Re-Training Deep Neural Networks
to Facilitate Boolean Concept Extraction

Camila González ✉(0000-0002-4510-7309), Eneldo Loza Mencía, and
Johannes Fürnkranz (0000-0002-1207-0159)

Knowledge Engineering Group, TU Darmstadt
Hochschulstrasse 10, 64289 Darmstadt, Germany
`camilag.bustillo@gmail.com, [eneldo,juffi]@ke.tu-darmstadt.de`

Abstract. Deep neural networks are accurate predictors, but their decisions are difficult to interpret, which limits their applicability in various fields. Symbolic representations in the form of rule sets are one way to illustrate their behavior as a whole, as well as the hidden concepts they model in the intermediate layers. The main contribution of the paper is to demonstrate how to facilitate rule extraction from a deep neural network by retraining it in order to encourage sparseness in the weight matrices and make the hidden units be either maximally or minimally active. Instead of using datasets which combine the attributes in an unclear manner, we show the effectiveness of the methods on the task of reconstructing predefined Boolean concepts so it can later be assessed to what degree the patterns were captured in the rule sets. The evaluation shows that reducing the connectivity of the network in such a way significantly assists later rule extraction, and that when the neurons are either minimally or maximally active it suffices to consider one threshold per hidden unit.

Keywords: Deep neural networks · Inductive rule learning · Knowledge distillation

1 Introduction

Deep neural networks [11] achieve state of the art performance in a variety of different fields, like computer vision, speech recognition and machine translation. They can be leveraged both in supervised and unsupervised problem formulations, as they automatically learn insightful features out of unprocessed data. In the last few years, they have considerably risen in popularity as advancements in the training practices and availability of user friendly frameworks have made it much simpler to train accurate models, as long as sufficient data is available.

However, the fact that the models are governed by a high number of parameters makes tracing the path that led to a classification an arduous process, which is why they are often regarded as *black boxes*. This is a significant shortcoming, as it makes them unsuitable for safety critical applications and domains where there are juridical barriers which either explicitly forbid their use or implicitly discourage it by making the user liable for the model's decisions. Amongst the legislation that aims for more comprehensible prediction models is the *General Data Protection Regulation (GDPR)*[1]

---

[1] EU Regulation 2016/679: http://eur-lex.europa.eu/eli/reg/2016/679/oj, http://www.eugdpr.org

planned to take effect in 2018. There is also an ongoing European legislative initiative on *Civil law rules on robotics*[2].

In fields such as health care and criminal sentencing, comprehensible models like decision lists or trees are favored because they provide understandable evidence to support their predictions [17, 18]. *Decision support systems (DSS)* aim at integrating machine learning models into a human-centered decision process. Here, interpretability is of particular advantage because a justified decision is more likely to convince the human to support or disregard the machine's recommendation. Besides, the extent to which the model is used in practice depends heavily on how easily interpretable it is, as this is a relevant criteria for eliciting trust [15].

Compared to neural networks, *if-then* rule sets are a representation with a good trade-off between human and machine interpretability [10]. This is partly because they provide a symbolic representation which more closely resembles the way humans model logic. Also, each rule can be observed individually, so only a limited amount of information must be considered at any time. This advantage, together with the fact that they can be more flexibly pruned, sometimes makes them preferable over decision trees [8].

Consequently, researchers have looked into converting neural networks into a rule-based representation. One problem with such approaches is that much information is lost when the continuous range of activation levels of the internal neurons is mapped to a two-valued logical representation. In this paper, we investigate ways for retraining deep neural networks with the goal of encouraging sparse connectivity and minimally or maximally active hidden units, with the idea of facilitating a later extraction of rules from the network. We study the problem on the task of reconstructing Boolean functions, because there we can see whether the use of the network's structure really helps to recover the logical structure in the target function.

We will start our discussion with a brief general overview of prior work on rule extraction from neural networks (Sec. 2), with a particular focus on the DEEPRED algorithm, upon which which forms the basis of our work (Sec. 3). The core contribution of this paper, an algorithm for retraining DNNs to extract better representations, is described in Sec. 4, and experimentally evaluated on the problem of reconstructing Boolean functions in Sec. 5.

2 Knowledge Distillation from Neural Networks

Much of the predictive strength of deep neural networks originates from their ability to form latent concepts in the hidden layers, and the high connectivity betweeen these layers makes it difficult to distill the meaning of these concepts. One approach is to rely on visualization in order to analyze the behavior of the learned network (see, e.g., [3]). However, another line of research concentrates on ways of making the knowledge that is implicitly captured in these nodes explicit and amenable to human inspection. Typically this is done by transforming the neural network into more interpretable knowledge representations such as rules or decision trees. A prerequisite for such work is often to simplify the network by pruning unnecessary connections and neurons. We will briefly recapitulate work in these areas in Secs. 2.1 and 2.2, respectively.

---

[2] http://www.europarl.europa.eu/oeil/popups/ficheprocedure.do?reference=2015/2103(INL)

2.1 Rule Extraction

Many approaches have been developed to extract symbolic representations from neural networks. However, most either do not consider the network's internal structure or are only applicable to shallow architectures. A distinction can be made between *pedagogical* methods, which regard the network as a black box and map relationships between the outputs and the inputs, *decompositional* ones that observe the contribution of individual parameters or neurons and *eclectic* methods which fall between the other two [5]. Other categorizations refer to the computational complexity of the approach, what data is used to build the model and whether a particular training regime is performed [2].

