Mapping Ensembles of Trees to Sparse, Interpretable Multilayer Perceptron Networks

Dalia Rodríguez-Salas¹ · Nina Mürschberger¹ · Nishant Ravikumar² · Mathias Seuret¹ · Andreas Maier¹

Received: 30 April 2020 / Accepted: 21 July 2020 / Published online: 6 August 2020
© The Author(s) 2020

Abstract

Tree-based classifiers provide easy-to-understand outputs. Artificial neural networks (ANN) commonly outperform tree-based classifiers; nevertheless, understanding their outputs requires specialized knowledge in most cases. The highly redundant architecture of ANN is typically designed through an expensive trial-and-error scheme. We aim at (1) investigating whether using ensembles of decision trees to design the architecture of low-redundant, sparse ANN provides better-performing networks, and (2) evaluating whether such trees can be used to provide human-understandable explanations for their outputs. Information about the hierarchy of the features, and how good they are at separating subsets of samples among the classes, is gathered from each branch in an ensemble of trees. This information is used to design the architecture of a sparse multilayer perceptron network. Networks built using our method are called *ForestNet*. Tree branches corresponding to highly activated neurons are used to provide explanations of the networks' outputs. *ForestNets* are able to handle low- and high-dimensional data, as we show on an evaluation using four datasets. Our networks consistently outperformed their respective ensemble of trees and had similar performance to their fully connected counterparts with a significant reduction of connections. Furthermore, our interpretation method seems to provide support for the *ForestNet* outputs. While *ForestNet's* architectures do not allow them yet to capture well the intrinsic variability of visual data, they exhibit very promising results by reducing more than 98% of connections for such visual tasks. Structure similarities between *ForestNets* and their respective tree ensemble provide means to interpret their outputs.

Keywords Multilayer perceptron · Random forest · Interpretable models · Sparse neural networks · Network architecture · Interpretability

This article is part of the topical collection “Machine Learning in Pattern Analysis” guest edited by Reinhard Klette, Brendan McCane, Gabriella Sanniti di Baja, Palaiahnakote Shivakumara, and Liang Wang.

✉ Dalia Rodríguez-Salas
dalia.rodriguez@fau.de

Nina Mürschberger
nina.muerschberger@fau.de

Nishant Ravikumar
n.ravikumar@leeds.ac.uk

Mathias Seuret
mathias.seuret@fau.de

Andreas Maier
andreas.maier@fau.de

¹ Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg, Martensstr. 3, 91058 Erlangen, Germany

² School of Computing, University of Leeds, Leeds LS2 9JT, UK

Introduction

Multilayer perceptron (MLP) networks have been successfully used for solving many complicated learning tasks, whether they involve visual or non-visual data. MLP have been first used mainly as classifiers, and the traditional methodologies involving them are mainly based on the extraction of hand-crafted features. These features, which are designed by experts in the problem and/or in machine learning, are combined to produce feature vectors which are used to train the MLP, or to classify data samples.

A more recent approach is to feed the networks directly with the raw data, e.g., pixel maps, and have them learn on their own representative features in their different hidden layers [11].

While feeding the MLP with raw data has the advantage of removing the need of engineering features, this approach has two main drawbacks, which were highlighted

in [12]. They are enumerated for their future reference. (1) The dimensionality of the input data impacts the number of parameters of the MLP, and therefore its complexity, as well. Visual data consisting of pixel maps are typically high-dimensional, and then, the first hidden layer of the MLP, for which every neuron is connected to all channels of all pixels, would have a very large number of trainable parameters. This implies that to avoid over-fitting, a very large amount of training samples is required for training. These both lead to high memory requirements and to a large amount of labeling work to produce the data. (2) The MLP does not take into account the topology of visual data. Therefore, MLP does not learn spatial relationships, and cannot learn shift-invariant representations. They also do not deal well with local distortions of the input data.

We can also mention two more general drawbacks of MLP networks which are not specific to visual data. (3) For a given task, it is not simple to decide how many hidden layers which they should contain, nor how many units should be in each of these layers. (4) Trying to interpret why the network gives a specific output is not understandable for non-specialists in most cases, although with standard deep learning approaches, some methods can be used to fight these drawbacks. Only drawback (2) is tackled from its root, due to the incorporation of convolutional filters and pooling operations among the hidden layers. For the other three drawbacks, in most cases, the usual techniques are independent to the classification models themselves, working as additional procedures. To give an example of such methods, one can think of augmentation techniques which are often used to reduce the over-fitting problem caused by drawback (1); they are totally independent to the deep learning model, and if the correct transformations are applied to the training samples, then the problem of having an insufficient amount of data is partially solved.

Our aim is to start paving the way for the development of deep learning models which address all these four drawbacks. Our first stone towards this aim consists of a method to automatically design MLP architectures which tackle drawbacks (1), (3), and (4) from their root. Our sparse MLPs are not meant to compete with deep neural networks trained on augmented data.

Our method is based on the usage of randomized trees to define the architecture of MLP networks. We start by building an ensemble of randomized trees (ERT), and then base the MLP architecture on the structure of the trees, particularly on their branches. Each branch of a tree is associated with a set of features. The features of a single branch, combined together, have the ability to classify accurately a subset of samples. Thus, a neuron taking these features as input can, as well, have such discriminating abilities. Therefore, for each of the tree's branches, we create a hidden neuron which is fed only using the features associated

with its corresponding branch. In a similar way, the neuron is connected only to the output neurons corresponding to the classes separated by the branch. The resulting MLP has a structure matching the one of the ERT, it is highly sparse and the amount of neurons and connections is totally defined by the trees, tackling on this way drawbacks (1) and (3). Drawback (4) is addressed by taking advantage of the similarity between the forest and the structure of the MLP, providing a way to interpret the outputs of the MLP using information from the tree branches the network was built with.

The main contributions of our work are:

- A method which enables automatic design of network architecture and complexity, i.e., an appropriate number of neurons in the hidden layer of an MLP, and of connections between layers of MLP networks.
- Foundation for developing easy-to-interpret models or architectures of artificial neural networks (ANN), which either use or are based on MLP networks, such as convolutional neural network (CNNs).
- A methodology for interpreting outputs of MLP built out of ERT.

