Weightless Neural Trees

Leandro Santiago de Araújo[1]

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)
Niterói – RJ – Brazil

leandro@ic.uff.br

Abstract. *This work introduces Weightless Neural Tree (WNT), a neuro-symbolic model based on the fusion of Decision Tree (DT) and Weightless Neural Network (WNN). A new learning algorithm is proposed to extract rules learned by a DT and design flexible WNN architectures based on tree discriminator generation and binary conditional mapping. Single-depth, boundary, and neighborhood mappings are also proposed to optimize the capture of inter-partition patterns. Finally, the WNT model is extended in the Weightless Neural Forest (WNF) with the addition of two ensemble methods that utilize rules learned by a Random Forest (RF). Experiments on ten classification datasets evaluate accuracy, F1-score, and model size under different hyperparameter configurations. WNT substantially reduces model size relative to the evaluated DT and WiSARD baselines, with reductions of up to 96%, while achieving competitive accuracy on most datasets. WNF further improves the accuracy of WNT and achieves up to 96.25% lower model size than the evaluated Random Forest baseline, while remaining competitive with Random Forest and XGBoost on the evaluated datasets. These results indicate that tree-guided architecture design is a promising direction for memory-efficient weightless neural models.* https://github.com/leandro-santiago/wnt

1. Introduction

The integration of decision trees (DTs) with artificial neural networks (NNs) has been widely investigated as a strategy to reconcile interpretability and high predictive performance by consolidating models known as Neural Trees (NTs) [Li et al. 2025]. Both are popular machine learning models that operate on widely different paradigms, namely connectionist (NN) and symbolic (DT), with their main difference in the form of knowledge learning. The combination of these models aims to explore their complementary characteristics, using decision trees to structure the decision-making process, select relevant attributes or provide interpretable explanations, while neural networks improve the generalization capacity and accuracy of the system. Thus, hybrid models based on DT and NNs have demonstrated the potential to produce more robust, efficient and reliable solutions to a broad spectrum of real-world problems, encompassing classification, regression [Tanno et al. 2019], computer vision [Tanno et al. 2019], and natural language processing [Isonuma et al. 2020].

Weightless Neural Networks (WNNs) [Ludermir et al. 1999] are connectionist models that perform pattern recognition and perceptual learning that replace weighted connections with memory-based lookup tables (LUTs). Inspired by n-tuple and RAM-based architectures [Santiago et al. 2020], WNNs store input–pattern associations di-

rectly in discrete memory structures, enabling fast training, low computational complexity, and efficient hardware implementation. Unlike deep neural networks trained through gradient-based optimization, WNNs do not rely on multiplications or weighted accumulations. Instead, each neuron, called a RAM node, concatenates binary inputs to form an address in an internal LUT and outputs the binary value stored at that address. Recent advances have demonstrated the potential of WNNs to achieve high efficiency in neural computing by adapting multi-pass gradient-based [Susskind et al. 2023] and multi-layered learning strategies [Bacellar et al. 2024], but the model's good performance depends on the choice of the input size n of each LUT, which directly impacts the amount of memory required by each neuron. Furthermore, the hyperparameter n is defined empirically and fixed for all neurons in the model.

To overcome the inherent limitations of traditional weightless architectures, this work introduces the Weightless Neural Tree (WNT) (Figure 1), a Neuro-Symbolic model that combines DT with WNN, addressing strategies on how to use the hierarchical structure learned by a decision tree to design a weightless neural network. The decision tree is not used as an inference component of WNT. Instead, its learned partitions are transformed into an architectural specification for the WNN. The specific contributions of this article are as follows:

1. WNT, a novel neuro-symbolic model that combines rules learned by a DT to enhance the construction of WNNs, allowing the model's discriminators to have different numbers of neurons and tuple sizes. A new learning algorithm based on tree discriminator construction and binary conditional mapping is introduced. Furthermore, three conditional mapping optimizations are proposed: single-depth mapping, boundary mapping, and neighborhood mapping.
2. Weightless Neural Forest (WNF), an extension of the WNT model that uses an ensemble method to combine the predictions of a set of WNT models. The rules used in the model are learned by a Random Forest, and two algorithms are used to aggregate the predictions of the submodels: majority voting and additive submodeling.
3. Comparison of WNT and WNF with state-of-the-art WNN, DT, and RF models in terms of accuracy, performance and memory size.