A first group is made up of *subset* approaches [9,28,29]. These are decompositional and typically assume a polarization of the activations and the use of exclusively binary inputs. They search the entire feature space and construct one expression per neuron of interest. Typically, a threshold is applied to the neuron's output to define an active and an inactive state. Rules are then learnt for the active state by finding combinations of the incoming weights that cause the bias of the hidden unit to be exceeded.

A shortcoming of these methods is that considering all subset combinations grows at an exponential rate with the number of incoming connections, which limits their applicability to larger networks. It also cannot be assumed that any network can be converted to one with only maximally or minimally active neurons while maintaining the initial accuracy. An even more difficult requirement to fulfill is that inputs should be discrete so they can be binarized without information loss.

Another problem that arises when sampling all possible inputs is that the way the network reacts to implausible instances may not be meaningful, so capturing this logic may result in an overly complicated rule model which is not better at classifying unseen, naturally occurring examples. Some methods thus focus primarily on the instances used to train the network when building the symbolic model.

Such is the case for the pedagogical TREPAN algorithm [6], which explains the outputs of the network with respect to the inputs by building decision trees directly between these layers. The tree building process makes use of queried instances, generated from the marginal distribution of each attribute, to avoid low amounts of data as the tree branches. However, in a comparison of different variants [19], that which did not generate new examples performed best.

CRED [21] also builds decision trees between network layers using the train data, but it works in a decompositional manner. First, a target condition is specified to discretize the class values, and decision trees are built to explain this output pattern with the hidden units as attributes, using the corresponding activation values of the training instances. The range of each hidden unit is partitioned in an online manner, so several thresholds may be considered per unit, and some units may be ignored. The trees are then converted into sets of decision rules. Redundant and unsatisfiable rules and terms are deleted, and rules are merged by forming their least general generalization (lgg) by selecting the most general condition of the attributes they share, and dropping all conditions of attributes they do not share. For instance, $a \leq 0.3 \wedge b > 4 \wedge c > 2 \rightarrow C_1$ and $a \leq 0.2 \wedge b > 3 \wedge d > 2 \rightarrow C_1$ would become $a \leq 0.3 \wedge b > 3 \rightarrow C_1$. Afterwards, analogous rule sets are built which explain each split value considered for a the hidden

4

units with respect to the inputs, which now make up the attributes. Finally, the *total* rules are formed by substituting the hidden split values with these new rule sets.

2.2 Connection Pruning

Pruning connections or whole neurons of trained neural networks is a common way to adapt the topology of the network to the effective size of the problem, thus discouraging overfitting and increasing the generalization capabilities. It can also be leveraged to require less time and resources when making classifications [13, 16, 27].

A connection $w_{j,k}^l$ between two neurons $h_{l-1,k}$ and $h_{l,j}$ can be pruned by equaling the weight entry to zero. This is similar to applying dropout [26] but whereas dropout temporarily removes randomly chosen connections for one epoch at a time, pruning permanently removes selected connections from the network. Connections can be pruned iteratively by retraining the network after each pruning step, which allows to discard a considerably higher number [12]. Note that pruning connections can result in indirectly pruning whole inputs or hidden units, as a neuron without output connections is disconnected from the network.

In [24] a method is introduced to prune connections from shallow neural networks. First, the networks are trained with a weight-decay penalty. Connections $w_{p,j}^2$ between the hidden and output layers are then pruned if

$$\left| w_{p,j}^2 \right| \leq \eta \,, \tag{1}$$

and connections $w_{j,k}^1$ between the inputs and hidden units are pruned if

$$max_p \left| w_{p,j}^2 . w_{j,k}^1 \right| \leq \eta \,. \tag{2}$$

If no connection fulfills one of those conditions, then the entry $w_{j,k}^1$ for which the minimum of the maximum products is lowest is pruned. Afterwards, the network is retrained. If the final error falls below an acceptable level, the pruning step is repeated; otherwise the last acceptable parameters are restored and the process is stopped.

The author uses this approach extensively as a preprocessing step before applying rule extraction algorithms [22, 25]. The pruning phase is usually followed by a discretization of the hidden unit activations. In [23], the connectivity of the hidden units is further reduced by 'splitting' those with many input connections. Each new unit is treated as an output and a hidden layer is inserted in the middle between the inputs and the new output layer. The network is retrained and the new subnetwork pruned, and the process is repeated until each neuron only has a small number of inputs.

3 The DEEPRED Algorithm

In order to extract representations from deep neural networks which not only explain the network's predictive behavior but also uncover hidden features, we make use of the DEEPRED algorithm [30], which extends CRED (Sec. 2.1) to deep neural networks. It is scalable to large architectures, works in a decompositional manner and has been shown to be capable of extracting accurate models from deep neural networks. We

F : A ∨ B

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

train →

$x_0 \rightarrow$

$x_1 \rightarrow$

$\rightarrow y_0$

$\rightarrow y_1$

sample →

x_0	x_1	$h_{1,0}$	$h_{1,1}$	$h_{1,2}$	$h_{2,0}$	$h_{2,1}$
0	0	-0.9	0.9	-0.9	0.8	0.2
0	1	-0.9	-0.9	-0.9	0.1	0.9
1	0	0.9	0.9	-0.9	0.2	0.8
1	1	0.9	0.9	-0.9	0.1	0.9

convert to DNF and simplify   C4.5

$\mathbf{h_{1,1} \leq 0 \vee (h_{1,1} > 0 \wedge h_{1,0} > 0) \rightarrow h_{2,1} > h_{2,0}}$

