Deep Neural Networks Constrained by Decision Rules

Yuzuru Okajima, Kunihiko Sadamasa
NEC Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa 211-8666, Japan
y-okajima@bu.jp.nec.com, k-sadamasa@az.jp.nec.com

Abstract

Deep neural networks achieve high predictive accuracy by learning latent representations of complex data. However, the reasoning behind their decisions is difficult for humans to understand. On the other hand, rule-based approaches are able to justify the decisions by showing the decision rules leading to them, but they have relatively low accuracy. To improve the interpretability of neural networks, several techniques provide post-hoc explanations of decisions made by neural networks, but they cannot guarantee that the decisions are always explained in a simple form like decision rules because their explanations are generated after the decisions are made by neural networks.

In this paper, to balance the accuracy of neural networks and the interpretability of decision rules, we propose a hybrid technique called *rule-constrained networks*, namely, neural networks that make decisions by selecting decision rules from a given ruleset. Because the networks are forced to make decisions based on decision rules, it is guaranteed that every decision is supported by a decision rule. Furthermore, we propose a technique to jointly optimize the neural network and the ruleset from which the network select rules. The log likelihood of correct classifications is maximized under a model with hyper parameters about the ruleset size and the prior probabilities of rules being selected. This feature makes it possible to limit the ruleset size or prioritize human-made rules over automatically acquired rules for promoting the interpretability of the output. Experiments on datasets of time-series and sentiment classification showed rule-constrained networks achieved accuracy as high as that achieved by original neural networks and significantly higher than that achieved by existing rule-based models, while presenting decision rules supporting the decisions.

Introduction

Deep neural networks have achieved great success when applied to a wide variety of tasks. Their high predictive accuracy stems from their ability to learn latent representations automatically from complex input data. However, the success of deep neural networks has renewed interest and discussion on their long-standing problem: "uninterpretability." While the decisions of neural networks are highly accurate, the reasons behind their decisions are difficult for humans to interpret. This lack of interpretability could hinder practical use of neural networks in real-world applications; thus, many techniques to give post-hoc explanations of the decisions of deep neural networks have been proposed (Montavon, Samek, and Müller 2018). As for these approaches, an explanation of a decision made by a trained neural network is generated by analyzing input features relevant to the decisions (Simonyan, Vedaldi, and Zisserman 2013; Bach et al. 2015) or approximating the behavior of the neural network with simpler models (Ribeiro, Singh, and Guestrin 2016; Zilke, Loza Mencía, and Janssen 2016).

On the other hand, the growing interest in interpretability has also brought renewed attention to rule-based models. The high computational power of modern processors has enabled non-greedy optimization of decision lists (Letham et al. 2015; Wang and Rudin 2015; Angelino et al. 2017; Yang, Rudin, and Seltzer 2017). Such optimized decision lists are highly interpretable and achieve better accuracy than traditional rule-based models for several datasets. Despite the recent improvement of rule-based models, however, they are less accurate than uninterpretable models because they cannot exploit the power of uninterpretable features.

In this study, which aims to balance the accuracy of neural networks and the interpretability of decision rules, we propose a new model called a rule-constrained network (RCN). RCNs are neural networks trained to make decisions by selecting decision rules. Given an observed instance, the RCNs do not directly predict its class label; instead, they select a decision rule from a given decision rule set so that the observation satisfies the antecedent of the rule and the consequent gives a high probability to the correct class. An attention mechanism using latent representations of rules is incorporated into the RCN to exploit similarity between rules and to reduce the number of parameters. In addition, we propose a method for optimizing the rule set from which the neural networks select a rule. The optimization is achieved by combining a RCN with a rule-sampling model that models the ruleset size and the probabilities of the rules being sampled.

An advantage of RCNs is that the decisions are confined within the reach of the decision rules. For every decision, a RCN is forced to select one rule that supports the decision. Thus, any decisions that cannot be justified by decision rules are never output. This guarantee of the existence of supporting rules is a feature of RCNs in contrast to post-hoc ex-

planation methods, which cannot guarantee the existence of such rules because their explanations are generated after the decisions are made by neural networks. Although the decisions are limited by decision rules, a RCN has the potential to overcome the poor accuracy of pure rule-based models because rules are selected by a neural network considering uninterpretable features. Finally, the decisions are controllable by changing the hyper parameters of the rule-sampling model with respect to the ruleset size and the probabilities of the rules being sampled. This feature makes it possible to keep the distinct number of rules lower than a given limit or give priority to human-made rules over automatically acquired rules for promoting the interpretability of the output.

RCNs were evaluated in experiments on time-series and sentiment classification. They showed accuracy as high as the original neural networks and significantly higher than existing rule-based models, while presenting rules supporting the decisions. The attention mechanism improved the accuracy of the RCN. The ruleset was optimized according to the hyper parameters that limit the distinct numbers of rules or prioritize human-made rules over automatically acquired rules.

Rule-constrained networks

In this section, we will explain how RCNs are constructed from a base neural network given a rule set. We propose two variants, RCN-A and RCN+A.

Let us consider a classification task in which we are given a raw or unprocessed data representation, $\boldsymbol{x}$, and its class, y. Feature engineering is the task of designing a mapping function, g, so that the classifier can predict y given a feature representation, $\boldsymbol{x'} = g(\boldsymbol{x})$, as input. Here, we focus on sequence classification problems like time-series or sentiment classification. Most traditional classification models, like decision trees or random forests, cannot handle sequences directly. Instead, they require the sequences to be abstracted into an unsequenced vector composed of feature values beforehand. For example, in the case of time series classification, $\boldsymbol{x}$ is a time series consisting of observed values for all time points. Then, we extract from $\boldsymbol{x}$ a vector composed of feature values as $\boldsymbol{x'}$, e.g., the mean, variance or slope of each interval extracted from the time series with a certain window size. In the case of sentiment classification, $\boldsymbol{x}$ is a sequence of words in the target text, and $\boldsymbol{x'}$ is its bag-of-words representation. If the features required for classification are simple enough, a human expert may be able to design features manually so that the classification model attains good performance. However, practical problems often requires such complex features that manual feature engineering fails to achieve good performance.

Neural networks can avoid this difficulty of feature engineering by representation learning. When a network is trained to output y given $\boldsymbol{x}$, its hidden layers learn abstract latent representations of $\boldsymbol{x}$. Representation learning not only eliminates the cost of manual feature engineering but also achieve better classification accuracy in many tasks. However, automatically created latent representations are difficult for humans to understand and significantly degrade the interpretability of the decisions made by neural networks.

