ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Implicit Neural Decision Trees

Francesco Spinnato[1,2,†], Antonio Matropietro[1,†] and Riccardo Guidotti[1,2] *

1 - University of Pisa, Italy
2 - ISTI-CNR Pisa, Italy

Abstract. Representation learning is a central topic in machine learning, with significant efforts dedicated to encoding structured data such as sequences, trees, and graphs for various downstream tasks. A branch of these studies focuses on functional data analysis, which views data not as discrete arrays but as continuous functions. When these functions are parameterized using neural networks, they are called Implicit Neural Representations (INR). INRs have been successfully applied to represent diverse data types but, to the best of our knowledge, have not been used for encoding decision models. This work addresses the novel challenge of using INRs to represent decision trees. We introduce a tailored coordinate system and train INRs to reconstruct decision trees with a loss function to minimize node reconstruction errors. We benchmark implicit neural decision trees on several datasets, showing that they can effectively represent individual trees, and show potential extensions to tree forests through meta-learning.

1 Introduction

Representation learning for structured data is currently one of the most studied topics in Machine Learning (ML) and Artificial Intelligence (AI). Various subfields of ML aim to create representations of complex data types such as sequences, trees, and graphs, enabling their use in various unsupervised and supervised tasks [1]. Traditionally, signal data has been represented as arrays through discrete coordinates: for instance, time series are vectors recorded over time, while images are matrices of pixel intensities. Recently, the concept of modeling data as functions has gained traction in functional data analysis [2], shifting the focus from data to so-called *functa*, i.e., mappings treated as instances within a dataset [3]. For example, a time series can be thought of as a function from the timestep to its value, whereas an image can be represented as a mapping from 2D pixel coordinates to RGB values. When a Neural Network (NN) parameterizes such a function, it is called an Implicit Neural Representation (INR) [4]. INRs have broad applicability across complex modalities such as audio, videos, 3D scenes, and even arbitrary topological spaces [5]. This functional representation offers advantages over array-based representations, including memory efficiency and summarization of diverse data modalities through a unified architecture.

*Supported by: Fondo Italiano per la Scienza FIS00001966 *MIMOSA*, ERC-2018-ADG G.A. 834756 *XAI*, G.A. 101070212 *FINDHR*, EU NextGenerationEU programme, PNRR-PE-AI FAIR, PNRR-SoBigData.it - Strengthening the Italian RI for Social Mining and Big Data Analytics - Prot. IR13. †*Authors contributed equally, first author chosen by coin toss.*

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Here, we address the novel challenge of representing predictive models by investigating whether INRs can represent classifiers as parameters of an NN. In particular, we focus on decision trees [6] due to their highly interpretable structure, where inner nodes represent feature-based split, branches correspond to split conditions, and leaf nodes indicate final decisions. This transparent decision-making format allows users to trace how specific inputs lead to outputs. Hence, representing decision trees as functional data enables a unified representation for diverse downstream tasks, particularly generative induction [7]. This approach could facilitate scalable model generation while preserving the transparency and interpretability of decision trees. As a first step toward this goal, we evaluate whether decision trees can be represented using INRs. To this aim, we define Implicit Neural Decision Trees (INDTs) by introducing a tailored coordinate system as input to the INR, which is trained to reconstruct the decision tree via a loss function, minimizing the node reconstruction error. INDTs are benchmarked by reconstructing decision trees from several datasets. Moreover, we propose alternatives for representing ensembles of such trees through meta-learning [8].

2 Setting The Stage

Implicit Neural Representations. A signal is seen as a function $x\colon \mathcal{C} \to \mathcal{X}$, mapping coordinates $c \in \mathcal{C}$ to feature values $x(c) \in \mathcal{X}$. Therefore, the signal is the collection of pairs, $\{c, x(c)\}_{c\in\mathcal{C}}$. Representing a signal through an INR requires to learn f_θ, such that $f_\theta(c) = \hat{x}$, where θ are the parameters of a NN [3]. For example, in the image domain, c are the pixel coordinates, i.e., its location, $x(c)$ its corresponding pixel value, i.e., its color and intensity in the RGB case. In this setting, each INR is trained on a signal, e.g., a single picture, to minimize a given loss: $\min_\theta \sum_c \mathcal{L}(x(c), f_\theta(c))$. In simple terms, the INR is trained to reproduce data given its coordinates, e.g., an image from its pixel coordinates. In a way, the INR creates this mapping by "overfitting" on a single instance.