$\mathbf{h_{1,1} \leq 0 \vee h_{1,0} > 0 \rightarrow h_{2,1} > h_{2,0}}$

$\mathbf{h_{2,1} > h_{2,0}}$

$\mathbf{h_{1,0} \leq 0}$ — F / T

$\mathbf{h_{1,1} \leq 0}$ — F / T

$\mathbf{h_{2,1} \leq h_{2,0}}$

$\mathbf{h_{2,1} > h_{2,0}}$

$\mathbf{h_{1,1} > 0}$

$\mathbf{x_0 \leq 0.5}$ — F / T

$\mathbf{x_1 \leq 0.5}$ — F / T

$\mathbf{h_{1,1} > 0}$

$\mathbf{h_{1,1} \leq 0}$

$\mathbf{h_{1,0} > 0}$ — F

$\mathbf{x_0 \leq 0.5}$ — T

$\mathbf{h_{1,0} \leq 0}$

$\mathbf{x_0 \leq 0.5 \wedge x_1 > 0.5 \rightarrow h_{1,1} \leq 0}$

$\mathbf{x_0 > 0.5 \rightarrow h_{1,0} > 0}$

$\boxed{\mathbf{x_0 > 0.5 \vee x_1 > 0.5 \rightarrow h_{2,1} > h_{2,0}}}$

substitute

Fig. 1. The general workflow of DEEPRED.

extended DEEPRED with a post-pruning step (Sec. 3.3) meant to contain error propagation and reduce the complexity. This is carried out each time a rule set is generated from a decision tree, and between substitution steps when building the expression of the target with regards to the inputs.

3.1 Overview

The DEEPRED algorithm extracts rules between any two layers by building decision trees for layer h_l using the activations from layer h_{l-1} as attributes. The trees are then transformed to rule sets, and a merging step converts the intermediate representations into a single rule set connecting the inputs with the outputs. Redundant and unsatisfiable rules and conditions are deleted, but unlike in CRED no further pruning takes place. There is a version of the algorithm that performs a feature selection prior to rule extraction by considering the contribution of each input for correctly classifying the training data and removing inputs that do have a great impact. This proves to be very advantageous when the network is used for high-dimensional data.

Fig. 1 exemplifies how DEEPRED would extract rule representations from a shallow neural network which emulates a Boolean function. The model is sampled to obtain activation values for each training instance. A first tree is built to predict under what activation settings of layer h_1 the target concept is fulfilled, namely that class C_1 has a higher probability than C_0. The tree is converted into a DNF representation and the expression is simplified. A tree is then built for each literal which appears in the simplified expression, using the input values as attribute data. Each of these expressions is extracted and simplified, and a last step substitutes the literals with regards to the hidden layer so that the expression which predicts class C_1 is in terms of the inputs.

3.2 Extraction of DNF Formulas from Trees

A rule can be regarded as a conjunction of terms, where a term is a condition indicating whether the activation state of a neuron falls or not above some threshold, and a rule set as an expression in disjunctive normal form (DNF). Each tree built by DEEPRED determines whether an input or the activation of a hidden unit fulfills one such term, using terms with respect of the adjacent shallower layer. For instance, a tree could determine if the value 0.5 is exceeded by the second neuron in the first hidden layer. Such a tree would have two possible classes, $h_{1,2} > 0.5$ and $h_{1,2} \leq 0.5$. A DNF for each of them can be obtained by joining the respective terms on all paths from the root to the corresponding leafs of the tree.

A separate DNF formula is maintained for each class, so there are two DNFs per split value, each of which fires as soon as one of its rules fires. A DNF formula for the event of neuron $h_{2,2}$ exceeding the threshold 0.5 may look like $(h_{1,1} > 0.5 \land h_{1,2} \leq 0.3) \lor (h_{1,2} > 0.3 \land h_{1,3} > 0.7) \rightarrow h_{2,2} > 0.5$.

The expressions for opposite class conditions, as would be those for $h_{2,2} > 0.5$ and $h_{2,2} \leq 0.5$, are complementary after being extracted for the tree, but this may no longer hold once pruning is applied. Thus, both may fire for a given example, or neither may do so. Usually, additional criteria such as a priority list for tie-breaking between multiple predictions or a default rule for the latter case are employed. However, in our case only one expression is maintained for the selected class, so the inconsistencies within intermediate expressions do not translate to ambiguities in the final class prediction.

3.3 Simplification and Post-pruning of Expressions

Transforming a decision tree into a rule set, as well as the process of building the expression of the target with regards to the inputs by sequentially substituting terms by DNFs, can result in expressions with redundant and unsatisfiable rules and redundant conditions. These are removed each time an expression is extracted from a decision tree and between substitution steps.

Very similar rules may still remain which do not provide more information than a simpler rule would. This affects the comprehensibility and can promote error propagation between layers. Yet, a too strong generalization of the intermediate concepts should be avoided as its repercussions on the final expression cannot be observed until the end. CRED (cf. Sec. 2.1) employs a pruning approach which is advantageous in shallow networks but proved in preliminary experiments to be too aggressive for deep networks. Instead, we use a method of reduced error pruning that only makes a change if this positively affects the accuracy with respect to the head of the rule.

