SOFT CLASSIFICATION TREES WITH CONDITIONAL COMPUTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Decision trees are widely used supervised learning models for tabular data. Multivariate deterministic trees provide interpretable decision rules and conditional computation but typically rely on expensive mixed-integer optimization, whereas soft trees are amenable to continuous optimization at the cost of losing single-path routing and conditional computation. We propose a soft classification tree that restores single-path routing and conditional computation while retaining the favorable properties of continuous optimization. Our approach relies on a new activation function for routing samples at branch nodes, defined as a novel scaled variant of the negative log-likelihood that combines its benefits with those of the sigmoid, is differentiable almost everywhere, and is proven to be classification-calibrated. The resulting model is suitably trained using a projected adaptation of the Adam optimizer to handle class assignment constraints. Unlike prior differentiable trees, our method needs only a single first-order run, with no scaling, annealing, warm start, or multi-start. Experiments on 61 classification datasets show that our single-tree method is better than all the compared multivariate single-tree alternatives, attaining essentially the same predictive performance as OCT-H, the strongest multivariate tree baseline, while requiring much lower average training time. It also achieves predictive performance competitive with Random Forest and within about 1% on average of XGBoost. On synthetic datasets, it substantially outperforms OCT-H in recovering the underlying ground-truth tree structure.

1 INTRODUCTION

The black-box nature of many advanced Machine Learning (ML) models limits their use in high-stakes domains requiring interpretable and accountable decisions. Post-hoc explainability methods have been proposed to address this issue but often fail to provide reliable insights (Rudin, 2019). As a result, inherently interpretable models, such as decision trees (DTs), maintain strong interest due to their transparent decision processes and effectiveness.

DTs are simple rule-based models that are interpretable because the decision path of each sample can be explicitly identified, and they achieve good accuracy at low computational cost. At each branch node, a sample $\mathbf{x}$ is routed using either a deterministic (hard) threshold on a linear feature combination or a continuous (soft) split, resulting in deterministic or soft DTs, respectively. Deterministic decision trees (DDTs) exhibit conditional computation (Bengio et al., 2015), i.e. only a small portion of the model is activated for a given input sample. This leads to faster inference, reduced per-sample parameter usage, and implicit regularization with improved statistical properties (Breiman et al., 1984; Hastie et al., 2009; Bengio et al., 2015). However, their discrete binary routing complicates optimization and limits scalability, so large-scale approaches typically adopt univariate trees with binary features, trading expressiveness for simpler training (see e.g., Lin et al., 2020; Babbar et al., 2025; Aghaei et al., 2025). Soft decision trees (SDTs) are smooth and differentiable and can be trained using standard continuous optimization methods. Although these nonconvex models do not guarantee global optimality, they scale well with features and samples. However, to compute predictions, soft routing assigns samples to multiple leaf nodes and aggregates their contributions, leading to the loss of conditional computation and the interpretability of DDTs.

In this work, we combine the optimization benefits of soft classification trees (SCTs) with the interpretability and conditional computation of deterministic classification trees (DCTs) to build

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

multivariate classification trees. Building on Blanquero et al. (2021; 2020), we propose a new margin-based activation function together with a specialized training procedure within a constrained nonlinear soft formulation. Unlike Blanquero et al. (2021; 2020), our method enforces single-path routing (enhancing interpretability), restoring conditional computation, and enabling margin-based confidence estimates. Moreover, whereas Blanquero et al. (2021; 2020) rely on a general-purpose nonlinear optimization (NLO) solver with a multistart strategy, our SCTs are trained using a single-run projected first-order Adam method with mini-batches, achieving substantially higher accuracy with the proposed activation than with the classical sigmoid.

Contributions: Our contributions can be summarized as follows: **(i)** We propose a variant of SCTs that naturally enables conditional computation at inference time and can be trained efficiently using first-order gradient-based optimization methods. To our knowledge, it is the first single soft-tree method to attain single-path routing in a single training run, without the scaling, annealing, routing regularization, warm start, or multi-start on which prior differentiable trees each rely, thus removing a family of mechanisms considered necessary so far. **(ii)** We introduce a new activation function derived from the negative log-likelihood and differentiable almost everywhere. We further prove that, although not convex, the proposed activation is such that the derived binary margin-based loss is classification-calibrated (Bartlett et al., 2006). **(iii)** We devise a projected version of the Adam algorithm that enables mini-batch training of the proposed SCTs while enforcing the class assignment constraints. **(iv)** We evaluate the proposed method on 61 real-world datasets, comparing it with five single-tree alternatives, whose modeling and training characteristics are summarized in Table 1: ORCT (Blanquero et al., 2021), DTSemNet (Panda et al., 2024), LatentTree (Zantedeschi et al., 2021), RADDT (Mao et al., 2025) and the state-of-the-art deterministic multivariate classification trees OCT-H (Bertsimas & Dunn, 2017; Dunn, 2018). We also compare our SCT with the ensemble methods Random Forest (RF) (Breiman, 2001) and XGBoost (Chen & Guestrin, 2016). Our method uses neither hyperparameter tuning to induce tree sparsity nor any initialization procedure, yet on real-world datasets it outperforms the sigmoid-based SCT alternative (ORCT), DTSemNet, LatentTree and RADDT while achieving essentially the same predictive performance as the strongest deterministic multivariate tree baseline, with one order of magnitude lower average wall-clock training time in the reference implementations. **(v)** We further compare our approach with OCT-H on synthetic datasets, showing that it more accurately recovers the ground-truth DT.

Related Work: Relevant prior work on DCTs and SCTs is discussed, with greater emphasis on SCTs (details on DCTs are in Appendix A). While the NP-hard DTs training (Hyafil & Rivest, 1976) has been early faced via greedy heuristics (e.g., Breiman et al., 1984; Quinlan, 1986; 1993), recent hardware and algorithmic advances have stimulated the spread of methods that optimally train DTs, with local or global optimality guarantees.

For DCTs, most approaches rely on mixed-integer programming (MIP) (e.g., Verwer & Zhang, 2019; Hu et al., 2019; Firat et al., 2020), dynamic programming (DP) (e.g., Demirović et al., 2022; Demirović et al., 2023), branch-and-bound (BnB) techniques (e.g., Aglin et al., 2020; Mazumder et al., 2022), or a combination of the previous strategies (e.g., Hua et al., 2022; Babbar et al., 2025). While all these works focus on univariate splits, multivariate splits have also been considered using different approaches (e.g., Carreira-Perpiñán & Tavallali, 2018; Boutilier et al., 2022; Bennett et al., 2000; Tibshirani & Hastie, 2007; D'Onofrio et al., 2024; Blanco et al., 2023). Seminal works on multivariate DCTs are Bertsimas & Dunn (2019) and Dunn (2018), proposing MIP formulations and local search for learning optimal multivariate classification trees (OCT-H).

