DeepRED – Rule Extraction from Deep Neural Networks*

Jan Ruben Zilke, Eneldo Loza Mencía, and Frederik Janssen

Technische Universität Darmstadt
Knowledge Engineering Group
j.zilke@mail.de, {eneldo,janssen}@ke.tu-darmstadt.de

Abstract. Neural network classifiers are known to be able to learn very accurate models. In the recent past, researchers have even been able to train neural networks with multiple hidden layers (deep neural networks) more effectively and efficiently. However, the major downside of neural networks is that it is not trivial to understand the way how they derive their classification decisions. To solve this problem, there has been research on extracting better understandable rules from neural networks. However, most authors focus on nets with only one single hidden layer. The present paper introduces a new decompositional algorithm – *DeepRED* – that is able to extract rules from deep neural networks.
The evaluation of the proposed algorithm shows its ability to outperform a pedagogical baseline on several tasks, including the successful extraction of rules from a neural network realizing the XOR function.

1 Introduction

To tackle classification problems, i.e.,deciding whether or not a data instance belongs to a specific class, machine learning offers a wide variety of methods. If the only goal is to accurately assign correct classes to new, unseen data, neural networks (NN) are able to produce very low error rates that yet could not be achieved by other machine learning techniques [8]. Sufficiently deep NNs, so-called deep neural networks (DNN), are even able to realize arbitrary functions. And lately, researchers improved the training of these structures such that they can generalize better and better from training data, for instance in the research fields of speech recognition and computer vision.

However, there is a major downside of NNs: For humans, it is not easy to understand how they derive their decisions [11]. However, understanding a NN's function can be essential. For instance, this is the case in safety-critical application domains such as medicine, power systems, or financial markets, where a hidden malfunction could lead to life-threatening actions or enormous economic loss. A better understanding of the way NNs derive their decisions could also push NN training research. Making NNs

* This is the authors' version of the work retrieved from http://www.ke.tu-darmstadt.de. The final publication is available at Springer at http://dx.doi.org/10.1007/978-3-319-46307-0_29 and appeared in T. Calders, M. Ceci, D. Malerba (Eds.): *Discovery Science 19th International Conference, DS 2016, Bari, Italy, October 19–21, 2016, Proceedings*, LNAI 9956, pp. 457–473, 2016.

more transparent, for instance, could help to discover so-called hidden features that might be formed in DNNs while learning. Such features are not present in the plain input data, but emerge from combining them in a useful way.

In contrast to NNs, rule-based approaches like decision trees or simple IF-THEN rules are known to be better understandable. Since most humans tackle classification problems in a manner very similar to the one implemented by many rule-based learning techniques, i.e., listing simple conditions that need to be met, their models and decision processes are more comprehensible.

To overcome the weakness of NNs being black boxes, especially in the 1990s researchers have worked on the idea of extracting rules from them. Since then, a lot of rule extraction techniques have been developed and evaluated – and for many approaches quite good results have been reported. However, most algorithms to extract rules focus on small NNs with only one hidden layer. Surprisingly little work has been done on analysing the challenges of extracting rules from the more complex DNNs. Although some authors have presented rather general approaches, to the best of our knowledge, there does not exist any algorithm that has explicitly been tested on the task of extracting rules from DNNs. But indeed, having a large amount of layers makes the extraction of comprehensible rules more difficult.

In this paper, we introduce and evaluate *DeepRED*, a new algorithm that is able to extract rules from DNNs. The decompositional algorithm extracts intermediate rules for each layer of a DNN. A final merging step produces rules that describe the DNN's behaviour by means of its inputs. The evaluation shows *DeepRED*'s ability to successfully extract rules from DNNs and to outperform a pedagogical approach.

The work is structured as follows: First, we give an overview of the current state-of-the-art. Afterwards, we describe our proposed algorithm in more detail (Sec. 3). We evaluate *DeepRED* on several tasks (Sec. 4 and 5) before we summarize and conclude our work.

2 Related Work

It is commonly believed that "concepts learned by neural networks are difficult to understand because they are represented using large assemblages of real-valued parameters" [4]. To transform NNs into a form that is better comprehensible for humans, many concepts are conceivable. One famous choice is to extract rules from NNs. In [1], the authors have introduced a widely used taxonomy to distinguish between different rule extraction algorithms. The taxonomy, for instance, comprises the *translucency* dimension that defines the strategy used: decompositional (considering NN's inner structure), pedagogical (black box approach), or eclectic (mixture of both).

Examples of eclectic methods are the *MofN* algorithm [19] as well as the *FERNN* algorithm [15]. *MofN*'s main approach is to find NN connections with similar weights. *FERNN* focusses on identifying a NN's relevant inputs and hidden neurons.

The works dealing with pedagogical rule extraction, for instance, comprise the *VIA* method discussed in [17, 18], different sampling-based approaches [5, 16, 13, 21, 14], and the *RxREN* algorithm [2]. *VIA* uses validity interval analysis to find provably correct rules. Sampling-based methods try to create extensive artificial training sets where rule learning algorithms later can extract rules from. *RxREN* provides interesting ideas to prune a NN before rules are extracted (cf. Sec. 3.4).

Examples for decompositional approaches are the *KT* method that heuristically searches for input combinations that let a neuron fire [7], a more efficient implementation of the latter [20], and an algorithm that transforms NNs to fuzzy rules [3]. However, the most important basis for this present work is the (decompositional) *CRED* algorithm presented in [12]. *CRED* uses decision trees to describe a NN's behaviour based on the units in its hidden layer. In a second step it builds up new decision trees to describe the split points of the first decision trees. Afterwards rules are extracted and merged.