In the following section, a review of related work is presented. Our method to build Sparse MLP is detailed in Sect. 3, followed by the Sects. 4 and 5 which contain the experiments and results, respectively. Section 6 focuses on the interpretation of our networks, in particular on how to use the ERT for interpretation purpose, and why it is possible to do so. Finally, Sect. 7 concludes this article and presents the next steps which we intend to investigate.

Related Work

Our work is related to two main research topics: automatic methods for creating ANN architectures and ANN output interpretation techniques. This section presents a selection of relevant publications.

Architecture Design

Relating decision trees to ANN has been done already in several works. More than 30 years ago, *Perceptron Trees* [28], as called by the authors, were used for classification tasks, having the characteristic of comprising Rosenblatt's perceptrons [23] on their leaves. These perceptron trees, however, do not have an MLP structure.

The first algorithm to create sparse architectures of MLP with two hidden layers using decision trees was published in [25]. These MLP were called *entropy nets*. In [21], another method for mapping decision trees to MLP consisting of only one hidden layer is presented. While having

significantly fewer connections and hidden units than the entropy nets, these networks with a single hidden layer achieved comparable classification performance.

The design of MLP networks using decision trees is still being investigated, as publications on this topic keep appearing. For example, in [2], the authors use ERT as base to build two different kinds of MLP. They create one MLP with two hidden layer for each of the trees of the ensemble of trees, and then concatenate these networks into a wide network with two or three hidden layers. Finally, a method to create ensembles of MLP networks out of ERT, with as many hidden layers as there are levels in the trees, has been presented in [8]. This method was evaluated only evaluated with low-dimensional, non-visual data, and a maximum tree depth of 5.

While most work relating decision trees to ANNs is about constructing the latter, the opposite is also possible. For example, in [6] and [32], decision trees are built from information gathered from CNNs, and in both works, the resulting trees provide a interpretation mean of the decisions taken by the CNNs which they were built with. On one hand, the former work uses soft trees, each decision node contemplates a filter, and a bias learned by the network. On the other hand, the latter work uses the feature maps obtained by a disentangled CNN to train the trees.

We can note that using trees is not the only approach for designing ANN architectures automatically. For example, in [26], an evolutionary algorithm for designing the architecture of recurrent neural networks, called NeuroEvolution of Augmenting Topologies (NEAT), was developed. It has then been modified [15], so that it can be applied to deep neural networks, as well. In [10], weights of CNN are learned through an evolutionary process. To decrease the size of the chromosome, this is done in a compressed Fourier space. This is possible as neighbor weights in a kernel have typically similar values. Such approaches based on evolution are, however, computationally expensive, as they require to train *populations* of ANN for several generations.

Although modern hardware can handle very large ANN, research on sparse networks is still ongoing, and has gained much attention recently, after a method allowing to prune ANN, while either preserving or even *increasing* their accuracy [5] has been presented. This result was obtained by iterating over training and pruning phases, with resetting the remaining weights to their initial (i.e., random) value before each training phase. This is, however, time-consuming, as the network has to be trained from scratch many times.

ANN Interpretation

To follow up on the subject of interpreting ANNs, which also concerns us, some works have used tree-based approaches to this aim. In [33], the filters in the top-conv layer of a

pre-trained CNN are associated with a specific semantic part or region of the objects which the network was train to classify. Per class, a parse tree is used to explain which object regions/filters were used as well as their contribution to the prediction. In [31], several families of ANN were trained to mimic shallow decision trees through tree regularization [30]; thereby, those trees can be used to interpret the outputs of such tree regularized networks. Nevertheless, expertise knowledge might be required to partition the space to train trees with a minimum fidelity and provide faithful explanations.

Techniques using decision trees are not the only ones; much more widely used in the computer vision community, one can find algorithms such as grad-CAM [24], layer-wise relevance propagation [1], and other visualization techniques per layer such as the work proposed in [14]. Either using decision trees or gradient-based techniques, all works mentioned up to now focus their efforts on interpreting deep ANNs. Nevertheless, the interpretation of shallow ANNs is also open to research, and in the case of MLP networks, the generation of interpretable architectures would also allow means of interpretation in deeper models which are based or make use of MLP networks, such as CNNs.

To conclude this section, we would like to point out the difference between interpreting the results through the learned model itself and interpreting the results through external tools. Examples of the former include, among others, models like linear logistic regression, linear regression, and decision trees; for the latter, grad-CAM is a example, since it provides means of interpretation which is separated from the model. A broad study about interpretability in machine learning is out of the scope of this work; nevertheless, we would like to address interested readers to the works in [17] and [16].

Method to Build a Sparse Multilayer Perceptron Network

In this section, we present our approach for building an MLP from a ERT. Let X be the labeled training data for a given task and T be a tree built using X . A class label is associated with each terminal node, or leaf. Let us consider a node of T which has two leaves, l_{left} and l_{right} , as children, and a sample which reaches this node during its classification process. Then, to decide whether the sample belongs to l_{left} or l_{right} , i.e., to decide to which class the sample belongs, then only the features which are on the path between the root of T and the father node of l_{left} and l_{right} are necessary. This implies that a subset of the samples can be classified using only a—small—subset of the features. Thus, each pair of leaves define their own subspace; therefore, the number of subspace is linearly proportional to the amount of fathers of

leaves. Hence, the maximum number of subspaces defined by the leaves of a tree grows exponentially with the depth of the tree.

Our approach for designing MLP architectures is based on the idea that the subspaces defined by the leaves of a tree ensemble can lead to efficient sparse MLP. We make the hypothesis that such an MLP has the ability to learn better class boundaries than its corresponding ERT, as the latter is limited to have axis-aligned boundaries [8].

The proposed methodology is composed of three main steps. First, we build an ensemble of randomized trees. Then, we use them to define the architecture of the corresponding sparse multilayer perceptron (SMLP) network. Finally, we initialize the weights of the SMLP.

