Neural-Symbolic Temporal Decision Trees for Multivariate Time Series Classification

Giovanni Pagliarini
Department of Mathematics and Computer Science, University of Ferrara, Italy
Department of Mathematics, Physics, and Computer Science, University of Parma, Italy

Simone Scaboro
Department of Mathematics, Physics, and Computer Science, University of Udine, Italy

Giuseppe Serra
Department of Mathematics, Physics, and Computer Science, University of Udine, Italy

Guido Sciavicco
Department of Mathematics and Computer Science, University of Ferrara, Italy

Ionel Eduard Stan
Department of Mathematics and Computer Science, University of Ferrara, Italy
Department of Mathematics, Physics, and Computer Science, University of Parma, Italy

Abstract

Multivariate time series classification is a widely known problem, and its applications are ubiquitous. Due to their strong generalization capability, neural networks have been proven to be very powerful for the task, but their applicability is often limited by their intrinsic black-box nature. Recently, temporal decision trees have been shown to be a serious alternative to neural networks for the same task in terms of classification performances, while attaining higher levels of transparency and interpretability. In this work, we propose an initial approach to neural-symbolic temporal decision trees, that is, an hybrid method that leverages on both the ability of neural networks of capturing temporal patterns and the flexibility of temporal decision trees of taking decisions on intervals based on (possibly, externally computed) temporal features. While based on a proof-of-concept implementation, in our experiments on public datasets, neural-symbolic temporal decision trees show promising results.

2012 ACM Subject Classification Theory of computation → Theory and algorithms for application domains

Keywords and phrases Machine learning, neural-symbolic, temporal logic, hybrid temporal decision trees

Digital Object Identifier 10.4230/LIPIcs.TIME.2022.13

Supplementary Material *Software (ModalDecisionTrees.jl)*: `https://zenodo.org/record/7040420`

Funding The authors acknowledge the partial support by the Italian INDAM GNCS project *Natural Language Elaboration and Temporal Logic for Text Formalization.*

1 Introduction

A *multivariate time series* is a collection of time-stamped tuples, each composed by the value of several attributes. Time series describe a variety of situations, and *classification* of time series is an active area of research across many scientific disciplines: air quality control and prediction in climate science, prices and rates of inflation in economics, infectious diseases trends and spreading patterns in medicine, pronunciation of word signs in linguistics, sensor recordings of systems in aerospace engineering, among many others [29].

As it is true for any other classification problem, the classification of multivariate time series too can be approached by means of both *symbolic* and *functional* (or *parametric*) machine learning, which are two fundamental pillars of modern machine learning; in short, one may say that functional learning is the process of learning a *function* that represents the theory of the underlying problem (functional methods range from simple *regression* techniques to the modern *neural network* models, in their several variants), while symbolic learning is the process of learning a *logical description* that represents that theory (typical symbolic methods are *rule-based classifiers* and *decision trees*). Whether one or the other approach should be preferred raised a long-standing debate among experts, whose roots lie in the fact that functional methods tend to be more accurate than symbolic ones, as they are capable to better generalize the problem at hand, while symbolic methods are able to extract models that can be explained, interpreted, and integrated with human expert knowledge. The higher *interpretability* degree of symbolic approaches over functional ones, both for political (consider, e.g., the General Data Protection Regulation (GDPR) that highlights the need for interpretable/explainable automatic decision processes) and technical reasons, are sometimes used as arguments for preferring a symbolic approach over a functional one. As suggested in [9, 10, 24] (among others), however, in order to solve the functional/symbolic duality, one can think of an *hybrid* approach: hybrid systems combine the strengths of both symbolic and functional methods, with the aim of guaranteeing highi degrees of interpretability of the learned models, while retaining high enough statistical performances.

On the one side, we shall consider the native, symbolic time series classification method proposed by Sciavicco and Stan [31]. The *temporal decision tree* prediction model, as it is called, is an extension and generalization the decision tree paradigm. Temporal decision trees work as the classical ones, but decisions are taken on *intervals* of time series, instead of the series as a whole; therefore, there is no need of an initial feature extraction phase, but, on the contrary, features are extracted dynamically, following the standard greedy approach. Temporal decision trees have already been shown to be competitive for time series classification. On the other side, we shall take, as a representative example, a *time-preserving autoencoder* neural network model [37, 39], which is characterized by being able to play both the role of time series classifier and the one of feature extractor, essentially without any modification. As it turns out, in the pursue of an hybrid decision tree model there are at least three independent parameters which can be combined, namely, the possibility of using a network for an initial screening of the dataset, to be later dealt with in a more precise way by one of several potentially different trees (*root hybridization*), the possibility of querying an external network as a feature extractor in order to take split decisions in a single tree (*split hybridization*), and the possibility of consulting one of several potentially different networks at the leaves of a decision tree before deciding the class (*leaf hybridization*). These techniques give rise to eight possible hybrid models, the simplest one of which is just a decision tree. In this paper, we propose to increase the generalization capacity of temporal decision trees by leveraging the power of a pre-trained autoencoder to partition instances in a decision node (split hybridization). To assess the value of our proposal, we perform several experiments on three public datasets. In particular, we consider the problem of establishing if the neural-symbolic approach is, in fact, beneficial, when compared with the pure neural network-based one as well as the pure temporal symbolic one. Therefore, our experiments are not designed to achieve the highest absolute performances (such as in Bagnall et al.'s [3] work, from which our datasets are taken), which is often the results of a very intensive hyperparametrization, complex feature extraction, and model stacking/bagging; instead, they are structured in such a way as to highlight the advantages of the hybrid approach over its constituents.

The paper is organized as follows: in Section 2 we review the most important contributions in the area of hybridization decision trees with neural networks; in Section 3 we present neural-symbolic temporal decision trees; then, in Section 4 we benchmark the proposed methodology based on a proof-of-concept implementation against public datasets, before concluding.

2 Related Work

Decision trees and neural networks are well-known alternatives for pattern recognition, and their strengths and weaknesses have been studied for more than three decades [2, 35]. Notoriously, decision trees favor the interpretability of their decisions which, due to their symbolic nature, represent coarse concepts in numeric domains, whereas neural networks are more difficult to interpret (they are often referred to as *black-box models*), but have a better generalization capacity. Let us focus on the recent literature concerning the hybridization of these two models.