Typically, f_θ is parameterized by a multilayer perceptron with positional encodings or, more commonly, with sinusoidal activation functions (SIRENs) [4], which have proven to be better for fitting high-frequency signals. Given a collection of *functa*, training a different function for each signal becomes increasingly computationally expensive. Various meta-learning approaches have been applied to provide an easier way to approximate collections of signals [3]. In the experiments, we will focus in particular on α-MAML, an adaptive version of MAML [8] incorporating an online hyperparameter adaptation scheme that auto-tunes itself to find a common network initialization, i.e., a meta-model, such that each INR can be fitted in a few gradient steps to the tasked signal. For example, given a dataset of images, this approach would provide a base NN, such that when fine-tuning on a single image, only a few adaptation steps are required to achieve good reconstruction. Besides images, INRs are commonly employed to functionally represent diverse data types, and even arbitrary topological spaces [5]. However, to the best of our knowledge, INRs have never been used to represent decision models, such as decision trees.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Decision Trees. A Decision Tree (DT) classifier is an interpretable predictive model representing decisions through nodes and branches [6]. Formally, a DT, x, can be defined as a set of m nodes, i.e., $x = \{x_i\}_{i=1}^{m}$. There are two types of nodes: inner nodes and leaf (or outer) nodes. Inner nodes contain the index of the feature $a_i \in \mathbb{N}$ and the threshold $t_i \in \mathbb{R}$, which represent the split condition used to route input data through the tree, i.e., $x_i^{\text{in}} = (a_i, t_i)$. Instances are routed through the inner nodes, starting from the root. At each inner node, a split condition $a_i \leq t_i$ is evaluated, directing the instance to one of its child nodes until it reaches a node with no children, i.e., a leaf. A leaf node contains information about $y_i \in \mathbb{N}$, which is the index of the majority class of instances in the training set that ended up in that specific leaf, i.e., $x_i^{\text{out}} = (y_i)$.

Tree induction algorithms commonly use a top-down greedy search for possible splits [6]. In contrast, alternative approaches include evolutionary algorithms or deep learning-based methods such as autoencoders, which can generate decision trees by discovering latent representations [7]. Regardless, representing the structure of a DT in a tabular format remains a significant challenge. In [9], the *Jankowski* encoding was proposed, which represents a tree structure as a $2 \times (2^{d+1} - 1)$ matrix, where each row of the matrix contains node-specific information, and d is the depth, i.e., the number of edges from the root to the deepest leaf. The first row stores the split features a_i for internal nodes or *null* for leaf nodes. The second row holds either the split thresholds t_i or the classification labels y_i. Tree navigation follows a breadth-first approach, with the i-th node in column i and its children in columns $2i$ and $2i + 1$.

3 Implicit Neural Decision Trees

In this paper, we propose Implicit Neural Decision Trees (INDT), a framework for representing decision trees using INRs. Consequently, we need to tackle the problem of representing DTs as functional data. The first step when designing an INR is to define the coordinate system, and the feature value the INR should map to. These choices are obvious for data such as images, where it is natural to identify pixel location using a grid lattice. However, for DTs, there is no canonical setting. Here, we propose using the nodes in the DT as a target of the INR, i.e., we consider the nodes $\{x_i\}_{i=1}^{m}$ of a tree in the same way as the pixels of an image, with the INR trying to reconstruct the node based on its coordinates.

To design a suitable coordinate system, we draw inspiration from the *Jankowski* encoding [9], which indexes nodes starting from the root using a breadth-first search (BFS). However, this coordinate system is suboptimal for our case, as it treats a DT like a sequence and lacks explicit information about node depth. To overcome this limitation, we introduce a binary coordinate system where the BFS index of a node, i_{BFS}, is converted to its binary representation: $i_{\text{INDT}} = bin(i_{\text{BFS}})$. Then, each binary digit becomes an entry of the coordinate vector $c \in \{0, 1\}^{d+1}$, where d is the tree's depth. For example, as illustrated in Figure 1 (left), the root node with $i_{\text{BFS}} = 1$ is represented as 001, its left child, $i_{\text{BFS}} = 2$, becomes 010, and so on. This coordinate system introduces leading zeros in the binary