Rules are ordered in terms of increasing precision. For each rule, the change in accuracy is calculated in case the rule is deleted, a term is removed from the rule (calculated for all terms) and the rule being merged with another one from the set (calculated for all remaining rules). The modification which leads to the highest accuracy is performed if it improves the accuracy over the current rule set. Unless the modification consists on removing the rule completely, the precision is calculated for the new rule, which is regarded as unseen. This is repeated until there are no unseen rules left.

4 Retraining DNNs to Extract Better Representations

One problem with rule extraction from neural networks is that the activations assume continuous values within some range, whereas a mapping to a decision tree or rule set reduces them to a discrete setting. The key idea of our work is that the transformation process may be supported by forcing the network weights to assume more extreme values. In this section, we therefore present methods for retraining a deep neural network in order to encourage sparseness in the weight matrices and make the hidden units be either maximally or minimally active. For this work, we consider the accuracy on the entire dataset for guiding the retraining, because our goal is to train the networks to exactly emulate predefined concepts. However, in a different setting it might be advisable to use a separate validation set, which is not used to optimize the parameters.

4.1 Weight Sparseness Pruning

We employ a connection pruning technique that is quite similar to that described in Sec. 2.2. In contrast to that method, ours can be applied to deep networks, and its aim is to sparsify all weight matrices so that the total number of connections between any two layers is reduced. This has the effect that single neurons are neither connected to a majority of the neurons of the following layer nor to a majority of those from the previous layer. The expectation is that, as observed by [23], rules extracted from minimally connected neurons will be simpler and more accurate.

The motivation for targeting such connections also comes from the performance of DEEPRED when applied to a network manually constructed to emulate the parity function with eight inputs [30]. The network constructed for this experiment has a recursive structure from the eight inputs to the output layer and is minimally connected. DEEPRED is not only able to extract the modeled DNF representation using a significantly lower number of instances than a pedagogical approach, but its intermediate rules also exactly replicate the recursive features.

In preliminary experiments, we could not repeat this effect on fully connected networks of the same topology trained with backpropagation, even if all combinations were used for training. When rules are extracted from such networks using a reduced set of instances, the majority of the logic is concentrated between the inputs and the first hidden layer. The rule sets extracted between these layers overfit the train data, and each depends on the majority of the inputs. Therefore, the accuracy on the unseen instances never exceeds fifty percent and actually decreases with increasing amounts of training data (a phenomenon that also affects C4.5). If, on the other hand, the number of connections is reduced, the network may be encouraged to learn a reduced amount of hidden features that are more abstract and apply to a greater percentage of examples.

General Methodology. A connection $w_{j,k}^l$ is represented by the index of the weight matrix l and the row and column indices j and k. The number of entries that have already been pruned in each row or column of each weight matrix is maintained in order to calculate the *neighborhood sparseness* of the remaining entries. This value is determined by the sum of entries that have been pruned in row j of matrix W^l plus those that have been pruned in column k of the same matrix.

On each step, a list of all existing connections is sorted in terms of increasing neighborhood sparseness, and it is attempted to prune the next head element, which is likely to be surrounded by unpruned entries. The target training accuracy that must be reached after retraining for the connection to remain pruned is the original train accuracy minus an allowed decrease. If the accuracy is satisfactory, the connection is pruned, the counts for column and row pruned entries are updated, and the list is re-sorted. Otherwise, the last accepted parameters are restored, and the next connection is removed from the list.

Iterations Used for Retraining. Preliminary experiments showed that often a small number of iterations suffice to determine whether a connection can be pruned, because the network gets stuck in a local minimum. On the other hand, some connections cause a steep decrease in accuracy when they are first removed, but the network later adapts. To allow the latter connections to be eliminated while not considerably increasing the retraining time, the retraining epochs are divided into smaller sets. If after a set the accuracy is equal or greater than the target accuracy, the connection is pruned, otherwise the retraining continues until either n retraining steps were performed from the time a connection was pruned or there were no improvements in the last m steps, with $n \geq m$.

Re-exploration of a Connection. It was also observed that if a connection could not be pruned at some stage, it was unlikely that it could be pruned later on, even if other connections had been pruned which affected the neurons it joined. Therefore, in the experiments outlined in Sec. 5 there was only one attempt of pruning per connection.

4.2 Activation Polarization

The activation range of the hidden units being continuous has several negative repercussions, such as making it more costly to classify new instances, so techniques have been developed for binarizing the parameters and activation values [1,4]. Yet, most networks are trained in such a way that the hidden units can take any value within the range.

As representing each neuron with only one expression is a more comprehensible way to illustrate that neuron's purpose in the network than if different expressions have to be regarded for an array of activation intervals, many decompositional rule extraction approaches reduce the possible states each neuron takes to being either at the bottom or the top of the activation range (cf. Sec. 2.1).