SCTs replace deterministic splits with soft, multivariate splitting rules based on sigmoid functions that route samples to child branches with different proportions. Hierarchical Mixture of Experts (HMEs), introduced by Jacobs et al. (1991), are an early instance of this paradigm and are trained via the Expectation–Maximization (EM) algorithm. Building on HMEs, several multivariate SCTs use sigmoid functions for soft splits on linear combinations of tabular features or neural network blocks at branch nodes. Among methods for tabular data, Suárez & Lutsko (1999) uses backpropagation-like gradient training, Irsoy et al. (2012) proposes bottom-up tree growth with node parameters optimized by gradient descent, and Blanquero et al. (2021) formulates training as a constrained nonlinear problem solved via a multistart strategy and an off-the-shelf NLO solver, relying on tunable sigmoid scaling for effective optimization. For this approach, Amaldi et al. (2023) studies sparsity and a decomposition training strategy. For SCTs with neural representations, Frosst & Hinton (2017) formulate them as an HME, where learned filters route samples to specific experts along the most probable path using continuous, mini-batch, gradient-based training. Hehn & Hamprecht (2019); Hehn et al.

(2020) adopt EM-based optimization and use annealing by introducing a multiplicative term on the sigmoid input to recover hard routing. Tanno et al. (2019) integrates neural representation learning with DT–based feature partitioning, enabling adaptive architectural growth via backpropagation. More recently, Bertsimas et al. (2024) studies soft DTs with sigmoid-based splits over convolutional features and introduces two regularization terms promoting single-path routing per sample.

All such works rely on sigmoid for soft routing, except Hazimeh et al. (2020), which proposes an ensemble of soft DTs as a neural network layer and an alternative activation with a hyperparameter to induce single-path predictions. To the best of our knowledge, our work is the first to introduce an activation function for training single soft trees that enable single-path routing without requiring hyperparameter tuning or annealing. Moreover, in all the aforementioned works except Hehn & Hamprecht (2019); Hazimeh et al. (2020), predictions are computed by selecting the leaf with the highest probability path or by aggregating all leaf outputs, which involves the full tree in each prediction and increases inference cost. In contrast, our approach routes each sample along a single path by selecting the branch that maximally activates the activation function, avoiding full-tree evaluation while preserving both conditional computation and differentiability.

Table 1: Additional mechanisms the compared differentiable tree methods rely on. "No per-dataset search" means a single global configuration: no multi-start, warm start, or per-dataset optimizer.

Method	No scale tuning	No annealing	Direct tree optimization	No per-dataset search
ORCT (Blanquero et al., 2021)	✗	✓	✓	✗
LatentTree (Zantedeschi et al., 2021)	✓	✓	✗	✗
DTSemNet (Panda et al., 2024)	✓	✓	✗	✗
RADDT (Mao et al., 2025)	✓	✗	✓	✗
sNLL-CT (ours)	✓	✓	✓	✓

2 Model and formulation

In this section, we present the SCT model and training formulation proposed in Blanquero et al. (2021; 2020) in a general setting, without assuming a specific activation function. We then introduce sNLL, a new margin-based activation based on a scaled variant of the classical negative log-likelihood (NLL), which promotes conditional computation and simplifies training without introducing hyperparameter dependence. Finally, we establish the Fisher consistency Lin (2004); Bartlett et al. (2006) of sNLL.

2.1 Soft classification trees

Consider a dataset $I = \{(\mathbf{x}_i, y_i)\}_{1 \le i \le N}$ of N samples, where $\mathbf{x}_i \in \mathbb{R}^p$ denotes the vector of p features and $y_i \in \{1, \dots, K\}$ the corresponding class label. In line with Jordan & Jacobs (1994); Suárez & Lutsko (1999); Hazimeh et al. (2020); Blanquero et al. (2021), we study maximal (perfect) binary SCTs with prescribed depth D, where each internal branch node has exactly two children and all leaf nodes lie at depth D. Let τ_B and τ_L denote the branch and leaf node sets, respectively. For each $j = 1, \dots, p$ and $t \in \tau_B$, $\mathrm{w}_{jt} \in \mathbb{R}$ denotes the coefficient of feature j in the soft oblique split at node t, while $\mathrm{w}_{0t} \in \mathbb{R}$ denotes the corresponding intercept. Sample routing at each branch node $t \in \tau_B$ is defined by an activation $\psi : \mathbb{R} \to [0, +\infty)$. In the SDT literature, ψ is usually a cumulative distribution function (CDF), i.e., a right-continuous, nondecreasing map to $[0, 1]$. Let $\beta_l = -1$ and $\beta_r = 1$ denote the left and right directions, respectively, and define the routing activation for branch $b \in \{l, r\}$ as $\mu_{it}^b = \psi(\beta_b(\mathbf{w}_t^T \mathbf{x}_i))$. Then

$$\mu_{it}^l = \psi\left(\beta_l\left(\mathrm{w}_{0t} + \frac{1}{p}\sum_{j=1}^{p} \mathrm{w}_{jt} x_{ij}\right)\right), \qquad \mu_{it}^r = \psi\left(\beta_r\left(\mathrm{w}_{0t} + \frac{1}{p}\sum_{j=1}^{p} \mathrm{w}_{jt} x_{ij}\right)\right). \tag{1}$$

The routing rule in (1) can be any of the routing rules proposed in the literature, such as the sigmoid function or the truncated smooth-step function of Hazimeh et al. (2020). Moreover, writing the routing rule as in (1) highlights its margin-based interpretation, since left and right decisions result from applying an activation to the functional margin $\beta(\mathbf{w}^T \mathbf{x})$. Consequently, μ_{it}^l and μ_{it}^r are margin-based confidence scores, coinciding with the left and right routing probabilities whenever ψ is a CDF such that its derivative is symmetric about the y-axis. Given (1), we define the activation level of a sample along the path from the root to each leaf $t \in \tau_L$. For $t \in \tau_L$, let $A_L(t)$ and $A_R(t)$ denote

the sets of ancestor nodes whose left and right branches, respectively, lie on this path. The routing activation level of sample $\mathbf{x}_i$ for leaf t, $i = 1, \ldots, N$, is then $M_{it} = \prod_{t_l \in A_{L(t)}} \mu^l_{it_l} \prod_{t_r \in A_{R(t)}} \mu^r_{it_r}$. For class label prediction at the leaf nodes, following several works (see e.g., Blanquero et al., 2021; Hazimeh et al., 2020), we consider a variable c_{kt} for each leaf node t and class label k. Consistent with Blanquero et al. (2021) and in contrast to Hazimeh et al. (2020), we treat c_{kt} as a binary variable which is equal to 1 if all samples routed to leaf t are assigned class k, and 0 otherwise. This allows to impose constraints ensuring that each class is represented by at least one leaf node.