Although we encounter a wide variety of interesting rule extraction approaches, there is a major shortcoming: Most decompositional algorithms introduced so far cannot deal with NNs with more than one hidden layer. Furthermore, to the best of our knowledge, there does not exist any algorithm that has explicitly been tested on the task of extracting rules from DNNs. This is the case even though most pedagogical approaches should be able to perform this task without any major modifications. However, due to the lack of pedagogical techniques to include the knowledge present in a DNN's inner structure, we focus on a decompositional method to extract rules from DNNs.

3 The DeepRED Algorithm

With *DeepRED* (Deep neural network Rule Extraction via Decision tree induction), in this paper we present an algorithm that is able to overcome the shortcoming of most state-of-the-art algorithms: *DeepRED* is applicable to DNNs. Since *DeepRED* is a decompositional approach, in contrast to pedagogical methods, the algorithm is able to easily extract hidden features from a DNN.

This section is structured as follows: First, we introduce our notation and assumptions. After describing the *CRED* algorithm in more detail, we present *DeepRED*'s way to extract rules from DNNs as well as the pruning technique used.

3.1 Preliminaries

We assume a multiclass classification problem with m training examples $x_1,\dots,x_m$, each x_j associated with one class $y_j \in \{\lambda_1,...,\lambda_u\}$. We represent the input values of a NN as $i = i_1,\dots,i_n$ and the output values as $o = o_1,\dots,o_u$, one for each possible class. The hidden layers are abbreviated as $h_i \in \{h_1,\dots,h_k\}$ while the hidden layer h_i consists of the neurons $h_{i,1}\dots,h_{i,H_i}$ (in the case of a single-hidden-layer NN, the hidden neurons are written as $h_1,\dots,h_H$). For convenience, we set $h_0 = i$ and $h_{k+1} = o$ and let $h_i(x)$ denote the specific layer values for input instance x. *DeepRED* produces intermediate rule sets $R_{a\to b}$ that include rules that describe layer b by means of terms based on layer a. A rule r : IF *body* THEN *head* is constituted by a set of terms in the body (conditions on attribute values, or, in our case, also conditions on activation values) and an assignment term in the head.

3.2 CRED as Basis

As mentioned earlier, *DeepRED* is based on *CRED* (Continuous/discrete Rule Extractor via Decision tree induction). In a first step, the original algorithm introduced by Sato and Tsukimoto [12] uses the well-known *C4.5* algorithm [10] to transform each output unit of a NN (with one hidden layer) into a decision tree. Such a tree's inner nodes are tests on the values of the hidden layer's units. The leaves represent the class such an example would belong to (cf. Figure 1). Afterwards, intermediate rules are extracted directly from these trees. In the case illustrated in Figure 1, if the task is to find rules for $class_1$, a single intermediate rule would be extracted, i.e. IF $h_2 > 0.5$ AND $h_1 \leq 0.6$ THEN $\hat{y} = \lambda_1$. This leads us to a rule set that describes the behaviour of a NN on the basis of the hidden layer's units.

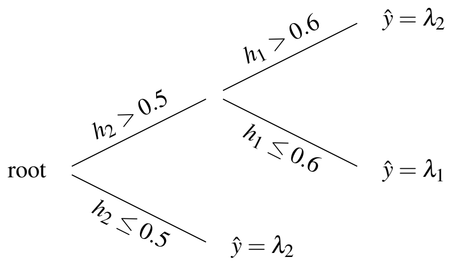

Fig. 1. Result of CRED's first step: A decision tree providing rules for the neural network's output based on its hidden layer (adapted from [12]).

Next, for each term used in these rules, another decision tree is created using split points on the NN's input layer. In these new decision trees, the leaves do not directly decide for an example's class, but rather on the tests used in the first tree's node, as exemplarily depicted in Figure 2. Extracting rules from this second decision tree leads us to a description of the hidden neurons' state by terms consisting of input variables. In case of the state $h_2 > 0.5$, the following intermediate rules could be extracted: IF $i_0 > 0.5$ THEN $h_2 > 0.5$, IF $i_0 \leq 0.5$ AND $i_1 = a$ THEN $h_2 > 0.5$, and IF $i_0 \leq 0.5$ AND $i_1 = b$ THEN $h_2 > 0.5$. Further decision trees must be extracted for all the other split points found in the first tree (in our example for $h_1 \leq 0.6$).

As a last step, the intermediate rules that describe the output by means of the hidden layer and those that describe the hidden layer based on the NN's inputs are substituted and merged to build rules that describe the NN's output on the basis of its inputs. Note that the resulting rules from the decision tree are disjunctive, i.e., only one will fire at a time for any test instance.

3.3 Extending CRED to DeepRED

Since *CRED* is a decompositional rule extraction approach, it is not possible to apply the original implementation directly to NNs with multiple layers of hidden neurons.

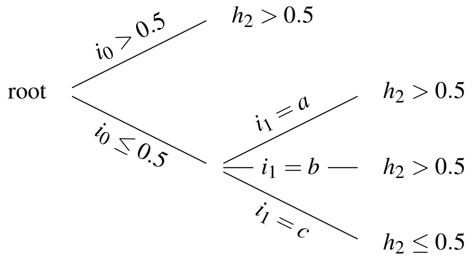

Fig. 2. Result of one iteration of CRED's second step: A decision tree providing rules for $h_2 > 0.5$ based on the neural network's inputs (adapted from [12]).

