretization can be used to encode numerical values into binary vectors[1].

Hidden layer of threshold functions: Each of the k units in the hidden layer is a threshold function that is trainable with arbitrary (positive or negative) full-precision weights and biases. This is implemented using a binary step activation function. The blue dashed lines in Fig. 1 indicate that the corresponding features have zero weights, which means the corresponding threshold function is not dependent on them. As discussed in the next section, each threshold function implicitly encodes an underlying Boolean logic function of inputs that will yield to a positive result.

Output disjunction layer: The output layer is designed to implement a *disjunction* of the k hidden layer threshold functions, which consists of a single neuron with all weights and the bias fixed at 1 and $-\epsilon$, respectively, where ϵ is a small constant between 0 and 1 (we use $\epsilon = 0.5$ in our experiments). This output threshold unit implements a logical-OR operation since by default, it makes a negative prediction if none of the threshold functions in the hidden layer is activated, whereas any activated threshold function is sufficient to cause the output unit to make a positive prediction. Since each threshold function essentially encodes an underlying Boolean logic function, the whole network also implicitly implements a Boolean logic function by taking the disjunction of these

[1]Widely studied interpretable rule-learning methods [4], [5], [14] on tabular classification problems also commonly assume the datasets to be binarized through pre-processing.

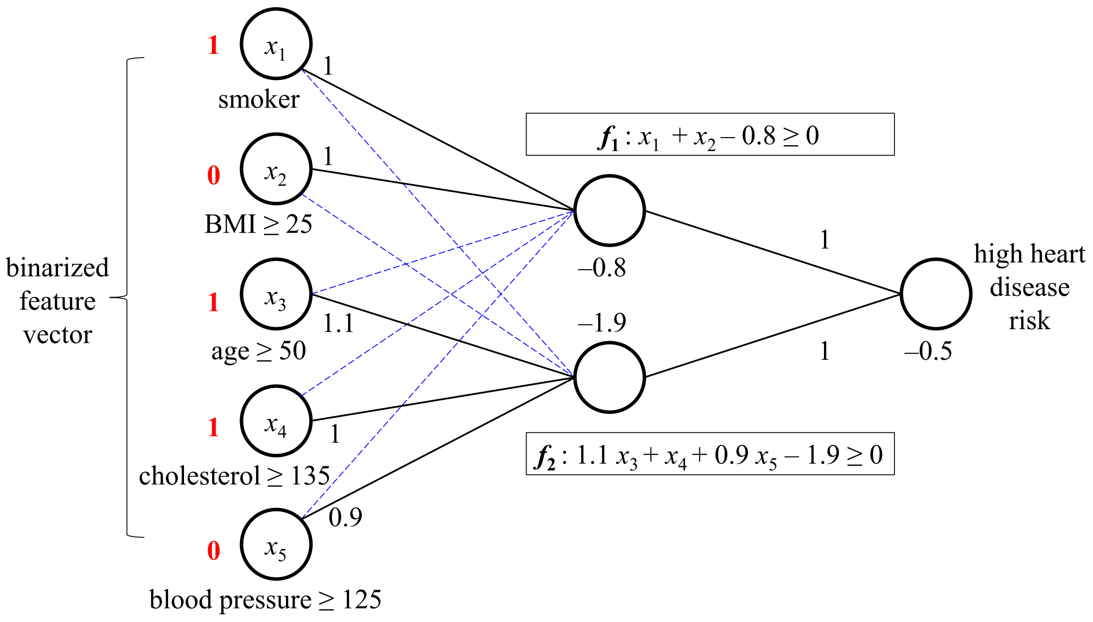


FIGURE 1. An example of the DT-Net architecture. Each hidden layer unit implements a threshold logic function, and the output unit implements a disjunction of these threshold functions. Explanations can be readily derived from the network to explain positive predictions.

threshold functions. We note that these two layers together compose a logic function in disjunctive normal form, which is capable of encoding any possible Boolean logic function. In other words, our proposed model is applicable to any binary classification problem.

Straight-through estimator: As previously mentioned, the outputs of the threshold functions are fed into a step activation function, which has an impulse derivative function that prevents the gradients from propagating through. In this work, we adopt the straight-through estimator with the gradient clipping technique to address this issue, which is detailed as follows:

$$g_{\hat{z}_i} = \begin{cases} 0, & \text{if } z_i < 0 \text{ or } (z_i > 1,\, g_{z_i} < 0) \\ g_{z_i}, & \text{otherwise} \end{cases} \tag{1}$$

where $g_{\hat{z}_i} = \frac{\partial L}{\partial \hat{z}_i}$ and $g_{z_i} = \frac{\partial L}{\partial z_i}$ are respectively the gradients of classification loss with respect to $\hat{z}_i$ and z_i.

Similar to the ReLU activation function, the step function only produces non-negative outputs. Therefore, we follow ReLU and clip the gradient w.r.t. negative outputs. Moreover, since the step function has an upper bound of 1 for its output, further increasing an activation that is already greater than 1 does not make any improvement, which empirically can even lead to an explosion of the weights. Therefore, we propose to clip such gradient that tries to further increase an activation greater than 1.