Rule	Antecedent (Condition)	Consequent (Class probability)
r_0	$f_0 =$ "A"	$[0.2, 0.8]$
r_1	$f_0 =$ "B"	$[0.8, 0.2]$
r_2	$f_1 > 2$	$[0.3, 0.7]$
r_3	$f_1 > 2.5$	$[0.2, 0.8]$
r_4	$f_0 =$ "A" AND $f_1 > 2$	$[0.1, 0.9]$
r_5	$f_0 =$ "B" AND $f_1 \leq 2$	$[0.9, 0.1]$

Figure 1: An example of a decision-rule set. The antecedents are defined for feature representation $\boldsymbol{x'} = [f_0, f_1]^T$. The consequents are binary class probabilities.

We propose RCNs as a solution to this trade-off between interpretability and accuracy. RCNs are prohibited from predicting a class label directly; instead, they predict a rule so that the observation satisfies the antecedent and the consequent gives high probability to the correct class.

We are given a set of n training instances $D = \{(\boldsymbol{x}_i, y_i)\}_{i=0}^{n-1}$ and manually designed feature function g. Furthermore, it is assumed that a set of decision rules, $R = \{r_j\}_{j=0}^{|R|-1}$, are given. The antecedents are conditions about features, and the consequents are class probabilities. An example of a decision-rule set is shown in Fig. 1. Such rule sets can be premined from D by frequent itemset mining techniques like Eclat (Zaki et al. 1997) and FP-Growth (Han, Pei, and Yin 2000). Recent decision list algorithms (Letham et al. 2015; Wang and Rudin 2015; Angelino et al. 2017; Yang, Rudin, and Seltzer 2017) also assume the existence of premined rules. Note that it is also possible to use random forests built from training data as a decision-rule set because each path from the root to a leaf can be seen as a decision rule. Any other resources can be used as a rule set. For example, a rule set for sentiment classification can be built from a dictionary of positive and negative words.

The consequents of rules are determined as follows. Let $n_{j,l}$ denote the count of training instances that satisfy the antecedent of rule r_j and belong to class $y = l$. The consequent of rule r_j is the class probability defined by

$$p(y = l | z = r_j) = \frac{n_{j,l} + \alpha}{\sum_{l'} n_{j,l'} + \alpha} \tag{1}$$

where α is a smoothing factor. This simple count-based definition makes it easy for humans to check the correctness of rules by counting the instances satisfying the antecedents.

Given a rule set, the RCN takes raw data $\boldsymbol{x}$ and its feature representation $\boldsymbol{x'} = g(\boldsymbol{x})$ as input. Then, it selects a rule from the rules that are included in R and have antecedents satisfied by $\boldsymbol{x'}$. While the rules are just defined over the unsequenced feature representation, the RCN selects a rule by exploiting information in the original data, $\boldsymbol{x}$, that may be lost in $\boldsymbol{x'}$. This procedure helps achieve higher performance than only using the rules about manual features.

Let us explain how the RCN is constructed from the base network. The base network is a neural network for k-class classification of a given task, composed of a base layer of size $(|x|, h)$, a linear layer of size (h, k), and a softmax function, as shown in Fig. 2 (a). In this paper, we describe a layer

has size (n_{in}, n_{out}) when it receives a vector of size n_{in} and outputs a vector of size n_{out}. The base layer is a layer or a neural network of any type suitable for the given task, for example, a CNN for time-series classification and a bi-directional LSTM (Bi-LSTM) for sentiment classification.

A RCN is constructed by changing the base network so that it does not predict the class label directly, but predicts the rule instead. We propose two variants of the RCN. The first variant, called RCN-A, which means RCN *without attention*, is constructed as follows. First, the linear layer of size (h, k) in the base network is replaced with an linear layer of size $(h, |R|)$ whose output vector corresponds to the rules in R. Next, we change the softmax activation. We only apply softmax activation to the elements of the output vector corresponding to the rules having antecedents satisfied by $\boldsymbol{x'}$ and simply ignore the other elements. We call this operation as "filtered softmax." This operation ensure that the antecedent of the selected rule is always true for the input instance. By changing the linear layer and softmax activation of the base network, we obtain RCN-A shown in Fig. 2 (b). RCN-A receives $\boldsymbol{x}$ and $\boldsymbol{x'}$ and returns $p(z|\boldsymbol{x}, \boldsymbol{w})$ [1], namely, the probability of the rules being selected by the network, where z is a probabilistic variable for rules, and $\boldsymbol{w}$ represents the parameters of the neural network. $p(z|\boldsymbol{x}, \boldsymbol{w})$ is set to 0 if rule z has antecedents not satisfied by $\boldsymbol{x'}$.

Here, we explain how to train RCNs without optimization of the rule set. The RCN can be trained by maximizing the log likelihood by using backpropagation as follows:

$$\max \sum_{(\boldsymbol{x}_i, y_i) \in D} \log p(y_i|\boldsymbol{x}_i, \boldsymbol{w})$$

The conditional probability of y is computed by marginalizing out rules as follows:

$$p(y_i|\boldsymbol{x}_i, \boldsymbol{w}) = \sum_{z \in R} p(y_i|z) p(z|\boldsymbol{x}_i, \boldsymbol{w})$$

In the test phase, when a test instance is given as $\boldsymbol{x}$, the rule z^* with the maximum conditional probability given $\boldsymbol{x}$ is selected via $z^* = \arg\max_z p(z|\boldsymbol{x}, \boldsymbol{w})$, and then the class y^* is predicted by taking the class with the maximum conditional probability given z^* via $y^* = \arg\max p(y|z^*)$. Thus, the predicted class is always presented with one rule supporting the decision.

Next, we introduce the second variant, called RCN+A, which means RCN *with attention*. RCN+A has an attention mechanism that select rules based on the latent representations of the rules. RCN+A is shown in Figs. 2 (c-1) and (c-2). Consider matrix V of shape $(|R|, h)$ where the j-th row represents the latent representation of rule r_j. The attention over the rules is obtained by computing the product of V and the output vector of the linear layer. Then, the attention is used as the input of the filtered softmax. The key point of RCN+A is how to obtain V, the latent representations of the rules. We propose to learn the latent representation of a rule in the form of the mean of the linearly transformed latent representations of the instances that satisfies the precedent of the rule. The aim of this formulation is to give rules similar latent representations if they have precedents satisfied by similar instances. V is computed as shown in Fig. 2 (c-2). We precompute truth value matrix T of shape $(|R|, n)$, where $T_{j,i} = 1$ if $\boldsymbol{x}'_i$ satisfies the precedent of r_j and $T_{j,i} = 0$ otherwise. Let $Norm(T)$ denote T normalized as the sum of each row equals 1. Finally, let X denote a matrix with n rows, where the i-th row is $\boldsymbol{x}_i$. Then, we obtain $V = Norm(T)Linear(Base(X))$, which can be calculated in neural networks as shown in Fig. 2 (c-2). RCN+A can be trained in the same way as RCN-A. RCN+A can take into account the similarity between rules in training. Furthermore, introducing the attention mechanism reduces the number of parameters because the size of the linear layer is changed from $(h, |R|)$ to (h, h). These features improved the accuracy in the experiments.