Building the Ensemble of Randomized Trees

Let m be the number of features of the training dataset X , and y be the targets for a classification problem of X . We can build an ensemble of randomized trees which can identify decision boundaries suited for the classification task. Many algorithms for building such tree ensembles exist, and although they can lead to a wide range of different tree structures, a broad analysis of such algorithms is out of the scope of this study. Nevertheless, most of these algorithms consider a subset of the features for each tree of the ensemble. Indeed, using a different subset of the features for every tree of the forest has two main impacts. First, as all trees base their decision on different data, then it is possible to majority voting. Indeed, if all trees had exactly the same input, their decisions would be extremely similar, and thus voting, or even having several trees, would make little sense. Second, it decreases over-fitting, as every tree has a limited access to information. This is extremely useful in case of high-dimensional data.

Random forests [3], and extremely randomized trees [7] are the two main algorithms for producing ERT which use only a subset of the features for each tree. Typically, if X has m features, then $\sqrt{m}$ features are in the subsets. Note that it is not enforced that a tree uses all of its inputs.

When used for classification, an ensemble of trees can potentially under-fit or over-fit the data, depending on how many trees it is composed of, as well as on the depth of the trees. Under-fitting means that the model is not complex enough to encompass all subtleties of the data. Over-fitting happens when the model is so complex with regard to the amount of available training data, that it loses its generalization capabilities. Thus, the size of the forest should ideally be just sufficient to cover the whole data space. This also means that the MLP created from the forest will be less likely to over-fit the training data thanks to the sparsity of its inputs.

Defining the SMLP Network Architecture

Our aim is to provide to our MLP architecture the ability of capturing all the information of the subspaces found by an ERT. Thus, we create it, such that every node parent of a leaf produces one hidden neuron. This neuron uses as input only features which are used on the path of the tree which goes from the root to the corresponding node. This implies that a neuron can have from one input, to as many as there are levels in the decision tree. Also, as hidden neurons are connected to features which are known, thanks to the decision tree, to be good for separating a subset of the samples of two classes, it makes sense to connect the hidden neurons only to the output neurons corresponding to these classes. In the case of a node which has a single leaf (and another node), then we proceed in the same way, but connect the corresponding hidden neuron to only one output neuron. This methodology produces an MLP with sparse connections in both its hidden and output layers.

Henceforth, any SMLP networks which architecture is built out of an ERT following the proposed method will be referred to as *ForestNet*.

The pseudo-code to build a ForestNet for a dataset (X, y) , where X is a set of samples, and y is the corresponding labels, given an already built tree ensemble for the same dataset is shown in Algorithm 1.

Fig. 1 Mapping of a tree ensemble to a *ForestNet* for a problem with nine features and three classes. The ten fathers of leaves (in color) have a corresponding hidden neuron (of the same color) in the network. The branch of a father of leaf, its corresponding hidden neuron, and connections are highlighted in bold

Algorithm 1: Pseudo-code to build a *ForestNet* given a tree ensemble. [22]

Input: Tree ensemble ET built using the training set (X, y) , Training matrix X , and Labels y

Output: *ForestNet* FN

$leaves \leftarrow getLeaves(ET);$

$nodeList \leftarrow unique(getFathersOfAll(leaves));$

$numHiddenUnits \leftarrow size(nodeList);$

$numOutputUnits \leftarrow countUniqueValues(y);$

$numSamples \leftarrow countSamples(X);$

$FN \leftarrow emptyForestNet(numHiddenUnits, numOutputUnits);$

$i \leftarrow 0;$

for $node \in nodeList$ **do**

$neuronInputs \leftarrow featuresOnThePath(ET, node);$

$neuronOutputs \leftarrow getLeavesAndLabels(node);$

$connectHiddenNeuron(FN, i, neuronInputs, neuronOutputs);$

$i \leftarrow i + 1;$

end

This methodology is illustrated in Fig. 1, where a (tiny) ensemble of trees is mapped to a *ForestNet*. Let us assume that we have a three-class problem (“red”, “blue”, and “green”), and nine features (A, B, ..., I) as input. In the three trees of Fig. 1, there are ten nodes which have leaves as children. Thus, we have to create an MLP with ten hidden neurons. Let us have a look at the branch of the yellow tree marked in bold. It has two features, I and F, and leads to one leaf, for the green class. The corresponding hidden neuron and its connection are also bold in the network.

It is possible to get a rough estimation of the maximum number of neurons use a specific feature, depending on where this feature is located in the tree. Let us assume that

the tree is fully grown, and that it has a depth D . We know that the feature corresponding to the root node is used by all neurons, because all paths to a father of leaf do go through the root node first. A fully grown tree of depth D has 2^D leaves, and therefore, 2^{D-1} fathers of leaves—all of which produce neurons using the feature of the root node. Let us now consider a feature in the second level of the tree, i.e., used by a node which is a direct child of the root. This node is the root of a subtree of depth $D - 1$, and therefore, its feature is used 2^{D-2} times. Applying this counting method recursively shows that a feature of the d th level of a tree of depth D would be used by at most 2^{D-d-1} times.

The *ForestNet* corresponding to an ERT has an architecture which reflects the hierarchy of the features in the trees. The deeper a feature is used in the tree, the fewer neurons use it as input. This leads to the observation that it is important to use algorithms that create trees, such that the most discriminative features are close to the root.

Initialization of the Weights

The nodes of a tree split the data which they receive into two subsets. Typically, when trees are built, nodes are being added until either the maximum depth is reached, or a vast majority of the samples reaching a leaf belong to the same class. For a node, the class impurity can be defined as a measurement of which fraction of the samples do not belong to the majority's class. When a node splits a set of samples into two subsets, then the impurity of these subsets is smaller than the one of the input set. A frequently used measurement for impurity is the Gini. In [27], the importance of a feature in a tree is computed by adding the Gini of each decision node containing that feature, divided by the total of nodes. In our model, we use this Gini-based importance to set the initial value of the weights of the hidden layer of the MLP.

As the output layer does not make a direct use of the input features, their Gini-based importance cannot be used there. The connections between the hidden and output layers can be thought of as connections between subspaces corresponding to the hidden neurons, and the classes outputted by the MLP. Let us assume that a subspace S corresponding to a hidden neuron (and therefore to a father of leaves) is associated with N_S training samples (i.e., this amount of samples go through the node during classification with the tree). If the node separates classes y_{left} and y_{right} , then we initialize the respective weights between the hidden and output layers to be $N_{S-\text{left}} \cdot N_S^{-1}$ and $N_{S-\text{right}} \cdot N_S^{-1}$, where $N_{S-\text{left}}$ and $N_{S-\text{right}}$ are the amount of training samples in S which belong to y_{left} and y_{right} , respectively.