The remainder of this paper is organized as follows. Basic concepts related to this work are presented in Section 2, providing a foundational understanding of Decision Tree and Weightless Neural Networks. Section 3 introduces the details of the novel Weightless Neural Tree. Section 4 describes the experiments and results, evaluating the performance of WNT against already established state-of-the-art models. Finally, Section 5 concludes the paper, discussing the potential future work.

2. Basic Concepts

2.1. Decision Tree

Decision Tree (DT) [Rana et al. 2014] is a non-parametric supervised learning model that builds a hierarchical decision structure from data by recursively partitioning the feature space based on purity criteria (such as entropy, information gain, or Gini index) in order to maximize the homogeneity of the samples in each partition. To construct a DT, a training set composed of feature vectors and their respective class labels is partitioned

by learning hierarchical data groupings using internal (or decision) nodes and leaf (or terminal) nodes connected by branches, through a three-step procedure: splitting nodes, determining which nodes are leaves, and assigning class labels to the leaf nodes. The most important and time-consuming step is splitting the nodes.

Each internal node contains a routing function of type $x_j \leq b_i$ (is the feature x_j less than or equal the threshold b_i?) to choose which child node to visit next, while a leaf node is associated with a class label. The prediction of a DT is given by traversing a path from the root to a leaf according to the sequence of true tests of the internal nodes. In this way, DTs grow recursively and greedily based on criteria related to informativeness. The most representative DT algorithms are CART (Classification and Regression Trees), ID3 and C4.5, with CART being the most used for ensemble strategies based on combining multiple decision trees, such as Random Forest (RF) and XGBoost [Rana et al. 2014], to increase performance at the expense of interpretability.

2.2. Weightless Neural Network

Weightless Neural Networks (WNNs) [Ludermir et al. 1999] are memory-based machine learning models where each neuron is defined by a RAM node or lookup tables (LUTs) with binary values, which differ from traditional neural networks where neurons react to input by calculating adjustable synaptic weights or weighted sum operations. Instead, their learning occurs through the explicit storage of patterns in their neurons, associating combinations of inputs with previously observed outputs. As a neuron is a RAM node with n bits of input, it is capable of learning 2^n patterns, called tuples, selected by a pseudo-random mapping of a binary input containing all the features of a given sample.

WiSARD (Wilkie, Stoneham and Aleksander Recognition Device) is the pioneering commercially distributed WNN [Aleksander et al. 1984] that provides a simple and efficient implementation, allowing the deployment of learning resources in embedded and real-time systems. WiSARD model is based on multi-discriminators to deal with M-class classification problem represented by M discriminators. A discriminator is composed of a set of N RAM-based neurons indexed by N tuples with size of n bits. Given an I-input model with n-bit tuple size, there are $N = I/n$ RAM nodes with total of 2^n addresses per discriminator, and therefore a total of $MN2^n$ learned patterns. Once the input is binarized, it is assigned to RAM nodes of a discriminator through a pseudo-random mapping and is submitted for the training or inference phase. During training, inputs are sent only to discriminators corresponding to their known classes, and the activated inputs of their RAM nodes are updated to 1. During inference, inputs are sent to all discriminators, and their responses are calculated by summing activated locations. The predicted label of the model is determined as the class of the discriminator with the highest response.

Thus, samples identical to the training data will map addresses of all RAM nodes with output 1. On the other hand, samples that are not identical to the training data will map RAM node addresses with output 0 for unlearned patterns and output 1 for input patterns shared with examples learned in training. Therefore, although many RAM nodes produce 0, some should still produce 1, allowing WiSARD to generalize to new data. The prediction will be correct if the correct discriminator has a larger number of RAM nodes that produce 1 than any other discriminator. The tuple size is a crucial hyperparameter that should normally be chosen by trial and error. Small values of n restrict the complexity