[1] $\boldsymbol{x'}$ is omitted from the condition of probability because $\boldsymbol{x'}$ is determined by $\boldsymbol{x}$

Ruleset optimization

In this section, we optimize the rule set from which the RCN selects rules. The RCN and the rule set are trained jointly to maximize the log likelihood of correct classifications.

An automatically acquired rule set may include abundant similar or redundant rules. If the RCN is allowed to select any rules, similar rules may be inconsistently selected, and the distinct number of rules increases as new instances are given. This could harm the consistency of explanations and increase the cost of checking if the rules are reliable. Thus, we propose a framework for extracting a sufficient subset of rules for the predictions, within the limit of the distinct number of rules, from the original, possibly redundant ruleset.

The rule set is optimized by combining the RCN with a rule-sampling model. First, we sample a subset of the original rule set R, which is called the selected rule set. Then, the RCN makes a decision for each instance by selecting a rule from the selected rule set. Our goal is to optimize the selected rule set in the training phase so that it contains useful rules allowing the RCN to make decisions as much as possible under the given sampling model. The sampling model consists of the following two stages:

1. Choose $\boldsymbol{\theta} = \boldsymbol{c}/\boldsymbol{\lambda}$ where $\boldsymbol{c} \sim Multi(\boldsymbol{\theta}_0, \lambda)$
2. For each instance:
   - Choose $z \sim Categorical(\boldsymbol{\theta})$

First, the user specifies $\boldsymbol{\theta}_0$, which represents the parameters of the multinomial distribution that represent the probability of the rules to be sampled, and λ, which represents the number of the rules to be sampled. The selected rule set R_θ is sampled by extracting λ rules according to $\boldsymbol{\theta}_0$ with replacement from the original rule set R. Let $\boldsymbol{c}$ denote a count vector with length $|R|$, where the j-th element is the count of r_j being sampled. Then, we set $\boldsymbol{\theta} = \boldsymbol{c}/\lambda$. For each instance in the training data, rule z is sampled from $Categorical(\boldsymbol{\theta})$. Thus, rules are sampled twice. The first sample determines the subset of rules to be used, and the second sample determines the rules used for instances.

The above-described rule-sampling model is combined with the RCN as follows.

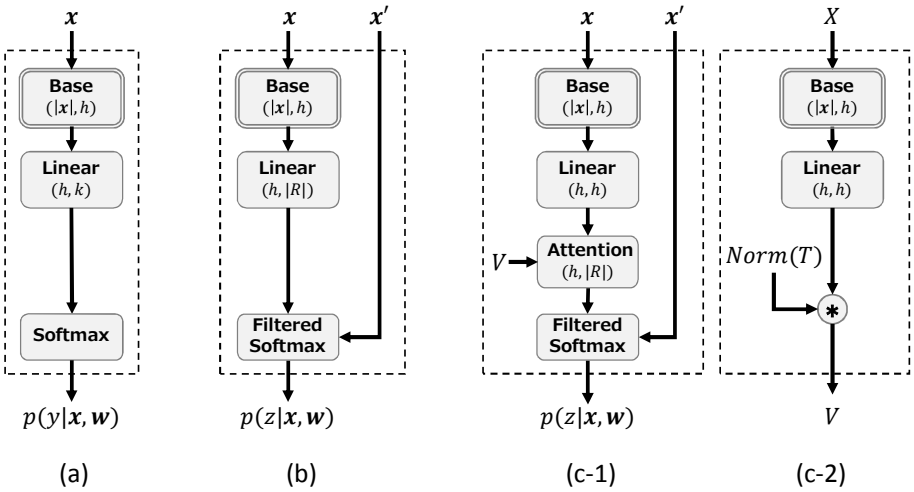


Figure 2: (a) Base network, (b) RCN-A and (c) RCN+A

$$p(y|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta}_0, \lambda) = \sum_{\boldsymbol{\theta}} p(y|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\theta}_0, \lambda) \quad (2)$$

$$p(y|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta}) = \sum_{z \in R} p(y|z) p(z|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta}) \quad (3)$$

$$p(z|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta}) = \frac{p(z|\boldsymbol{x}, \boldsymbol{w}) p(z|\boldsymbol{\theta})}{\sum_{z' \in R} p(z'|\boldsymbol{x}, \boldsymbol{w}) p(z'|\boldsymbol{\theta})} \quad (4)$$

Equations (2) and (3) indicate that the posterior probability of y is marginalized over $\boldsymbol{\theta}$ and z. Equation (4) indicates that the posterior probability of z is a normalized product of $p(z|\boldsymbol{x}, \boldsymbol{f}, \boldsymbol{w})$, namely, the posterior based on the RCN, and $p(z|\boldsymbol{\theta})$, namely, the posterior based on the sampling model. The normalized product is called the "product of experts" (Hinton 2002). If rule r_j is not included in $R_{\boldsymbol{\theta}}$, $p(z = r_j|\boldsymbol{\theta})$ is zero. Thus, the RCN combined with the sampling model as in (4) is forced to select a rule from $R_{\boldsymbol{\theta}}$.