However, the algorithmic approach can be extended relatively straight-forward by deriving additional decision trees and intermediate rules for every supplementary hidden layer.

The general process of *DeepRED* is the following: The algorithm extracts rules for each class/NN output one after another. For each class it processes every hidden layer in a descending order. *DeepRED* extracts rules for each hidden layer that describes its behaviour based on the preceding layer. In the end, all rules for one class are getting merged such that we arrive at the rule set $R_{i\to o}$.

Our modified version of the *CRED* algorithm – *DeepRED* – starts just like the original implementation by using *C4.5* to create decision trees consisting of split points on the activation values of the last hidden layer's neurons and the regarding classifications in the trees' leaves. Figure 3 provides the pseudo code for *DeepRED*. To exemplify *DeepRED*'s approach we consider, without loss of generality, a NN with k hidden layers. As a result of the first step we get a decision tree that describes the output layer by split points on values regarding h_k, i.e., the rule set $R_{h_k\to o}$. The data to run *C4.5* on is generated by feeding the training data to the NN and recording the outputs of the hidden neurons (line 8).

In the next step, in contrast to the algorithm presented by Sato and Tsukimoto, we do not directly refer to the input layer, but instead process the next shallower hidden layer, i.e. h_{k-1} (loop starting in line 1). For every term present in one of the rules in $R_{h_k\to o}$, we need to apply the *C4.5* algorithm to find decision trees that describe layer h_k by means of h_{k-1} and can be transformed to the rule set $R_{h_{k-1}\to h_k}$ (line 10). We also see that the proposed algorithm also implements a procedure to prevent itself from performing redundant runs of *C4.5* since we only learn a decision tree for terms which were not already extracted (line 6). Just like in the *CRED* example presented earlier, the terms in $R_{h_k\to o}$ are directly used to differentiate positive and negative examples for the regarding *C4.5* runs (line 9).

We proceed in the same manner until we arrive at decision trees/rules that describe terms in the first hidden layer h_1 by terms consisting of the original inputs to the NN, i.e. $R_{i\to h_1}$.

Now we have rule sets that describe each layer of the NN by their respective preceding layer, i.e. we have the sets of intermediate rules $R_{i\to h_1}, R_{h_1\to h_2}, \ldots, R_{h_{k-1}\to h_k}$, and

```
Input: Neural network $h_0, h_1, \ldots, h_k, h_{k+1}$, training examples $x_1, x_2, \ldots, x_m$
Output: Set of rules representing the NN
1  foreach class $\lambda_v \in \lambda_1, \ldots, \lambda_u$ do
2      $R^v_{h_k \to o} \leftarrow$ IF $h_{k+1,v} > 0.5$ THEN $\hat{y} = \lambda_v$    // initial rule for NN's prediction
3      foreach hidden layer $j = k, k-1, \ldots 1$ do
4          $R^v_{h_{j-1} \to h_j} \leftarrow \emptyset$
5          $T \leftarrow$ extractTermsFromRuleBodies ($R^v_{h_j \to h_{j+1}}$)
6          $T \leftarrow$ removeDuplicateTerms ($T$)
7          foreach $t \in T$ do
8              $x'_1, \ldots, x'_m \leftarrow h_j(x_1), \ldots, h_j(x_m)$
                   // obtain activation values for all neurons in the $j-1$-th
               layer for each training example
9              $y'_1, \ldots, y'_m \leftarrow t(h_{j+1}(x_1)), \ldots, t(h_{j+1}(x_m))$
                   // apply term $t$ on respective activation in layer $j$ and get
               binary outcome, the training signal, for each training example
10             $R^v_{h_{j-1} \to h_j} \leftarrow R^v_{h_{j-1} \to h_j} \cup$ C4.5 $((x'_1, y'_1), \ldots, (x'_m, y'_m))$
                   // learn rules for $t$ and add to rules representing the layer
11         end
12     end
       // merging to conjunctive rules and cleaning up
13     foreach hidden layer $j = k, k-1, \ldots 1$ do
14         $R^v_{h_{j-1} \to o}$ = mergeIntermediateTerms( $R^v_{h_{j-1} \to h_j}$ , $R^v_{h_j \to o}$)
                                            // dissolve rules of remaining two layers
15         $R^v_{h_{j-1} \to o}$ = deleteUnsatisfiableTerms($R^v_{h_{j-1} \to o}$)
16         $R^v_{h_{j-1} \to o}$ = deleteRedundantTerms($R^v_{h_{j-1} \to o}$)
17     end
18 end
19 return $R^1_{i \to o}, R^2_{i \to o}, \ldots, R^u_{i \to o}$    // describes output of NN by rules consisting of
   terms on input values of first layer
```

Fig. 3. Pseudo code of DeepRED.

$R_{h_k \to o}$. To get a rule set $R_{i \to o}$ that describes the NN's outputs by its inputs, these rules need to be merged. This merging process proceeds layer-wise (block starting at line 13). First, *DeepRED* substitutes the terms in $R_{h_k \to o}$ by the regarding rules in $R_{h_{k-1} \to h_k}$ to get the rule set $R_{h_{k-1} \to o}$ (line 14). As mentioned in Figure 3, unsatisfiable intermediate rules as well as redundant terms are deleted (lines 15 and 16).[1] This happens to reduce computation time and memory usage drastically. Next, we merge the rule sets $R_{h_{k-1} \to o}$ and $R_{h_{k-2} \to h_{k-1}}$. Step by step we go through all layers until we arrive at rules that describe the classification/outputs according to the inputs to the NN, which is the result we were looking for.