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

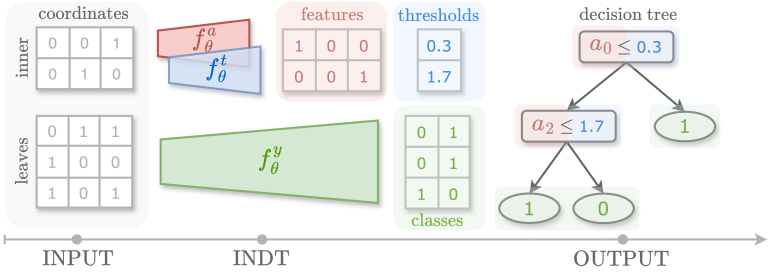

Fig. 1: Schema of our proposal: tree node coordinates are the input of three functions within the INDT to reconstruct feature and threshold arrays for inner nodes, and classes for leaf nodes, enabling decision tree representation.

representation, which expose direct information about the depth of the node. The INDT can leverage this information to improve its ability to represent DTs. Furthermore, unlike [9], we encode features and classes as one-hot vectors. An example of our feature, threshold, and class representation is shown in Figure 1.

Given the different domains of these representations, it would be challenging to model these outputs with a single parametrization. Therefore, the INDT is composed of three functions, $f^a_\theta, f^t_\theta, f^y_\theta$, taking binary coordinates as input and returning one output each: the predicted feature, $f^a_\theta(c) = \hat{a}$, the threshold, $f^t_\theta(c) = \hat{t}$, and the class, $f^y_\theta(c) = \hat{y}$. For inner nodes, we need to minimize both the feature reconstruction error, $\ell^a(a, \hat{a})$, measured with cross-entropy loss, and the threshold reconstruction error, $\ell^t(t, \hat{t})$, measured with *mse*. For leaf nodes, we need to minimize the cross-entropy $\ell^y(y, \hat{y})$. The weighted sum of the losses gives the total loss for the whole tree:

$$\mathcal{L} = \sum_{i=1}^{m} \mathbb{1}_{\text{inner}(i)} \left(\beta \ell^a_i + \gamma \ell^t_i\right) + \mathbb{1}_{\text{leaf}(i)} (\delta \ell^y_i), \tag{1}$$

Once trained, the INDT can sequentially process tree coordinates and get the encoded tree output, which in turn is decoded using the reverse *Jankowski* encoding. A set of INDTs forms an Implicit Neural Forest. This forest can be modeled either as many separate INDTs or as a more compact meta-model of INDTs. Clearly, we can expect greater computational complexity in the latter than the former, but also more generalization capabilities. Note that the functional representation of the INDT enables us to look at the trees through the lens of the parameters θ of the learned NNs. In the case of separate INDTs, the obtained representation is the collection of the parameters of each NN: the representation space is the product of the single-tree INDT parameter spaces. On the other hand, in the case of a meta-model, the parameter of a single NN represents the entire forest or, if the forest is a representative sample, a distribution of trees. Trees in the forest share the same representation space, enabling adaptation in a few gradient steps, the generation of new trees from the common distribution, and the summarization and analysis of the forest within the unique meta-model.

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

		bank	breast	cars	glass	pima	wine
orig	RF$_{acc}$	1.00±0.00	0.98±0.02	0.96±0.02	0.77±0.05	0.70±0.02	0.61±0.01
	DT$_{acc}$	0.97±0.00	0.95±0.01	0.86±0.01	0.67±0.04	0.65±0.01	0.52±0.01
single	*loss*	0.53±0.00	0.85±0.00	0.90±0.00	1.21±0.00	0.81±0.01	1.41±0.01
	RF$_{acc}$	<u>0.98</u>±0.01	0.99±0.02	<u>0.86</u>±0.04	0.65±0.07	0.66±0.02	<u>0.52</u>±0.04
	DT$_{acc}$	0.94±0.02	0.94±0.01	0.85±0.01	0.56±0.07	0.64±0.01	0.46±0.01
α-maml	*loss*	1.18±0.03	1.92±0.02	2.11±0.06	2.81±0.06	1.83±0.02	3.27±0.01
	RF$_{acc}$	0.97±0.01	<u>0.98</u>±0.01	0.74±0.02	<u>0.67</u>±0.07	<u>0.72</u>±0.05	<u>0.52</u>±0.01
	DT$_{acc}$	0.88±0.02	0.95±0.00	0.72±0.00	0.52±0.05	0.68±0.02	0.47±0.00

Table 1: Loss, RF accuracy, and DT accuracy for 10 trees of depth 5. (top) Average performance of the original model; (bottom) average performance of the reconstructions. The closest RF accuracy to the original is underlined.