Experimental Settings

We evaluate the performance of our method on four different datasets, using tenfold cross-validation. For each dataset, we built an ERT using the Extremely Randomized Trees algorithm [7], and the corresponding *ForestNet*. This allows us to compare the ensembles of trees and the *ForestNet*. Then, to compare our sparse networks with fully connected one, we create, for each *ForestNet*, a non-sparse (i.e., fully connected) version of it. Henceforth, we will refer to the latter as *Fully* net.

When comparing the three different types of classifiers, we use the Wilcoxon rank sum test [29] to determine whether there is a statistically significant difference between each pair of them.

The datasets used in the evaluation of our approach are the following. The Pima Indians consist of medical data of 768 Indian females, and the goal is to determine which ones of them suffer from diabetes. This dataset is composed of eight features, such as the age and the blood pressure. The Wisconsin dataset consists of 30 hand-crafted features which are supposed to discriminate malignant from benign tumors. Such features were extracted from digitized images of a fine needle aspirate of a breast mass. The widely used MNIST dataset is a well-known benchmark for image classification. It consists of binary images of handwritten digits which are size-normalized to 28×28 pixels and centered in the image. The SVHN, or Street View House Numbers, dataset consists of 32×32 pixels images of digits cropped from house number photos. There is a large variety of styles for the digits, and, because of the position of the camera, there are also some perspective distortions, as well. Numeric information about the datasets are given in Table 1.

Hyperparameter Settings for Tree Ensembles

The two main hyperparameters are the maximum depth of the trees, and the quantity of trees in the ensemble. In our previous work on the topic [22], we found that for the considered datasets, a depth of 10 is sufficient, as deeper trees have little or no impact on the performances. It can be noted that a depth of 10 would already produce up to 2^9 subspaces

Table 1 Brief description of the datasets used on the experiments. [22]

Dataset ID	No. of samples	No. of features	No. of classes	Visual?	Pre-processing	Source
Pima Indians	768	8	2	No	Nomalization	[4]
Wisconsin	569	30	2	No	Nomalization	[4]
MNIST	60000	784	10	Yes	Nomalization	[13]
SVHN	73257	3072 ^a	10	Yes	Nomalization, Gray-scale	[18]

^aOnly 1024 were used as a result of the Gray-scale pre-processing step

Fig. 2 Average accuracy and standard deviation of using tenfold cross-validation on four datasets with ERT, its respective ForestNet, and its Fully version. For all datasets, the maximum depth of the trees was set to 10 and the number of trees to 6

Table 2 Test of the null hypothesis that there is no significant difference between the accuracies gotten by our ForestNet and those gotten by ERT and fully net, respectively

Compared pairs	<i>p</i> value	Null hypothesis ($\alpha = 0.01$)
ForesNet—ERT	0.00000037	False
ForesNet—fully net	0.03569436	True

The two-sided *p* values for the Wilcoxon signed-rank test was applied with a significance level α of 0.01

per tree (if the tree is fully grown). We built ensembles of six trees using the ERT algorithm [7] as implemented in [20]. Each tree was built using a random subset of $\sqrt{m}$ features, where m is the total amount of features in the considered dataset.

Hyperparameter Settings for Networks

The connections of the *ForestNet* and of the *Fully* net are directly derived from the tree ensemble. However, some more had to be taken in consideration when building and training the networks. We used the rectified linear unit (ReLU) activation in the hidden layer, and log-softmax in the output layer. During the training phase, we used the Adam optimizer [9], with a learning rate of 0.01, and cross-entropy as loss function. Both kinds of networks, i.e., the *ForestNet* and the *Fully* net, are trained with early stopping, using a patience of 35 epochs.

All networks are implemented and trained using PyTorch [19]. The sparsity is enforced by multiplying the weights of fully connected layers by binary masks. Their gradients are also set to 0 in the backward pass, to prevent them from impacting the previous layers.

Table 3 Average amount of connections when using tenfold cross-validation on four dataset with ForestNet and its fully version

Dataset	Average amount of connections		
	ForestNet	Fully net	Reduction (%)
Pima Indians	3209 ± 206	5235 ± 347	38.70
Wisconsin	1492 ± 134	6373 ± 545	76.60
MNIST	30,908 ± 269	2,146,986 ± 17170	98.56
SVHN	31,350 ± 628	2,814,972 ± 50633	98.89

Table 4 Average amount of hidden neurons when using tenfold cross-validation on four datasets

Dataset	Amount of hidden neurons
Pima Indians	476 ± 32
Wisconsin	199 ± 17
MNIST	2704 ± 22
SVHN	2722 ± 49

Classification Results

This section is composed of three parts. First, we present the classification accuracies of all three classifiers on all datasets. Second, we compare the amount of parameters in the *ForestNet* and *Fully net*. Finally, we compare the behavior of the two networks during the training process.

Classification Accuracy

We evaluate our classifiers on the four following datasets: Pima Indians, Wisconsin, MNIST, and SVHN. The results are summarized in Fig. 2. Each pair of columns corresponds to the accuracy of a classifier on, respectively, test, and training data. To have experimental protocols as close as possible for all datasets, and to compute the variance of the results not only on multiple runs, but on more sample diversity, we applied a tenfold cross-validation on the training subsets of the datasets.

We can see that the ERT tends to exhibit a significantly stronger over-fitting than the MLP. Indeed, the difference between the training and test accuracy is larger for the ERT than for the two MLP. The larger variances of the results obtained on Pima Indians and Wisconsin than on MNIST and SVHN can be explained by the larger amount of samples in the two latter datasets. The differences between the two MLP are small enough to require a significance test.

As we have four datasets of different sizes, and obtain results of different variances, we use the Wilcoxon [29] test to compare *ForestNet* architecture with the other classifiers. The results of the test are given in Table 2. There is no

doubt that *ForestNets* outperform ERT. For the comparison between *ForestNets* and their fully connected counterpart, for an $\alpha = 1\%$, the null hypothesis is accepted.

Thus, we can conclude that for these four datasets, the *ForestNets* significantly outperform the ERT from which they are built. Furthermore, they performed similarly as their fully-connected versions. We can see in Fig. 2 that this is more pronounced in the two smallest datasets, which are more prone to over-fitting.