Example: Consider the heart disease risk prediction example again, as depicted in Fig. 1. Each input instance corresponds to an individual and the features of this person, i.e., smoker, overweight, older than 50, cholesterol, and high blood pressure, are encoded as $x_1, x_2, \ldots, x_5$, respectively. In this toy example, threshold function (hidden neuron) f_1 can be activated by the individual being either a smoker or overweight, and threshold function f_2 evaluates to true if at least two out of the three features with non-zero weights (older than 50, high cholesterol, and high blood pressure) are 1, due to the fact that for any combination of at least two of these features, the summation of their weights is sufficiently greater than 1.9. Given the individual represented as $\langle x_1, x_2, x_3, x_4, x_5 \rangle =$ [10110], both neurons f_1 and f_2 produce a 1 for this instance. Therefore, the entire network produces a positive prediction (the individual has a high heart disease risk).

There can be several *explanations* as to *why* the individual is predicted to have a high heart disease risk. One explanation is that the individual is a *smoker*, which sufficiently explains the high heart disease risk prediction. This explanation is also the *simplest explanation* in that there is no other explanation that is more concise. A more complex explanation is that the person is *older than 50* with *high cholesterol*. This explanation is the simplest when only considering f_2, but it is not the simplest explanation overall as identifying the individual as a smoker is a more concise explanation. However, it is a *minimal explanation* in that no other condition can be removed from the explanation so that the explanation remains sufficient: i.e., *older than 50* by itself is insufficient to explain a high heart disease risk prediction. As detailed later in the article, given a positive prediction, we can easily *derive* the simplest explanation with respect to an activated threshold function. Unlike existing interpretable rule-learning methods [4], [5], [14] that explicitly generate sets of decision rules as classifiers, our approach does not require the generation of any specific decision rule set from the trained disjunctive threshold network model. Instead, predictions are made through standard neural network operations so that potentially complicated rules can still be implicitly encoded to achieve better generalizations, where simple explanations for each positive prediction can nonetheless be readily generated afterwards. In addition, due to the natural use of stochastic gradient descent (SGD), any state-of-the-art SGD training techniques can be applied to improve classification performance. In particular, we will discuss later in the article a well-developed sparsity-inducing method that we incorporate to simplify the network, which further leads to concise explanations. In the next section, we describe how human-readable explanations can be readily derived for positive predictions produced by the proposed network.

III. EXPLAINING DT-NET PREDICTIONS

An important feature of our DT-Net approach is that human understandable explanations can be easily derived from DT-Net predictions. We first prove several important properties about threshold functions that we will use to derive explanations from them. We then describe how explanations can be derived in the single threshold function case, followed by a discussion regarding how explanations can be derived from the overall DT-Net. All proofs to theoretical results in this section can be found in the supplementary material.

A. THRESHOLD FUNCTIONS AND PRIMES

A feed-forward neural network typically comprises layers of neurons. Further, a neuron with binary inputs and full-precision weights performs the following computation:

$$f(\mathbf{x}) = \varphi\left(\mathbf{w}^T\mathbf{x} + b\right), \tag{2}$$

where $\mathbf{w} \in \mathbb{R}^n$ is a weight vector $\langle w_1, w_2, \ldots, w_n\rangle$, $\mathbf{x} \in \mathbb{R}^n$ is an input vector $\langle x_1, x_2, \ldots, x_n\rangle$, $b \in \mathbb{R}$ is a bias term, and $\varphi(\cdot)$ is a non-linear activation function. Common activation functions include ReLU activation, the sigmoid function, and the step function.

When the n inputs are binary features, and the step function is used for activation, the neuron $f(\mathbf{x})$ corresponds to a *threshold function*[2], where

$$z(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b, \tag{3}$$

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } z(\mathbf{x}) \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

A threshold function f implements an underlying Boolean logic function $f : \{0,1\}^n \to \{0,1\}$. As such, terminologies and properties from Boolean algebra apply. An *instance* $\alpha \in \{0,1\}^n$ is a specific assignment to the input features. With respect to the threshold function f, a positive instance is one such that $f(\alpha) = 1$, and a negative instance is one such that $f(\alpha) = 0$. A *literal* ℓ_i is a feature (positive literal) or its negation (negative literal), denoted as $\ell_i = x_i$ and $\ell_i = \bar{x}_i$, respectively. A *term* π is a consistent conjunction of literals, e.g., $x_1 \wedge \bar{x}_2 \wedge x_3$, or simply $x_1\bar{x}_2x_3$[3]. The *length* of π, denoted as $|\pi|$, is the number of literals that it includes. We say that a term π_i *covers* or *contains* another term π_j, written as $\pi_j \Rightarrow \pi_i$, if and only if π_j includes all the literals in π_i (e.g., $x_1\bar{x}_2$ covers $x_1\bar{x}_2x_3$).

An *implicant* π of a Boolean function f is a term that *satisfies* f, written as $\pi \Rightarrow f$, meaning all instances covered by π are positive instances. A *prime implicant* (or simply a *prime*) is an implicant that is not covered by any other implicant. A prime is *essential* if it covers an instance that is not covered by any other prime. A set of primes $\{\pi_1, \ldots, \pi_m\}$ is a *prime cover* for f if $\vee_{i=1}^m \pi_i$ is equivalent to f, and it is a *prime and irredundant cover* if no prime π_i can be removed from $\{\pi_1, \ldots, \pi_m\}$ such that the set remains a prime cover.

Several concepts are introduced next to prove several important properties about deriving prime implicants from threshold functions.