We want to train the RCN and find θ that maximizes the likelihood under this combined model. We use the generalized EM algorithm. Let X and Y denote the set of $\boldsymbol{x}_i$ and y_i in D. We calculate the expected log likelihood with respect to $p(\boldsymbol{\theta}|Y, X, \boldsymbol{w}, \boldsymbol{\theta}_0, \lambda)$ in the E-step, and we update $\boldsymbol{w}$ in the M-step. However, the posterior probability of $\boldsymbol{\theta}$ is intractable because $\boldsymbol{\theta}$ can take $|R|^\lambda$ values. The posterior is thus approximated by sampling $\boldsymbol{\theta}$ based on the Metropolis–Hastings algorithm. The acceptance rate is given as

$$A(\boldsymbol{\theta}'|\boldsymbol{\theta}) = min\left(1, \frac{p(\boldsymbol{\theta}'|Y, X, \boldsymbol{w}, \boldsymbol{\theta}_0, \lambda))}{p(\boldsymbol{\theta}|Y, X, \boldsymbol{w}, \boldsymbol{\theta}_0, \lambda))} \frac{g(\boldsymbol{\theta}|\boldsymbol{\theta}')}{g(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right) \quad (5)$$

$$= min\left(1, \frac{p(Y|X, \boldsymbol{w}, \boldsymbol{\theta}') p(\boldsymbol{\theta}'|\boldsymbol{\theta}_0, \lambda))}{p(Y|X, \boldsymbol{w}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\theta}_0, \lambda))} \frac{g(\boldsymbol{\theta}|\boldsymbol{\theta}')}{g(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right) \quad (6)$$

where $g(\boldsymbol{\theta}'|\boldsymbol{\theta})$ is a proposal distribution, and

$$p(Y|X, \boldsymbol{w}, \boldsymbol{\theta}) = \prod_{(\boldsymbol{x}_i, y_i) \in D} p(y_i|\boldsymbol{x}_i, \boldsymbol{w}, \boldsymbol{\theta})$$

is computed by (3) and (4).

The proposal distribution, $g(\boldsymbol{\theta}'|\boldsymbol{\theta})$, generates $\boldsymbol{\theta}'$ from $\boldsymbol{\theta}$ in the following way. Because $\boldsymbol{\theta} = \boldsymbol{c}/\lambda$ is generated by $Multi(\boldsymbol{\theta}, \lambda)$, $\boldsymbol{c}$ has λ counts in total. $\boldsymbol{c}'$ is generated from $\boldsymbol{c}$ by removing one count chosen randomly from λ counts, and one count is added to the rule that is chosen according to probabilities proportional to

$$p(z|\boldsymbol{\theta}_0) \sum_{\boldsymbol{x}_i \in D} p(z|\boldsymbol{x}_i, \boldsymbol{w}). \quad (7)$$

Then, $\boldsymbol{\theta}' = \boldsymbol{c}'/\lambda$ is obtained. The ratio of the proposal probabilities in (6), $g(\boldsymbol{\theta}|\boldsymbol{\theta}')/g(\boldsymbol{\theta}'|\boldsymbol{\theta})$, is calculated based on (7). This proposal distribution is designed to add a likely rule with respect to both the sampling model and the RCN. Thus, generated proposals are more likely to be accepted than uniformly random choices. We repeat generation until we obtain a set of s accepted rules, Θ. Here, s represents the sample size. Then, $\boldsymbol{w}$ is updated to maximize the following expected log likelihood:

$$-\frac{1}{|\Theta|} \sum_{\theta \in \Theta} \log p(Y|X, \boldsymbol{w}, \boldsymbol{\theta}) \quad (8)$$

A pseudocode is given in Algorithm 1. After the training, we take $\boldsymbol{\theta}$ with the maximum posterior probability. In the test phase, just one rule $z = \arg\max_z p(z|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta})$ is chosen for each test instance. The distinct number of rules used in the test phase is guaranteed to be less than or equal to λ.

Experiments

Setup

We evaluated RCNs from the viewpoint of two sequence-classification tasks: time-series and sentiment classification. RCNs were compared with CART, random forest (RF), SVMs with RBF kernel, and scalable Bayesian rule list (SBRL) (Yang, Rudin, and Seltzer 2017), and the base networks the RCNs were based on. SBRL is the state-of-the-art

Algorithm 1 Generalized EM algorithm

for $t = 0$ to $T - 1$ **do**
  for $i = 0$ to n **do**
    Obtain $p(z|\boldsymbol{x}, \boldsymbol{w})$ by forward propagation
  $\Theta \Leftarrow \emptyset$
  Choose $\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\boldsymbol{\theta}_0, \lambda)$
  while $|\Theta| < s$ **do**
    Choose $\boldsymbol{\theta}' \sim g(\boldsymbol{\theta}'|\boldsymbol{\theta})$
    for $i = 0$ to n **do**
      Obtain $p(y_i|\boldsymbol{x}_i, \boldsymbol{w}, \boldsymbol{\theta})$ by (3,4)
    Randomly accept $\boldsymbol{\theta}$ by (5,6)
    If accepted, add θ' to Θ, and $\theta \Leftarrow \theta'$
  Compute expected likelihood by (8)
  Update $\boldsymbol{w}$ by backpropagation

technique for rule list optimization scalable for thousands of rules. We used scikit-learn implementations of CART, RF and SVMs. The number of decision trees in RF was set to 100. We used the authors' R implementation of SBRL, which works with a rule set premined by Eclat (Zaki et al. 1997). For comparison, both SBRL and RCNs were given the same rule set mined by Eclat with the minimum support set to 0.1 and the maximum itemset length set to 2 for time series and 1 for sentiment classification. Furthermore, to check the effect of the difference of rule sets, we also ran RCNs with another rule set extracted by RF with the maximum depth set to 4, considering each path is a rule. Both the Eclat and RF rulesets were added a default rule whose antecedent is always true for any instances so that at least one rule is selected by RCNs for each instance. The consequents were computed by (1); the smoothing factor α was set to 1. RCNs are trained with the Adam optimizer with its default parameters (Kingma and Ba 2015). The hyperparameters of CART, RF and SVMs were selected from Table 2 by grid search with five-fold cross validation using training data. The hyperparameter λ of SBRL was also selected from $10^{0.5}$, 10, $10^{1.5}$, 10^2, $10^{2.5}$, and 10^3 by validation. Neural networks like CNN and LSTM can directly handle sequence data $\boldsymbol{x}$ as input. The other algorithms cannot directly handle sequence data and require feature representation $\boldsymbol{x'}$ as input. Only RCNs require both $\boldsymbol{x}$ and $\boldsymbol{x'}$ as input.

The base networks used were CNN for time series (Wang, Yan, and Oates 2016) and Bi-LSTM for sentiment classification (Johnson and Zhang 2016). The base networks were pretrained, and we built RCNs based on the base networks, while the weights of the base networks were fixed in the training of the RCNs. The hidden layer size, h, was 128 for CNN and 512 for Bi-LSTM. The RCNs were first trained without rule set optimization, i.e., using all rules in R for time-series and sentiment classification. After that, they were trained with rule set optimization with sample size s set to 100.