In order to extract rules which are true to the network while making this assumption, the networks must be trained in a way that the activations are polarized. There are several ways to achieve this. We propose a retraining step similar to that used in [20] to encourage sparse activations. The key idea is to penalize the loss function with a term based on the *KL divergence* between the mean absolute value of each activation

$$\hat{\rho}_{l,n} = \frac{1}{|D|} \sum_i \left| a_{l,n}^i \right| \tag{3}$$

and a ρ close to one, which results from the use of a hyperbolic tangent function. These terms are summed up over all hidden units, yielding

$$\sum_{l,n} KL(\rho \,\|\, \hat{\rho}_{l,n}) = \sum_{l,n} \rho \log \frac{\rho}{\hat{\rho}_{l,n}} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_{l,n}} \,. \tag{4}$$

This term introduces the additional parameters ρ for the optimal activation average and β for the weighting of the penalty term. Instead of having to define β, the retraining method as implemented for this work starts by setting $\beta = 0$, and increases this value iteratively. Between each increment, a number of epochs are conducted, for as long as there is no decrease in accuracy, and the divergence stays above some threshold. The last weight and bias parameters are stored before each increase of β. If the process stops because the accuracy falls, the parameters which were saved last are restored.

5 Experiments

In order to show that we can derive meaningful conceptual descriptions from deep neural networks, we performed experiments on artificial datasets. Our goal was to demonstrate that our algorithms can reconstruct Boolean input functions from networks trained to model them. For this, we first made a quality assessment of the concepts extracted when the entire dataset is available. After exploring the limits of each approach, we compared the generalization abilities of the different variants by utilizing a subset of the combinations as training data and analyzing the accuracy on the remaining instances.

5.1 Experimental Setup

Data. We used twenty datasets constructed from Boolean functions with six to fourteen literals. Nineteen were generated by joining groups of randomly selected literals with alternating OR and AND operators and choosing to apply negation over each group with a 0.2 probability, and one was the parity function with eight inputs. The expressions were simplified with the ESPRESSO heuristic logic minimizer [14]. Each dataset was made up of all combinations of literals in the simplified expression.

Network Architecture and Training. The networks had three hidden layers, the first with twice as many neurons as inputs, the second with the average of that number and two and the third and output layers with two neurons. The hyperbolic tangent was used as activation function, and a softmax function was applied on the last layer. The networks were trained using the *TensorFlow* framework and cross-entropy as the loss function. They were trained with all input combinations until achieving a perfect accuracy. In this way, it was certain that they mimicked the logic of the predefined formulas.

Compared Algorithms. We compared several variants of our approach where (i) no retraining took place, (ii) weights sparseness pruning was performed, and (iii) a polarization of the activations was followed by weight sparseness pruning. Also, as observing one expression per hidden neuron of interest, which predicts when that unit is in an active state, is more comprehensible than having to consider an array of expressions, it was analyzed how the models would be affected if instead of allowing the online discretization of C4.5 to select thresholds of the activation range, only the midpoint of this range was considered.

Hyperparameter setting. C4.5 was set to stop growing a tree when the percentage of the majority class exceeded 99% or only less than 1% of the original instances remained in a node. Only binary splits were allowed and the trees had a maximum depth of twenty nodes. For activation polarization, ρ was set to 0.99 and β was increased by 0.1 at a time. For weight sparseness pruning, a 1% decrease in accuracy was allowed. In both cases, each epoch set consisted of 1000 epochs. For the networks which were retrained in both manners, the penalty term from the activation polarization was added to the loss function used during connection pruning, multiplied by the last accepted value of β.

Evaluation Measures. The comprehensibility of the intermediate logic – which is to say that between subsequent layers – was assessed with the number of expressions and the total number of terms. The semantic quality was measured using the accuracy, which is to say the proportion of correctly classified instances among all classifications. Note that as the networks used perfectly replicate the Boolean functions, this corresponds to the fidelity of the extracted rules mimicking the network's behavior.

For determining whether observed performance differences between two classifiers were statistically significant, the sign and Wilcoxon signed ranks tests were used for a significance level of $p = 0.05$. Following [7], ties were distributed, and in the event of an uneven number of ties, the number *N* of datasets was reduced by one. Also subtracted from this number were comparisons which could not be performed because of uncompleted experiments. This occurred when the time or memory constraints for the extraction – set respectively at 24 hours and 5000 MegaBytes – were surpassed or when no model could be built using one threshold per hidden unit. At least one experiment could not be completed for a total of three datasets, including that of the parity function.

5.2   Characteristics of the Trained Networks

After retraining the networks it was observed how many weights had been pruned, and how well the neurons could be polarized by calculating the deviation of the activations from zero averaged over all hidden units and examples. The results suggest that a trade-off takes place between the divergence from the range center and the number of pruned connections, as can be observed in Fig. 2. The pruning approach eliminated a great percentage of the connections, but at the cost of distributing the activation values more evenly across the range. When the networks were first polarized and the penalty term was maintained during the latter connection pruning, the activation values gathered even closer to the range boundaries, but considerably less connections were eliminated.

5.3   Reconstruction Using the Entire Dataset

There was a noticeable change in the number of intermediate expressions which were extracted – as well as in their complexity measured with the number of terms – when the networks were retrained under weight sparseness pruning. As can be observed in Fig. 3, models taken from less connected networks were much more compact.