[1] The merging may produce rules of the form $i_1 < 0.1$ AND $i_1 > 0.2$, or $i_1 > 0.4$ AND $i_1 > 0.5$.

Table 1. Overview of deep neural networks used for evaluation, including the characteristics of the original data the NNs were trained on and the accuracies of the NN on the training and test set.

	#attributes	#training ex.	#test ex.	NN structure	acc(training)	acc(test)
MNIST	784	12056	2195	784-10-5-2	99.6%	98.8%
letter	16	1239	438	16-40-30-26	96.9%	97.3%
artif-I	5	20000	10000	5-10-5-2	99.5%	99.4%
artif-II	5	3348	1652	5-10-5-2	99.4%	99.0%
XOR	8	150	106	8-8-4-4-2-2-2	100%	100%

3.4 Pruning

One way to facilitate the rule extraction process is to prune NN components that can be considered as less important for the classification. While the authors of *CRED* did not report on any of these approaches, other researchers have invented several techniques that help to extract comprehensible rules. As a pre-processing step before *DeepRED* is applied, we have borrowed the input pruning technique from *RxREN* [2]. It proceeds by testing the NN's performance while ignoring individual inputs. Those inputs that where not necessary to still produce an acceptable classification performance are getting pruned.

4 Experiments

Since we are not aware of any algorithm that has been tested on extracting rules from DNNs, we want to fill this gap with a *DeepRED* evaluation. Although *DeepRED* is able to extract rules from DNNs with an arbitrary number of output neurons, in this study we limit ourselves to two-class-problems. Before we present and discuss the results more extensively, we first take a look at the evaluation data base.

To evaluate *DeepRED*, we need suitable input data, i.e. trained NNs and training sets. Unfortunately, it is not trivial to find already trained NNs in literature or online. Although NN research is a lot about training, usually only network structures, training methods and performances are reported, while specific weights are not of particular interest for most researchers. Therefore, we trained DNNs for real-world and artificial problems. Table 1 summarizes our data basis.

Both, the *MNIST* and the *letter* data are taken from real-world problems. The basis for our *MNIST* dataset is the dataset presented in [9]. We picked a subset of the original data, reduced the problem to distinguish only between two different classes (*1* vs. rest), and trained a DNN on it. The *letter* recognition problem is derived from the dataset introduced in [6]. Here, we first trained the DNN on a multi-class training set. Afterwards we reduced the training and test set to a binary classification problem (accuracy rates reported are based on the training set reduced to two classes).

The datasets *artif-I* and *artif-II* are artificial ones that we have created to be able to compare original and extracted models. Both databases comprise examples with five attributes (discrete and continuous domains). A challenging characteristic of *artif-I* is that it contains greater-than relations on real-valued attributes. These functions cannot easily be modelled by decision trees. This is not the case with *artif-II*, however, the second artificial dataset is based on a model where the value of one attribute has no effect on the class, which as well might be challenging for rule extraction algorithms.[2] As a fifth problem we manually constructed a DNN that solves the *XOR* problem with eight inputs. The parity function is well-known as a hard problem for rule learners. Top-down learner, for instance, need all possible input examples to correctly model *XOR*.

4.1 Rule Extraction Algorithms

To get an idea of how well the proposed algorithm performs the task of extracting rules from DNNs, in this evaluation we compare two variants of *DeepRED* – a version without and a version with *RxREN* pruning – with a pedagogical baseline.

To control the behaviour of *DeepRED*, we implemented two parameters that influence the included *C4.5* algorithm and one for the pruning mechanism. The first two parameters control when *C4.5* should stop further growing a decision tree. As an additional general setting we adjust *C4.5* to only perform binary splits and to produce a maximum decision tree depth of ten.

The first parameter – *class dominance* – is a threshold that considers the classes of the current data base. If the percentage of examples that belong to the most frequent class exceeds the value in the class dominance parameter, this class is getting predicted instead of further dividing the data base.

The second parameter is the minimum *data base size*. This value tries to stop *C4.5* further growing the decision tree when there is not enough data available to base dividing steps on. This parameter takes the number of training examples available in the first step as a reference value and defines a percentage of this size as the minimum data base size. If for the current step, there are less examples available than the parameter requires, *C4.5* produces a leaf with the most frequent class at this point.

The *RxREN* pruning parameter, namely the maximum *accuracy decrease*, controls the pruning intensity. Step by step, the technique prunes the inputs that are considered as the least significant ones, as long as the NN's accuracy does not drop below the decrease allowed by this parameter.

As a pedagogical rule extraction baseline we choose to use the *C4.5* algorithm, that is also used as a part of *DeepRED* itself. Like the sampling-based approaches, our baseline is provided with the training examples as well as the NN's classification for these instances (instead of the real classes). Using these values, *C4.5* extracts a decision tree that, afterwards, is transformed to a rule set. The two introduced *C4.5* parameters do also apply to the baseline.

Note that no rule rewriting, rule pruning or any other rule optimization mechanisms are implemented by the *DeepRED* variants or the baseline (except *DeepRED*'s reported steps to delete redundant terms and unsatisfiable rules).