In classical SCTs, the prediction for $\mathbf{x}$ is a weighted combination of leaf predictions based on routing activations. In contrast, in our SCT model, the prediction for $\mathbf{x}$ is deterministically defined as

$$\hat{y} = \arg\max_{k \in K} \sum_{t \in \tau_L} \left(\prod_{t_l \in A_{L(t)}} \mathbb{1}_+\big(\beta_l(\mathbf{w}_{t_l}^T \mathbf{x})\big) \prod_{t_r \in A_{R(t)}} \mathbb{1}_+\big(\beta_r(\mathbf{w}_{t_r}^T \mathbf{x})\big) \right) c_{kt} \tag{2}$$

where $\mathbb{1}_+(u) = 1$ if $u \geq 0$ ($u > 0$ for the right branch to avoid edge ambiguity) while $\mathbb{1}_+(u) = 0$ if $u < 0$. Thus, exactly one term in (2) is nonzero. When ψ is a CDF, (2) yields deterministic root-to-leaf routing, choosing at each node the branch with probability above 50%.

In Blanquero et al. (2021), SCT training is formulated as a nonconvex, constrained NLO problem minimizing the expected misclassification cost over the training set. We generalize this approach to arbitrary margin-based activation functions and train our SCT by solving the following formulation:

$$\min_{\mathbf{w},\mathbf{c}} \quad f(\mathbf{w}, \mathbf{c}) = \frac{1}{N} \sum_{i=1}^{N} \sum_{t \in \tau_L} M_{it} \sum_{k=1}^{K} e_{y_i k} c_{kt} \tag{3a}$$

$$\text{s.t.} \quad \sum_{k=1}^{K} c_{kt} = 1 \quad \forall t \in \tau_L, \tag{3b}$$

$$\sum_{t \in \tau_L} c_{kt} \geq 1 \quad \forall k \in \{1, ..., K\}, \tag{3c}$$

$$\mathrm{w}_{jt} \in \mathbb{R} \qquad \forall j \in \{0, \ldots, p\}, t \in \tau_B, \tag{3d}$$

$$c_{kt} \in [0, 1] \quad \forall k \in \{1, \ldots, K\}, t \in \tau_L, \tag{3e}$$

where $e_{y_i k} \geq 0$ is a fixed hyperparameter denoting the cost of assigning sample $\mathbf{x}_i$ to class k instead of its true class y_i (i.e., $e_{y_i k} > 0$ if $y_i \neq k$). Constraints (3b) assign exactly one class to each leaf, and (3c) ensure that each class is assigned to at least one leaf node. Although the integrality constraints on c_{kt} are relaxed in (3a)–(3e), the problem admits an optimal solution in which all c_{kt} are binary (see Blanquero et al., 2021). As anticipated, we consider formulation (3) to unconstrained alternatives (e.g., Suárez & Lutsko, 1999; Hazimeh et al., 2020) as (3c) ensures at least one leaf node per class.

Our goal is to propose an easily trainable classification tree with an Adam-like algorithm that satisfies conditional computation via the proposed sNLL activation. We refer to as sNLL-CT any SCT trained by solving (3) with our sNLL as activation.

2.2 The new sNLL activation function

In classical SDTs, branch nodes use the sigmoid to compute routing probabilities. Although this choice is practically ubiquitous in the literature (see Section 1), the sigmoid function suffers from several drawbacks, mainly of numerical nature, which make training SDTs challenging. These issues include vanishing gradients, as the gradient approaches zero for large positive or negative inputs, leading to saturation and slow convergence. Further numerical instabilities stem from the exponential computations, especially for large-magnitude inputs. The sigmoid is also sensitive to weight initialization, as poor initialization can rapidly drive branch nodes to saturation and to convergence to suboptimal solutions. Moreover, during training it favors local minima in which most samples follow a single path, collapsing into few leaf nodes and failing to exploit the tree topology (see e.g., Frosst & Hinton, 2017; Bertsimas et al., 2024).

As anticipated, to mitigate some limitations of the sigmoid, we propose a new margin-based activation that combines the advantages of the NLL and the sigmoid. Its properties enable the training of SCTs through a continuous formulation that supports conditional computation at inference time, so that each sample follows a single path. In practice, sNLL concentrates M_{it} on a single root-to-leaf path, so single-path and soft-aggregated predictions nearly coincide: across all 61 datasets and

tree depths, they agree on over 99% of test samples (see Appendix I). Thus, sNLL-CT recovers conditional computation while retaining soft-aggregation accuracy. Moreover, sNLL also preserves DCT interpretability and provides margin information interpretable as a confidence measure.