How this reflects in the extracted models is exemplified in Fig. 4 for the expression $(x_3 \wedge x_1 \wedge x_5) \vee (\bar{x}_3 \wedge \bar{x}_0 \wedge \bar{x}_4 \wedge x_2 \wedge \bar{x}_5)$. The model extracted from the original network finds an adequate representation for the second conjunction in $h_{1,6} > 0.32$, as
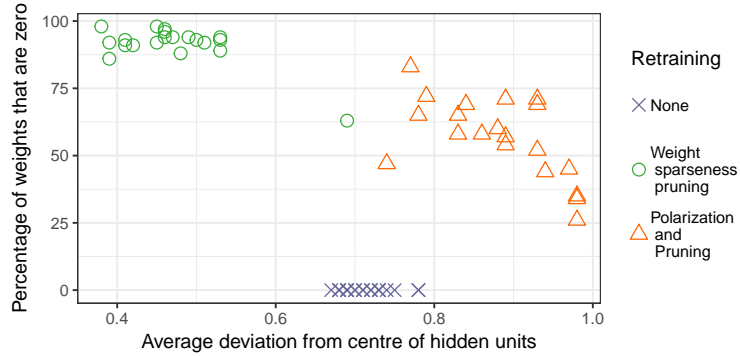
Fig. 2. Trade-off between deviation of the activations from the center of the range and weight entries that were pruned. Each point represents the result on one of the 20 datasets, either with no retraining, retraining after weight sparseness pruning or retraining after polarization and posterior weight sparseness pruning.
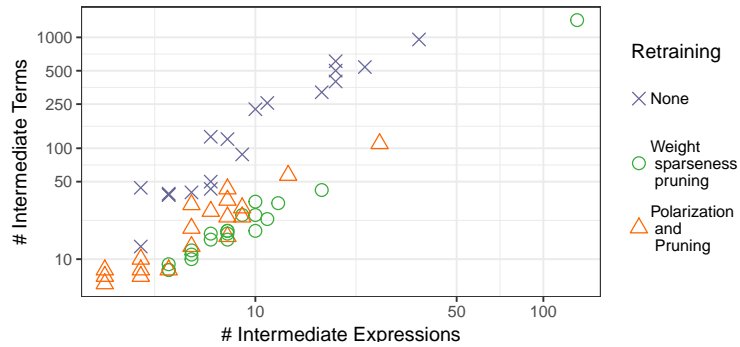

Fig. 3. Change in the complexity of the intermediate concepts when preceding extraction with weight sparseness pruning or polarization of the activations followed by pruning.

well as for the first part of the first conjunction in $h_{1,0} \leq -0.26$, but fails to do so for x_5. The model extracted from the polarized and pruned network includes instead only expressions with only a couple of literals each and it therefore much simpler to trace.

Applying the different retraining methods on the networks did not cause a substantial change in accuracy when any threshold could be considered. Neither was a significant difference found when only zero was used to partition the activation ranges of the hidden units, except for when the networks had been subjected to weight sparseness pruning but no activation polarization. This effect is shown in Fig. 5. The black circles, which refer to the models extracted from unpolarized and pruned networks which only include conditions of the hidden units with zero as threshold, illustrate a clear fall in accuracy when compared to the rest of the models.

These results reinforce the hypothesis that, when a high number of network connections are pruned and a retraining phase is performed between pruning steps, the logic

Model extracted from original network	
$h_{3,1} \leq 0.15$	$\rightarrow target$
$h_{2,6} > 0.32 \lor h_{2,4} > 0.56$	$\rightarrow h_{3,1} \leq 0.15$
$(h_{1,0} \leq -0.26 \land h_{1,7} \leq -0.65) \lor (h_{1,0} \leq -0.26 \land h_{1,8} > 0.12)$	$\rightarrow h_{2,4} > 0.56$
$h_{1,6} > 0.32$	$\rightarrow h_{2,6} > 0.32$
$(x_4 \land x_5) \lor (x_1 \land \bar{x}_2 \land x_5) \lor (x_1 \land \bar{x}_2 \land \bar{x}_3 \land x_4) \lor (x_1 \land \bar{x}_3 \land x_5)$	$\rightarrow h_{1,8} > 0.12$
$(\bar{x}_0 \land x_3 \land x_5) \lor (x_2 \land x_3 \land x_5) \lor (\bar{x}_1 \land x_2 \land x_4 \land x_5) \lor (\bar{x}_0 \land x_2 \land \bar{x}_3) \lor (x_3 \land x_4 \land x_5)$	$\rightarrow h_{1,7} \leq -0.65$
$x_1 \land x_3$	$\rightarrow h_{1,0} \leq -0.26$
$\bar{x}_0 \land x_2 \land \bar{x}_3 \land \bar{x}_4 \land \bar{x}_5$	$\rightarrow h_{1,6} > 0.32$

Model extracted from polarized and pruned network	
$h_{3,1} \leq -0.10$	$\rightarrow target$
$h_{2,6} > 0.88 \lor h_{2,6} \leq -0.99$	$\rightarrow h_{3,1} \leq -0.10$
$h_{1,0} \leq -0.01 \land h_{1,8} > -0.01$	$\rightarrow h_{2,6} \leq -0.99$
$h_{1,5} > -0.01 \land h_{1,8} \leq -0.01 \land h_{1,9} > -0.93$	$\rightarrow h_{2,6} > 0.88$
$\bar{x}_0 \land \bar{x}_4$	$\rightarrow h_{1,9} > -0.93$
$x_1 \land x_3$	$\rightarrow h_{1,0} \leq -0.01$
$\bar{x}_5$	$\rightarrow h_{1,8} \leq -0.01$
x_5	$\rightarrow h_{1,8} > -0.01$
$x_2 \land \bar{x}_3$	$\rightarrow h_{1,5} > -0.01$