The datasets we used are summarized in Table 1. For time-series classification, we used the top five largest binary classification data from the UCR time series repository (Chen et al. 2015). As feature representation $\boldsymbol{x'}$, we extracted the means, variances, and slopes of the intervals obtained by dividing the time series with different lengths. Because Eclat requires discretized input, the continuous-value features were discretized into five intervals with equal frequencies before applying Eclat. For sentiment classification, three review datasets were used: IMDB for movies, Elec for electronics products (Maas et al. 2011), and Yelp for local businesses[2]. Bag-of-words representations were used as $\boldsymbol{x'}$, composed of the top 1000 words with the highest ANOVA F-values with respect to classes.

Table 1: Five time-series and three sentiment datasets. The columns represent the number of training instances, test instances, features, and rules created by Eclat and RF. Straw, Prox and Phar represent Strawberry, ProximalPhalanxOutlineCorrect and PharangesOutlinesCorrect, respectively.

Name	#Train	#Test	#Feat.	#Eclat	#RF
Straw	370	613	208	83905	1754
Prox	600	291	48	4241	2200
FordB	810	3636	412	37789	2020
FordA	1320	3601	412	33817	2380
Phar	1800	858	48	3137	2364
IMDB	25000	25000	1000	2633	2690
Elec	25000	25000	1000	2543	2708
Yelp	25000	25000	1000	2551	2804

Table 2: Hyperparameters of CART, RF, and SVMs. 'None' means no restriction.

CART	
criterion	'gini', 'entropy'
max_depth	4, 8, 12, 16, None
min_samples_leaf	1, 5, 10, 20, 50, 100
RF	
criterion	'gini', 'entropy'
max_depth	4, 8, 12, 16, None
min_samples_leaf	1, 5, 10, 20, 50, 100
max_features	'sqrt', None
SVM	
C	0.1, 1, 10, 100, 1e+3, 1e+4, 1e+5, 1e+6

Comparison of error rates

We classified the algorithms to be compared into three classes: *transparent*, *rule-supported*, and *no-rule*. First, an algorithm is called transparent if its internal mechanism is simple enough for humans to understand. CART and SBRL are transparent. Second, an algorithm is called rule-supported if it is not transparent, but it is able to present at least one rule that supports the decision. RF is classified as rule-supported here, because its decisions are based on majority vote, and it can always find at least one rule that

[2]Training and test data were extracted from the original Yelp dataset (https://www.yelp.com/dataset/challenge) so that the Yelp dataset has the same sizes as the IMDB and Elec datasets.

supports the decision. A RCN is also a rule-supported one; however, in contrast to RF, RCNs can select just one rule that explains the decision for each instance, and they can limit the total number of distinct rules for the whole instance set. Third, an algorithm is called no-rule if it cannot show any decision rules. The base networks and SVM are no-rule.

Because we created two different rulesets, our proposed algorithms now consist of four variants: RCN+A with RF, RCN+A with Eclat, RCN-A with RF, and RCN-A with Eclat. For simplicity, we abbreviate "with the RF (Eclat) ruleset" as "with RF (Eclat)". Among these four, RCN+A with RF was considered as our primary algorithm because it can utilize the attention mechanism and the power of RF rules, and it was compared with the others.

The test error rates are listed in Table 3. The error rate is equal to 1.0 minus accuracy. Thus, low error rate means high accuracy. The error rates were averaged over 5 runs, and the standard errors were smaller than or on the order of the least significant digit. For the results listed in Table 3, the RCNs were trained without ruleset optimization; that is, they were allowed to select any rules in R without a restriction on the distinct number of rules. The numbers in parentheses represent the averaged distinct numbers of rules selected by the RCNs and contained in CART and SBRL. The minimum error rate for each data set is indicated in boldface. The averaged ranks are shown in the last row.

We tested the statistical significance by applying the Friedman test and the Holm procedure (Demšar 2006). As a result, RCN+A with RF was significantly better than SVM, RF, CART, SBRL, and RCN-A with RF and Eclat (significance level $\alpha = 0.05$). It showed no significant differences from the base network and RCN+A with Eclat.

The important points of the results are threefold. First, RCN+A with RF performed better than the existing algorithms that can present decision rules supporting their decisions (RF, CART and SBRL). This supports the effectiveness of our main idea, i.e., using neural networks to select decision rules. Second, RCA+A with RF performed better than RCN-A with RF even though they used the same RF ruleset. This means the attention mechanism used in RCN+A improved the accuracy of RCNs. Third, the difference between the rulesets, namely, either Eclat or RF, had little effect on the performance of RCN+A. Thus, the following experiments focused on RCN+A with RF.

In terms of the number of rules, CART and SBRL prefered less rules than the RCNs, even with the hyperparameters determined with validation, because their models are not designed to handle large rules. In particular, the probabilistic model of SBRL is designed to assign very low probability to large rule lists, and thus such rule lists are rejected by MCMC sampling.

Examples of the rules selected by RCNs

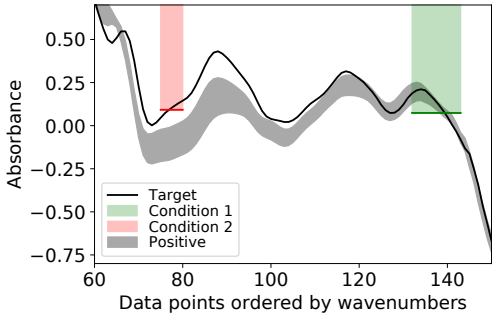
Here, we present several examples of the rules selected by RCN+A from the RF ruleset and give interpretations of them. First, let us start with the Strawberry dataset (Holland, Kemsley, and Wilson), one of the time-series datasets. The task is to distinguish the spectrographs of authentic strawberries (positive instances) from non-strawberry spectrographs (negative instances). For each instance, a time series was generated by measuring absorbance at 235 data points with different wavenumbers. The data points are ordered according to the wavenumbers.[3] An example of a target instance and the rule selected by the RCN are shown in Fig. 3. The graph at the top of the figure illustrates a negative or non-strawberry target instance. The solid black line represents the target, and the gray region represents the range of typical positive instances (i.e., mean $\pm$ standard deviation). The target seems to deviate from the range of the typical positive instances. The output of the base network is shown in the middle of the figure. The base network predicted that the target is negative with high probability (almost 1), but it did not give any reason for the decision. On the other hand, the output of the RCN is shown at the bottom of the figure. The RCN also gave high probability to the negative class, but it also presented a decision rule that explains the high probability. The antecedent is composed of "condition 1" (green) and "condition 2" (red). Feature "mean_from_s_to_e" represents the mean of the absorbance in interval $[s, e)$. By selecting this rule, the RCN says the target is negative because 95% of the instances that satisfy conditions 1 and 2 are negative. The green and red boxes represent the regions of the means satisfying conditions 1 and 2, respectively. The region for condition 2 deviates from the typical positive instances, justifying that 95% of the instances are negative. This example clearly explains the strength of the RCNs. The importance of features can be visualized by applying several existing algorithms (Simonyan, Vedaldi, and Zisserman 2013; Bach et al. 2015). However, the RCNs not only indicate the features to be paid attention but also give the thresholds of those features that lead to the consequent class probability. Furthermore, it is easy for humans to validate the correctness of the selected rule by checking and counting the instances satisfying the antecedent, because the consequent is calculated by a simple counting-based method shown in (1).