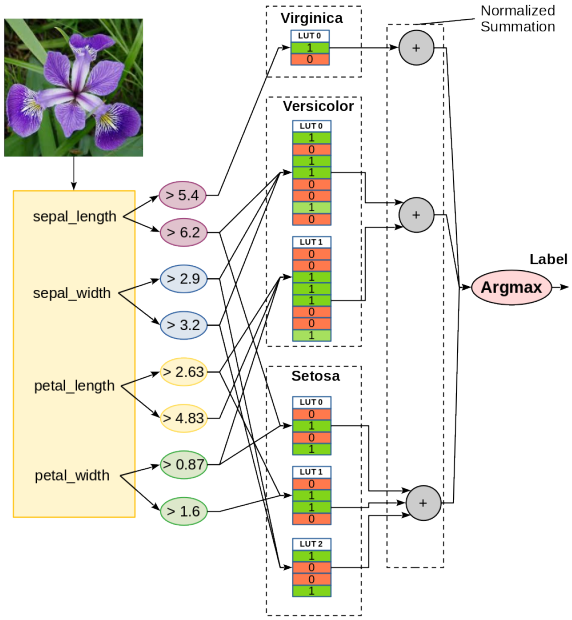


Figure 1. Simple WNT for the Iris dataset [Unwin and Kleinman 2021]. WNT performs computation using different numbers of lookup tables (LUT-1, LUT-2s, and LUT-3s, in this example) per discriminator, guided by a decision tree model T. The inputs are binarized using a unary "thermometer" encoding, conditionally mapped into tuples according to the partition space of T. The outputs of the LUTs are summed and normalized according to the number of neurons in each discriminator.

of the patterns the model can learn, which can improve the model's generalization ability but may impair overall accuracy due to RAM node saturation, where all their positions are filled with 1. Large values of n produce more specialized behavior but can also result in overfitting to the training data, as the model memorizes more complex patterns from the data instead of useful common patterns. The exponential relationship between n and the total size of the model parameters also restricts the range of viable options due to the amount of memory required by the model.

3. Weightless Neural Tree Design

The WNT model uses the rules learned from a decision tree T to select the range of values for relevant features to design the architecture of a weightless neural network as illustrated in Figure 1. The first step is to generate a tree discriminator to identify the minimum and maximum values of each feature associated by labels. Once the tree discriminator is obtained, it is used to perform conditional tuple mapping, which selects the relevant bits of each feature and determines the tuple size according to the maximum label depth achieved by T. Thus, each discriminator with label l of the designed WNN model will have the following characteristics: (i) tuple mapping considering only the most relevant bits of the features according to l, (ii) tuple size related to l, and (iii) number of neurons related to l.

The training of the WNT model follows the same procedures as a WNN discussed in Section 2.2, with the difference that each discriminator will have different tuple sizes and number of neurons. The WNT model inference adapts the traditional WNN model

by performing a normalized sum as the response for each discriminator. This is necessary because each discriminator now has a varying number of neurons, and to maintain uniform relevance among the discriminators, their response indicates the percentage of active neurons for a test sample.

3.1. Tree Discriminator

The main idea of the WNT model arose from the proposal to store the leaves corresponding to a class in a LUT. The path of a class can be formally represented by a logical conjunction (AND) between the nodes that make up that path, or the logical edges that connect the nodes of the path. In turn, the union of all paths can be obtained by the logical disjunction (OR) of all valid paths. Formally, if each path p_i is defined as a conjunction of nodes ($n_1 \wedge n_2 \wedge ... \wedge n_k$), then the class C can be expressed as $C = p_1 \vee p_2 \vee ... \vee p_m$. The total number of possible paths can be defined as the number of leaves of a perfect binary tree with height h, which is 2^h. This formulation allows the model to preserve the tree structure through well-defined logical operators, enabling association with a truth table with 2^h addresses, where each path results in the value 1 in the table. Based on this analogy, the tuple size in class C is defined as the height of the subtree representing C. Thus, the tree discriminator does not reproduce the complete decision tree. It summarizes, for each class, the feature intervals and maximum path depth required to represent the class partitions in the WNN.

Algorithm 1 presents the step-by-step procedure for generating the tree discriminator based on Depth-First Search (DFS). Based on the routing function response of each internal node, it is possible to delimit the partition of each feature. Let $x_i \leq b_i$ be the routing function; a true response indicates that x_i has the maximum value b_i, while a false response indicates that x_i has the minimum value b_i. Along the path of a tree, the feature x_i can be tested multiple times, defining new minimum and maximum values. Finally, the tree discriminator will contain the height (line 6) and the union of all partitions for each class subtree, by determining the maximum (line 10) and minimum (line 12) values of each feature x_i.