From decision trees to neural networks. First, we mention Sethi [32], Brent [6] and Ivanova and Kubat [15], who proposed a mapping algorithm of a decision tree into a 2-hidden-layers neural network, whose topology is inferred by the structure of the decision tree; then the weights can be retrained by error-backpropagation to increase generalization. Ivanova and Kubat's Tree-Based Neural Network has been further investigated by Setiono and Leow [33] by compressing and removing redundant units and connections in the resulting network, a method called Pruning-Based Neural Network. Radial-basis function networks can also be initialized by decision trees, where each region in the instance space discovered by the decision tree is then turned into a neuron, each of which representing a basis function, in the resulting network [19].

From neural networks to decision trees. Craven and Shavlik [7] observed that decision tree induction algorithms are limited by the fact that splits are performed over fewer and fewer instances recursively. The decision tree inducer algorithm that they have proposed alleviates such issue by querying the oracle (i.e., the trained neural network) which generates new instances aiming at performing better statistically solid m-to-n Boolean splits [38] (i.e., m out of n conditions must be satisfied) that are greedily evaluated using gain ratio [28]. Dancey et al. [8] proposed a similar method where the splits are univariate. Krishnan et al. [18], inspired by Craven and Shavlik's work, extracted a decision tree from inputs generated by the neural network instead of doing it directly from the data, that is, they also query the oracle. Unlike Craven and Shavlik's method, Schmitz et al. [30] proposed an approach that can be applied to inputs and outputs that are both discrete or continuous with a novel attribute significance analysis to perform splits. Zhou and Jiang [42], instead, proposed to extract a C4.5 decision tree from the input instances merged with new randomly generated instances from an ensemble of neural networks. Such an approach an be seen as a loop that uses a genetic algorithm to generate *prototypes* (i.e., representative input instances for which the neural network give a desired output classification) to train the decision tree; at this stage, the decision tree is tested on an independent test set and if the performances are acceptable then the procedure stops; otherwise, other prototypes are generated by the GA and the cycle continues.

Hybrid neural-symbolic models. Li et al. [21] proposed a top-down adaptive neural tree for hierarchical classification that can add and delete nodes incrementally while inducing the tree structure. To increase the generalization of CART-like decision trees [5], Guo and Gelfand's [12] method trained a small 1-hidden-layer perceptron with one output at each node of the decision tree to learn a non-linear, multivariate feature $f(\cdot)$ that splits the subset of the training instances at that node based on the test $f(\cdot) < 0$ (since the output's activation function is the *tanh* function) and showed that their method performed better

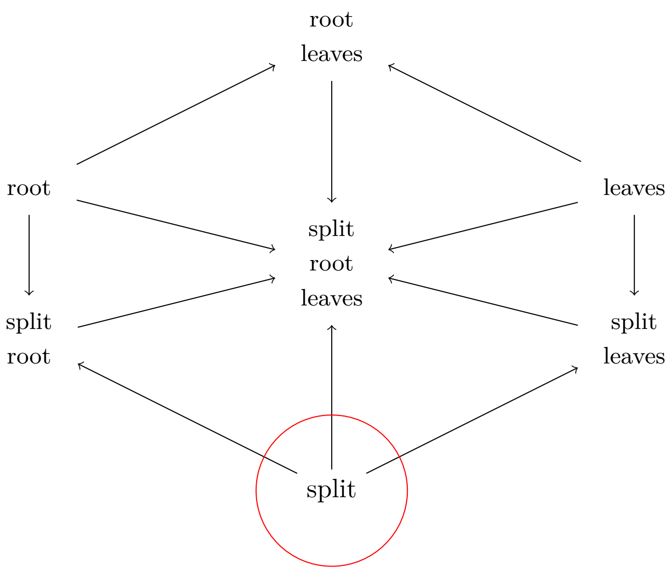
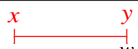
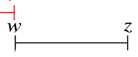
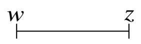
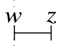
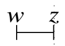
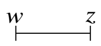
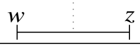


Figure 1 Different types of hybridization, partially ordered by the residual level of interpretability. In this paper, we focus our attention on temporal decision trees hybridized at the split level only.

than CART in terms of accuracy and better than a larger multi-layered neural network trained with backpropagation in terms of training time. A similar approach can be found in Setiono and Liu's [34] work for generating oblique decision trees. Zhou and Chen [41] proposed a methodology to induce a hybrid decision tree where, first, at the internal nodes of the decision tree the splits are done over unordered attributes only, if any, to perform qualitative analysis and, then, at the leaf nodes a virtual feed-forward neural network is embedded to perform quantitative analysis over the ordered attributes only, if any. More recently, Micheloni et al. [23] proposed a novel neural tree by using two innovations, namely perceptron substitution and pattern removal, to produce k-ary balanced trees ($k \geq 2$). In the field of computer vision, tree-structured neural networks have been proposed, among others, by Srivastava and Salakhutdinov [36] and Hinton et al. [14] to transfer knowledge, by Kontschieder et al. [17] where a decision tree has been introduced after a fully connected layer as part of the convolutional neural network, by Murthy et al. [26] where each decision node of the decision tree is a convolutional neural network, by Murdock et al. [25] where several layers are fused into the decision nodes of the decision tree, by Alaniz et al. [1] where the structure of the decision tree is encoded in the memory of a recurrent neural network jointly learned by two models acting as agents through message communication, and by Wan et al. [40] where the final layer of the network is replaced by a decision tree.

In an attempt at taxonomizing the existing work, one could argue that there are at least three independent parameters which can be combined, namely the possibility of using a network for an initial screening of the dataset, to be later dealt with in a more precise way by one of several potentially different trees (*root hybridization*), the possibility of querying an external network as a feature extractor in order to take split decisions in a single tree (*split hybridization*), and the possibility of consulting one of several potentially different networks at the leaves of a decision tree before deciding the class (*leaf hybridization*). As a result, there are at least eight types of hybridization (the simplest one of which is just a decision tree), which can be, in a sense, partially ordered by their residual level of interpretability, as

Table 1 Allen's interval relations and their representation.