Amount of Connections

One of the main aims of our method is to create very sparse MLP. While our implementation is based on non-sparse matrices with weights set to 0 where there is no connection (see Sect. 4), a sparse implementation has the potential not only to decrease risks of over-fitting, but also to be more energy-efficient, which can be very important for mobile applications or the industry.

We averaged the number of connections in *ForestNet*, on the four datasets individually, during a tenfold cross-validation. The results are given in Table 3. We can see that although the parameters for the ensemble of randomized trees were the same, the amount of connections vary greatly for the different datasets. We can also notice that the reduction for the two visual datasets is drastic: *ForestNets* have more than 98% less parameters than their corresponding *Fully nets*.

Thus, our method produces sparse MLP, with a sufficient amount of parameters, as they perform similarly as their fully connected versions. Additionally, the percentage of reduction and amount of connections are adapted to the datasets.

Amount of Hidden Neurons

As it has been proven in Sect. 3, the maximum number of neurons that can be create grows exponentially with the depth of the trees. Indeed, a fully grown tree of depth D produces 2^D neurons. Thus, when using an ensemble of t trees, we could potentially end-up with a hidden layer containing $t \cdot 2^D$ neurons.

We measured the size of the hidden layer of the *ForestNet* for the four datasets, and average these measurements with a tenfold cross-validation. The results are given in Table 4. We can see that while we could have had potentially up to 6000 neurons with the hyperparameters of the ERT algorithm, our methodology produces significantly less. For the two high-dimensional datasets, MNIST and SVHN, we obtain roughly 45% of the theoretical maximum neuron count, while for the two other datasets, Pima Indians and Wisconsin, we are below 8% of this limit.

Fig. 3 Training and validation loss on the MNIST and SVHN datasets. We used a middle-sized ForestNet obtained with six trees, and its fully connected version. The exhibited behavior is typical of what

we observed with other networks, regardless of the amount of trees used to create them

The dimensionality of the input data is not the only factor impacting the size of the hidden layer. Indeed, the number of features of Pima Indians is less than half than of Wisconsin. Thus, we can assume that the class separability is more complicated; more neurons will be produced. Indeed, if the trees need to use most of their available features to get a good Gini, then they will tend to have longer branches and more leaves.

This means that the maximum depth of the trees is not a critical parameter when aiming at producing small networks—if the data allow for it, then small MLP will be produced.

Training and Validation Losses

During our experiments, we have noticed that the *Fully* nets and *ForestNets* behave differently during the training phase. In Fig. 3, we show two typical sets of loss curves.

The dashed lines correspond to *Fully* nets, while the continuous ones are for *ForestNets*. The vertical lines indicate where the lowest validation loss is reached. We can notice that the validation losses all start increasing after a while, and thus, there is clearly some over-fitting. The validation loss of the *ForestNets* is, however, not increasing as fast as for the *FullyNets*. While this could be expected, this shows that the lower amount of trainable parameters shields partially against over-fitting.

Furthermore, the smoother and flatter validation loss of the *ForestNet* makes less critical the choice of the right time to stop the training. Indeed, training for a few epochs more or less as the optimum has little impact on the validation loss.

Discussion of Classification Results

The statistical significance tests showed that while our method produces very sparse MLP, they have similar performances as their fully connected versions. The reduction of the number of parameters depends on the dataset, and not on the hyperparameters of the ERT algorithm. While the number of hidden neurons remains proportional to the number of trees in the ensemble, it does not depend much on the maximum tree depth allowed. The main factor seems to be the data itself. Thus, while deeper trees are able to produce large hidden layers, the size of the latter is adapted to the datasets' requirements, and significantly smaller than their theoretical maximum. Furthermore, their lower amount of parameters allow the *ForestNets* to exhibit less over-fitting than the *Fully* nets.

While SVHN and MNIST have similar tasks (digit identification), we can see a clear difference between the accuracy of our method on these two datasets. This is due to one of the main differences between these datasets: SVHN is composed of digit cropped from photos, and MNIST is composed of pre-processed digits acquired on a tablet pen. The variance in location and orientation of the digits in MNIST is lower than for SVHN. Additionally, the low image quality of the latter dataset makes it more difficult, even for humans.

Interpretation Technique for a ForestNet

An accurate model is not always the only required feature of a classifier. In some applications, being able to explain decisions is highly desirable. Such can be the case of many medical applications. For instance, knowing the reason of

Fig. 4 Interpretation technique for a *ForestNet*. A trained *ForestNet* is used to classify a sample. Per tree, the branch corresponding to the neuron with the most impact on the network result is used to build a human-understandable report

why a model decided that a given patient has a malignant tumor, and not a benign one, could provide radiologists with more insights to either confirm or change their diagnosis. But above all, such explanations have to be understandable by humans, experts in the problem in question. This latter is an important feature which decision trees have and ANNs lack. Indeed, decisions based on feature thresholds do not require machine learning knowledge, while understanding the inner mechanisms of an ANN does. This opens the following question: can *ForestNet*'s decisions be explained using the simple thresholding approach of the trees it was built with?

Given that the architecture of a *ForestNet* matches the structure of its respective ERT, a branch in a tree is associated with each hidden neuron. Our proposal consists of looking, for a specific input sample, at the activation values of the hidden neurons, and focuses only in the ones which have the highest impact on the network's output which correspond to the predicted class. As these neurons are mainly responsible for the decision of the MLP, then we assert that their corresponding branches can be used as surrogate for interpretation purpose.

Let us assume that the network output class k , N_i is the set of indices of hidden neurons corresponding to the i th tree of the ensemble, h_j is the output of the j th hidden neuron, and w_{kj} is the weight of h_j for the k th output of the network.