[2] Input instances are drawn randomly from $x \in \{0, 0.5, 1\} \times \{0, 0.25, 0.5, 0.75, 1\} \times [0,1]^3$. For *artif-I* $y = \lambda_1$ if $x_1 = x_2$, if $x_1 > x_2$ AND $x_3 > 0.4$, or if $x_3 > x_4$ AND $x_4 > x_5$ AND $x_2 > 0$, else $y = \lambda_2$, whereas for *artif-I* $y = \lambda_1$ if $x_1 = x_2$, if $x_1 > x_2$ AND $x_3 > 0.4$, or IF $x_5 > 0.8$.

4.2 Evaluation Measures

To compare the results of the algorithms presented using the trained DNNs introduced, we derive two central measures from the extracted rule sets, that are used to assess their quality. First, we measure the comprehensibility by the number of terms in the extracted rule set. Second, we use the fidelity (ratio of prediction matches) to compare the mimicking performance of the extracted rules in contrast to the original NN's behaviour.

4.3 Evaluation Setup

Before we setup the evaluation, we collected our expectations of *DeepRED*'s performance. Our expectations were:

1. *DeepRED is able to extract rules from deep neural networks.*
   As stated earlier, to the best of our knowledge, no rule extraction algorithm has ever been tested on a DNN. We believed that in general the proposed algorithm can manage this task. However, we also expected *DeepRED*'s memory and computation time demands to sometimes make rule extraction processes intractable. We assumed that this was especially true for classification problems where an instance is described by a large number of attributes.
2. *DeepRED outperforms the baseline on more complex problems.*
   We believed that our algorithm can profit from a NN's structure when the training data model a concept that cannot be described/learned by a simple decision tree – including but not restricted to the *XOR* problem. On the other hand, for decision tree problems, i.e. classification tasks that can more easily be solved by a decision tree, our assumption was that the baseline could outperform *DeepRED*.
3. *RxREN pruning facilitates the extracted rules.*
   For problems where not all inputs are relevant, we expected the extracted rules to be more comprehensible when *RxREN* pruning has been applied. Depending on the problem, we even thought that the fidelity rates could be improved. However, since the optimal setting of the pruning parameters is not trivial, the extracted rules probably would not reach the possibly best tradeoff between comprehensibility and fidelity.
4. *DeepRED extracts more accurate rules if more data is available.*
   We believed that the *DeepRED* algorithm would need a certain amount of data to extract reasonable rules. However, we also thought that our algorithm is less dependant on a sufficiently large dataset than *C4.5* is, because *DeepRED* focusses on the setup of the NN's inner structure using instances, which is a more efficient representation compared to the pedagogical point of view.

To check these expectations, our evaluation setup includes three different pairs of *C4.5* parameter values, three different pruning settings, as well as four values that control the number of training examples visible to the rule extraction algorithm. Table 2 summarizes the different parameter settings chosen.

The number of experiments for the proposed algorithm sums up to 180 (or 36 per dataset) while the baseline is executed in 60 experiments (or 12 per dataset) – *RxREN* pruning never is applied here.

Table 2. The parameter settings used for the evaluation.

C4.5 parameters	RxREN pruning	Training set
92% / $\leq 2\%$	No pruning	10%
95% / $\leq 1\%$	5%	25%
99% / $\leq 0\%$	10%	50%
		100%

Table 3. Statistics of performed experiments for DeepRED.

	artif-I	artif-II	letter	MNIST	XOR
Executed	36	36	36	21	12
Successful	11	23	26	4	12
Aborted (memory)	24	13	10	7	0
Aborted (time)	1	0	0	10	0

5 Evaluation

The results of *DeepRED*'s evaluation shows that most of our expectations are met. The main results are:

DeepRED can Successfully Extract Rules from Deep Neural Networks – Table 3 gives an overview of the general success of applying *DeepRED* to different DNNs. For every dataset, we list the number of experiments that were executed[3] as well as the number of successful and aborted attempts[4].

One observation is that for every DNN tested there is at least one parameter setting that enables *DeepRED* to successfully extract rules. Except for the *MNIST* dataset, this also holds true for every *RxREN* pruning variant evaluated. On the other hand is the high abortion rate of approximately 46% an indication that the right parameters are important when using *DeepRED*. Especially for classification problems with a large number of inputs, i.e. *MNIST*, the success rate is very low.

A further investigation concerning the abortion reasons shows that most of the time memory limits are exceeded while merging if a lot of intermediate rules are present in the first two layers. The abortions due to violation of time constraints can all be ascribed to large training sets passed to the algorithm – the implemented *C4.5* is not efficient enough when dealing with large datasets.

DeepRED can Extract Comprehensible Rules for Complex Problems – Taking a closer look at the results on the datasets *artif-I* and *artif-II* lets us assess *DeepRED*'s performance on more complex problems. As described earlier, the data in *artif-I* model a function that cannot easily be realized by a decision tree. We consider this a difficult problem for the baseline. The *artif-II* data, on the other hand, can quite easily be

[3] You might notice that, earlier, we mentioned that there are 36 experiments per dataset. However, to avoid sophisticating the outcomes, we discard those experiments where the *RxREN* pruning results in no pruned inputs at all.

[4] An abortion could either be the case if the experiment exceeds the allocated memory space (10000MB) or if *DeepRED* needs more than the maximum execution time (24 hours).