HS modality	Definition w.r.t. the interval structure			Example
				
$\langle A \rangle$ (after)	$[x,y]R_A[w,z]$	$\Leftrightarrow$	$y = w$	
$\langle L \rangle$ (later)	$[x,y]R_L[w,z]$	$\Leftrightarrow$	$y < w$	
$\langle B \rangle$ (begins)	$[x,y]R_B[w,z]$	$\Leftrightarrow$	$x = w \wedge z < y$	
$\langle E \rangle$ (ends)	$[x,y]R_E[w,z]$	$\Leftrightarrow$	$y = z \wedge x < w$	
$\langle D \rangle$ (during)	$[x,y]R_D[w,z]$	$\Leftrightarrow$	$x < w \wedge z < y$	
$\langle O \rangle$ (overlaps)	$[x,y]R_O[w,z]$	$\Leftrightarrow$	$x < w < y < z$	

in Fig. 1. When the architecture of the underlying neural networks are comparable, different hybridization types become comparable as well. In our initial exploration, we focus on the hybridization at the split level only.

3 Neural-Symbolic Temporal Decision Trees

A single *multivariate time series* has n (temporal) attributes $A_1, \ldots, A_n$ evolving through a time axis, whose values are time-stamped by N integers (the *length* of the series), that is, its underlying domain of interest is $(\{0, 1, \ldots, N-1\}, <)$. A *temporal labelled dataset* is a set of m multivariate time series, each labelled with a *class*. The multivariate time series *classification problem* is the problem of automatically extract a classifier from a temporal labelled dataset. Existing multivariate time series classification methods can be divided into *feature-based* (see, e.g., [20]), *instance-based* (see, e.g., [3]) and *native* ones (see, e.g., [11]).

Designed as a native time series classification method, *temporal decision trees* have been recently proposed [22, 31]. Let $\mathcal{T} = \{T_1, \ldots, T_m\}$ be a temporal dataset of m instances, where each is a multivariate time series described by n attributes $\{A_1, \ldots, A_n\}$. Given $T \in \mathcal{T}$ and a time point t, we denote by $A(t)$ the value of A at the point t, and by $dom(A)$ the domain of A. Now, let f a *dynamic feature* of the variable A (e.g., the average value of A over an interval); in its simplest form, f is a *scalar descriptor* for A within any non-point interval of the series.

The key idea of interval temporal decision trees is that decisions are taken over *strict* intervals, that is, intervals of the type $[x, y]$ with $x < y$, and for time series whose length is N, there are $N \cdot (N-1)/2$ such intervals. A temporal decision tree starts off by looking at the whole dataset from the point of view of the first temporal value, and searches through all possible non-point intervals, and, for each one of them, it computes a predetermined set of dynamic features; then, it searches through all possible interval-interval relations, and it establishes which other interval, and which other dynamic feature over that interval, is most informative in the considered sub-dataset. In this way, it applies the same abstract approach of the classical static decision tree up until a dataset is small enough, or pure enough, so that a stopping criteria can be applied and a leaf can be created. For each possible feature f, let $dom(f(A))$ denote the set of possible values that f takes over A throughout $\mathcal{T}$. The temporal dataset $\mathcal{T}$ entails a propositional alphabet $\mathcal{AP}$ defined as follows:

$$\mathcal{AP} = \{f(A) \bowtie a \mid A \in \mathcal{A}, \bowtie \in \{<, \leq, =, \geq, >\} \text{ and } a \in dom(f(A))\}.$$

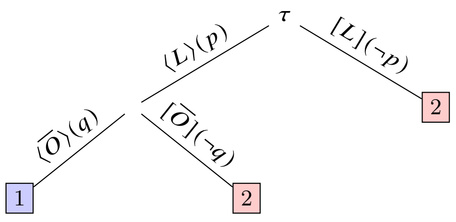


Figure 2 Example of temporal decision tree.

The set $\mathcal{AP}$ is the natural generalization of the set of propositional letters that implicitly emerges in inductive processes from static data; for example, in a static dataset, the propositional letter *fever greater than* 38 *degrees* may emerge. The main difference between the two cases, propositional and temporal, is that in the latter case propositions in $\mathcal{AP}$ are given an interval semantics, that is, they are evaluated over intervals of time; this is a natural choice that depends from the fact that time series describe continuous processes, in which evaluations based on point-wise values have little sense. Intuitively, consider an interval of time $[x, y]$ and an attribute A that varies on it. We can ask the question $A \bowtie a$ over the entire interval, which is positively answered if *every value* of A in the interval $[x, y]$ respects the given constraint; but, to enhance an interval-based semantics we replace this question with $f(A) \bowtie a$, which, in general, allows us to extract more information from the interval $[x, y]$; in the above example, we may have the proposition *average fever greater than 38 degrees*. As it turns out, many dynamic features have been studied in the literature of time series to extract features from *whole* series, and these range from simple functions such as *average, minimum,* or *maximum* to very complex ones such as *number of local minima* or *number of local maxima*; we generalize this concept by applying them to *intervals*, which are, themselves, series. Thus, in temporal decision trees the *univariate split-decisions* (or, simply, *decisions*) that partition a set of instances at a specific node are of the type:

$$\mathcal{S} = \{\langle X\rangle(p) \mid X \in \mathcal{X} \cup \{=\} \text{ and } p \in \mathcal{AP}\},$$

where $\mathcal{X} = \{A, L, B, E, D, O, \overline{A}, \overline{L}, \overline{B}, \overline{E}, \overline{D}, \overline{O}\}$, contains the possible interval-interval relations, whose informal semantics is depicted in Tab. 1 (for a formal definition of the semantics, see [13]). In conclusion, binary *temporal decision trees* τ are formulas of the following grammar:

$$\tau ::= (S \wedge \tau) \vee (\neg S \wedge \tau) \mid C,$$

where $S \in \mathcal{S}$ is a decision and $C \in \mathcal{C}$ is a class. An example of temporal decision tree is depicted in Fig. 2. Recall that time series of length N have $(\{0, 1, \ldots, N-1\}, <)$ as underlying domain of interest. The learning starts by splitting the dataset at the root, where each time series is fixed on the *dummy* interval $[-2, -1]$ so that the only feasible interval-interval operators are $\langle L\rangle$ (for the left branch) and $[L]$ (for the right branch); then, recursively, for the instances that fall into the left branch, the intervals that are *witnesses* for the decision $\langle X\rangle(p)$ are the new intervals to which the time series are fixed to find the next interval-interval relation with a proposition (from $\mathcal{A}P$) as argument; similarly, if the instances fall on the right branch by some decision $[X](\neg p)$, the witnesses remain the same as before the split. What is more, just as static decision trees induce propositional formulas on their branches, temporal decision trees induce *interval temporal logic* [13] formulas. Continuing the parallelism with the static case, a static decision tree may give rise to a branch whose associated formula is *fever greater than 38 degrees and cough as a symptom*, whereas a formula on a temporal