Definition 1 (Slack): The *slack* of an instance α with respect to a threshold function f corresponds to $z(\alpha)$ in (3). Therefore,

$$f(\alpha) = \begin{cases} 1 & \text{if the slack is non-negative} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The *slack* of a term π is defined as the minimum slack among the instances that π covers:

$$z(\pi) = \min_{\alpha} z(\alpha), \quad \text{s.t. } \alpha \Rightarrow \pi. \tag{6}$$

Note $z(\pi)$ can be directly computed by setting every feature x_i that does not appear in the term π to its worst-case value, which minimizes $z(\pi)$: i.e., if $w_i > 0$, set $x_i = 0$; otherwise, set $x_i = 1$.

Definition 2 (Maximum Slack): We define the *maximum slack* of a threshold function f to be the largest slack among all possible assignments. In other words,

$$z_{\max} = \max_{\alpha} z(\alpha), \quad \forall \alpha \in \{0,1\}^n. \tag{7}$$

This maximum slack can be directly computed by setting a feature x_i to its best-case value to maximize $z(\alpha)$ if it appears in f with a non-zero weight: i.e., set $x_i = 1$ if $w_i > 0$ and $x_i = 0$ otherwise.

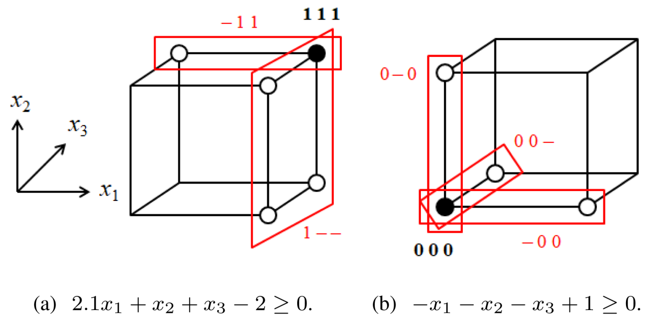
Definition 3 (Base Term): For a threshold function f, we define the *base term* π_{base} to be a term that includes the literal x_i if $w_i > 0$ and the literal $\bar{x}_i$ if $w_i < 0$ (no literal for x_i is included if $w_i = 0$).

Proposition 1: The base term always achieves the maximum slack. In other words, $z(\pi_{base}) = z_{\max}$.

Next, we illustrate the above definitions with two examples, as depicted in Fig. 2. Consider the first example depicted in Fig. 2(b) The base term is [111] as it achieves the maximum slack of $2.1 + 1 + 1 - 2 = 2.1$. The base term is shown in a black circle, while the remaining positive instances are shown

[2]A threshold function is also commonly written in the form $\mathbf{w}^T\mathbf{x} \geq \theta$, which is equivalent to $\mathbf{w}^T\mathbf{x} - \theta \geq 0$, where θ is referred to as the *threshold*. This is equivalent to Equations 3 and 4, with $\theta = -b$. We will use the form expressed in Equations 3 and 4, as this is the common expression form for describing neurons.

[3]As a shorthand, [101] is used to denote the term $x_1\bar{x}_2x_3$. Here is another shorthand example: [10−] is used to denote the term $x_1\bar{x}_2$, where "−" means the corresponding feature is excluded from the term.



−1 1
1 1 1
x_2
x_3
x_1
1 − −
0 − 0
0 0 −
0 0 0
− 0 0
(a) $2.1x_1 + x_2 + x_3 - 2 \geq 0$.
(b) $-x_1 - x_2 - x_3 + 1 \geq 0$.

FIGURE 2. Threshold function examples with the corresponding base terms (e.g., 111) and primes (e.g., 1 − −).

in white circles. There are two primes in this example (shown in red). One prime $[-11]$ can be derived by *expanding* the base term $[111]$ in the x_1 direction (by removing the literal x_1), whereas the other prime $[1--]$ can be derived by expanding the base term $[111]$ in both the x_2 and x_3 directions. In both primes, the slack of each is non-negative, but removing one more literal would cause the corresponding slack to become negative.

Intuitively, all primes can be generated *from* the base term π_{base} with the maximum slack. If there exists a non-zero w_i such that $|w_i| \leq z_{\max}$, then the corresponding literal for x_i can be removed from $\pi_{\max}$ to produce an intermediate implicant $\tilde{\pi}$. This process can be repeated by removing each additional literal as long as there is a corresponding non-zero w_i such that $|w_i| \leq z(\tilde{\pi})$, the remaining slack, until a prime is produced.

In the second example depicted in Fig. 2(b), the base term is $[000]$ as it achieves the maximum slack of $0 - 0 - 0 + 1 = 1$. There are three primes in this example, corresponding to expanding in each of the three directions, to produce $[-00]$, $[0-0]$, and $[00-]$.

Theorem 2: All primes of a threshold function cover the base term.

Proof: We prove this by contradiction. Assume a prime π does not cover the base term. Then, there must exist a literal $\ell_i \in \{x_i, \bar{x}_i\}$ that is in π but not in π_{base}. Consider the following two cases. First, if $\bar{\ell}_i$ is present in π_{base}, then ℓ_i corresponds to the worst-case value and removing it from π will increase the slack. Second, if π_{base} does not include ℓ_i or $\bar{\ell}_i$, then it implies $w_i = 0$ and removing ℓ_i from π does not change the slack. In both cases, since π is a prime, we have $z(\pi) \geq 0$, and hence $z(\pi \setminus \{\ell_i\}) \geq 0$, which is contradictory to the definition of primes. ■

Theorem 3: All primes of a threshold function are essential.