Then, the index of the neuron selected for the i th tree, n_i , is given by Eq. 1:

$$n_i = \operatorname{argmax}_{j \in N_i} (|h_j \cdot w_{kj}|). \quad (1)$$

Thus, we get for each tree the associated neuron which has the highest impact on the output of the MLP. Finally, we get the tree branches from which these neurons were built, as explained in Sect. 3. Up to this point, we have a sample, a classification result, and one branch per tree. Our aim is to produce, out of these data, a short, human-understandable report hinting for which reasons that the network had this specific output. The intended reader of this report is a specialist in the domain of the task, e.g., a doctor in medicine in the case of medical data. The processes to obtain such reports are illustrated in Fig. 4, where one can see a trained *ForestNet* which was build using trees T1 and T2. The network is used to classify a new sample. For the associated neurons to each of the trees, the white neurons have the most impact on the classification result for a given sample; only their respective branches will be used to build the report. In our example, the branch corresponding to the most impacting neuron for the tree T2 contains only feature B with 3487 as threshold. To follow the path of that branch, one has to go to the right, meaning that a sample would take that path

if its value for feature B is greater or equal than 3487. This is shown on the final report, the value of B for our classified sample is 3561 which matches the condition of that branch node, and it is indicated with a check-mark; if the condition is not matched, it is indicated with a cross-mark.

For each selected branch, we have a sequence of internal or decision nodes which lead to a leaf node. A decision node has a feature and a threshold associated with it; the decision to make takes the sample to either the left, if the sample's feature value is smaller than the threshold, or to the right otherwise. The hypothesis of a branch is that, if a sample meets all the conditions that make it go through each of its nodes, it means that the object belongs to the class associated with the branch's leaf node.

The ERT and the *ForestNet* do not produce the exact same results, as the latter is more accurate, as shown in Sect. 3. Thus, the features of a sample might not match all of the conditions of the branch corresponding to a neuron highly activated by this sample. Indeed, the neural network is able to learn more-refined class boundaries than the ERT.

This leads to the following question: does the *ForestNet* still have a behavior *close enough* to the one of the ERT so that the branches can be used for interpreting the network's results? This is investigated in Sects. 6.1 and 6.2. Also, assuming that this question can be answered positively, then as the *ForestNet* has more-refined class boundaries, some tolerance has to be given to the interpretation. Indeed, if a feature is on the *wrong* side of the threshold, but very close to it, we could assume that this is due to the less accurate class boundary of the ERT. This can be addressed by relaxing the comparison to the threshold with a small tolerance margin. Thus, the comparison is assessed as true if either the feature is on the correct side of the threshold, or if the error is smaller than the tolerance margin. This is as well investigated in our experiments.

Experimental Settings for the Interpretation Technique

In this section, we describe the various experiments used to investigate the explainability *ForestNets*. In the first two experiments, for simplicity reasons, we focused on the Pima Indians dataset, as it contains features which can be understood by non-specialists.

First, we produced a human-understandable report for a randomly selected sample using our method. Due to space limitations, a smaller *ForestNet* than those in our previous experiments was created. Since smaller depths would lead to smaller branches, we reduced the maximum depth of the trees to 6 and only 3 trees were built to have fewer branches, features, thresholds, and conditions to assess. For training the network, all hyperparameters remained the same.

Second, we investigated the similarity of class boundaries between ERT and *ForestNets*. From an arbitrary fold, the information of a randomly selected patient from the validation set was used to display the decision boundaries of both models. The two most relevant features (the body mass index and the glucose concentration in blood) were selected using the same feature importance described in Sect. 3.3. All other features were fixed to run the classification for many value combinations of the two selected features.

Finally, using the models obtained from our previous experiments described in Sect. 4, per each of the datasets, we measured the similarity between ERT and *ForestNet* using information gathered from the interpretation reports of all samples in the dataset. For the generation of such interpretation reports, the same tenfold cross-validation training-validation sampling than in our previous experiments was used. For each tree, we considered the branch corresponding to the hidden neuron which has the highest impact on the classification result of the *ForestNet*. When classifying a sample, there are two possible paths to follow at each decision node. If the features value is smaller than the threshold of the decision node, then it follows the left tree branch; otherwise, it follows the right branch. It might then happen that the sample exits the considered branch. In such a case, the *ForestNet* and the ERT differ in their decision-making. We counted each such occurrence as a dissimilarity; comparably, a counting of the similarities was performed. Thus, the addition of the similarity- and dissimilarity-counting equals the amount of condition nodes in the considered branches. If the similarity-counting is majority, one can assume that there exists a similarity between an ERT and its corresponding *ForestNet*. On the contrary, if the dissimilarity-counting is majority, it can be said that the *ForestNet* is learning completely different boundaries to the ERT, and thus, the latter could not be used as interpretation base for the results of the *ForestNet*. We can note that a similarity of 100% is not desirable, as this would imply that the *ForestNet* has exactly the same output as the ERT.

Results on the Interpretation Technique

An example of an explanation table produced by our method is given in Table 5. First, to contextualize the data, some statistics about the dataset are presented in Table 5a. To show what can be expected of data from healthy patients, the mean value and standard deviation have been computed for all patients who have not been diagnosed with diabetes. The features are as follows: the number of pregnancies; the concentration of glucose measured during a test; the diastolic blood pressure, in mmHg; the thickness of the skin over the triceps when it is pinched; the concentration of insulin measured during a test; the body mass index, which is the weight of the patient in kg divided by her height in meter;

Table 5 Example of a Forestnet results' report for a given sample; 3 trees with a maximum depth of 6 were used in the ensemble

Feature	Mean	Std	One patient	
(a) The means and standard deviations of non-diabetes patients from the Pima Indian dataset are given in the 2nd and 3rd columns as reference values ^a				
Pregnancies	3.86	2.91	5	
Glucose	110.64	24.75	130	
Blood pressure	70.88	12.15	82	
Thickness	27.24	10.01	–	
Insulin	130.29	102.29	–	
BMI	30.86	6.55	39.1	
Predisposition	0.43	0.30	0.96	
Age	31.19	11.66	37	
Forestnet's result:			Positive	
Name	Value	Operator	Threshold	Assessment
(b) Trees' features and thresholds associated with the most activated neurons per tree are shown in the first and fourth columns ^b				
Most activated neuron for Tree 1				
Pregnancies	5	≥	6.31	X
Predisposition	0.96	≥	0.63	✓
Predisposition	0.96	<	1.51	✓
Glucose	130	≥	110.83	✓
Most activated neuron for Tree 2				
Glucose	130	≥	81.69	✓
Glucose	130	≥	108.77	✓
Age	37	≥	58.92	X
Most activated neuron for Tree 3				
Insulin	0	<	360.15	✓
Glucose	130	≥	143.82	X
Age	37	<	58.8	✓
Insulin	0	≥	33.51	X
BMI	39.1	<	47.71	✓
BMI	39.1	<	38.65	X

^aThe fourth column shows the features' value of a specific patient; the symbol "–" indicates missing values which are set to 0 during training and testing

^bPatient's values for those features in the second columns are compared with the branches' condition operators in the third column; the assessment' results are shown in the last column

the diabetes pedigree function, which hints at how predisposed to diabetes a patient is; and the age of the patient.