Examples from the IMDB dataset are shown in Fig. 4. The decision rules are defined for the bag-of-words representations. "*word* <= 0.5" means the word does not appear in the review and "*word* $>$ 0.5" means the word appears more than once. The corresponding words in the reviews and the rules are highlighted in green. Fig. 4 (a) shows an example of a negative review that is correctly classified as negative by the selected rule. The selected rule says the review is negative because it includes two negative words, "horrible" and "bad", and it does not include two positive words, "best" and "perfection". On the other hand, Fig. 4 (b) shows a positive review that is incorrectly classified as negative by the selected rule. The selected rule says the target review does not include obviously positive and negative words, namely "perfect", "ridiculous", and "bad". It concludes that it is negative, but with low confidence of 57%. The target review actually does not include any positive or negative words. Instead, it includes "not to be missed", which is obviously a positive phrase for humans, but such a long phrase is hard to be detected by machine learning models. It can thus be

[3] Strictly speaking, the Strawberry dataset are not genuine time-series data because they are not ordered according to time.

Table 3: Test error rates of baseline algorithms and RCNs (without ruleset optimization) .

	No-rule		Rule-supported					Transparent	
	Base	SVM	RCN-A set=Eclat	RCN-A set=RF	RCN+A set=Eclat	RCN+A set=RF	RF	CART	SBRL
Straw	.041	.049	.062 (55)	.069 (29)	**.034** (229)	.038 (117)	.053	.091 (14)	.103 (11)
Prox	.099	.192	.129 (73)	.142 (43)	**.095** (96)	.098 (82)	.147	.165 (10)	.167 (6)
FordB	.123	.303	.203 (84)	.355 (239)	.129 (189)	**.120** (444)	.286	.355 (35)	.363 (4)
FordA	**.094**	.207	.155 (117)	.327 (293)	.096 (305)	.096 (565)	.272	.335 (14)	.331 (9)
Phar	.178	.235	.210 (86)	.191 (125)	**.176** (230)	**.176** (225)	.208	.284 (52)	.206 (20)
IMDB	.136	**.133**	.160 (75)	.145 (440)	.134 (114)	.134 (690)	.169	.263 (368)	.300 (18)
Elec	.115	.127	.124 (61)	.130 (376)	.117 (97)	**.114** (595)	.156	.257 (368)	.283 (24)
Yelp	**.054**	.075	.075 (76)	.072 (382)	.077 (128)	**.054** (652)	.093	.153 (627)	.217 (278)
Ave. rank	2.375	5.250	5.125	5.563	2.438	**1.813**	6.125	8.063	8.250



Output of base network

[Pos:0.00, Neg:1.00]

Output of RCN

mean_from_132_to_143 > 0.074
AND mean_from_75_to_80 > 0.092

→ [Pos:0.05, Neg:0.95]

Figure 3: An example of outputs given an instance from the Strawberry dataset

inferred that the RCN gives incorrect decision with low confidence because the review has no obviously positive or negative words.

Limiting ruleset size

The performance of the RCN without ruleset optimization have been shown so far. Next, let us see the effect of ruleset optimization. In particular, we evaluated the effect of λ, which limits the ruleset size. It is expected that choosing small λ reduces the cost of checking whether the rules are reliable, because the distinct number of selected rules is reduced. However, if we choose too small λ, the ruleset may be too small, and the decision accuracy may be degraded. The purpose of this experiment is to evaluate this effect. We used a uniform distribution over R as $\boldsymbol{\theta}_0$ in this experiment.

Measured error rates of RCN+A with RF for different λ are listed in Table 4. The numbers in parentheses represent the distinct numbers of selected rules. The column named *Without ruleset opt.* is copied from Table 3 for comparison with the result obtained under the same setting but without ruleset optimization. For $\lambda = 1000$, the RCNs achieved almost the same error rate as that achieved without ruleset optimization. However, the distinct number of selected rules is much lower. This result means many redundant rules were included in the rules that were selected without ruleset optimization, and the ruleset optimization could remove such redundant rules and gave more succinct explanations in terms of the ruleset size. The distinct numbers of rules for $\lambda = 100$ are lower than those for $\lambda = 1000$. Finally, for $\lambda = 10$, the distinct numbers of rules are very low, but the worst error rates were observed for all of datasets. Thus, changing λ provides a tradeoff between succinctness and accuracy; removing redundant rules gives succinct explanations, but small rulesets may degrade the accuracy of decisions.

Table 4: Test error rates of RCN+A with RF for different λ

	Without ruleset opt.	λ=1000	λ=100	λ=10
Straw	.038(117)	.035(92)	**.029**(55)	.053(10)
Prox	**.098**(82)	.119(65)	.103(46)	.219(10)
FordB	.120(444)	**.109**(287)	.112(82)	.348(10)
FordA	.096(565)	**.092**(304)	.114(79)	.193(10)
Phar	**.176**(225)	.182(163)	.191(65)	.297(10)
IMDB	.134(690)	**.119**(361)	**.119**(79)	.162(10)
Elec	**.114**(595)	.116(342)	.118(85)	.213(10)
Yelp	**.054**(652)	.059(331)	.091(80)	.187(10)

Mixing human-made rules

As the final experiment, we mixed human-made rules with automatically acquired rules for sentiment classification. In terms of interpretability, it is reasonable to give higher prior probability to human-made rules than automatically acquired rules. However, giving them too high prior probabilities may degrade the accuracy of RCNs. The purpose of this experiment was to experimentally confirm this tradeoff.