Proof: We prove this by contradiction. Assume a prime π_1 is not essential, which means there exists an instance covered by π_1 that is also covered by another prime. Consider an instance α covered by π_1 that disagrees with every literal ℓ_i that is in π_{base} but not in π_1. Suppose α is covered by another prime π_2, implying that for every such ℓ_i, π_2 either includes $\bar{\ell}_i$ or excludes both ℓ_i and $\bar{\ell}_i$. According to Theorem 2, we have $\pi_{base} \Rightarrow \pi_1$ and $\pi_{base} \Rightarrow \pi_2$. Since ℓ_i is covered by π_{base}, we must have π_2 excludes every such ℓ_i and $\bar{\ell}_i$, which implies all literals removed from π_{base} to produce π_1 are also removed to produce π_2. As a result, $\pi_1 \Rightarrow \pi_2$. This means either $\pi_1 = \pi_2$ or π_1 is not a prime, which are contradictory in both cases. ■

Corollary 4: The prime cover of a threshold function is unique and irredundant.

Proof: It follows from the proof of Theorem 3. ■

B. EXPLAINING A SINGLE THRESHOLD FUNCTION

We first consider the problem of deriving an explanation for the single threshold function case. A threshold function f is equivalent to a Boolean *classifier*, where $f(\alpha) = 1$ means the *decision* is positive, and $f(\alpha) = 0$ means the decision is negative. For a positive prediction, an explanation can be thought of as some subset of its literals. Referring to the example depicted in Fig. 1, an explanation why an individual is at high heart disease risk may be that the individual is *older than 50* and has *high cholesterol*. Another explanation may be that the individual is a *smoker*. We formalize below what explanations are and how they can be readily derived in the case of a single threshold function.

Definition 4 (Explanation): An *explanation* for a positive decision on an instance α is an implicant that contains the instance.

Definition 5 (Minimal Explanation): A *minimal explanation* is a prime that contains the instance.

Definition 6 (Simplest Explanation): A *simplest explanation* a shortest length minimal explanation.

Note that minimal and simplest explanations are not unique. As shown in [17], for a threshold function f and a positive instance α, finding minimal explanations corresponds to finding prime implicants[4] of f that contain α. The prime associated with a minimal explanation corresponds to a minimal subset of features that are sufficient for the positive prediction. This can be achieved by first converting the threshold function f into a logic representation, followed by using known prime generation algorithms to generate all minimal explanations, where the simplest explanation (shortest prime containing α) can be found, but this approach is worst-case exponential in time and space. Fortunately, the simplest explanation can be directly derived from the threshold function f, as discussed below.

Definition 7 (Base Explanation): Given a threshold function f as a classifier and a positive instance α, we define the *base explanation*, written as $\pi_{base-exp}$, to be the *supercube* of the base term π_{base} and the instance α, written as $super(\pi_{base}, \alpha)$.

The *supercube* of two terms, $super(\pi_i, \pi_j)$, is a new term derived by removing literals from π_i that do not appear in π_j.

[4] In [17], minimal explanations are referred to as PI-explanations, where PI refers to prime implicants. The work in [17] was developed for Naive Bayes Classifiers, but the same PI-explanation concept also applies to threshold functions. In [20], PI-explanations are referred to as abductive explanations.

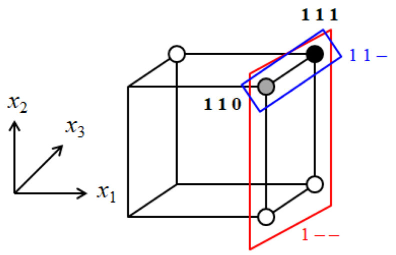


FIGURE 3. An example of base and minimal explanations, with f defined as $2.1x_1 + x_2 + x_3 - 2 \geq 0$ and $\alpha = [110]$.

The operation is symmetric in that the new term can also be derived by removing literals from π_j that do not appear in π_i.

Theorem 5: The set of minimal explanations of a positive instance for a threshold function includes only essential primes.

Proof: It follows from the proof of Theorem 3. ■

Theorem 6: All minimal explanations of a positive instance for a threshold function cover the base explanation.

Proof: We prove this by contradiction. Assume a minimal explanation π has a literal ℓ_i that the base explanation $\pi_{base-exp}$ does not have. Then there are two possibilities. The first is that $\pi_{base-exp}$ has the literal $\bar{\ell}_i$ instead of ℓ_i. Based on the definition of the base explanation, the instance α must also have $\bar{\ell}_i$. Since ℓ_i and $\bar{\ell}_i$ cannot both appear in π, π does not have $\bar{\ell}_i$ and thus does not contain instance α. This contradicts the definition of an explanation. The second possibility is that $\pi_{base-exp}$ does not have ℓ_i or $\bar{\ell}_i$. Since π is an explanation of α, π contains α and thus α also has the literal ℓ_i. Then, we know that the base term of the threshold function must have $\bar{\ell}_i$, so that $\pi_{base-exp}$, as a supercube of π_{base} and α, does not have ℓ_i or $\bar{\ell}_i$. According to the definition of the base term, the corresponding weight w_i of the threshold function is negative. At this point, it is obvious that a new term $\pi \setminus \{\ell_i\}$ is still a valid explanation because removing ℓ_i from π does not change the slack of π, which is contradictory to the premise that π is a minimal explanation. ■