3.2. Binary Conditional Mapping

The linear thermometer [Bacellar et al. 2022] is the most widely used technique in WNN models to convert floating-point input values into binary format. The range of values for a feature is divided into equal parts, and each part receives a corresponding code. It is based on a unary code using more 1 bits as the feature value approaches its maximum value. Similarly, the closer to the minimum value of the range, the fewer 1s and the more 0 bits the representation will have.

Since the number of bits equal to 1 in the thermometer preserves an order relation of successor numbers, the ranges of partitions in the tree discriminator can be mapped directly in binary space, as illustrated in Figure 2. Let n be the number of bits in the thermometer code, n_i and n_j the number of 1 bits for binary representation of the values b_i and b_j, respectively, there are 3 possible cases of a partition:

1. if $x_i \leq b_i$, the valid bit range of $x_i = [1, n_i]$;
2. if $x_i > b_i$, then valid bit range of $x_i = [n_i, n]$;
3. if $b_i < x_i \leq b_j$, then valid bit range of $x_i = [n_i, n_j]$.

Algorithm 1 Tree Discriminator Generation

```
Input: Trained tree T from Decision Tree model
Output: Tree discriminator D
procedure GEN_TREE_DISCRIMINATOR(T, node, depth)
    if T[node] is leaf then
        l ← T[node].label
        D[l].depth ← max(D[l].depth, depth)
        for f in feature_stack do
            t ← f.threshold
            if f.branch is LEFT then
                D[l][f].max ← max(D[l][f].max, t)
            else if f.branch is RIGHT then
                D[l][f].min ← min(D[l][f].min, t)
            end if
        end for
        return
    end if

    feature_stack.push(T[node].threshold, LEFT)
    gen_tree_discriminator(T, T.left[node], depth+1)
    feature_stack.pop()
    feature_stack.push(T[node].threshold, RIGHT)
    gen_tree_discriminator(T, T.right[node], depth+1)
    feature_stack.pop()
end procedure
```

After selecting the valid bit range for each feature x_i, they are concatenated to form the binary input to be submitted to the WNT model. Each discriminator has its own valid bit ranges and tuple mapping. A special case is the conditional mapping of a class subtree with height 1, called **single-depth mapping**. In this case, its discriminator selects only one feature and is optimized to have a single bit stored in a single neuron with two possible cases. If $x_i \leq b_i$, then it is sufficient to check if the bit $n_i + 1$ (first bit has index 1) is equal to 0. If $x_i > b_i$, then it is sufficient to check if the bit n_i is equal to 1. Furthermore, two more conditional mappings were developed, called boundary and neighborhood mapping.

Boundary mapping select only the two bits closest to the partition boundaries. If $x_i \leq b_i$, then bits n_i and $n_i + 1$ are selected. Bit n_i represents the maximum value of b_i, that is, it will always be 1 when $x_i = b_i$. In this way, it is possible to differentiate values greater than b_i, which will have the value 1 in both bits. Values less than b_i will always have bit $n_i + 1 = 0$. Similarly, if $x_i > b_i$, then bits n_i and $n_i - 1$ are selected. Thus, values less than b_i are detected if bit $n_i = 0$.

Neighborhood mapping select the neighboring bits of the threshold. In this mapping, all thresholds (internal nodes) are used instead of the partition union range. For every threshold b_i, 3 bits are selected: n_i, $n_i + 1$ and $n_i - 1$. In this way, it is possible to

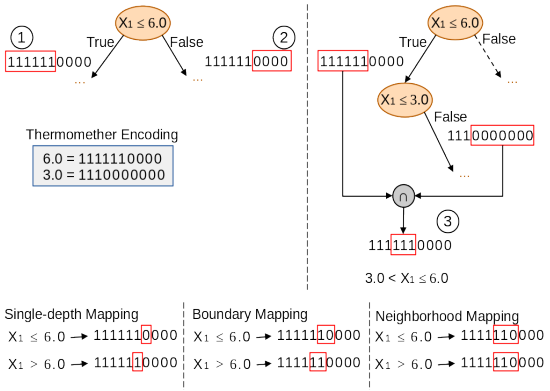


Figure 2. Binary partition mapping of thermometer encoding. Feature values are encoded with 10 bits using thermometer encoding, and valid ranges are defined according to the 3 possible cases. Single-depth mapping corresponds to a class subtree with height 1. Boundary and neighborhood mappings are optimizations to differentiate patterns within and outside partitions. Valid ranges are within the red-bordered block.

differentiate the pattern of values outside the threshold range.