4 Experiments

We assess here the ability of INRs to correctly represent and reconstruct DTs and forests of DTs. We selected six tabular datasets commonly used when experimenting with novel DTs approaches [7]. We first generate the trees by training three `scikit-learn` Random Forest (RF) classifiers consisting of 10 trees each with maximum depth $d = 5$, and assess the performance on the test set of the RF, and the underlying DT estimators. Then, we build both individual INDTs trained separately on each tree, and meta-INDTs trained using α-MAML, and benchmark them on the test set. The individual INDTs and meta-learning are implemented using the `jax` and `learn2learn` libraries[1]. We set the network structure to a SIREN model [4], with one hidden layer containing 32 units. As optimizers, we use ADAM with learning rate $lr = 1\text{e}{-3}$ for individual SIRENs, and meta-SGD with $lr = 1\text{e}{-5}$ in α-MAML, with 3 adaptation steps. Further, we set the loss weights β, γ, δ to 1. Each model is trained for 10 000 epochs.

The results presented in Table 1 provide insights into the reconstruction performance of DTs and RF when implemented using individual INDTs or meta-INDTs. The first two rows report the performance of the original models. The benchmark evaluation focuses on two aspects: *(i)* reconstruction loss, measuring how well INDTs or meta-INDTs reconstruct trees, and *(ii)* test accuracy of the reconstructed DTs and RFs. Accuracy serves as a proxy for assessing reconstruction fidelity, as reconstructed models should ideally match the performance of the original models. Overall, the results indicate that small deviations reflect consistent performance in both accuracy and loss metrics. Training separate INDTs for each DT usually leads to lower reconstruction losses than meta-INRs trained with α-MAML. This is expected, as the generalization goal of meta-learning introduces complexity, making precise reconstruction more challenging. Notably, lower reconstruction losses do not always correlate with better accuracy, highlighting the need to distinguish loss minimization from the practical accuracy of the reconstructed models. The decision thresholds of inner nodes, derived from the gradients of

[1]Code available at: `https://github.com/fismimosa/indtree/`

ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

the loss function ℓ^t, influence node split precision without altering tree structure. Reconstruction quality depends on calibrating these thresholds, not just on loss reduction. We hypothesize that finetuning the loss weights could improve the accuracy of the reconstructed tree and forest, although this is beyond our scope, and we leave it for future experiments. This sensitivity to multi-output loss is corroborated by preliminary experiments showing that a single parametrization of the outputs leads to a more unstable optimization than keeping separate NNs. In summary, while individual INDTs demonstrate better performance regarding reconstruction loss, meta-learning with α-MAML shows competitive accuracy.

5 Conclusion

We have introduced Implicit Neural Decision Trees (INDT), a framework for representing decision trees using Implicit Neural Representations (INRs). Our findings show that INRs can effectively model DTs, also extending to Random Forests through meta-learning. Thus, we have empirically demonstrated the potential of INRs as a functional representation of DTs. A limitation of this study is the computational complexity of meta-learning with current libraries, which restricted testing on larger datasets and deeper DTs. Nonetheless, we believe our proposal provides a promising unified representation for DT classifiers. For future work, we propose exploring alternative coordinate systems, such as hyperbolic spaces, to better capture the hierarchical properties of DTs. Additionally, we aim to leverage these functional representations to induce interpretable DTs in a generative and scalable manner and to evaluate them against traditional and neural-based ensembles, enhancing their utility in ML applications.

References

[1] Benjamin Paaßen, Claudio Gallicchio, Alessio Micheli, and Alessandro Sperduti. Embeddings and representation learning for structured data. In *ESANN*, 2019.

[2] Jan Gertheiss, David Rügamer, Bernard XW Liew, and Sonja Greven. Functional data analysis: An introduction and recent developments. *Biometrical Journal*, 66(7), 2024.

[3] Emilien Dupont, Hyunjik Kim, S. M. Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In *ICML*, volume 162 of *PMLR*, pages 5694–5725. PMLR, 2022.

[4] Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020.

[5] Daniele Grattarola and Pierre Vandergheynst. Generalised implicit neural representations. In *NeurIPS*, 2022.

[6] Leo Breiman, J. H. Friedman, Richard A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[7] Riccardo Guidotti, Anna Monreale, Mattia Setzu, and Giulia Volpi. Generative model for decision trees. In *AAAI*, pages 21116–21124. AAAI Press, 2024.

[8] Harkirat Singh Behl, Atilim Günes Baydin, and Philip H. S. Torr. Alpha maml: Adaptive model-agnostic meta-learning. In *6th ICML Workshop on AutoML*, 2019.

[9] Dariusz Jankowski and Konrad Jackowski. Evolutionary algorithm for decision tree induction. In *CISIM*, volume 8838 of *Lecture Notes in Computer Science*, pages 23–32. Springer, 2014.