We created human-made rules composed of 963 positive and 1260 negative rules based on the VADER sentiment lex-

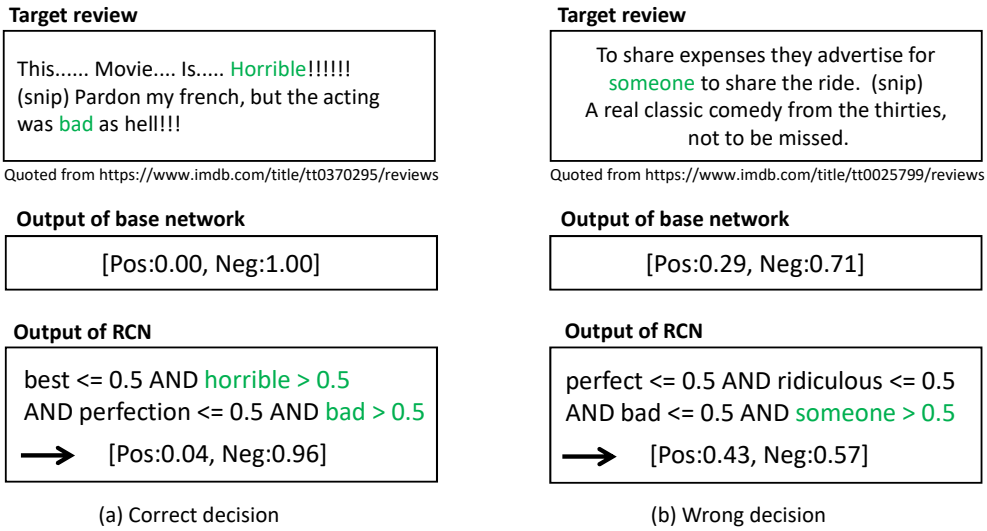


Figure 4: Examples from the IMDB dataset

icon (Hutto and Gilbert 2014) by considering the occurences of positive and negative words as the antecedents of decision rules. These human-made rules were appended to the automatically acqured rules in the RF ruleset. The original automatically acquired rules were assigned weight 1.0, and the newly appended human-made rules were assigned weight ρ. Then, $\boldsymbol{\theta}_0$ was obtained by normalizing these weights so that the sum equals 1. In this experiment, λ was set to 100. We measured the test error rates of RCN+A with the mixed ruleset, averaged over 5 runs, for different ρ. The distinct numbers of rules selected by RCN+A were also measured.

The results on the IMDB dataset are listed in Table 5. "#RF" represents the distinct number of selected RF rules, and "#HM" represents the distinct number of selected human-made rules. "HM ratio" represents the ratio of human-made rules, i.e., #HM/(#RF+#HM). For $\rho = 1$, which means the human-made and RF rules were given equal prior probabilities, the ratio of human-made rules were very low. On the other hand, for $\rho = 100$, which means the human-made rules are given much higher prior probabilities than those given to the RF rules, the ratio of human-made rules were high. However, the test error rate for $\rho = 100$ was worse than that for $\rho = 1$. This result means, for $\rho = 100$, a larger portion of decisions were explained by human-made rules at the expense of accuracy. Moderate results were obtained for $\rho = 10$. The error rate for $\rho = 10$ was just slightly worse than for $\rho = 1$ while the ratios of human rules were quite high. This experiment showed that the parameters of RCNs could change the ratio of human-made rules in the selected rules.

The most frequently selected rules for $\rho = 1$ and $\rho = 100$ are listed in Table 6. For $\rho = 1$, the top ten rules were all selected from the RF ruleset, and they include quite a few unintelligible conditions like "minutes <= 0.5", "also <= 0.5" and "anything <= 0.5". It is hard to understand how these words are related to the sentiment of reviews. On the other hand, for $\rho = 100$, many human-rules were frequently selected. They are simple and intelligible, but they have week predictive power as observed in Table 5.

Table 5: Results of mixing human-made rules

ρ	Error rate	#RF	#HM	HM ratio
1	0.134	77	13	0.14
10	0.145	33	54	0.62
100	0.23	7	72	0.91

Related work

Improving the interpretability of machine learning is a rising research area that includes techniques applicable to general machine learning models (Lipton 2016; Doshi-Velez 2017) or techniques especially for neural networks (Montavon, Samek, and Müller 2018). These techniques can be classified into three groups. The first group analyzes which input features are relevant to decisions. SHAP (Lundberg and Lee 2017) is a general framework for presenting feature importance based on game theory given a particular decision. Gradient-based sensitivity analysis (Simonyan, Vedaldi, and Zisserman 2013) and layer-wise relevance propagation (Bach et al. 2015) are common approaches for neural networks. The second group approximates neural networks with simpler models. LIME (Ribeiro, Singh, and Guestrin 2016) is a general framework for building a simper model to approximate the local decision boundaries around a given decision. Rule extraction from shallow or simple neural networks was mainly studied in the 1990s (Andrews, Diederich, and Tickle 1995; Zilke, Loza Mencía, and Janssen 2016), but it cannot be applied to modern complex neural networks like CNN or LSTM. The first and second groups are just post-hoc analysis, so they never affect the decisions of neural networks and cannot guarantee that a simple explanation for the decisions always exists. The third group constrains the neural network with models

Table 6: Most frequently selected rules

(a) $\rho = 1$

Rank	Freq.	Antecedent	Consequent [Pos,Neg]	Human -made
1	2394	lame <= 0.5 AND minutes <= 0.5 AND no <= 0.5 AND bad <= 0.5	[0.63,0.37]	
2	1711	bad <= 0.5 AND great > 0.5 AND worst <= 0.5 AND pointless <= 0.5	[0.76,0.24]	
3	1182	beautiful <= 0.5 AND poor <= 0.5 AND movie > 0.5	[0.55,0.45]	
4	1122	acting <= 0.5 AND boring <= 0.5 AND poor <= 0.5 AND anything <= 0.5	[0.57,0.43]	
5	997	bad <= 0.5 AND also > 0.5 AND terrible <= 0.5 AND worst <= 0.5	[0.68,0.32]	
6	972	both <= 0.5	[0.51,0.49]	
7	929	bad > 0.5 AND also <= 0.5 AND steals <= 0.5	[0.21,0.79]	
8	692	excellent > 0.5 AND rent <= 0.5 AND thing <= 0.5	[0.84,0.16]	
9	686	no > 0.5 AND plot <= 0.5	[0.40,0.60]	
10	671	save <= 0.5 AND bad > 0.5 AND bad > 1.5 AND also <= 1.5	[0.12,0.88]	