3.3. Weightless Neural Forest

Weightless Neural Forest (WNF) is an ensemble model that combines multiple predictions from WNTs to generate a more robust and reliable result. WNF uses the rules learned by a Random Forest to select the range of values for the relevant features and build an ensemble of weightless neural trees. Therefore, each WNT learns distinct data partitions, enabling flexibility in both the tuple size and the number of neurons assigned to each discriminator. Two ensemble aggregation strategies were implemented. The first is majority voting, in which the class receiving the highest number of votes among the trees is selected as the final prediction. The second strategy is the additive submodeling, proposed by [Susskind et al. 2023], which sums the response scores for each discriminator in all submodels before calculating the final prediction.

3.4. Limitations

WNT model relies on rules learned for all classes in a DT, which is determined by its path to each leaf of the class. However, there are cases where a DT does not have leaves corresponding to all classes in the dataset. In this case, if no leaf is found for a given class, then a discriminator is generated that receives the complete binarized sample, and the tuple size is determined by the height of the tree.

4. Experiments and Results

To demonstrate the performance of WNTs and WNFs, this work compares both with the Decision Tree, Random Forest, XGBoost, and WiSARD models.

4.1. Experimental Setup

The experiments were conducted on a desktop server equipped with Intel® Core™ i7-1070 (8C/16T) @ 2.9 GHz (x86_64) processor with 16 GB DDR4 RAM running Ubuntu Linux. All experiments were implemented in Python, and the complete source code is

publicly available[1] for reproducibility. Decision Tree, Random Forest and XGBoost models were implemented using the Scikit-learn library, employing default hyperparameter settings (random_state=42) with a total of 100 estimators and an objective function with "multi:softprob" for the ensemble models. The maximum depth hyperparameter was evaluated in different configurations. WiSARD, WNT and WNF models were implemented using the PyTorch framework considering each neuron implemented as dictionary. Since the work focuses on validating WNT, the dictionary structure was not optimized in C++, with each entry requiring 5 bytes (4 bytes for the key and 1 byte for the value). Thus, the reported size represents the logical storage of the Python dictionary rather than the memory footprint of an optimized hardware implementation.

4.2. Dataset

Table 1. Hyperparameters swept for all datasets.

Dataset Name	Max Depth	Tuple Size	Bits Encoding
MNIST	[10, 30]	28	[4, 20]
FashionMNIST	[10, 30]	28	[4, 20]
Ecoli	[3, 10]	20	[4, 40]
Iris	[3, 10]	20	[4, 40]
Letter	[10, 30]	28	[4, 20]
Satimage	[3, 10]	20	[4, 30]
Shuttle	[3, 30]	20	[4, 40]
Vehicle	[10, 30]	20	[4, 40]
Vowel	[5, 30]	20	[4, 30]
Wine	[5, 30]	8	[4, 30]

A total of 10 datasets were chosen, 9 selected from work [Santiago et al. 2020] and the Fashion MNIST dataset. For each dataset, a hyperparameter sweep was performed related to the maximum tree depth, tuple size, and number of bits for thermometer encoding to identify the best configuration for the WNT model, according to the intervals presented in Table 1. Only MNIST and Fashion MNIST have separate training and test datasets. For the other datasets, the data was divided into training sets (70% of the samples) and test sets (30% of the samples).

4.3. Accuracy and Model Size for Weightless Neural Trees

All results are reported as the mean over 10 runs. Four mapping strategies were evaluated for WNT: partition (includes all bits between the intervals of the route functions), partition+boundary, boundary and neighborhood. Table 2 compares the accuracy, F1-score and model size of WNT with those of DT and WiSARD.

The results demonstrate that WNT can substantially reduce model size, although this reduction may come at the cost of predictive performance. The accuracy and F1-score results indicate that WNT performance is strongly dependent on the quality and granularity of the partitions induced by the underlying decision tree. This dependency is

[1]Github repository: `https://github.com/leandro-santiago/wnt`

Table 2. Accuracy, F1-score, and memory results for WNT. The WNT mapping strategy with the highest accuracy is reported in parentheses.