NLL, formally defined as $\psi(\alpha) = \log(1 + e^{\alpha})$, is widely used in ML because it links model outputs to likelihood maximization and underlies the sigmoid in logistic regression. Its convex form provides smooth, non-saturating gradients, improving numerical stability and mitigating vanishing gradients. It also induces an SVM-like margin and acts as a smooth approximation of the hinge loss. Its unbounded growth for severely misclassified samples follows from the margin-based form, as it increasingly penalizes margin violations, promotes strong class separation, and yields calibrated predictions with meaningful confidence scores. Motivated by NLL's numerical and margin-based properties, we introduce a new activation, sNLL, combining the benefits of NLL with those of a CDF[1]. The sNLL function is defined as $\psi(\alpha) = \frac{\log(1+e^{\alpha})}{2\log 2+|\alpha|}$, and can be interpreted as a scaled NLL that reconstructs a CDF. Like the NLL, it has a derivative that is not even, promotes margin-based separation, and yields calibrated predictions. At each branch node $t \in \tau_B$, sNLL defines the routing activations, such that for a sample $\mathbf{x}$ the left (right) branch is activated with level $\frac{\log(1+e^{\beta_l(\mathbf{w}^T\mathbf{x})})}{2\log 2+|\mathbf{w}^T\mathbf{x}|} = \frac{\log(1+e^{-\mathbf{w}^T\mathbf{x}})}{2\log 2+|\mathbf{w}^T\mathbf{x}|}$ (respectively, $\frac{\log(1+e^{\mathbf{w}^T\mathbf{x})}}{2\log 2+|\mathbf{w}^T\mathbf{x}|}$). Hence, the function's behavior depends on the side of the hyperplane on which the sample lies, i.e., on the sign of $\mathbf{w}^T\mathbf{x}$. Since its derivative is not even, unlike the sigmoid's, the function exhibits different rates of variation for samples lying on opposite sides of the hyperplane at the same distance from it. As clarified below, this asymmetrc derivative is numerically advantageous in terms of vanishing gradient. Note that, although sNLL can be viewed as a CDF, it does not define a true routing probability, since routing depends on the distance from the hyperplane. In particular, $\psi(-\mathbf{w}^T\mathbf{x}) + \psi(\mathbf{w}^T\mathbf{x}) = 1$ holds only for $\mathbf{w}^T\mathbf{x} = 0$ and as $|\mathbf{w}^T\mathbf{x}| \to \infty$, i.e. when the margin is very large. Still, $\psi(\mathbf{w}^T\mathbf{x})$ can be used as a margin-dependent confidence value in $[0, 1]$.

For additional intuition, we consider the binary classification setting and compare the margin-based losses induced by the standard sigmoid and the proposed sNLL in terms of loss values and gradient magnitude. Given the binary margin-based loss for labels $y \in \{0, 1\}$,

$$\ell(y, z) = y\,\psi(-z) + (1 - y)\,\psi(z), \tag{4}$$

we rewrite it in the equivalent margin form for labels $y \in \{-1, +1\}$[2]. With $z = y(\mathbf{w}^T\mathbf{x})$, using the sigmoid and sNLL for ψ gives the two margin losses $f(z) = \sigma(-z) = \frac{1}{1+e^z}$ and $g(z) = \frac{\log(1+e^{-z})}{2\log 2+|z|}$; Figure 1 shows both losses and their gradient magnitudes. Both losses are bounded, strictly decreasing in z, and satisfy $f(0) = g(0) = 1/2$, but they differ sharply in their gradient behavior. As $z \to -\infty$, $|f'(z)| \sim e^z$ vanishes exponentially, whereas $|g'(z)| \sim C|z|^{-2}$ decays polynomially for some $C > 0$. Thus, f yields high loss but almost no corrective gradient on severely misclassified samples, while g preserves stronger gradient information and mitigates saturation for large negative margins. Furthermore, for strictly negative margins (sufficiently away from zero), g is relatively flat but reacts more strongly to larger errors, whereas f assigns substantial weight to samples near the decision boundary, whether barely correct or incorrect. Hence, g is less aggressive on slightly incorrect examples but more sensitive to large errors and correctly classified samples near the hyperplane. This may be beneficial under feature noise near the boundary, as confirmed by our synthetic experiments.

Classification-calibrated property: In binary classification, many loss functions have been proposed as computationally tractable surrogates for the 0–1 loss. Among them, margin-based classification-calibrated losses are widely studied for ensuring statistical consistency with the Bayes classifier, i.e., alignment with the underlying classification objective (Bartlett et al., 2006).
Let (X, Y) denote a random input-label pair with $Y \in \{-1, 1\}$, let $f : \mathcal{X} \to \mathbb{R}$ be a real-valued scoring function whose prediction is based on the sign of $f(X)$, and let $\phi : \mathbb{R} \to [0, +\infty)$ be a loss function depending on the margin $Yf(X)$. For margin-based loss functions, classification calibration guarantees Fisher consistency, namely, any minimizer of the ϕ-risk $R_\phi(f) = \mathbb{E}[\phi(Yf(X))]$ yields a classifier that agrees with the Bayes-optimal decision rule minimizing the 0–1 risk $R(f) = \mathbb{P}(Y \neq \text{sign}(f(X)))$, where $\text{sign}(\alpha) = 1$ if $\alpha > 0$ and -1 otherwise. Informally, fix an input x and consider the score assigned by the classifier to that sample. The conditional ϕ-risk $\mathbb{E}(\phi(Yf(X)) \mid X = x)$,

[1] We also evaluated NLL as an activation function, but it underperformed sNLL on both real-world and synthetic datasets, as summarized in Table 2 and detailed in Appendix H.

[2] It is easy to verify that (4) is equivalent to (3) for a depth $D = 1$ tree with two classes.

i.e. the ϕ-risk for that fixed input value x, evaluates the average loss at x as a function of that score. Classification calibration requires that, if $P(Y = 1 \mid X = x) > 1/2$, then this conditional ϕ-risk is minimized by choosing a positive score, whereas if $P(Y = 1 \mid X = x) < 1/2$, it is minimized by choosing a negative score. Hence, the sign of the optimal score agrees with the Bayes decision rule. Furthermore, classification calibration yields explicit bounds linking the excess surrogate risk $R_\phi(f) - \inf_f R_\phi(f)$ to the excess 0–1 risk (see Bartlett et al., 2006).
Convex classification-calibrated losses are widely studied for favorable computational properties (Bartlett et al., 2006). Although sNLL is non-convex, the result below concerns the margin-based loss induced by sNLL and establishes that it is classification-calibrated in the standard sense, thus retaining the associated statistical consistency guarantees (formal definition and proof in Appendix B).