Fig. 4. Effect of preceding DEEPRED with activation polarization and weight sparseness pruning in the model that reconstructs the expression $(x_3 \land x_1 \land x_5) \lor (\bar{x}_3 \land \bar{x}_0 \land \bar{x}_4 \land x_2 \land \bar{x}_5)$.
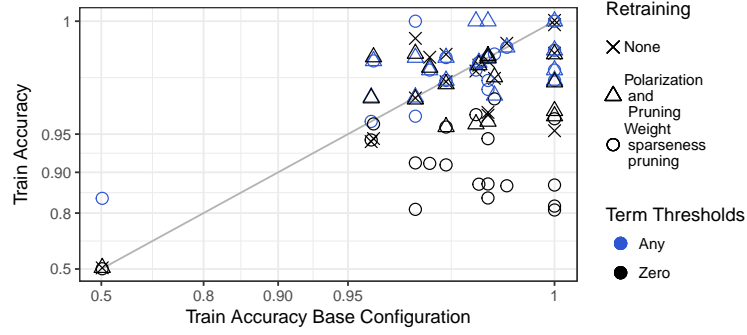

Fig. 5. Extent to which the concepts modeled by the neural networks were reconstructed when using all input combinations. Different configurations are compared to that where the network is not retrained and the term thresholds are selected by C4.5 in an online manner.

modeled by the network is more heavily concentrated in the remaining neurons, thus needing to subdivide the neuron range into more intervals to describe it.

5.4 Reconstruction Using Part of the Dataset

For these experiments, the data was split into ten, four or two stratified folds and a cross-validation was performed. Also, the folds were inverted to observe the situation where lower data percentages were available. This resulted in ten experiments where 10% and 90% of the data was used, four with 25% and 75% being available, and two for the 50% case. The evaluation measure was averaged over all folds.

Again, the analysis focused on the effect of the different retraining methodologies. As models extracted from networks that had been pruned but not polarized using the range midpoint as sole threshold had a very low train accuracy, the effect of enforcing
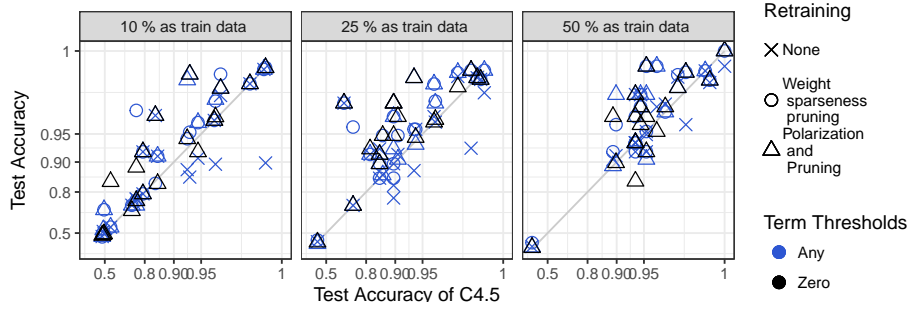
Fig. 6. Test accuracy of the models when using 10%, 25% and 50% of the dataset as train data and the remaining instances as test data. Different configurations are compared to the C4.5 algorithm, which disregards the internal structure.

this constraint was only analyzed for the variant including both activation polarization and weight sparseness pruning.

Generally, the best performing models were those for which weight sparseness pruning had been performed, but significant differences were only found when less data was available. When more data was used to build the models, the predictive accuracy approached the accuracy on the train data. A special case was the parity function, for which none of the approaches extracted a well generalizing model, with the accuracies laying at 0.5 or below. Though the models extracted from pruned networks displayed slightly higher accuracies, we were eventually not able to resolve the issue for this very special case, which was our initial motivation for the pruning and re-training approaches (cf. Sec. 4.1), and leave further investigation for future work.

The results when using 50% of the data or less are illustrated in Fig. 6. The performances are shown in direct comparison to using plain C4.5 between the input and output data. Compared to C4.5, the variant which did not perform network pruning did not show any significant difference. That which only included weight sparseness pruning outperformed C4.5 when 50% and 10% of the data was present according to the Wilcoxon test (with $Z = 2.22$ and $Z = 2.63$) and for both tests when 25% was used (16 wins out of $N = 20$, $Z = 3.06$). The same held for the polarized and pruned variant (with, respectively, $Z = 2.31$, $Z = 2.52$, 14 wins out of $N = 19$ and $Z = 2.75$). This variant also outperformed C4.5 using only one threshold per neuron according to both tests when 25% of the data was used (with 16 wins from $N = 19$ and $Z = 3.40$).

The unpolarized pruned variant outperformed that with no retraining according to both significance tests when 25% of the data was used (with 16 from $N = 19$ wins, $Z = 3.11$). The variant for which both retraining methods had been applied was deemed better by the Wilcoxon test when using 10% of the data ($Z = 2.07$).

6 Conclusion

Reducing the connectivity of the network proved to be a robust way for extracting simpler intermediate concepts, which were also better at classifying unseen instances. Yet

it seems that encouraging low connectivity not only identifies irrelevant logic created from training too large architectures, but also concentrates the hidden features which are in fact relevant for the classification into fewer neurons. Thus a finer-grained partitioning of the activation ranges is required to regain the hidden patterns.

This was partly shown by an analysis of the characteristics of the network, which exposed a trade-off between the extent to which the activation values could be polarized and the percentage of connections that could be pruned. The negative consequences of this effect for rule extraction were confirmed by the dismal performance of models which combined connection pruning with only considering the center of the activation range as threshold.