Dataset	Model	Max Dep	Bits /Input	Accuracy	F1	Size (KiB)	Stats. signif.
MNIST	DecisionTree	28	-	87.91 ± 0.0	87.75	**942.38**	
	WiSARD	-	4	**94.33** ± 0.22	94.32	4475.98	✓
	WNT (boundary)	28	4	91.71 ± 0.0	91.6	3082.44	✓
F-MNIST	DecisionTree	28	-	79.43 ± 0.0	79.46	**1164.88**	
	WiSARD	-	4	**81.59** ± 0.22	81.42	9866.16	✓
	WNT (boundary)	28	4	80.44 ± 0.0	80.38	4910.95	✓
Ecoli	DecisionTree	20	-	77.95 ± 3.41	47.35	8.84	
	WiSARD	-	20	**79.39** ± 3.65	50.44	2.64	✗
	WNT (partition)	20	20	72.28 ± 4.7	36.86	**1.25**	✓
Iris	DecisionTree	3	-	94.67 ± 2.27	94.73	0.56	
	WiSARD	-	20	**94.89** ± 3.59	94.49	0.89	✗
	WNT (part_bound)	3	20	92.93 ± 4.22	92.82	**0.26**	✗
Letter	DecisionTree	26	-	**86.73** ± 0.49	86.7	893.05	
	WiSARD	-	15	81.54 ± 0.77	81.74	100.92	✓
	WNT (partition)	26	15	66.59 ± 1.38	66.87	**69.15**	✓
Satimage	DecisionTree	20	-	**85.47** ± 0.69	82.96	70.84	
	WiSARD	-	16	85.45 ± 1.63	83.04	56.93	✗
	WNT (neighbor)	20	16	79.86 ± 1.55	77.37	**26.06**	✓
Shuttle	DecisionTree	19	-	**99.81** ± 0.02	74.96	27.5	
	WiSARD	-	35	87.33 ± 2.06	40.39	2.6	✓
	WNT (neighbor)	19	35	92.15 ± 2.62	48.89	**1.1**	✓
Vehicle	DecisionTree	15	-	**70.28** ± 1.84	70.43	16.67	
	WiSARD	-	16	67.17 ± 1.58	66.9	12.7	✓
	WNT (neighbor)	15	16	59.21 ± 2.97	58.25	**3.96**	✓
Vowel	DecisionTree	20	-	77.9 ± 2.54	77.82	33.22	
	WiSARD	-	18	**88.83** ± 0.73	88.75	13.76	✓
	WNT (neighbor)	20	18	43.95 ± 6.25	42.55	**5.6**	✓
Wine	DecisionTree	5	-	89.81 ± 4.55	89.88	1.07	
	WiSARD	-	10	**97.22** ± 1.71	97.34	2.46	✓
	WNT (neighbor)	5	10	85.0 ± 8.34	81.94	**0.26**	✗

particularly evident in the Letter, Vehicle, and Vowel datasets, where WNT exhibits substantial gaps in both accuracy and F1-score relative to the other evaluated models. WNT produces a smaller model than the evaluated baselines in most datasets, with MNIST and F-MNIST being the main exceptions when compared with the corresponding DT models. In both datasets, the generated trees have a maximum depth of 28 for all leaves, resulting in a tuple size of 28 for all WNT discriminators. Consequently, the number of stored patterns increases substantially, limiting the memory savings obtained through the tree-guided architecture. Nevertheless, WNT improves upon DT in both accuracy and F1-score, achieving 91.71% accuracy and an F1-score of 91.60% on MNIST, compared with 87.91% and 87.75% for DT. On F-MNIST, WNT achieves 80.44% accuracy and 80.38% F1, compared with 79.43% and 79.46% for DT. Furthermore, WNT reduces the

model size relative to WiSARD by 31.13% on MNIST and 50.22% on F-MNIST, while maintaining relatively competitive predictive performance. In particular, WNT surpasses WiSARD in accuracy on the Shuttle dataset, while achieving a maximum model-size reduction of 96% relative to DT on Shuttle and 89.39% relative to WiSARD on Wine.

Statistical hypothesis tests were conducted to determine whether the differences in accuracy between WNT or WiSARD and DT were statistically significant. The Wilcoxon signed-rank test was used with a significance level of 5%. The results are reported in the last column of Table 2. A check mark indicates that the observed difference in accuracy between the corresponding model and DT is statistically significant at the 5% level.