Consider the example depicted in Fig. 3. In this example, $\pi_{base} = [111]$ and $\alpha = [110]$ (shown as a gray circle), then the base explanation is $[11-]$ (shown in blue). The generation of explanations can be performed in a similar way as prime generation. According to Theorem 6, all minimal explanations, which are primes containing the instance α, can be generated *from* the base explanation $\pi_{base-exp}$ with the available slack $z(\pi_{base-exp})$. Consider again the example depicted in Fig. 3. The base explanation $[11-]$ can be expanded into a minimal explanation by *expanding* in the x_2 direction (by removing the literal x_2) to obtain the prime and minimal explanation $[1--]$. There are no other minimal explanations, making $[1--]$ also the simplest explanation. In particular, if there exists a non-zero w_i such that $|w_i| \leq z(\pi_{base-exp})$, then the corresponding literal for x_i can be removed from $\pi_{base-exp}$ to produce an intermediate implicant. This process can be repeated as long as there is a corresponding non-zero w_i such

Algorithm 1: Smallest-absolute-weights-first removal

```
Input: Threshold function $f$, base explanation $\pi_{base-exp}$
Output: Simplest explanation $\pi$
1: $L \leftarrow \{\ell_i \in \pi_{base-exp}\}$ sorted by $|w_i|$ in ascending order
2: $\pi \leftarrow \pi_{base-exp}$
3: for $\ell_i \in L$ do
4:     if $z(\pi \setminus \{\ell_i\}) \geq 0$ then
5:         $\pi \leftarrow \pi \setminus \{\ell_i\}$
6:     else
7:         break
8:     end if
9: end for
10:    return $\pi$
```

that $|w_i|$ is less than or equal to the remaining slack, until a minimal explanation is produced.

Based on this intuition, we propose the smallest-absolute-weights-first removal algorithm, which is summarized in Algorithm 1. This is a very fast and simple greedy algorithm that can guarantee the simplest explanation.

Theorem 7: Algorithm 1 finds a simplest explanation for a positive instance of a threshold function.

Proof: We prove this by contradiction. Assume π_1 is the explanation generated by Algorithm 1 and π_2 is a shorter explanation. According to Algorithm 1, π_1 is a minimal explanation (prime) since further removing any literal from π_1 would cause its slack to become negative. Consider two sets of literals $\{\ell_i \mid \ell_i \in \pi_2, \ell_i \notin \pi_1\}$ and $\{\ell_j \mid \ell_j \in \pi_1, \ell_j \notin \pi_2\}$. Since π_2 is shorter than π_1, we have $|\{\ell_i \mid \ell_i \in \pi_2, \ell_i \notin \pi_1\}| < |\{\ell_j \mid \ell_j \in \pi_1, \ell_j \notin \pi_2\}|$. For π_2, keep replacing such ℓ_i with ℓ_j until there does not exist such ℓ_i and denote by π_3 the produced term. Since π_1 is generated by Algorithm 1, we always have $w_i > w_j$ for w_i and w_j corresponding to any combinations of ℓ_i and ℓ_j, respectively. Therefore, we must have $z(\pi_3) \geq 0$ and π_3 is an implicant. Further, we have $\pi_1 \Rightarrow \pi_3$, which is contradictory to the premise that π_1 is a prime. ■

C. EXPLAINING THE DISJUNCTIVE THRESHOLD NETWORK

We next consider the problem of deriving an explanation for the entire disjunctive threshold network. Since all threshold functions are combined using a logical OR operator, an explanation of a positive instance for one of the activated threshold functions is also an explanation for the whole network. Therefore, we can simply enumerate Algorithm 1 on each of the activated threshold functions and return the shortest explanation among them as an explanation for the overall network. This enumeration algorithm is also very fast and simple, as depicted in Algorithm 2.

We note that the explanation generation algorithms are neat and efficient. In particular, the time complexity of Algorithm 1 is $O(n \log n)$, which comes from the sorting algorithm, where n is the number of literals, and the total time complexity of Algorithm 2 is $O(nk \log n)$, where k is the number of threshold

Algorithm 2: Deriving explanations from DT-Net

```
  Input: Set of threshold functions $F = \{f_1, f_2, \ldots, f_k\}$,
  positive instance $\alpha$
  Output: DT-Net explanation $\tilde{\pi}$
 1: $S \leftarrow \{\}$
 2: for $f_i \in F$ do
 3:     if $f_i(\alpha) = 1$ then
 4:         $\pi_{base-exp} \leftarrow$ get base explanation of $f_i$
 5:         $\pi \leftarrow$ Algorithm 1$(f_i, \pi_{base-exp})$
 6:         $S \leftarrow S \cup \{\pi\}$
 7:     end if
 8: end for
 9: $\tilde{\pi} \leftarrow \arg\min_{\pi \in S} |\pi|$
10: return $\tilde{\pi}$
```

functions in the hidden layer. In practice, it takes less than 10 ms of CPU time to generate succinct explanations in all test cases.

IV. SIMPLIFYING EXPLANATIONS THROUGH SPARSITY-INDUCING REGULARIZATION

DT-Net can be accurately trained using well-developed stochastic gradient descent training algorithms. We use a binary cross-entropy loss function at the output, and we use a straight-through estimator with gradient clipping [24] in the hidden layer to backpropagate gradient updates through the binary step activations. It should be clear from the previous section that zero weights in a threshold function mean that the corresponding inputs will not have any effect on the logic of the threshold function, which means those input features can be removed from any explanation derived from that threshold function. Therefore, promoting the sparsity of hidden layer threshold functions indirectly simplifies explanations. Further, as shown in [27], neurons with zero input connections (meaning all its weights are zero) can be safely removed since these dead neurons will have no effect on the output classification. Besides a training strategy that maximizes the number of zero weights, encouraging weights with small absolute magnitudes is also beneficial in deriving simpler explanations. This is because more input features can be removed from an explanation if the corresponding weights have small absolute values relative to the available slack.