Proposition 2.1. *The sNLL loss defined by* $\phi(x) = \frac{\log(1+e^{-x})}{2\log 2+|x|}$ *is classification-calibrated.*

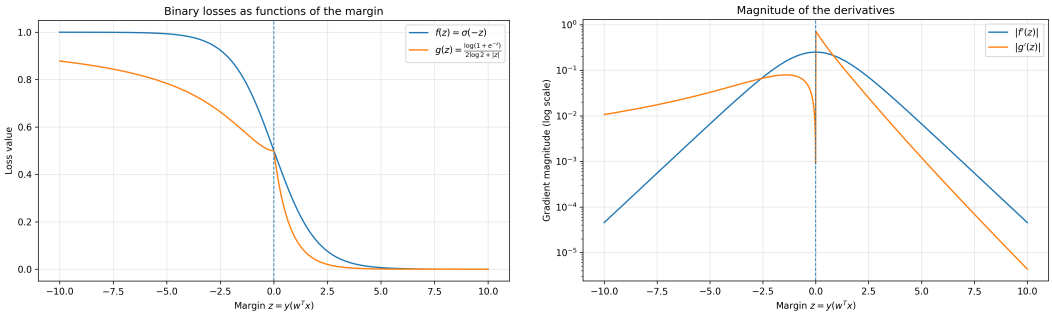


Figure 1: Binary margin losses induced by the sigmoid (blue) and the proposed sNLL (orange) as functions of the margin $z = y(\mathbf{w}^\top \mathbf{x})$. Left: loss. Right: gradient magnitude (log scale).

3 Training Algorithm

Training SCTs is challenging due to their highly non-convex structure. Although SCTs use fewer parameters than modern deep networks, optimization remains difficult because of inherent characteristics. Beyond the sigmoid-related issues discussed in Section 2, the binary assignment of classes to leaf nodes significantly contributes to this difficulty. To mitigate sigmoid-induced low-quality solutions, some works add highly non-convex regularizers to avoid local minima where most samples are routed to few leaves (see e.g., Frosst & Hinton, 2017; Bertsimas et al., 2024). To address class-assignment issues, other approaches combine gradient-based or EM optimization with continuous latent variables for leaf node assignment (see e.g., Suárez & Lutsko, 1999; Hehn et al., 2020; Hazimeh et al., 2020). In Blanquero et al. (2021; 2020), the nonconvex formulation (3) with sigmoid activation is optimized via IPOPT (Wächter & Biegler, 2006), a general purpose off-the-shelf primal–dual interior-point solver exploiting exact first and second order derivatives. To mitigate the sigmoid limitations, the authors introduce a scaling hyperparameter in its argument and combine IPOPT with a multistart strategy, which markedly increases training time.

In this work, we propose a training algorithm based on Adam (Kingma & Ba, 2015), designed to handle the class assignment linear constraints. Adam is a first-order optimizer combining momentum with adaptive per-parameter step-sizes via exponentially weighted estimates of the first and second gradient moments, applied to both full-batch and stochastic settings. It scales well to large problems, where second-order information is out of reach. The adaptive step-sizes improve robustness in hard settings where parameters exhibit heterogeneous sensitivities, such as the SDTs training. To explicitly handle classification and ensure that each class is assigned to at least one leaf, we adopt a projected mini-batch variant of Adam. At each update, the unconstrained parameters are optimized through an Adam step on a training mini-batch, while the assignment variables c_{kt} are projected onto the convex set defined by (3c)–(3b)–(3e) [3]. Moreover, the validation loss is evaluated every E iterations, the learning rate is adaptively reduced when no improvement is observed for P consecutive evaluations, and the final solution is selected as the one yielding the lowest validation loss over the training. Algorithm 1 is the mini-batch projected Adam procedure for training (3).

[3] Although projection onto the relaxed set may yield nonbinary c_{kt}, predictions are unaffected: by definition, (2) depends only on $\arg\max_k c_{kt}$ at the reached leaf, not on their exact values. Accordingly, rounding the learned c_{kt} to integer assignments never changes a test prediction (see Appendix J).

Algorithm 1 Mini-batch Projected Adam with validation-based model selection

Input: training set $\mathcal{T}$, validation set $\mathcal{V}$, loss function $f(\mathrm{w}, \mathbf{c}; \mathcal{D})$, feasible set C for $\mathbf{c}$ defined by (3c)–(3e), initial learning rate η, decay rates β_1, β_2, tolerance ϵ, batch size B, evaluation frequency E, decay factor $\rho \in (0,1)$, plateau patience P, maximum number of iteration T

Initialize: $\mathrm{w}^{(0)}, \mathbf{c}^{(0)}$, $\mathbf{m}_{\mathrm{w}}^{(0)} = \mathbf{0}$, $\mathbf{v}_{\mathrm{w}}^{(0)} = \mathbf{0}$, $\mathbf{m}_c^{(0)} = \mathbf{0}$, $\mathbf{v}_c^{(0)} = \mathbf{0}$, Set plateau counter $p \leftarrow 0$

Compute $\ell^\star = f(\mathrm{w}^{(0)}, \mathbf{c}^{(0)}; \mathcal{V})$, set $(\mathrm{w}^\star, \mathbf{c}^\star) \leftarrow (\mathrm{w}^{(0)}, \mathbf{c}^{(0)})$

for $s = 1, \ldots, T$ **do**

  $\mathcal{B} \in \textsc{MiniBatches}(\mathcal{T}, B)$

  Compute $\mathbf{g}_{\mathrm{w}}^{(s)} = \nabla_{\mathrm{w}} f(\mathrm{w}^{(s-1)}, \mathbf{c}^{(s-1)}; \mathcal{B})$, $\quad \mathbf{g}_c^{(s)} = \nabla_{\mathbf{c}} f(\mathrm{w}^{(s-1)}, \mathbf{c}^{(s-1)}; \mathcal{B})$

  w update

  $\mathbf{m}_{\mathrm{w}}^{(s)} = \beta_1 \mathbf{m}_{\mathrm{w}}^{(s-1)} + (1-\beta_1)\mathbf{g}_{\mathrm{w}}^{(s)}$, $\quad \mathbf{v}_{\mathrm{w}}^{(s)} = \beta_2 \mathbf{v}_{\mathrm{w}}^{(s-1)} + (1-\beta_2)(\mathbf{g}_{\mathrm{w}}^{(s)})^2$

  $\hat{\mathbf{m}}_{\mathrm{w}}^{(s)} = \mathbf{m}_{\mathrm{w}}^{(s)}/(1-\beta_1^s)$, $\quad \hat{\mathbf{v}}_{\mathrm{w}}^{(s)} = \mathbf{v}_{\mathrm{w}}^{(s)}/(1-\beta_2^s)$

  $\mathrm{w}^{(s)} = \mathrm{w}^{(s-1)} - \eta\, \hat{\mathbf{m}}_{\mathrm{w}}^{(s)}/(\sqrt{\hat{\mathbf{v}}_{\mathrm{w}}^{(s)}} + \epsilon)$

  c update

  $\mathbf{m}_c^{(s)} = \beta_1 \mathbf{m}_c^{(s-1)} + (1-\beta_1)\mathbf{g}_c^{(s)}$, $\quad \mathbf{v}_c^{(s)} = \beta_2 \mathbf{v}_c^{(s-1)} + (1-\beta_2)(\mathbf{g}_c^{(s)})^2$