The third column of Table 5a corresponds to a specific patient. We can see that two measurements, the skin thickness and the insulin concentration, are not available. This is represented by zero values in the dataset. Our *ForestNet* classified this patient as having diabetes. We can then look at Table 5b to get a better understanding of this result.

The selected branch from the first tree diagnoses diabetes if the following three conditions are met. First, the woman has had more than six pregnancies. Second, her diabetes predisposition value is between 0.63 and 1.51. And, finally, her glucose concentration is above 110. Our patient does not meet the first condition, as she has had only five pregnancies, but she does meet the other two.

The second branch has two conditions, with some redundancy, on the glucose concentration, and one on the age. Based on the data of healthy patients, this branch diagnoses diabetes for older patients with a glucose concentration above the average—it seems to us that these criteria might not be very discriminative, and correspond to some overfitting of the tree. Thus, we believe, as persons not trained in medical diagnose, that this branch can be ignored in the interpretation. Also, while our patient has indeed an above-average glucose concentration, her age is significantly closer to the average than to the branch's threshold.

The third branch is longer, with six condition nodes. It is important to have a deeper look at the first condition of this branch, on the insulin concentration. While it is marked as met, its value actually indicates a lack of data for our patient.

Fig. 5 Decision boundaries for glucose and body mass index (bmi), computed for using the other features from a random patient. The green area corresponds to values of these two features for which the classifiers estimate the patient to be healthy. The red areas correspond to diabetes diagnoses

Table 6 Average percentage of similarity between four ForestNets and their respective ERT with different levels of tolerance assigned to the trees' thresholds

Dataset	Average similarity			
	0.00	0.05	0.10	0.15
Pima Indians	63.77 ± 2.02	71.71 ± 2.20	78.23 ± 1.85	83.18 ± 1.77
Wisconsin	69.83 ± 2.23	80.28 ± 2.07	87.23 ± 1.83	91.88 ± 1.51
MNIST	67.31 ± 1.01	70.15 ± 1.21	72.72 ± 1.15	75.29 ± 1.12
SVHN	52.17 ± 0.80	59.84 ± 0.86	67.04 ± 0.91	73.57 ± 0.91

As the tree was created using some patient data missing the insulin concentration, as well, it could be argued that the condition is fully met. However, as there is uncertainty on the real value, it is safer to ignore this criterion. The same reasoning can, of course, be applied to the second usage of this feature, as well. The glucose concentration, while much higher than for healthy patients, is, however, too low for the branch's condition. Thus, there is here a disagreement

between the ERT and the *ForestNet*. The body mass index was used twice in the branch. Once the value meets the condition, and the second time, it does not. Thus, we can say that this feature is partially compliant with the branch's conditions.

We saw that in some cases, the *ForestNet* and the ERT tend to have dissimilarities. For example, the glucose concentration in the third branch has a value relatively far from what the branch expected. In other cases, such as for the second body mass index condition of the same branch, the difference can be considered as small. Thus, we can question whether the *ForestNet* and the ERT use similar class boundaries. To investigate this, we can display them, as explained in Sect. 6.

An example is given in Fig. 5, where green corresponds to healthy and red to diabetes. We can observe that the boundaries, while more refined in the case of the *ForestNet*, are very similar, yet not identical. The ERT also exhibits some overfitting, as for glucose concentrations of 160 and above, it has very quick (and apparently meaningless) class transitions when the body mass index is changed. Due to the higher accuracy always reached by the *ForestNet*, we can safely assume that the majority of these differences are in favor of the neural network.

The class boundaries which the *ForestNet* and the ERT found, while of different qualities, separate the input space in similar ways. In Fig. 5, the horizontal boundary at roughly 130 for the glucose corresponds to a non-linear boundary for the *ForestNet*. Thus, if we accept a small imprecision, some tolerance, then the boundary found by the *ForestNet* can be explained by the ERT.

In Table 6, we give, for all datasets, the average similarity over a tenfold cross-validation, i.e., the fraction of check marks in interpretation tables such as Table 5b over all test samples of all folds. We computed it with four tolerance levels, between 0 and 15% of the values of the thresholds. This means that for a tolerance τ , and a threshold T in the ERT, we use $T \cdot (1 + \tau)$ when comparing with the $<$ operator, and $T \cdot (1 - \tau)$ otherwise.

We can see that without tolerance, the similarity is above 60% for three of the datasets, and increases above 70% with a tolerance of 5%. With at least 5% of tolerance, the similarity values for the non-visual datasets get higher than for the visual datasets.

In the case of the visual datasets, the interpretation tables are not optimal for explaining classification results, as they would flood the user with data about pixel values. With these results, it appears that in addition to this, the *ForestNets* produce as well-class boundaries which are sufficiently different from the ones of the ERT to hinder the ability of ERT to explain the output of *ForestNets*.

Conclusion and Future Work

In this article, we have presented a method for producing automatically sparse MLP architectures, called *ForestNet*, using the structure of tree ensembles. We showed that the *ForestNet* significantly outperformed the tree ensembles on four different datasets. We also compared these sparse networks with their fully connected counterparts, and while *ForestNets* have significantly less parameters, the accuracy results were comparable on all four datasets. Furthermore, we showed that the low amount of parameters of the *ForestNet* has a beneficial impact on over-fitting issues.

We also showed that the class boundaries produced by *ForestNets* share similarities with the boundaries produced by the corresponding ERT. Thanks to this property, we could introduce an explanation methodology of the networks' outputs based on few branches of the ERT. Due to the simple inner working of decision trees, non-experts in machine learning can be provided with easy-to-understand definitions of the class boundaries involved in the processing of a specific sample.