We can encourage sparsity by including a regularization term into the overall loss function of the form

$$\mathcal{L} = \mathcal{L}_{BCE} + \lambda \mathcal{L}_R(W), \tag{8}$$

where $\mathcal{L}_{BCE}$ is the binary cross-entropy loss, $\mathcal{L}_R(\cdot)$ is the regularization loss over the weight matrices W in the network, and λ is the regularization coefficient. Fortunately, we can encourage both zero weights and weights with small values in absolute magnitude by means of sparsity-inducing regularization. In particular, we use the reweighted L_1 regularization [28] approach that penalizes smaller absolute value weights so that they are driven towards zero faster, resulting in more weights near zero. We also incorporate a pruning method [27] to eliminate weights with absolute magnitudes below a certain threshold. Weights near this threshold that remain tend to be small so that they are more likely to be eliminated in our algorithms to derive explanations. As shown in [28], a log-sum penalty term,

$$\mathcal{L}_R(W) = \log(\|W\|_1 + \epsilon), \tag{9}$$

can be used to achieve reweighted L_1 minimization, where $\epsilon > 0$ is a small value (e.g., $\epsilon = 0.1$) added to ensure numerical stability. As shown in the evaluation section, this sparsity-inducing regularization approach not only simplifies the explanations, but it also leads to the removal of many dead neurons.

V. EVALUATION

Benchmarks: The numerical experiments were evaluated on 8 publicly available binarized classification datasets, most of which have more than 10,000 instances and comprise categorical and numerical attributes for each instance before binarization. We used three datasets from the UCI Machine Learning Repository [29], namely adult (Adult Census), magic (MAGIC Gamma Telescope), and chess (Chess: King-Rook vs. King). Two of the selected datasets are from Kaggle: churn (Telco Customer Churn) and airline (Airline Passenger Satisfaction). The other three datasets are: house (House_16H) [30], retention (TED Dataset) [31], and recidivism (Predicting Recidivism) [32]. These datasets were shuffled (with a fixed seed to ensure the consistency for all approaches) and split into 5 sets of training and test datasets using 5-fold cross-validation. All experimental results are derived by running the classifiers on 5 test sets and averaging the results.

DT-Net Configurations: For DT-Net, we used the Adam optimizer with a fixed learning rate of 10^{-2} and no weight decay across all experiments. There are 100 neurons in the hidden layer to ensure there is an efficient search space for all datasets, and the network is trained for 1,000 epochs to guarantee complete convergence. For simplicity, the batch size is fixed at 2,000 and the weights are uniformly initialized within the range between 0 and 1. Other parameters were selected according to the nested 5-fold cross-validation, which will be discussed in the following subsections.

A. CLASSIFICATION PERFORMANCE

Baselines and Pre-processing: In this evaluation, we compare our approach with four rule learners, including Decision Rule Net (DR-Net) [14], the Column-Generation-Based algorithm (CG) [5], RIPPER [2], and Bayesian Rule Sets (BRS) [4]. We also compare our approach with decision trees (CART), random forests (RF), and gradient boosting trees (XGB). RIPPER is a greedy rule mining approach based on sequential covering. DR-Net, BRS and CG are recent rule-set-generation classifiers that optimize both for accuracy and interpretability, and CART [9] is a decision tree learning algorithm. We use random forest (RF) [12] and XGBoost (XGB) [10] to provide

TABLE 1. **Average Test Accuracy and Complexity (In Parentheses) Based on the 5-Fold Cross-Validation**

dataset	DT-Net	DR-Net	CG	RIPPER	BRS	CART	RF	XGB
adult	**83.64** (11.83)	82.55 (11.46)	82.60 (6.54)	82.25 (5.16)	78.78 (3.00)	82.44 (13.11)	84.03	84.41
magic	**85.34** (4.70)	83.91 (2.73)	83.33 (2.82)	82.86 (4.55)	81.37 (3.00)	83.18 (11.98)	86.71	87.16
house	**87.61** (10.43)	86.07 (3.56)	83.80 (2.90)	81.43 (5.74)	83.26 (3.00)	85.10 (12.22)	88.49	88.92
recidivism	**65.39** (6.31)	64.09 (2.06)	64.57 (2.98)	64.84 (4.22)	61.98 (3.00)	62.85 (9.86)	66.77	64.33
chess	**89.30** (7.28)	84.47 (7.70)	81.93 (6.97)	85.46 (9.80)	74.66 (3.00)	85.36 (16.02)	92.63	94.98
retention	**93.47** (3.64)	87.78 (3.23)	90.77 (3.68)	88.92 (3.73)	89.37 (3.00)	90.11 (11.86)	93.43	94.29
churn	**79.51** (8.92)	78.85 (6.80)	79.21 (2.71)	78.27 (4.81)	76.74 (3.00)	79.00 (9.82)	80.35	77.45
airline	**94.41** (4.71)	93.32 (3.45)	90.10 (3.50)	93.08 (4.28)	90.71 (2.90)	90.21 (12.64)	94.79	95.94

baselines for typical performances that black-box models can achieve on the datasets evaluated. For all datasets, we adopted the pre-processing approach discussed in [14] to binarize numerical and categorical features. BRS and CG do not directly consider the negation of features. Therefore, we followed the procedures described in their articles to append the negations of binarized features so that they can be considered in their rule sets.