  $\hat{\mathbf{m}}_c^{(s)} = \mathbf{m}_c^{(s)}/(1-\beta_1^s)$, $\quad \hat{\mathbf{v}}_c^{(s)} = \mathbf{v}_c^{(s)}/(1-\beta_2^s)$

  $\tilde{\mathbf{c}}^{(s)} = \mathbf{c}^{(s-1)} - \eta\, \hat{\mathbf{m}}_c^{(s)}/(\sqrt{\hat{\mathbf{v}}_c^{(s)}} + \epsilon)$

  $\mathbf{c}^{(s)} = \mathcal{P}_C(\tilde{\mathbf{c}}^{(s)})$

  if $s = 1$ **or** $s \bmod E = 0$ **then** $\quad$ **compute** $\ell^{(s)} = f(\mathbf{w}^{(s)}, \mathbf{c}^{(s)}; \mathcal{V})$

    if $\ell^{(s)} < \ell^\star$ **then** $\quad \ell^\star \leftarrow \ell^{(s)}$, $\quad (\mathbf{w}^\star, \mathbf{c}^\star) \leftarrow (\mathbf{w}^{(s)}, \mathbf{c}^{(s)})$, $\quad p \leftarrow 0$

    else $\quad p \leftarrow p + 1$; **if** $p \geq P$, **set** $\eta \leftarrow \rho\eta$ **and** $p \leftarrow 0$

Output: $(\mathrm{w}^\star, \mathbf{c}^\star)$

4 Experiments

On a broad collection of real-world benchmark datasets, we compare the test performance of sNLL-CT with: (i) OCT-H (Bertsimas & Dunn, 2017), a state-of-the-art multivariate DCT, considered both at maximum depth without pruning (OCT-H single) and with tuned pruning to explore the accuracy–complexity trade-off (OCT-H grid); (ii) ORCT (Blanquero et al., 2021); (iii) DTSemNet (Panda et al., 2024) ; (iv) RADDT (Mao et al., 2025); (v) LatentTrees (Zantedeschi et al., 2021) for binary classification (reported in Appendix F). We further compare sNLL-CT with the ensemble methods RF Breiman (2001) and XGBoost Chen et al. (2016). Finally, we evaluate sNLL-CT on synthetic data to assess ground-truth recovery against OCT-H, the best multivariate single-tree baseline.

4.1 Experimental setting

The real-world datasets include all the 54 ones used in Bertsimas & Dunn (2017), and 7 additional datasets from the UCI repository (Asuncion & Newman, 2007) with up to 245,000 samples and 6 classes, addressing the fact that many datasets in Bertsimas & Dunn (2017) focus on binary classification. We evaluated several depth values D for single-tree methods to assess how D affects accuracy, while ensuring at least one leaf node per class. We considered 4 settings: i) $D = \lceil \log_2 K \rceil + 1$, i.e., the minimum depth allowing at least one leaf per class, plus one extra level to increase representational capacity; ii) $D = 3$ for datasets with $K \leq 8$ and $D = 4$ otherwise; iii) $D = 4$ for all datasets; iv) $D = 5$ for all datasets. Test accuracy is assessed using 5-fold cross-validation, and the reported performance corresponds to the average accuracy across the five folds.
For the synthetic data, we adapt to the multivariate case the experimental setting introduced in Murthy & Salzberg (1995) and later adopted in Bertsimas & Dunn (2017). Synthetic data are generated by first building a decision tree (ground truth) with randomly selected splits and then creating datasets for which this tree is optimal. sNLL-CT and OCT-H are evaluated in terms of accuracy and clustering consistency with the ground truth. We consider 5 settings based on the baseline $D = 3$, $N = 1000$, $p = 5$, modifying one experimental factor at a time. In **Exp1**, D varies in $\{2, 3, 4\}$. In **Exp2**, the training size N varies in $\{1000, 5000, 10000, 20000\}$. In **Exp3**, feature noise is introduced by perturbing the features of $5, 10, 15, 20\%$ of the training samples with uniform noise in $[-0.1, 0.1]$. In **Exp4**, label noise is added by modifying the class labels of $5, 10, 15, 20\%$ of the training samples. In **Exp5**, p varies in $\{5, 10, 15, 20\}$. While the training set size depends on the setting, the test set size

is fixed to $2^D(p-1)500$. In addition to train and test accuracy, we assess the quality of the learned trees by computing the Rand Index (RI) (Rand, 1971), which measures the similarity between the clusters induced by the leaf nodes of the ground truth tree and those obtained by the two methods. In each setting, we generate 10 random ground-truth trees and construct 10 train–test splits for each tree, so that all results are averaged over 100 runs. Appendix C reports detailed results for all settings and additional information on the datasets, synthetic data generation, and training procedures.

4.2 Experimental results

Real-world dataset experiments. Table 2 reports average test accuracy and training time across 61 real-world datasets for all 4 depth settings, including two activation ablations in the rightmost columns, using the sigmoid (Sig) or standard NLL (NLL-CT) instead of sNLL, all else unchanged; complete per-dataset results and additional tables are provided in Appendix D. Table 2 shows that sNLL-CT has the shortest average training time among all single-tree models and, together with OCT-H grid, achieves the highest test accuracy. Table 3 confirms this result through pairwise Wilcoxon signed-rank tests (Demšar, 2006), with raw and Holm–Bonferroni adjusted p-values ordered by significance. Since the results are highly consistent across settings, we report only setting i) for brevity. As to the ensemble methods (detailed in Appendix D), Table 3 shows that XGBoost performs statistically better than all single-tree methods, including sNLL-CT and OCT-H grid, although the p-values of the latter two are closer to the significance threshold. Notably, RF outperforms all single-tree methods except sNLL-CT and OCT-H grid, while sNLL-CT requires lower average training time than RF (36 vs. 88.5 seconds). sNLL-CT and OCT-H grid are statistically more accurate than all other single-tree methods. In particular, sNLL-CT outperforms both OCT-H single (the maximum-depth variant without pruning) and the sigmoid-based ORCT soft tree, suggesting that both the activation and proposed training procedure contribute to improved accuracy. As for training time, Figure 2, showing boxplots for all methods, indicates that sNLL-CT consistently has training times on the order of 10^1 seconds across datasets, whereas the other methods may exceed 10^3 seconds (such results should be viewed as wall-clock comparisons between the reference native implementations, see Appendices C.3-D). As to RADDT, sNLL-CT's advantage does not stem from evaluating RADDT at its most expensive configuration: sNLL-CT is more accurate and markedly faster than RADDT with 1, 5, and 10 starting points (Appendix E). More generally, sNLL-CT reaches higher accuracy than the alternative soft trees in less training time, and without any of the additional components they each rely on, such as multiple restarts, softmin annealing, sigmoid scaling, or off-the-shelf nonlinear solvers. Finally, ablations isolate the activation's role: relative to sNLL-CT, accuracy drops sharply with Sig and by 3–4 points on average with NLL-CT, confirming that the proposed tuning-free sNLL activation, not the optimizer or prediction rule, drives the gain (larger gaps on synthetic data, Appendix H).