However, when polarization of the activations was done jointly with connection pruning, the benefits of the latter could be leveraged while avoiding the undesired effect of concentrating more logic into less neurons. Although the number of connections which could be pruned in these networks was substantially lower, the intermediate models were not significantly more complex, and in terms of accuracy these approaches consistently performed within the highest.

Acknowledgements

References

1. Aizenberg, I., Aizenberg, N.N., Vandewalle, J.P.: Multi-Valued and Universal Binary Neurons: Theory, Learning and Applications. Springer Science & Business Media (2013)
2. 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)
3. Bischof, H., Pinz, A., Kropatsch, W.G.: Visualization methods for neural networks. In: Proceedings of the 11th International Conference on Pattern Recognition (ICPR-92). vol. 2, pp. 581–585. IEEE (1992)
4. Courbariaux, M., Bengio, Y., David, J.: BinaryConnect: Training deep neural networks with binary weights during propagations. In: Advances in Neural Information Processing Systems 28 (NIPS-15). pp. 3123–3131. Montreal, Quebec, Canada (2015)
5. Craven, M., Shavlik, J.W.: Using sampling and queries to extract rules from trained neural networks. In: Proceedings of the 11th International Conference on Machine Learning (ICML-94). pp. 37–45. Morgan Kaufmann, New Brunswick, NJ, USA (1994)
6. Craven, M., Shavlik, J.W.: Extracting tree-structured representations of trained networks. In: Advances in Neural Information Processing Systems 8 (NIPS-95). pp. 24–30 (1995)
7. Demšar, J., Schuurmans, D.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research 7(1), 1–30 (2006)
8. Freitas, A.A.: Comprehensible classification models: a position paper. SIGKDD Explorations 15(1), 1–10 (2013)
9. Fu, L.: Rule learning by searching on adapted nets. In: Proceedings of the 9th National Conference on Artificial Intelligence (AAAI-91), Volume 2. pp. 590–595. Anaheim, CA, USA (1991)
10. Fürnkranz, J., Gamberger, D., Lavrač, N.: Foundations of Rule Learning. Springer-Verlag (2012)

11. Goodfellow, I.J., Bengio, Y., Courville, A.C.: Deep Learning. Adaptive computation and machine learning, MIT Press (2016), http://www.deeplearningbook.org/
12. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems 28 (NIPS-15). pp. 1135–1143. Montreal, Quebec, Canada (2015)
13. Hassibi, B., Stork, D.G.: Second order derivatives for network pruning: Optimal brain surgeon. In: Advances in Neural Information Processing Systems 5 (NIPS-92). pp. 164–171. Morgan Kaufmann, Denver, Colorado, USA (1992)
14. Hayes, J.P.: Digital Logic Design. Addison Wesley (1993)
15. Kayande, U., Bruyn, A.D., Lilien, G.L., Rangaswamy, A., van Bruggen, G.H.: How incorporating feedback mechanisms in a DSS affects DSS evaluations. Information Systems Research 20(4), 527–546 (2009)
16. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Advances in Neural Information Processing Systems 2 (NIPS-90). pp. 598–605. Denver, Colorado, USA (1989)
17. Liu, J., Li, M.: Finding cancer biomarkers from mass spectrometry data by decision lists. Journal of Computational Biology 12(7), 971–979 (2005)
18. Malioutov, D.M., Varshney, K.R.: Exact rule learning via Boolean compressed sensing. In: Proceedings of the 30th International Conference on Machine Learning, (ICML-13). pp. 765–773. Atlanta, GA, USA (2013)
19. Milaré, C.R., de Carvalho, A.C.P.L.F., Monard, M.C.: Extracting knowledge from artificial neural networks: An empirical comparison of Trepan and symbolic learning algorithms. In: Proceedings of the 2nd Mexican International Conference on Artificial Intelligence (MICAI-02). pp. 272–281. Merida, Yucatan, Mexico (2002)
20. Ng, A.: Sparse autoencoder. CS294A Lecture Notes, Stanford University (2011)
21. Sato, M., Tsukimoto, H.: Rule extraction from neural networks via decision tree induction. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN-01). vol. 3, pp. 1870–1875. Institute of Electrical and Electronics Engineers (2001)
22. Setiono, R.: Extracting rules from pruned neural networks for breast cancer diagnosis. Artificial Intelligence in Medicine 8(1), 37–51 (1996)
23. Setiono, R.: Extracting rules from neural networks by pruning and hidden-unit splitting. Neural Computation 9(1), 205–225 (1997)
24. Setiono, R.: A penalty-function approach for pruning feedforward neural networks. Neural Computation 9(1), 185–204 (1997)
25. Setiono, R., Liu, H.: Symbolic representation of neural networks. IEEE Computer 29(3), 71–77 (1996)
26. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)
27. Thodberg, H.H.: Improving generalization of neural networks through pruning. International Journal of Neural Systems 1(4), 317–326 (1991)
28. Towell, G.G., Shavlik, J.W.: Extracting refined rules from knowledge-based neural networks. Machine learning 13(1), 71–101 (1993)
29. Tsukimoto, H.: Extracting rules from trained neural networks. IEEE Transactions on Neural Networks and Learning Systems 11(2), 377–389 (2000)
30. Zilke, J.R., Loza Mencía, E., Janssen, F.: DeepRED – Rule extraction from deep neural networks. In: Proceedings of the 19th International Conference on Discovery Science (DS-16). pp. 457–473. Bari, Italy (2016)