Complexity Measurement: For our model and other interpretable models, the classification performance was evaluated using both accuracy and interpretability. The accuracy was evaluated on the test set and the interpretability was measured by the average explanation complexity. We note that while rule learners generally consider the complexity of generated rules, our model carries out the prediction without pre-learning any rules, but derives the explanation afterwards. Therefore, we proposed a new complexity metric, namely explanation complexity, as the average length of explanations for all positive instances in the test dataset. For DT-Net, the explanations were produced according to the algorithm discussed in Section III-C and therefore the complexity is the length of the explanation. For rule learners, the complexity was computed based on the simplest rule that covers the test instance. For CART, the explanation was derived by tracing down the decision path from the root node, and the complexity is measured by the number of nodes in the decision path.

Parameter Tuning: We evaluated the predictive performance of DT-Net by comparing both test accuracy and complexity with other state-of-the-art machine learning models. For parameter selection in all models, we used a 5-fold nested cross-validation to improve training accuracy. Specifically, the best accuracy is achieved by tuning the regularization coefficient λ for DT-Net, minimum number of samples per leaf for CART and RF, and the regularization term for XGBoost. We tuned the same parameters mentioned in [14] for DR-Net, CG, RIPPER and BRS. We take the average performance over the 5 training-testing pairings as the final reported results. We summarize the accuracies of all models and the complexities of the interpretable models in Table 1, where the best accuracies among interpretable models are highlighted in bold.

As can be observed in Table 1, DT-Net achieves the best accuracy among all interpretable models on all datasets, indicating that DT-Net has a significant advantage over other interpretable models on generalization capability. At the same time, DT-Net achieves an accuracy very close to the uninterpretable models on most datasets (except for the chess dataset) and it even outperforms them in some cases (see churn and airline datasets). Moreover, since DT-Net is a neural network, the performance of DT-Net can possibly be further improved with finer parameter tuning and more advanced training techniques. Regarding complexity, it can be seen that though DT-Net does not produce the simplest explanations, it still shows admissible interpretability in that its complexity is within the same magnitude of other approaches. In particular, we note that DT-Net always outperforms the traditional decision tree on all datasets and thus in real-world applications, DT-Net can generally substitute decision trees with both higher accuracy and lower complexity.

TABLE 2. **Statistics of DT-Net After Pruning. # pruned: The Number of Neurons Pruned in the Hidden Layer**

dataset	# pruned	Sparsity	# activated
adult	96.2	81.21%	1.49 (0.68)
magic	87.8	82.81%	2.84 (1.63)
house	90.2	70.95%	3.63 (1.78)
recidivism	95.0	76.00%	1.12 (0.36)
chess	62.2	58.50%	2.75 (1.82)
retention	67.8	80.27%	3.91 (2.77)
churn	96.8	70.90%	1.34 (0.51)
airline	87.2	87.92%	4.48 (2.03)

Sparsity: The Percentage of the Zero Weights in Each Neuron. # activated: The Number of Activated Neurons for Each Test Instance. The Average Over 5 Partitions is Reported for Every Dataset (Standard Deviation in Parentheses).

B. EFFECTS OF SPARSITY-INDUCING REGULARIZATION

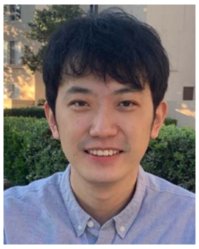
In our experiments, the networks are composed of a large number of threshold functions, e.g. 100 neurons in the hidden layer, to ensure enough capacity. Simplifying the neural network using the regularization and pruning methods mentioned earlier helps reduce both the complexity and the computation time of the explanations. We show in Table 2 that our neural network achieves very high sparsity, which partially explains why our explanation generation procedure has relatively low computational cost.As can be observed from Table 2, most threshold functions are disabled after pruning, which verifies the effectiveness of our regularization method in excluding the redundant capacity. Further, the remaining neurons generally achieve an average sparsity over 50%. This means that more than half of the literals are directly removed before applying our algorithms, which explains how we remove most literals and generate explanations with reasonable complexity in our experiments. In addition, individual positive instances in general are only produced by about 2 or 3 neurons for all datasets. The fact that each instance only activates a few neurons explains why our explanation generation algorithm generally produces explanations that are minimal for the network.

VI. CONCLUSION

We proposed in this work a neural network architecture called DT-Net for tabular data classification that provides both high accuracy performance and interpretability. An important feature of the proposed solution is that only a simple greedy algorithm is required to provide an explanation with the prediction that is human-understandable. We further employ a sparsity-inducing regularization approach to sparsify the threshold functions so that the derived explanations are simple. In comparison with other explainable decision models, our evaluation shows that our proposed approach can achieve superior predictive performance on a broad set of tabular data classification datasets.