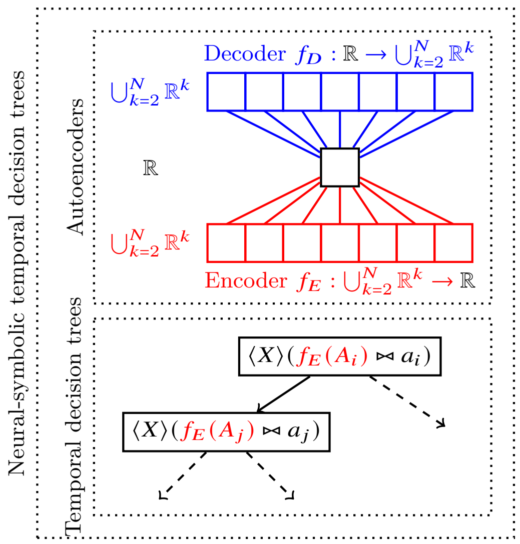

Neural-symbolic temporal decision trees
Autoencoders
Decoder $f_D : \mathbb{R} \to \bigcup_{k=2}^{N} \mathbb{R}^k$
$\bigcup_{k=2}^{N} \mathbb{R}^k$
$\mathbb{R}$
$\bigcup_{k=2}^{N} \mathbb{R}^k$
Encoder $f_E : \bigcup_{k=2}^{N} \mathbb{R}^k \to \mathbb{R}$
Temporal decision trees
$\langle X \rangle (f_E(A_i) \bowtie a_i)$
$\langle X \rangle (f_E(A_j) \bowtie a_j)$

Figure 3 Neural-symbolic temporal decision trees pipeline. At the top level, an autoencoder is trained for each attribute A_i on each of its non-point interval of a multivariate time series. At the bottom level, a temporal decision tree queries the encoder of each trained autoencoder to partition instances in a decision node.

decision tree may be *(a period of) average fever greater than 38 degrees overlapping (a period of) cough as a symptom*; to complete the example, following the left-most branch of the temporal decision tree illustrated in Fig. 2, the corresponding interval temporal logic formula is:

$$\langle L \rangle (p \wedge \langle \overline{O} \rangle (q)),$$

from which, by interpreting p as *average fever greater than 38 degrees* and q as *cough as symptom*, we obtain the above sentence in natural language.

An *autoencoder* is a neural network architecture typically used for extracting significant feature representations from unlabeled data. The feature extraction is achieved by training the model to reproduce its input (i.e., to learn the identity function) while introducing an information bottleneck throughout the model. This generally results in an encoder-decoder architecture where the encoder (that is, the first part of the network), ends with a layer with the smallest number of neurons, which is, then, the only input to the remaining part of the network. Ultimately, the training phase forces the encoder to learn to extract a succinct representation of the input, performing a non-linear dimensionality reduction, and the decoder to learn to retrieve the original information from this representation. After the training phase, the encoder can be used as a feature extractor, that is, a model that provides a succinct abstract description of its input.

In the case of time series, a fundamental distinction among autoencoders is between *time preserving* models, which provide a description that also extends along the time axis, and *non-time preserving* ones, which provide a static description, only consisting of a fixed-size vector of scalar values. The architectures used for time series are typically based on convolutional neural networks, that are effective for shift-invariant pattern recognition, or recurrent neural networks, that are specifically designed for temporal data.

Table 2 Dataset specifications.

Dataset	# train+test instances	# points	# attributes	# classes
Libras	180 + 180 = 360	45	2	15
NATOPS	180 + 180 = 360	51	24	6
RacketSports	151 + 152 = 303	30	6	4

In this first attempt at split hybridization of temporal decision trees, autoencoders are used to derive attribute-specific feature extractors; that is, once an autoencoder is trained, the encoder part, seen as a function whose input is mapped to a real number, plays the same role that the average, minimum, and maximum functions play. With respect to these simpler function, the learned encoder has a black-box nature, and is, therefore, less interpretable; however, as we shall see, it can yield higher specificity for a given attribute, thus providing scalar descriptions that are more relevant. For the purpose of this work, we consider a sequence-to-sequence [37] architecture (referred to as *S2S*), and a transformer-based [39] architecture (referred to as *transformer autoencoder, TSA*); these are two time preserving autoencoders which found fruitful application in many contexts, and represented a major breakthrough in the field of natural language processing. Note that, being time preserving architectures, the descriptions computed by the encoders extend along a time axis, whereas the framework presented above requires a scalar feature extraction. To solve this issue, we operate a choice commonly adopted in contexts where a scalar feature extraction is needed: we only consider a single temporal slice of the description. For both architectures, we considered the one temporal slice that is used by the decoder to reproduce the input series but, because of structural differences between the two architectures, different policies are required: for S2S, the last temporal slice is considered while, for TSA, the first slice is considered. Fig. 3 depicts an abstraction of the structure of the autoencoders in use, and the deployment of the trained encoders within the split-decisions of a temporal decision tree. Recall that the encoder maps its input series (seen as a vector of real numbers) to a single real number, therefore reducing the original size.

4 Experiments

In order to assess the performances of hybrid temporal decision trees, we carried out several experiments using three publicly available datasets that are commonly employed for benchmarking multivariate time series classification models. They are known, respectively, as Libras, NATOPS, and RacketSports [3], and their specifications are shown in Tab. 2. The Libras dataset consists of sensor recordings of hand movements in a bidimensional space, extracted from videos of Brazilian sign language speakers performing different gestures; NATOPS data consists of 3D recordings of hands, elbows, wrists and thumbs of people performing different actions; and RacketSports contains 3D recordings for both a gyroscope and an accelerometer mounted on a smartwatch worn by several subjects while playing badminton and squash games.