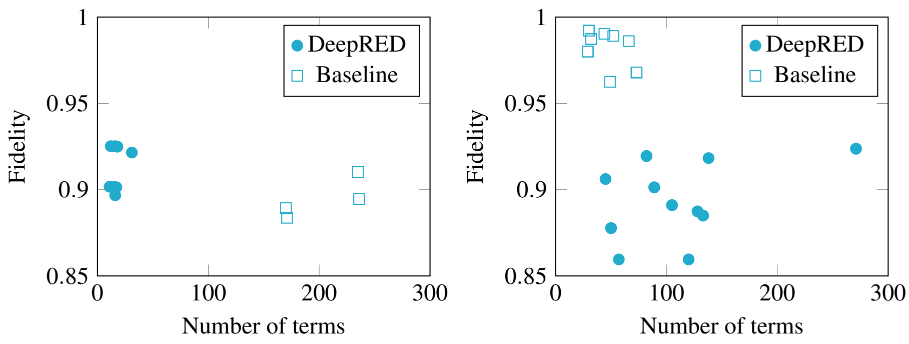

Fig. 4. Evaluation results for DeepRED and the baseline on artif-I (left) and artif-II (right).

modelled by a decision tree. However, please notice that the functions the DNNs have learned are not equivalent to the true models.

Figure 4 shows the results. We clearly see that the best *DeepRED* result outperforms the baseline on the DNN representing the *artif-I* dataset. Especially the comprehensibility of the rules extracted by our approach is much better. It seems like the proposed algorithm could use the inner structure of the DNN to create rules that efficiently model the NN's behaviour.

The *artif-II* dataset gives a different picture: The baseline produces much better results than *DeepRED* does. Regarding fidelity, the outcomes of our approach are more or less similar to those of the *artif-I* data. However, the baseline is able to find a more efficient way to model the NN's outputs – and does so much more accurate. For some reason, the inner structure of the DNN seems to confuse *DeepRED* such that it cannot extract high-quality rules.

Part of our expectation also was, that *DeepRED* can outperform the baseline by far when extracting rules from an *XOR* NN. Figure 5 gives an insight into the performance on this task. Please notice, that compared to Figure 4 we needed to adjust the axes to draw a meaningful graph. As we expected it and it is well-known, *C4.5* cannot generalize from *XOR* training examples to correctly classify distinct test instances, i.e. it never gets better than random. However, as we see in the graph, *DeepRED* is able to use the knowledge embedded in the DNN, to extract rules that correctly classify unseen examples. As we will discuss in a moment, *DeepRED* needs a sufficiently large data basis to perform this task. But if it is given, an error rate of 0% can be achieved (only having the training examples).

You might argue that *DeepRED* is not extracting comprehensible rules. In fact, rules in the form as we use them are not able to efficiently model the *XOR* problem. To correctly classify all possible 256 input combinations for this problem, 128 rules with 8 terms each are necessary to cover all 128 positive examples (the default rule covers the negative examples). In sum this leads to a number of 1024 terms in the rule set – which is the number of terms, the most accurate rules extracted by *DeepRED* have.

RxREN Pruning Helps DeepRED to Extract More Comprehensible Rules – *RxREN* pruning can facilitate the rule extraction process for *DeepRED*. Here, we want

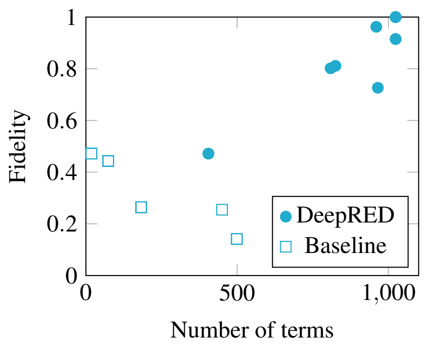

Fig. 5. Evaluation results for DeepRED and the baseline on XOR.

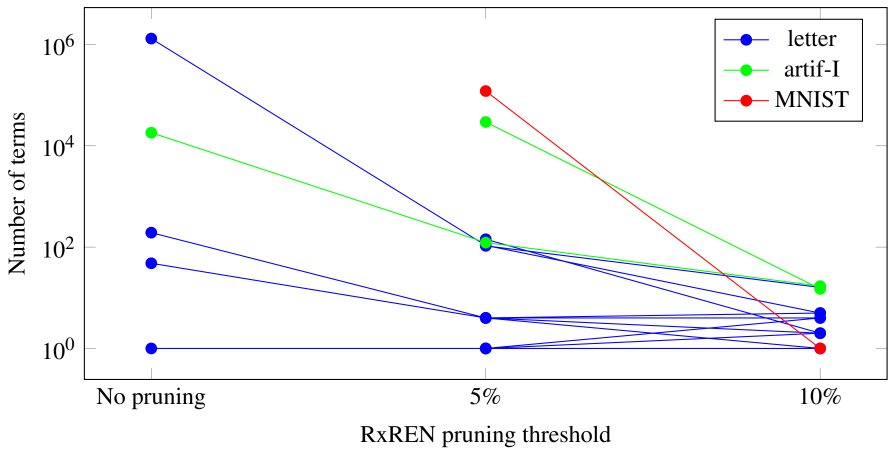

Fig. 6. The effect of DeepRED's different RxREN pruning parameters on the extracted rules' comprehensibility.

to analyse how the *RxREN* pruning affects a rule set's comprehensibility – and what it means for its fidelity. Figure 6 offers some insights for this question. We only show the datasets where not all inputs are important, i.e. *letter*, *artif-I*, and *MNIST*. The connected points represent experiments with the same *C4.5* parameters and training set sizes. Please note, that the figure only shows groups of experiments where at least two different pruning settings were successful.