Table 2: Average test accuracy over 61 datasets for four experimental settings. Training time (seconds) is in parentheses, and the highest average accuracy per row is in bold. The two rightmost columns are ablations: Sig uses the sigmoid and NLL-CT the unscaled NLL instead of sNLL, all else equal.

	Comparison methods						Ablations	
Setting	sNLL-CT	OCT-H single	OCT-H grid	ORCT	DTSemNet	RADDT	Sig	NLL-CT
i)	**86.89 (36)**	85.56 (169)	86.73 (297)	78.78 (1077)	84.55 (49)	85.9 (400)	51.29 (23)	82.45 (49)
ii)	**86.67 (35)**	84.72 (195)	86.37 (345)	78.51 (600)	83.65 (49)	85.21 (414)	55.47 (22)	82.55 (51)
iii)	85.95 (35)	84.22 (215)	**86.43 (386)**	78.61 (1301)	84.1 (49)	84.91 (507)	58.11 (23)	83.01 (53)
iv)	**86.72 (39)**	84.42 (210)	86.7 (386)	80.29 (2535)	84.6 (49)	85.72 (789)	64.62 (25)	82.62 (53)

Synthetic dataset experiments. Table 4 reports, for the 5 experiment types, how many configurations significantly favor sNLL-CT or OCT-H on train and test ccuracy and RI. For each experiment and metric, significance is assessed at level $\alpha = 0.05$ using an exact paired permutation test on the aggregated mean differences (Pesarin & Salmaso, 2010), treating each ground truth as an independent block to account for intra-block dependencies. sNLL-CT is generally better than OCT-H, with a clear and consistent advantage in **Exp1**, **Exp3**, and **Exp5** and statistically significant wins in both Accuracy and RI on train and test sets. The only exception is **Exp4**, where OCT-H is more often significantly preferred, indicating greater robustness to label noise. Consistent with the analysis in Section 2.2, this result suggests that sNLL-CT is more robust to feature perturbations, while label noise remains more challenging due to its asymmetric weighting on the two sides of the decision boundary. In **Exp2**, no method is statistically dominant, but median scores across all metrics consistently favor sNLL-CT

(see Appendix G). Overall, these results suggest stronger predictive performance and more faithful recovery of the ground-truth tree structure by sNLL-CT in most scenarios.

Table 3: Two-sided Wilcoxon signed-rank tests on test accuracy over 61 datasets (setting i). Bold marks the better method. Raw and Holm-adjusted p-values are in increasing order; rows above the line remain significant at the 5% level after Holm correction.

Comparison	p-value	Adj. p-value
XGBoost vs. DTSemNet	1.76e-7	4.92e-6
XGBoost vs. OCT-H single	3.50e-7	9.46e-6
sNLL-CT vs. ORCT	6.10e-7	1.59e-5
XGBoost vs. ORCT	1.26e-6	3.16e-5
OCT-H grid vs. ORCT	3.07e-6	7.38e-5
XGBoost vs. RADDT	1.20e-5	2.75e-4
sNLL-CT vs. DTSemNet	5.05e-5	0.001
RF vs. ORCT	6.12e-5	0.001
RF vs. OCT-H single	2.19e-4	0.004
RF vs. DTSemNet	2.26e-4	0.004
OCT-H grid vs. DTSemNet	5.40e-4	0.010
OCT-H grid vs. OCT-H single	8.12e-4	0.014
sNLL-CT vs. OCT-H single	0.002	0.031
XGBoost vs. sNLL-CT	0.002	0.031
RADDT vs. ORCT	0.003	0.043
XGBoost vs. OCT-H grid	0.006	0.075
RF vs. RADDT	0.006	0.077
OCT-H single vs. ORCT	0.022	0.237
DTSemNet vs. ORCT	0.025	0.247
sNLL-CT vs. RADDT	0.030	0.269
RADDT vs. DTSemNet	0.048	0.383
RF vs. OCT-H grid	0.084	0.588
OCT-H grid vs. RADDT	0.111	0.667
RF vs. sNLL-CT	0.126	0.667
XGBoost vs. RF	0.195	0.779
RADDT vs. OCT-H single	0.289	0.866
sNLL-CT vs. OCT-H grid	0.467	0.934
OCT-H single vs. DTSemNet	0.576	0.934

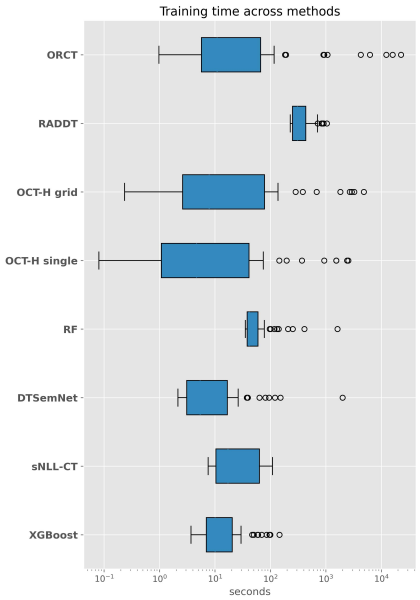


Figure 2: Setting i) training-time boxplots across the 61 datasets, ordered from highest to lowest mean training time.