As future work, we intend to investigate more the usage of *ForestNets* on 2D, visual data. We will focus on two aspects: classification and interpretability. Currently, *ForestNets* are outperformed by CNN on MNIST and SVHN. Indeed, due to their nature, they are unable to achieve invariance/equivariance to translation and scaling. We intend to investigate the feasibility of creating convolutional *ForestNets* and stacking them. Information from the ERT or the complexity of the top layer of the stack might be a hint about whether additional layers are required. These networks will also require the development of new approaches for explaining their classification outputs in a visual, human-understandable way.

Acknowledgements Open Access funding provided by Projekt DEAL. This research was partially funded by The German Academic Exchange Service (DAAD—German: Deutscher Akademischer Austauschdienst).

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Bach S, Binder A, Montavon G, Klauschen F, Müller KR, Samek W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLoS One*. 2015;10(07):1–46. <https://doi.org/10.1371/journal.pone.0130140>.
- Biau G, Scornet E, Welbl J. Neural random forests. *Sankhya A*. 2019;81(2):347–86. <https://doi.org/10.1007/s13171-018-0133-y>.
- Breiman L. Random forests. *Mach Learn*. 2001;45(1):5–32. <https://doi.org/10.1023/A:1010933404324>.
- Dheeru D, Karra Taniskidou E. UCI machine learning repository. 2017. <http://archive.ics.uci.edu/ml>.
- Frankle J, Carbin M. The lottery ticket hypothesis: finding sparse, trainable neural networks. In: International conference on learning representations. 2019. <https://openreview.net/forum?id=rJl-b3RcF7>.
- Frosst N, Hinton G. Distilling a neural network into a soft decision tree. 2017. [arXiv:1711.09784](https://arxiv.org/abs/1711.09784).
- Geurts P, Ernst D, Wehenkel L. Extremely randomized trees. *Mach Learn*. 2006;63(1):3–42. <https://doi.org/10.1007/s10994-006-6226-1>.
- Humbird KD, Peterson JL, Mcclarren RG. Deep neural network initialization with decision trees. *IEEE Trans Neural Netw Learn Syst*. 2019;30(5):1286–95. <https://doi.org/10.1109/TNNLS.2018.2869694>.
- Kingma DP, Ba J. Adam: a method for stochastic optimization. 2014. [arXiv:1412.6980](https://arxiv.org/abs/1412.6980).
- Koutník J, Cuccu G, Schmidhuber J, Gomez F. Evolving large-scale neural networks for vision-based reinforcement learning. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation, 2013. p. 1061–8.
- LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7553):436–44.
- LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. *Proc IEEE*. 1998;86(11):2278–324.
- LeCun Y, Cortes C. MNIST handwritten digit database. 2010. <http://yann.lecun.com/exdb/mnist/>
- Mahendran A, Vedaldi A. Understanding deep image representations by inverting them. 2014. [arXiv:1412.0035](https://arxiv.org/abs/1412.0035) [CoRR abs].
- Miikkulainen R, Liang J, Meyerson E, Rawal A, Fink D, Francon O, Raju B, Shahrzad H, Navruzyan A, Duffy N, et al. Evolving deep neural networks. In: Artificial intelligence in the age of neural networks and brain computing. Elsevier. 2019. p. 293–312.
- Molnar C. Interpretable machine learning. 2019. <https://christophm.github.io/interpretable-ml-book/>
- Murdoch WJ, Singh C, Kumbier K, Abbasi-Asl R, Yu B. Definitions, methods, and applications in interpretable machine learning. *Proc Natl Acad Sci*. 2019;116(44):22071–80. <https://doi.org/10.1073/pnas.1900654116>.
- Netzer Y, Wang T, Coates A, Bissacco A, Wu B, Ng AY. Reading digits in natural images with unsupervised feature learning. 2011.
- Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A. Automatic differentiation in pytorch. 2017.
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E. Scikit-learn: machine learning in Python. *J Mach Learn Res*. 2011;12:2825–30.
- Rodríguez-Salas D, Gómez-Gil P, Olvera-López A. Designing partially-connected, multilayer perceptron neural nets through information gain. In: The 2013 international joint conference on neural networks (IJCNN), 2013. p. 1–5. <https://doi.org/10.1109/IJCNN.2013.6706991>.

22. Rodríguez-Salas D, Ravikumar N, Seuret M, Maier A. Forest-net—automatic design of sparse multilayer perceptron network architectures using ensembles of randomized trees. In: Asian conference on pattern recognition. Springer; 2019. p. 32–45.
23. Rosenblatt F. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev.* 1958;65(6):386.
24. Selvaraju RR, Das A, Vedantam R, Cogswell M, Parikh D, Batra D. Grad-cam: why did you say that? visual explanations from deep networks via gradient-based localization. 2016. [arXiv:1610.02391](https://arxiv.org/abs/1610.02391) [CoRR abs]
25. Sethi IK. Entropy nets: from decision trees to neural networks. *Proc IEEE.* 1990;78(10):1605–13.
26. Stanley KO, Miikkulainen R. Evolving neural networks through augmenting topologies. *Evolut Comput.* 2002;10(2):99–127.
27. Steinberg D, Colla P. Cart: classification and regression trees. *Top Ten Algorithms Data Min.* 2009;9:179.
28. Utgoff PE. Perceptron trees: a case study in hybrid concept representations. *Connect Sci.* 1989;1(4):377–91.
29. Wilcoxon F. Individual comparisons by ranking methods. *Biometr Bull.* 1945;1(6):80–3.
30. Wu M, Hughes MC, Parbhoo S, Zazzi M, Roth V, Doshi-Velez F. Beyond sparsity: tree regularization of deep models for interpretability. In: Thirty-second aaai conference on artificial intelligence. 2018.
31. Wu M, Parbhoo S, Hughes MC, Roth V, Doshi-Velez F. Optimizing for interpretability in deep neural networks with tree regularization. 2019.
32. Zhang Q, Yang Y, Ma H, Wu YN. Interpreting cnns via decision trees. 2019. p. 6261–70.
33. Zhang Q, Yang Y, Wu YN, Zhu S. Interpreting cnns via decision trees. 2018. [arXiv:1802.00121](https://arxiv.org/abs/1802.00121) [CoRR abs]

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.