REFERENCES

[1] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Mach. Intell.*, vol. 1, no. 5, pp. 206–215, 2019.
[2] W. W. Cohen, "Fast effective rule induction," in *Proc. 12th Int. Conf. Mach. Learn.*, 1995, pp. 115–123.
[3] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 1675–1684, doi: 10.1145/2939672.2939874.
[4] T. Wang, C. Rudin, F. Doshi-Velez, Y. Liu, E. Klampfl, and P. MacNeille, "A Bayesian framework for learning rule sets for interpretable classification," *J. Mach. Learn. Res.*, vol. 18, no. 70, pp. 1–37, 2017. [Online]. Available: http://jmlr.org/papers/v18/16-003.html
[5] S. Dash, O. Günlük, and D. Wei, "Boolean decision rules via column generation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31.
[6] M. Li et al., "An adaptive framework for confidence-constraint rule set learning algorithm in large dataset," in *Proc. 31st ACM Int. Conf. Inf. Knowl. Manage.*, 2022, pp. 3252–3261, doi: 10.1145/3511808.3557088.
[7] R. L. Rivest, "Learning decision lists," *Mach. Learn.*, vol. 2, no. 3, pp. 229–246, 1987.
[8] B. Letham, C. Rudin, T. H. McCormick, and D. Madigan, "Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction model," *Ann. Appl. Statist.*, vol. 9, no. 3, pp. 1350–1371, Sep. 2015, doi: 10.1214/2F15-aoas848.
[9] L. Breiman, J. Friedman, C. Stone, and R. Olshen, *Classification and Regression Trees (wadsworth and brooks-cole statistics-probability series)*. Taylor & Francis, New York, NY, USA, 1984. [Online]. Available: https://books.google.com/books?id=JwQx-WOmSyQC
[10] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 785–794.
[11] G. Ke et al., "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3149–3157.
[12] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324..
[13] J. Chen, K. Liao, Y. Wan, D. Z. Chen, and J. Wu, "DANets: Deep abstract networks for tabular data classification and regression," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 3930–3938. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/20309
[14] L. Qiao, W. Wang, and B. Lin, "Learning accurate and interpretable decision rule sets from neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 4303–4311.
[15] A. Abutbul, G. Elidan, L. Katzir, and R. El-Yaniv, "DNF-Net: A neural architecture for tabular data," 2020, *arXiv: 2006.06465*.
[16] S. Ö. Arik and T. Pfister, "TabNet: Attentive interpretable tabular learning," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 6679–6687.
[17] A. Shih, A. Choi, and A. Darwiche, "A symbolic approach to explaining Bayesian network classifiers," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 5103–5111.
[18] W. Shi, A. Shih, A. Darwiche, and A. Choi, "On tractable representations of binary neural networks," in *Proc. Int. Conf. Princ. Knowl. Representation Reasoning*, 2020, pp. 882–892.
[19] A. Choi, W. Shi, A. Shih, and A. Darwiche, "Compiling neural networks into tractable Boolean circuits," in *Proc. AAAI Spring Symp. Verification Neural Netw. (VNN)*, 2019.
[20] A. Ignatiev, N. Narodytska, and J. Marques-Silva, "Abduction-based explanations for machine learning models," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 1511–1519.
[21] Y. Izza and J. M. Silva, "On explaining random forests with SAT," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, 2021.
[22] G. Audemard, F. Koriche, and P. Marquis, "On tractable XAI queries based on compiled representations," in *Proc. 17th Int. Conf. Princ. Knowl. Representation Reasoning*, 2020, pp. 838–849, doi: 10.24963/kr.2020/86.
[23] S. Ruder, "An overview of gradient descent optimization algorithms," 2016, *arXiv:1609.04747*.
[24] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.
[25] C. Louizos, M. Welling, and D. P. Kingma, "Learning sparse neural networks through l_0 regularization," in *Proc. Int. Conf. Learn. Representations*, 2018.
[26] Y. Li and S. Ji, "l_0-arm: Network sparsification via stochastic binary optimization," in *Proc. Mach. Learn. Knowl. Discov. Databases: Eur. Conf.*, 2020, pp. 432–448.
[27] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.
[28] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted l_1 minimization," *J. Fourier Anal. Appl.*, vol. 14, pp. 877–905, 2008.
[29] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml
[30] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "OpenML: Networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013, doi: 10.1145/2641190.2641198..
[31] V. Arya et al., "One explanation does not fit all: A toolkit and taxonomy of AI explainability techniques," in *Proc. INFORMS Annu. Meeting*, 2021.
[32] P. Schmidt and A. D. Witte, *Predicting Recidivism in North Carolina 1978 and 1980*. Ann Arbor, MI, USA: Inter-Univ. Consortium Political Social Res., 1988.



WEIJIA WANG received the B.S. degree in electrical engineering from Zhejiang University, Hangzhou, China, in 2016, and the M.S. degree in electrical and computer engineering in 2018 from the University of California, San Diego, La Jolla, CA, USA, where he is currently working toward the Ph.D. degree in electrical and computer engineering. His research focuses on machine learning and deep learning, including the compression and acceleration of deep convolutional neural networks, algorithms of meta-learning and federated learning, and explainable artificial intelligence.

LITAO QIAO received the B.S. and M.S. degrees in computer and science and engineering in 2019 and 2020, respectively, from the University of California, San Diego, La Jolla, CA, USA, where he is currently working toward the Ph.D. degree in electrical and computer engineering. His research interests include data science, explainable machine learning, and deep learning.

BILL LIN received the B.S., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, Berkeley, CA, USA, in 1985, 1988, and 1991, respectively. He is currently a Professor in electrical and computer engineering with the University of California, San Diego, La Jolla, CA, where he is actively involved with the Center for Wireless Communications, Center for Networked Systems, and Qualcomm Institute in industry-sponsored research efforts. His research has led to more than 200 journal and conference publications, including a number of Best Paper awards and nominations. He also holds five awarded patents. He was the General Chair and on the executive and technical program committee of many IEEE and ACM conferences. He was an Associate or Guest Editor for several IEEE and ACM journals.