One can clearly see that for the vast majority of experiments, the number of terms gets smaller when the pruning intensity gets higher. This holds true for all experiments when we only consider no pruning versus pruning with a threshold of 5%.

Figure 7 provides fidelity rates for exactly the same experiments as depicted in Figure 6. For the *artif-I* problem, where the 10% pruning setting produced the shortest rules, we also see the best fidelity rates for the same pruning setting. For the *MNIST* problem, instead, the threshold of 10% is too greedy and leads to a much worse fidelity.

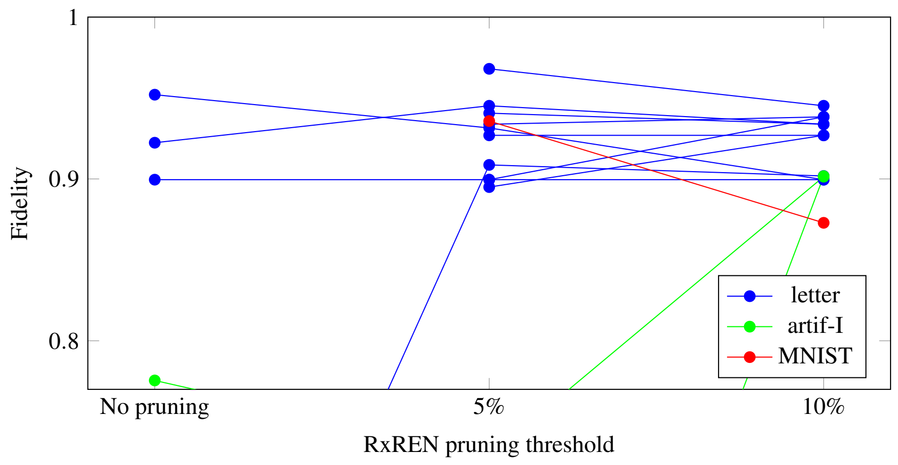

Fig. 7. The effect of DeepRED's different RxREN pruning parameters on the extracted rules' fidelity.

The results for the *letter* problem are less clear. We conclude that the pruning intensity needs to be adjusted in a more elaborate way to ensure the best achievable results. In total the results are better than expected since, although *RxREN* pruning overall increases the comprehensibility drastically, a clear trend towards lower fidelity rates cannot be recognized.

Another interesting observation is the fact that *RxREN* pruning in eight of 13 cases in the first place enables *DeepRED* to successfully extract rules. Without the pruning option, for instance, the algorithm would not be able to process the *MNIST* DNN at all.

DeepRED can Extract High-Quality Rules Independently from the Training Set Size – As a last measure we want to analyse the effect of the training set size on the results. Our belief was, that *DeepRED* profits from more available data. The graph in Figure 8 shows the results of all successful experiments on datasets with less than 4000 training examples (with a minimum fidelity of 77%).

Although we can clearly see that increasing the training set size from 10% to 25% has a positive influence on the fidelity, the overall results for the *artif-II* and *letter* datasets are surprising. Contrary to our expectations more data do not necessarily lead *DeepRED* to extract better rules. Often, the fidelity even gets worse when making more data available to the algorithm, while the baseline more often can profit from additional training data. A further investigation why this is the case is left for future research.

For the *XOR* dataset, Figure 8 shows us exactly what we expected – a strong correlation between the number of training examples and the fidelity of the rule sets. While the results for the baseline are not shown in the extract of the graph (since the results are never better than 50%), *DeepRED* manages to extract sensible rule sets with only 50% of the training data (which is about 29% of all possible input combinations). In our experiments with 100% training data, the proposed algorithm extracted rules that perfectly mimic the NN's behaviour. Again, we want to clarify that the 100% training data

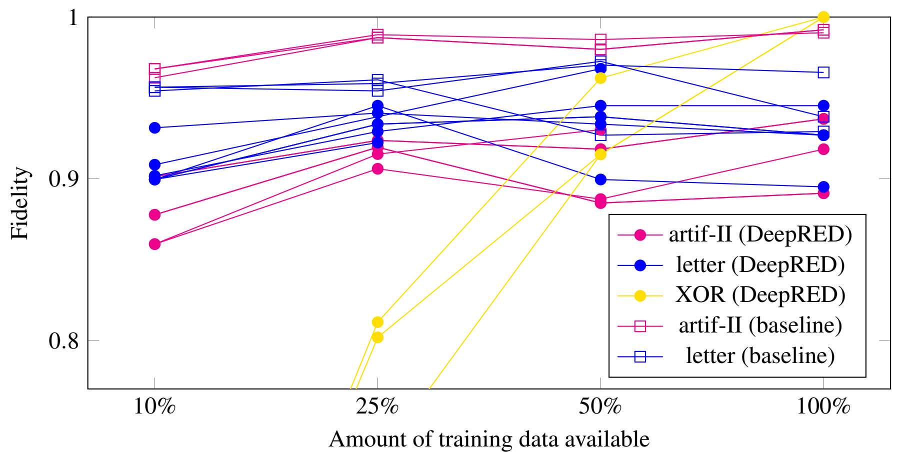

Fig. 8. The implications of different sizes of the available data on the extracted rules' fidelity.

are approximately 59% of all possible input combinations. The baseline would only be able to extract such a rule set, if all 256 possible instances are available for training.