All datasets are provided with pre-existent partitioning into training set and test set. The expertiments are conducted in a randomized cross-validation setting with 10 repetitions, where the training and test sets for each repetition are drawn from the union of the original sets, reproducing the class distributions of the original sets, with the sole exception that the first of the 10 repetitions uses the exact original training and test sets; this approach is similar to Ruiz et al. [29]. Six variations of decision trees are compared, which are obtained by using both a static decision tree model and the temporal decision tree one, each, in turn,

Table 3 Test results and training time of the decision tree models in comparison. Values for the performance metrics are shown in percentage points. For each measure, the table reports the average and standard deviation over 10 repetitions. For each dataset, the most performant model is highlighted.

			κ	OA	AA	F1	time (s)
Libras	DT	min, max	35.4 ± 3.4	39.7 ± 3.1	39.7 ± 3.1	39.3 ± 3.2	0.1
		neural	19.0 ± 4.3	24.4 ± 4.0	24.4 ± 4.0	23.8 ± 4.0	0.1
		min, max, neural	40.9 ± 5.9	44.8 ± 5.5	44.8 ± 5.5	44.2 ± 5.6	0.1
	TDT	min, max	54.6 ± 4.3	57.6 ± 4.0	57.6 ± 4.0	57.2 ± 3.8	6.3 ± 1.6
		neural	54.5 ± 3.7	57.5 ± 3.5	57.5 ± 3.5	56.7 ± 4.0	18.0 ± 5.1
		min, max, neural	**55.2 ± 4.1**	**58.2 ± 3.8**	**58.2 ± 3.8**	**57.6 ± 3.9**	**30.7 ± 6.5**
NATOPS	DT	min, max	65.1 ± 3.7	70.9 ± 3.1	70.9 ± 3.1	70.8 ± 3.3	0.7 ± 0.1
		neural	42.8 ± 4.3	52.3 ± 3.6	52.3 ± 3.6	52.1 ± 3.8	0.6 ± 0.1
		min, max, neural	65.7 ± 2.2	71.5 ± 1.8	71.5 ± 1.8	71.4 ± 1.9	1.0 ± 0.1
	TDT	min, max	84.0 ± 2.9	86.7 ± 2.4	86.7 ± 2.4	86.7 ± 2.4	37.0 ± 9.0
		neural	**87.1 ± 3.7**	**89.2 ± 3.1**	**89.2 ± 3.1**	**89.3 ± 3.1**	**118.3 ± 35.8**
		min, max, neural	86.7 ± 2.9	88.9 ± 2.4	88.9 ± 2.4	89.0 ± 2.4	252.1 ± 98.3
RacketSports	DT	min, max	55.4 ± 3.3	66.6 ± 2.4	68.0 ± 2.5	67.4 ± 2.3	0.2
		neural	44.2 ± 3.9	58.4 ± 3.0	59.6 ± 2.8	59.2 ± 3.1	0.2 ± 0.1
		min, max, neural	**57.5 ± 6.9**	**68.2 ± 5.2**	**69.7 ± 5.1**	**69.3 ± 5.3**	**0.3 ± 0.1**
	TDT	min, max	55.0 ± 5.8	66.3 ± 4.3	67.7 ± 4.2	67.5 ± 4.1	1.1 ± 0.9
		neural	56.0 ± 5.6	67.1 ± 4.2	68.2 ± 4.0	68.1 ± 4.3	2.7 ± 1.5
		min, max, neural	56.3 ± 5.8	67.3 ± 4.3	68.6 ± 4.2	68.3 ± 4.1	5.5 ± 5.4

in three versions: original (non-hybrid), split hybrid using only neural features, and split hybrid using both neural and non-neural features. As we have seen in the previous section, temporal decision trees are characterized by taking decisions on intervals of time, and then relating such decisions via temporal logic formulas. This is paradigm-shifting with respect to static ones, which, in the common literature, can only deal with dimensional data (for us, temporal data) by extracting global features for each attribute and then using them as decisions. Such a difference has been maintained in the hybrid version; in fact, in the static case both neural and non-neural features are computed on the whole series. For these experiments, in both the temporal and the static case, the non-neural feature functions were fixed to minimum and maximum. Temporal and static trees were trained using the ModalDecisionTrees.jl open-source Julia package [27], which implements the CART algorithm and its modal extensions.

After a preliminary study in which different decision tree parametrizations are tested, two tree-pruning conditions are fixed, namely, a minimum number of instances at the tree leaves of 2 for RacketSports and Libras, and 4 for NATOPS, and a minimum entropy gain of 0.01, which prevents less informative splits to be performed at any internal node. As neural feature extractors, we use S2S for Libras and TSA for NATOPS and RacketSports. For the training of both neural architectures, we use PyTorch and the following hyperparameters: AdamW optimizer with 10^{-5} learning rate and 10^{-8} epsilon factor, batches of size 256 with accumulation step equal to 4, L1Loss as loss function, 150 epochs, gradient clipping during training, and weights initialized using the default setting of PyTorch. The architecture of S2S is the same proposed in [4] with the difference that we used two Gated Recurrent Unit Networks as encoder and decoder instead of the LSTMs. A simpler architecture is used to

Table 4 Class-wise accuracies and average accuracy (AA) (shown in percentage points) of the decision tree models in comparison. For each entry, the table reports the average over 10 repetitions. The best result of each class is highlighted.

		Libras															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	AA
DT	min, max	29	41	26	45	53	38	34	39	**88**	28	38	38	36	16	**48**	40
	neural	17	12	13	46	38	22	34	8	30	14	28	32	19	22	30	24
	min, max, neural	35	57	25	59	53	31	42	50	87	36	41	43	44	28	42	45
TDT	min, max	48	77	**49**	70	**70**	49	**64**	55	**88**	**42**	**55**	51	41	**58**	**48**	58
	neural	**59**	76	42	68	59	**56**	62	**58**	**88**	35	52	**60**	**50**	50	47	58
	min, max, neural	56	**78**	46	**72**	69	51	62	55	86	39	54	53	53	52	47	**58**

		NATOPS						
		1	2	3	4	5	6	AA
DT	min, max	91	77	65	52	51	90	71
	neural	48	45	40	60	52	69	52
	min, max, neural	91	73	64	57	53	90	72
TDT	min, max	93	87	68	90	90	92	87
	neural	**95**	**88**	**70**	**91**	**94**	92	**89**
	min, max, neural	**95**	87	67	**91**	**94**	**94**	89

		RacketSports				
		1	2	3	4	AA
DT	min, max	50	**55**	83	84	68
	neural	54	36	85	64	60
	min, max, neural	55	52	**88**	85	**70**
TDT	min, max	51	54	80	85	68
	neural	**60**	49	77	**87**	68
	min, max, neural	54	53	82	84	69

build TSA, which is composed of a transformer encoder layer with two heads, both for the encoder and for the decoder. In the encoder, we used Time2Vec [16] to resize each temporal slice from 1 to 168 adding also information about the position of the slices, as done by classical transformers with the positional embeddings. Both the output of the encoder and of the decoder is resized from 168 to 1 with a linear layer. Note that, while static decision trees feature functions are computed on the whole series, features for temporal decision trees are evaluated on *all* the sub-intervals of the series; as such, the autoencoder models are trained using the raw training instances in the first case, and using all the sub-intervals of the training instances in the second case.