4.4. Accuracy and Model Size for Weightless Neural Forest

Table 3 compares the accuracy, F1-score, and model size of WNF with RF and XGBoost. Different WNF configurations were evaluated by combining the four WNT mapping strategies with additive (add) and majority voting (maj) aggregation. Overall, WNF improves upon the corresponding WNT models and achieves competitive predictive performance with RF and XGBoost on most datasets. The improvement is particularly evident in datasets where WNT exhibits lower performance, such as Letter, Vehicle, and Vowel, indicating that the ensemble of tree-derived WNTs can mitigate the limitations of a single tree partition. WNF also provides substantial model-size reductions compared with conventional ensembles. The maximum reduction reaches 96.25% relative to RF on Shuttle and 91.73% relative to XGBoost on Ecoli. However, MNIST and F-MNIST represent exceptions, as the WNF models require larger memory footprints due to the maximum tree depth of 28, as discussed in Section 4.3. Despite this increase in model size, WNF improves the accuracy and F1-score of the corresponding WNT models on these datasets.

Statistical hypothesis tests were conducted using the Wilcoxon signed-rank test with a significance level of 5%. As reported in the last column of Table 3, WNF shows statistically significant accuracy differences from RF on all datasets except Wine, while XGBoost differs significantly from RF on MNIST, F-MNIST, Ecoli, Letter, and Vowel. The WNF experiments used the same `max_depth` and `bits/input` configurations as WNT, with a fixed number of 100 estimators. Therefore, the reported memory sizes should not be interpreted as optimized WNF configurations. A dedicated search over the number of estimators and tree depth may yield substantially smaller models while preserving accuracy. Missing classes were observed in the generated trees for Ecoli and Shuttle, with 72 and 2 occurrences, respectively, which may affect class-wise performance. Since the number of estimators and tree depth were not optimized for WNF, further reductions in model size may be possible by adjusting these parameters. Overall, the results indicate that WNF can improve the predictive performance of WNT while retaining its memory-efficiency advantages.

5. Conclusion

This article introduced the novel Weightless Neural Tree (WNT), a neuro-symbolic model that uses the partitions learned by a Decision Tree to design WNN architectures. The proposed tree discriminator and binary conditional mapping mechanisms allow different discriminators to use different tuple sizes and numbers of neurons. Three mapping strategies: single-depth, boundary, and neighborhood mapping, were investigated to further

Table 3. Accuracy, F1-score, and memory results for WNF. The WNF mapping strategy with the highest accuracy is reported in parentheses.

Dataset	Model	Max Dep	Bits /Input	Accuracy	F1	Size (KiB)	Stats. signif.
MNIST	RandomForest	28	-	96.92 ± 0.05	96.9	124162	
	XGBoost	28	-	**97.56** ± 0.0	97.54	**9885**	✓
	WNF (bound_add)	28	4	94.63 ± 0.11	94.6	308632	✓
F-MNIST	RandomForest	28	-	87.64 ± 0.1	87.51	122045	
	XGBoost	28	-	**89.43** ± 0.0	89.34	**14864**	✓
	WNF (bound_maj)	28	4	84.17 ± 0.1	84.0	499996	✓
Ecoli	RandomForest	20	-	**85.24** ± 2.48	53.9	767.38	
	XGBoost	20	-	82.9 ± 2.9	52.95	534.47	✓
	WNF (bound_add)	20	20	82.9 ± 2.24	50.23	**44.2**	✓
Iris	RandomForest	3	-	**95.78** ± 2.32	95.53	60.2	
	XGBoost	3	-	94.22 ± 2.47	94.12	160.36	✗
	WNF (pbound_maj)	3	20	93.94 ± 2.34	93.89	**33.92**	✓
Letter	RandomForest	26	-	**95.88** ± 0.18	95.87	95379.65	
	XGBoost	26	-	95.4 ± 0.26	95.39	7251.20	✓
	WNF (part_add)	26	15	92.71 ± 0.36	92.71	**7017.43**	✓
Satimage	RandomForest	20	-	91.58 ± 0.55	89.58	6752.15	
	XGBoost	20	-	**91.85** ± 0.56	90.01	1602.71	✗
	WNF (bound_add)	20	16	88.78 ± 0.55	85.87	**950.19**	✓
Shuttle	RandomForest	19	-	**99.83** ± 0.02	73.6	3267.83	
	XGBoost	19	-	**99.83** ± 0.02	75.19	1041.38	✗
	WNF (neigh_add)	19	35	97.7 ± 0.08	65.76	**122.37**	✓
Vehicle	RandomForest	15	-	74.57 ± 1.68	73.97	1534.83	
	XGBoost	15	-	**75.08** ± 1.47	74.96	647.61	✗
	WNF (bound_add)	15	16	71.42 ± 0.97	70.36	**187.72**	✓
Vowel	RandomForest	20	-	**94.09** ± 1.2	94.04	3298.39	
	XGBoost	20	-	88.58 ± 1.61	88.58	982.81	✓
	WNF (bound_add)	20	18	84.08 ± 1.84	83.73	**193.16**	✓
Wine	RandomForest	5	-	**98.33** ± 1.75	98.38	116.17	
	XGBoost	5	-	96.48 ± 2.41	96.53	146.94	✓
	WNF (bound_add)	5	10	97.41 ± 2.89	97.46	**18.5**	✗