This clearly shows the benefit of having access to the NN's structure. *DeepRED* does not need every input combination of all attributes in the training data – the examples must only contain the relevant input combinations anywhere in the training set. In the best case, for an *XOR* with four inputs, only four examples are needed[5], which is only 25% of all 16 possible input combinations. The deeper a similar *XOR* network, the more *DeepRED* benefits from the NN's architecture.

6 Conclusion

In this paper, we presented and evaluated *DeepRED*, a new decompositional algorithm that solves the problem of extracting rules from DNNs to make their decision processes more comprehensible.

Since most state-of-the-art algorithms do not consider DNNs at all, we created a new rule extraction algorithm that is able to deal with DNNs. We proposed *DeepRED* which extends the *CRED* algorithm. Our approach uses *C4.5* to create rules that describe neurons on the basis of neurons in the preceding layer. *DeepRED* then merges these rules to produce a rule set that mimics the overall behaviour of the given DNN.

Our evaluation with different parameter settings on five datasets showed *DeepRED*'s advantages over a pedagogical baseline. The proposed algorithm even managed to successfully extract correct rules from an *XOR* NN despite not seeing all training examples that would usually be necessary to replicate such a function with a rule learning algorithm.

[5] An example of a sufficient training set with the instance notation $x = x_1x_2x_3x_4$ would be 0011, 1101, 1000, and 0110. It contains all combinations of x_1/x_2 and x_3/x_4.

Future research on *DeepRED* is needed to better adjust its parameters and select or sample suitable training examples. Not only optimizing *C4.5* to better fit DNN rule extraction characteristics, but also a replacement of *C4.5* by another rule learner could be very interesting. Additional evaluations that are not restricted to binary classification problems would help to further optimize *DeepRED*. Since, to the best of our knowledge, the DNN rule extraction problem by now has not been analysed more extensively by other researchers, developing alternatives for *DeepRED* also is an interesting open research opportunity.

References

[1] Andrews, R., Diederich, J., Tickle, A.B.: Survey and critique of techniques for extracting rules from trained artificial neural networks. Knowledge-based systems 8(6), 373–389 (1995)

[2] Augasta, M.G., Kathirvalavakumar, T.: Reverse engineering the neural networks for rule extraction in classification problems. Neural processing letters 35(2), 131–150 (2012)

[3] Benítez, J.M., Castro, J.L., Requena, I.: Are artificial neural networks black boxes? Neural Networks, IEEE Transactions on 8(5), 1156–1164 (1997)

[4] Craven, M., Shavlik, J.W.: Using sampling and queries to extract rules from trained neural networks. In: ICML. pp. 37–45 (1994)

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

[6] Frey, P.W., Slate, D.J.: Letter recognition using holland-style adaptive classifiers. Machine learning 6(2), 161–182 (1991)

[7] Fu, L.: Rule generation from neural networks. Systems, Man and Cybernetics, IEEE Transactions on 24(8), 1114–1124 (1994)

[8] Johansson, U., Lofstrom, T., Konig, R., Sonstrod, C., Niklasson, L.: Rule extraction from opaque models–a slightly different perspective. In: Machine Learning and Applications, 2006. ICMLA'06. 5th International Conference on. pp. 22–27. IEEE (2006)

[9] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)

[10] Quinlan, J.R.: C4.5: Programs for Machine Learning, vol. 1. Morgan Kaufmann (1993)

[11] Russell, S., Norvig, P.: Artificial intelligence: a modern approach. Pearson Education (1995)

[12] Sato, M., Tsukimoto, H.: Rule extraction from neural networks via decision tree induction. In: Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on. vol. 3, pp. 1870–1875. IEEE (2001)

[13] Schmitz, G.P., Aldrich, C., Gouws, F.S.: ANN-DT: an algorithm for extraction of decision trees from artificial neural networks. Neural Networks, IEEE Transactions on 10(6), 1392–1401 (1999)

[14] Sethi, K.K., Mishra, D.K., Mishra, B.: KDRuleEx: A novel approach for enhancing user comprehensibility using rule extraction. In: Intelligent Systems, Modelling and Simulation (ISMS), 2012 Third International Conference on. pp. 55–60. IEEE (2012)

[15] Setiono, R., Leow, W.K.: FERNN: An algorithm for fast extraction of rules from neural networks. Applied Intelligence 12(1-2), 15–25 (2000)

[16] Taha, I.A., Ghosh, J.: Symbolic interpretation of artificial neural networks. Knowledge and Data Engineering, IEEE Transactions on 11(3), 448–463 (1999)

[17] Thrun, S.: Extracting provably correct rules from artificial neural networks. Tech. rep., University of Bonn, Institut für Informatik III (1993)

[18] Thrun, S.: Extracting rules from artificial neural networks with distributed representations. Advances in neural information processing systems pp. 505–512 (1995)
[19] Towell, G.G., Shavlik, J.W.: Extracting refined rules from knowledge-based neural networks. Machine learning 13(1), 71–101 (1993)
[20] Tsukimoto, H.: Extracting rules from trained neural networks. Neural Networks, IEEE Transactions on 11(2), 377–389 (2000)
[21] Zhou, Z.H., Chen, S.F., Chen, Z.Q.: A statistics based approach for extracting priority rules from trained neural networks. In: Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on. vol. 3, pp. 401–406. IEEE (2000)