Tab. 3 gives an overview of the trained models in terms of the training time required (excluding the prior training of the neural networks), and the performance obtained on the test data. The results are given in terms of mean and standard deviations over the 10 repetitions. The performance itself is measured in terms of κ coefficient (which relativizes the overall accuracy to the probability of a random correct answer), overall accuracy (OA), average accuracy (AA), and average F1-score (F1). Note that the four metrics measure in different ways the overall performance of the models, but they all happen to be in agreement; as such, we focus on the values of the κ coefficients, which is invariant to the number of classes, and varies largely across the three datasets.

At a first look, Libras seems to enclose the hardest problem for the models at hand (κ equal to 55.2%), followed by RacketSports (57.5%) and then by NATOPS (87.2%). Libras and NATOPS reveal a performance gap between static and temporal trees; that is, when the best models for each group is considered, the second group has an average κ higher by ~15 and ~22 percentage points, respectively. Conversely, RacketSports represents a case where temporal decision trees are outperformed by classical decision trees; indeed, in both cases, the best accuracy is achieved using both simple and neural features, but temporal trees achieve an average κ of 56.3%, while classical trees achieve 57.5%. When it comes to neural features, classical DTs only benefit from them when these are used as supplementary information; in fact, for all datasets, classical decision trees using only neural features are

Table 5 Confusion matrixes for the tree generated with seed 6, dataset NATOPS; left: static, non-neural; right: temporal, neural.

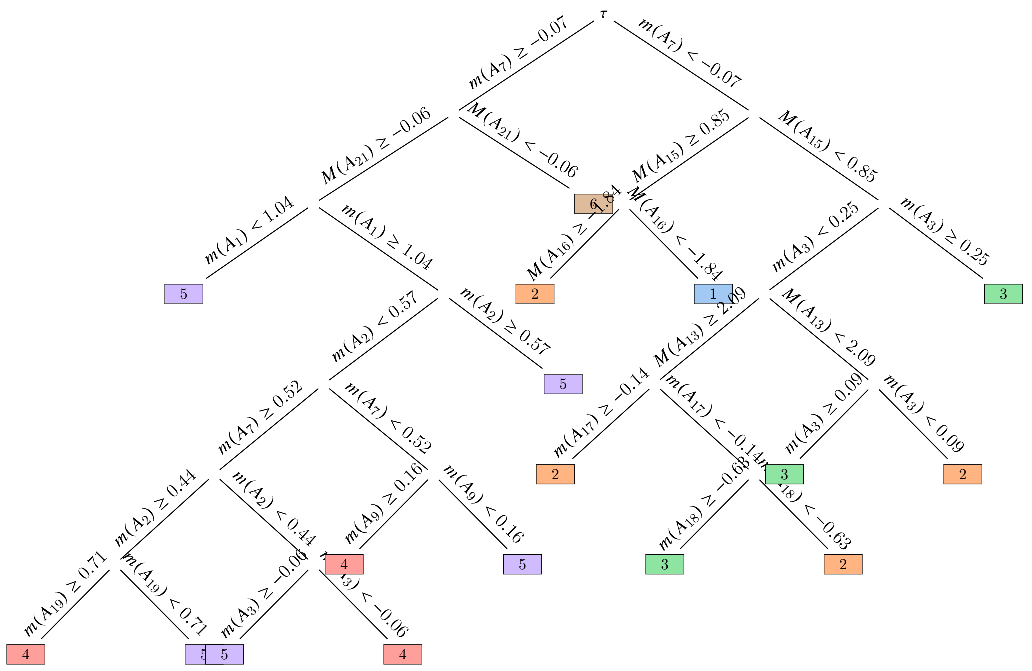
	static, non-neural					
	1	2	3	4	5	6
1	**24**	6	0	0	0	0
2	2	**26**	2	0	0	0
3	0	14	**16**	0	0	0
4	1	0	0	**19**	9	1
5	1	0	0	13	**16**	0
6	0	0	0	0	3	**27**

	temporal, neural					
	1	2	3	4	5	6
1	**30**	0	0	0	0	0
2	0	**28**	2	0	0	0
3	0	5	**25**	0	0	0
4	0	0	0	**26**	1	3
5	0	0	0	0	**28**	2
6	3	1	0	0	0	**26**

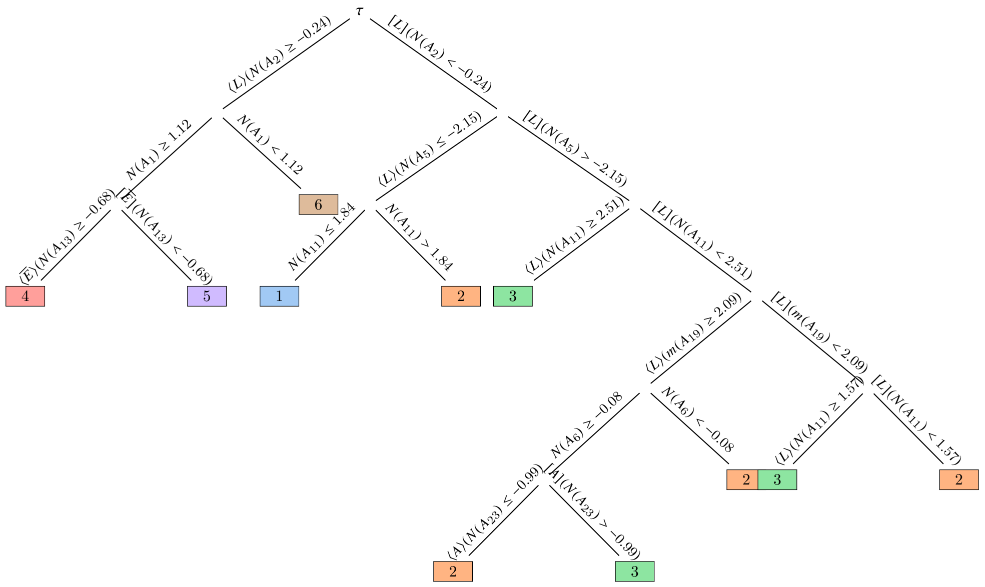
outperformed by classical DTs based on minimum and maximum, but the latter are in turn always outperformed by mixed trees that use both simple and neural features, which may indicate that our approach is promising. If we look separately at DTs and TDTs within the same experimental settings, we observe, once again, that adding neural features gives a greater advantage in the temporal case; a possible interpretation of this phenomenon is that, being able to perform qualitative temporal reasoning among intervals when partitioning instances, the inductive procedure becomes some kind of *symbolic attention* mechanism; however, as it must be acknowledged, TDTs generally require higher training times due to the interval-interval relations.