(b) $\rho = 100$

Rank	Freq.	Antecedent	Consequent [Pos,Neg]	Human -made
1	7895	DEFAULT	[0.50,0.50]	
2	1980	bad	[0.1,0.9]	✓
3	1709	no	[0.1,0.9]	✓
4	1663	worse <=0.5 AND favorite <= 0.5 AND great > 0.5 AND fantastic <= 0.5	[0.68,0.32]	
5	1014	worst > 0.5 AND of <= 4.5 AND sex <= 0.5 AND subtle <= 0.5	[0.06,0.94]	
6	619	wonderful <= 0.5 AND even <= 0.5 AND excellent > 0.5	[0.84,0.16]	
7	473	worst	[0.1,0.9]	✓
8	471	amazing	[0.9,0.1]	✓
9	433	fun	[0.9,0.1]	✓
10	409	enjoyed	[0.9,0.1]	✓

of simpler forms. Combining neural networks with graphical models to improve the interpretability was recently proposed (Al-Shedivat, Dubey, and Xing 2017); however this approach gives explanations in the form of the graphical models, which are not as interpretable as decision rules. For the structured prediction problem, a technique to add relational rules between labels to constrain the consistency of labels predicted by neural networks has been proposed (Hu et al. 2016), but its rules describe only a fraction of relations between labels, and it cannot give an explanation in the form of decision rules. RCNs belong to the third group and, to our knowledge, is the first approach that can always present a decision rule that justifies a decision made by neural networks.

Improving the accuracy of rule-based models has also been researched. Non-greedy optimization of decision lists (Letham et al. 2015; Wang and Rudin 2015; Angelino et al. 2017; Yang, Rudin, and Seltzer 2017) has been realized with the help of the high computational power of modern processors. In these approaches, given a rule set built by association mining, a decision rule list composed of rules extracted from the given rule set is optimized. However, even with modern processors, solving the optimization problem about the combination of rules has computational difficulty when the given rule set is large. Scalable Bayesian rule lists (Yang, Rudin, and Seltzer 2017), which alleviate this difficulty using fast bit-vector computation, is the first approach that is scalable to more than tens of thousands of rules.

Conclusion and future work

To combine the classification accuracy of neural networks with the interpretability of decision rules, we proposed a model called rule-constrained networks (RCNs). In this model, neural networks are forced to select decision rules from a rule set so that the rules predict correct classes. The rule set from which rules are taken is also optimized with given hyper parameters that specify the ruleset size and the probabilities of rules being selected. The proposed model guarantees that each decision is always supported by one decision rule, while the rules used for decisions are selected from a ruleset with a limited size. The experiments showed RCNs achieved higher accuracy than existing rule-based classifiers for time-series classification and sentiment classification while presenting rules that support the decisions.

The direction of our future work is two-fold. First, the evaluation of the RCNs will be extended to tasks other than sequential classification. Second, we should ask human experts to evaluate RCNs and add, delete or change decision rules to improve the quality of the ruleset. We believe rules added by human experts would amplify the interpretability of RCNs from the viewpoint of the experts themselves.

References

Al-Shedivat, M.; Dubey, A.; and Xing, E. P. 2017. Contextual explanation networks. *CoRR* abs/1705.10301.

Andrews, R.; Diederich, J.; and Tickle, A. B. 1995. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Know.-Based Syst.* 8(6):373–389.

Angelino, E.; Larus-Stone, N.; Alabi, D.; Seltzer, M.; and Rudin, C. 2017. Learning certifiably optimal rule lists. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, 35–44. ACM.

Bach, S.; Binder, A.; Montavon, G.; Klauschen, F.; Müller, K.-R.; and Samek, W. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE* 10(7):1–46.

Chen, Y.; Keogh, E.; Hu, B.; Begum, N.; Bagnall, A.; Mueen, A.; and Batista, G. 2015. The UCR time series classification archive. http://www.cs.ucr.edu/~eamonn/time_series_data/.

Demšar, J. 2006. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* 7:1–30.

Doshi-Velez, Finale; Kim, B. 2017. Towards a rigorous science of interpretable machine learning. In *eprint arXiv:1702.08608*.

Han, J.; Pei, J.; and Yin, Y. 2000. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, 1–12. ACM.

Hinton, G. E. 2002. Training products of experts by minimizing contrastive divergence. *Neural Comput.* 14(8):1771–1800.

Holland, J. K.; Kemsley, E. K.; and Wilson, R. H. Use of fourier transform infrared spectroscopy and partial least squares regression for the detection of adulteration of strawberry purées. *Journal of the Science of Food and Agriculture* 76(2):263–269.

Hu, Z.; Ma, X.; Liu, Z.; Hovy, E.; and Xing, E. 2016. Harnessing deep neural networks with logic rules. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2410–2420. Association for Computational Linguistics.

Hutto, C., and Gilbert, E. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text.

Johnson, R., and Zhang, T. 2016. Supervised and semi-supervised text categorization using LSTM for region embeddings. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, 526–534. JMLR.org.

Kingma, D. P., and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of the Third International Conference on Learning Representations*, ICLR '15.

Letham, B.; Rudin, C.; McCormick, T. H.; and Madigan, D. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Ann. Appl. Stat.* 9(3):1350–1371.

Lipton, Z. C. 2016. The mythos of model interpretability. In *ICML Workshop on Human Interpretability of Machine Learning*, WHI 2016.

Lundberg, S. M., and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 4765–4774.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, 142–150. Association for Computational Linguistics.

Montavon, G.; Samek, W.; and Müller, K.-R. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* 73:1 – 15.

Ribeiro, M. T.; Singh, S.; and 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*, KDD '16, 1135–1144. ACM.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR* abs/1312.6034.

Wang, F., and Rudin, C. 2015. Falling rule lists. In *Proceedings of Artificial Intelligence and Statistics (AISTATS)*.

Wang, Z.; Yan, W.; and Oates, T. 2016. Time series classification from scratch with deep neural networks: A strong baseline. *CoRR* abs/1611.06455.

Yang, H.; Rudin, C.; and Seltzer, M. 2017. Scalable Bayesian rule lists. In Precup, D., and Teh, Y. W., eds., *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, 3921–3930. PMLR.

Zaki, M. J.; Parthasarathy, S.; Ogihara, M.; and Li, W. 1997. New algorithms for fast discovery of association rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, KDD'97, 283–286. AAAI Press.

Zilke, J. R.; Loza Mencía, E.; and Janssen, F. 2016. DeepRED – rule extraction from deep neural networks. In Calders, T.; Ceci, M.; and Malerba, D., eds., *Discovery Science: 19th International Conference, DS 2016, Bari, Italy, October 19–21, 2016, Proceedings*, 457–473. Springer International Publishing.