reduce the amount of stored information. WNT was also extended to Weightless Neural Forest (WNF), which combines multiple tree-guided WNN models using majority voting or additive submodeling. Experiments on ten classification datasets show that WNT can substantially reduce model size while maintaining competitive accuracy on several datasets. The largest reductions reached 96% relative to the evaluated DT and WiSARD baselines. WNF further improved WNT accuracy and achieved up to 96.25% lower model size than the evaluated Random Forest baseline. However, the experiments also show that WNT accuracy is strongly dependent on the partitions produced by the underlying tree, with substantial degradation on some datasets.

These results suggest that tree-guided architecture design is a promising approach for memory-efficient weightless neural models. Future work will investigate improved

tree construction strategies, multi-stage learning to identify more informative partitions, optimized data structures such as Bloom filters, alternative thermometer encodings, and dedicated optimization of WNF ensemble size and tree depth.

Acknowledgements

I would like to thank the *National Council for Scientific and Technological Development (CNPq)* for funding this research through the grant 307770/2025-7.

References

Aleksander, I., Thomas, W., and Bowden, P. (1984). Wisard·a radical step forward in image recognition. *Sensor Review*, 4(3):120–124.

Bacellar, A., Susskind, Z., Villon, L., Miranda, I., Santiago, L., Dutra, D., Breternitz, M., JOHN, L., Lima, P., and França, F. (2022). Distributive thermometer: A new unary encoding for weightless neural networks. pages 31–36.

Bacellar, A. T. L., Susskind, Z., Breternitz Jr., M., John, E., John, L. K., Lima, P. M. V., and França, F. M. G. (2024). Differentiable weightless neural networks. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Isonuma, M., Mori, J., Bollegala, D., and Sakata, I. (2020). Tree-Structured Neural Topic Model. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 800–806, Online. Association for Computational Linguistics.

Li, H., Song, J., Xue, M., Zhang, H., and Song, M. (2025). A survey of neural trees: Co-evolving neural networks and decision trees. *IEEE Transactions on Neural Networks and Learning Systems*, 36(7):11718–11737.

Ludermir, T. B., Carvalho, A. C. P. d. L. F. d., Braga, A. P., and De Souto, M. C. (1999). Weightless neural models: a review of current and past works. *Neural Computing Surveys*, 2:41–61.

Rana, K. K. et al. (2014). A survey on decision tree algorithm for classification. *International journal of Engineering development and research*, 2(1):1–5.

Santiago, L., Verona, L., Rangel, F., Firmino, F., Menasché, D. S., Caarls, W., Breternitz Jr, M., Kundu, S., Lima, P. M., and França, F. M. (2020). Weightless neural networks as memory segmented bloom filters. *Neurocomputing*, 416:292–304.

Susskind, Z., Arora, A., Miranda, I. D. S., Bacellar, A. T. L., Villon, L. A. Q., Katopodis, R. F., de Araújo, L. S., Dutra, D. L. C., Lima, P. M. V., França, F. M. G., Breternitz Jr., M., and John, L. K. (2023). Uleen: A novel architecture for ultra-low-energy edge neural networks. *ACM Trans. Archit. Code Optim.*, 20(4).

Tanno, R., Arulkumaran, K., Alexander, D., Criminisi, A., and Nori, A. (2019). Adaptive neural trees. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6166–6175. PMLR.

Unwin, A. and Kleinman, K. (2021). The iris data set: In search of the source of virginica. *Significance*, 18.