A more specific analysis of the trends can be made by inspecting the ability of the models for correctly classifying each of the classes in the three datasets. Tab. 4 reports the class-wise accuracies, as well as the average accuracy; although they provide useful insights, class-wise accuracies are subject to a higher variance than metrics of overall performance, and, as such, reliable conclusions from them can be drawn only when comparing different groups of models. We observe that with the Libras dataset the step from static to temporal learning encompasses a general accuracy improvement over all classes except two (9 and 15), and in some classes, such as 14, such an improvement is impressive (from 28% to 58%); classes 1, 2, 12, and 13, moreover, show a clear benefit of the hybrid temporal-and-neural approach. The improvement due to the temporal approach in NATOPS emerges in all classes except class 1, and in all classes, though in a lesser amount, adding the neural features results in a further improvement. Finally, as for RacketSports, the improvement of the temporal approach is less clear, and only visible in class 1 and 4, and the same holds for the further, slight, improvement given by the addition of the neural features.

In addition to the simple analysis of the numerical performance, we can give a closer look at the generated trees. In order to do so, we consider the dataset NATOPS and a representative seed, namely seed 6, and compare the extracted trees in terms of structure, size, and ability to predict specific classes in two specific cases: the static tree without neural features and the temporal tree with neural features (Fig. 4 and Fig. 5, in which we used m to represent the minimum, M the maximum, and N the encoding by the encoder neural network, applied to an attribute A). As it can be observed, the static one is 40% bigger than the temporal one, the former having 17 leaves, versus the 11 leaves of the latter. Yet, the classification abilities of the static tree are clearly lower than those of the temporal one: on the one side, in average, the static non-neural approach presented 71% accuracy versus the 89% of the temporal one with neural and non-neural features; on the other side, these specific trees present 71% accuracy in the static case versus 91% in the temporal case. Even more interestingly, class 5 labels several leaves in the static tree, and only 1 in the temporal



■ **Figure 4** Static tree without neural features, seed 6, NATOPS.



■ **Figure 5** Temporal tree with neural features, seed 6, NATOPS.

one, indicating that the temporal tree was able to extract the essence, in some sense, of this class, in a single formula. Observe that class 5 is classified correctly only 51% of the times in the static tree versus 94% of the times in the temporal one, in average, and 53% in the static case versus 93% in the temporal one for the considered trees. More in particular, as it can be deduced from the observation of the confusion matrices in both cases (Tab. 5), the static tree confuses class 5 with class 4 very often, and, in lesser amount, with class 6, while

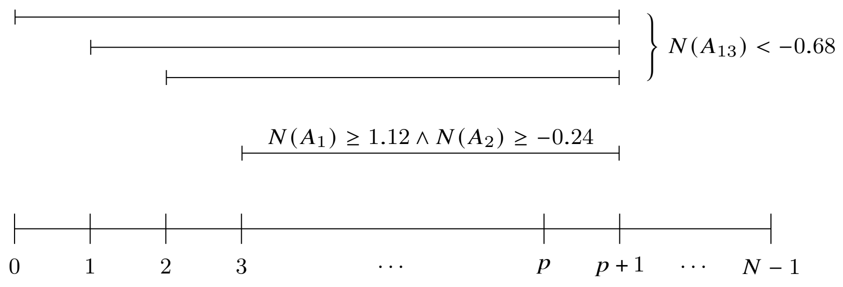


Figure 6 A model for class 5 of the NATOPS dataset.

the temporal neural tree presents similar mistakes a very reduced number of times. In other words, the temporal neural approach was able to extract a smaller and more precise logical description of class 5, which can be summarized into a single temporal formula:

$$\langle L\rangle(N(A_2) \geq -0.24 \wedge N(A_1) \geq 1.12 \wedge [\overline{E}]N(A_{13}) < 0.68) \Rightarrow 5,$$

which, in turn, can be expressed as a temporal model to visualize, in a sense, the class itself (see Fig. 6 and recall our discussion from the previous section on the witnesses for the left/right branch). To finalize this discussion, we observe that, in the static case, 7 variables were involved in the decision tree for class 5, while in the temporal case only 3 variables were sufficient.

5 Conclusions

In this paper we have presented a method for multivariate time series classification that combines the high generalization capacity of trained neural networks with the symbolic nature of temporal decision trees. This method is able to learn the structure of a temporal decision trees from raw multivariate time series, and, internally (at each decision node) performs both a qualitative analysis by means of entity-relation reasoning and a quantitative one by means of neural features extracted from pre-trained neural networks. Although based on a proof-of-concept implementation, our experiments performed on public datasets showed promising results, and allowed us to draw some initial conclusions: *(i)* the hybridization between temporal decision trees and neural networks seems quite natural; *(ii)* the obtained method offers a statistically significant improvement in performances over its constituents; *(iii)* such an improvement seems to be higher in more complicated problems.

As future work, we plan to perform a more systematic experimental benchmark, explore different neural network architectures, which have been successfully applied to the problem of multivariate time series classification, and investigate higher context sizes. More importantly, we plan to explore all different hybridization schemata, and compare them against each other and against the standard approach. Finally, a similar idea can be pursued in the spatial case, where neural networks have shown even more predictive ability than in the temporal one.

References

1 S. Alaniz, D. Marcos, B. Schiele, and Z. Akata. Learning decision trees recurrently through communication. In *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13518–13527, 2021.

2 L. Atlas, R. Cole, Y. Muthusamy, A. Lippman, J. Connor, D. Park, M. El-Sharkawai, and R.J. Marks. A performance comparison of trained multilayer perceptrons and trained classification trees. *Proc. of the IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 78(10):1614–1619, 1990.

3 A. J. Bagnall, J. Lines, A. Bostrom, J. Large, and E. J. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017.

4 D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2014. `doi:10.48550/arXiv.1409.0473`.

5 L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and regression trees*. Wadsworth Publishing Company, 1984.

6 R. P. Brent. Fast training algorithms for multilayer neural nets. *IEEE Transactions on Neural Networks*, 2(3):346–354, 1991.

7 M. W. Craven and J. W. Shavlik. Extracting tree-structured representations of trained networks. In *Proc. of the 8th Advances in Neural Information Processing Systems (NIPS)*, pages 24–30, 1995. URL: `http://papers.nips.cc/paper/1152-extracting-tree-structured-representations-of-trained-networks`.

8 D. Dancey, D. McLean, and Z. Bandar. Decision tree extraction from trained neural networks. In *Proc. of the 7th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 515–519, 2004.

9 A. S. d'Avila Garcez, M. Gori, L. C. Lamb, L. Serafini, M. Spranger, and S. N. Tran. Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning. *Journal of Applied Logics*, 6(4):611–632, 2019.

10 A. S. d'Avila Garcez, L. C. Lamb, and D. M. Gabbay. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer, 2009.

11 H. I. Fawaz, G. F., J. Weber, L. Idoumghar, and P.-A. Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

12 H. Guo and S. B. Gelfand. Classification trees with neural network feature extraction. *IEEE Transactions on Neural Networks*, 3(6):923–933, 1992.

13 J. Y. Halpern and Y. Shoham. A propositional modal logic of time intervals. *Journal of the ACM*, 38(4):935–962, 1991.

14 G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network, 2015. `doi:10.48550/arXiv.1503.02531`.

15 I. Ivanova and M. Kubat. Initialization of neural networks by means of decision trees. *Knowledge-Based Systems*, 8(6):333–344, 1995.

16 Seyed Mehran Kazemi, Rishab Goel, Sepehr Eghbali, Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart, and Marcus Brubaker. Time2vec: Learning a vector representation of time, 2019. `doi:10.48550/arXiv.1907.05321`.

17 P. Kontschieder, M. Fiterau, A. Criminisi, and S. Rota Bulò. Deep neural decision forests. In *Proc. of the International Conference on Computer Vision (ICCV)*, pages 1467–1475, 2015.

18 R. Krishnan, G. Sivakumar, and P. Bhattacharya. Extracting decision trees from trained neural networks. *Pattern Recognition*, 32(12):1999–2009, 1999.

19 M. Kubat. Decision trees can initialize radial-basis function networks. *IEEE Transactions on Neural Networks*, 9(5):813–821, 1998.

20 M. Längkvist, L. Karlsson, and A. Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

21 T. Li, L. Fang, and A. Jennings. Structurally adaptive self-organizing neural trees. In *Proc. of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 329–334, 1992.

22 F. Manzella, G. Pagliarini, G. Sciavicco, and I. E. Stan. Interval temporal random forests with an application to COVID-19 diagnosis. In *Proc. of the 28th International Symposium on Temporal Representation and Reasoning (TIME)*, volume 206 of *LIPIcs*, pages 7:1–7:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021.

23 C. Micheloni, A. Rani, S. Kumar, and G. L. Foresti. A balanced neural tree for pattern classification. *Neural Networks*, 27:81–90, 2012.

24 M. Minsky. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine*, 12(2):34–51, 1991.

25 C. Murdock, Z. Li, H. Zhou, and T. Duerig. Blockout: Dynamic model selection for hierarchical deep networks. In *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2583–2591, 2016.

26 V. N. Murthy, V. Singh, T. Chen, R. Manmatha, and D. Comaniciu. Deep decision network for multi-class image classification. In *Proc. of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2240–2248, 2016.

27 G. Pagliarini, F. Manzella, G. Sciavicco, and I. E. Stan. ModalDecisionTrees.jl: Interpretable models for native time-series & image classification, 2021. `doi:10.5281/zenodo.7040419`.

28 J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

29 A. Pasos Ruiz, M. Flynn, J. Large, M. Middlehurst, and A. J. Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35(2):401–449, 2021.

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

31 G. Sciavicco and I. E. Stan. Knowledge extraction with interval temporal logic decision trees. In *Proc. of the 27th International Symposium on Temporal Representation and Reasoning (TIME)*, volume 178 of *LIPIcs*, pages 9:1–9:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

32 I. K. Sethi. Entropy nets: From decision trees to neural networks. *Proc. of the IEEE*, 78(10):1605–1613, 1990.

33 R. Setiono and W. K. Leow. On mapping decision trees and neural networks. *Knowledge-Based Systems*, 12(3):95–99, 1999.

34 R. Setiono and H. Liu. A connectionist approach to generating oblique decision trees. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, 29(3):440–444, 1999.

35 J. W. Shavlik, R. J. Mooney, and G. G. Towell. Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, 6:111–143, 1991.

36 N. Srivastava and R. Salakhutdinov. Discriminative transfer learning with tree-based priors. In *Proc. of the 26th Advances In Neural Information Processing Systems (NIPS)*, pages 2094–2102, 2013.

37 I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Proc. of the 27th Advances in Neural Information Processing Systems (NIPS)*, pages 3104–3112, 2014.

38 G. G. Towell and J. W. Shavlik. Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.

39 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. of the 30th Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

40 A. Wan, L. Dunlap, D. Ho, J. Yin, S. Lee, S. Petryk, S. Adel Bargal, and J. E. Gonzalez. NBDT: Neural-Backed Decision Tree. In *Proc. of the 9th International Conference on Learning Representations (ICLR)*, 2021.

41 Z.-H. Zhou and Z. Chen. Hybrid decision tree. *Knowledge-Based Systems*, 15(8):515–528, 2002.

42 Z.-H. Zhou and Y. Jiang. NeC4.5: Neural Ensemble Based C4.5. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):770–773, 2004.