Table 4: Synthetic experiments: stars count configurations favoring sNLL-CT or OCT-H at $\alpha = 0.05$ per experiment/metric via an exact paired permutation test on ground-truth-level mean differences.

	sNLL-CT				OCT-H			
	Tr. Acc.	Te. Acc.	Tr. RI	Te. RI	Tr. Acc.	Te. Acc.	Tr. RI	Te. RI
Exp1	*	**	-	**	-	-	-	-
Exp2	-	-	-	-	-	-	-	-
Exp3	****	****	*	****	-	-	-	-
Exp4	-	**	-	-	**	*	**	*
Exp5	***	***	***	***	-	-	-	-

5 Conclusions

We propose sNLL-CT, a continuous SCT training method combining a novel activation function inducing a classification-calibrated binary loss, with a projected Adam algorithm enforcing class-assignment constraints, which empirically enables conditional computation via accurate predictions along one root-to-leaf path per sample. Experiments on 61 real-world datasets show that sNLL-CT outperforms all the considered single-tree alternatives and achieves essentially the same predictive performance as OCT-H, the strongest multivariate deterministic tree baseline, while requiring substantially lower average wall-clock training time in the available implementations. Notably, sNLL-CT, trained in one run without the additional components on which prior differentiable trees rely, achieves performance comparable to RF and within about 1% of XGBoost on average. Extensive synthetic experiments further show that sNLL-CT recovers the underlying ground-truth structure more effectively than OCT-H across a range of settings. These results are obtained with non-sparse maximal trees of fixed depth, without initialization or sparsity-promoting tuning, leaving room for further gains from sparsity and pruning, as considered by competing methods such as OCT-H. Single-path prediction already provides procedural interpretability, while sparsity may further improve generalization, accuracy, and interpretability. We also plan to extend the proposed approach to regression tasks.

AI USE STATEMENT

We used generative AI tools only for manuscript-preparation tasks and for verifying one of our proofs. Specifically, we used them to improve the grammar and readability of author-written English text, to shorten and rephrase passages in order to meet the page limit without altering their scientific content, and to assist with the LaTeX formatting and visual presentation of tables. In addition, an LLM was used to review one of our proofs and flagged minor inaccuracies, which the authors subsequently corrected; the corresponding claim and result were unaffected. We did not use generative AI to generate data, develop or provide feedback on the methodology or experimental design, formulate mathematical claims, implement the proposed method, process datasets, or interpret experimental results. All AI-assisted suggestions were manually reviewed and, where necessary, revised by the authors, who verified that they preserved the intended scientific meaning, numerical results, citations, and claims. We take full responsibility for the final content of the submission.

ETHICS STATEMENT

This work is methodological in nature and does not target a specific societal application. The experiments use publicly available benchmark datasets and synthetically generated data, and involve no new data collection, human-subject interaction, or crowdsourcing. The work does not introduce datasets or models that pose specific high-risk misuse concerns. Existing datasets, software, and methods are appropriately credited and used in accordance with their applicable terms and licenses.

REPRODUCIBILITY STATEMENT

We provide the information required to reproduce the theoretical and experimental results throughout the paper and appendices. Sections 2 and 3 describe the model, activation function, and training algorithm, with the complete theoretical results and proofs provided in Appendix B. Appendix C documents the datasets, synthetic-data generation procedure, preprocessing, computational resources, training settings, and hyperparameters, while Appendix D reports the detailed experimental results. An anonymized code repository accompanying the submission provides the implementation, experimental scripts, synthetic-data generation code, required dependencies, and instructions for reproducing the experiments.

AUTHOR CONTRIBUTIONS

If you'd like to, you may include a section for author contributions as is done in many journals. This is optional and at the discretion of the authors.

Use unnumbered third level headings for the acknowledgments. All acknowledgments, including those to funding agencies, go at the end of the paper.

REFERENCES

Sina Aghaei, Andrés Gómez, and Phebe Vayanos. Strong optimal classification trees. *Operations Research*, 73(4):2223–2241, 2025.

Gaël Aglin, Siegfried Nijssen, and Pierre Schaus. Learning optimal decision trees using caching branch-and-bound search. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, pp. 3146–3153. AAAI Press, 2020.

Edoardo Amaldi, Antonio Consolo, and Andrea Manno. On multivariate randomized classification trees: l0-based sparsity, VC dimension and decomposition methods. *Computers & Operations Research*, 151:106058, 2023.

Arthur Asuncion and David Newman. UCI Machine Learning Repository, 2007.

Varun Babbar, Hayden McTavish, Cynthia Rudin, and Margo Seltzer. Near-optimal decision trees in a SPLIT second. In *Proceedings of the 42nd International Conference on Machine Learning*. PMLR, Jul 2025. Oral.

Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Convexity, classification, and risk bounds. *Journal of the American Statistical Association*, 101(473):138–156, 2006.

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. In *Workshop track - ICLR*, 2015.

Kristin P Bennett, Nello Cristianini, John Shawe-Taylor, and Donghui Wu. Enlarging the margins in perceptron decision trees. *Machine Learning*, 41:295–313, 2000.

Dimitris Bertsimas and Jack Dunn. Optimal classification trees. *Machine Learning*, 106(7):1039–1082, 2017.

Dimitris Bertsimas and Jack Dunn. *Machine learning under a modern optimization lens*. Dynamic Ideas LLC Waltham, 2019.

Dimitris Bertsimas, Lisa Everest, Jiayi Gu, Matthew Peroni, and Vasiliki Stoumpou. Deep trees for (un) structured data: Tractability, performance, and interpretability. *arXiv preprint arXiv:2410.21595*, 2024.

Víctor Blanco, Alberto Japón, and Justo Puerto. Multiclass optimal classification trees with SVM-splits. *Machine Learning*, 112(12):4905–4928, 2023. URL https://link.springer.com/article/10.1007/s10994-023-06366-1.

Rafael Blanquero, Emilio Carrizosa, Cristina Molero-Río, and Dolores Romero Morales. Sparsity in optimal randomized classification trees. *European Journal of Operational Research*, 284(1): 255–272, 2020. ISSN 0377-2217.

Rafael Blanquero, Emilio Carrizosa, Cristina Molero-Río, and Dolores Romero Morales. Optimal randomized classification trees. *Computers & Operations Research*, 132:105281, 2021.

Justin J Boutilier, Carla Michini, and Zachary Zhou. Shattering inequalities for learning optimal decision trees. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 19th International Conference, CPAIOR 2022, Los Angeles, CA, USA, June 20-23, 2022, Proceedings*, pp. 74–90. Springer, 2022.

Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks/Cole, Monterey, CA, 1984.

Miguel A Carreira-Perpiñán and Pooya Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. *Advances in Neural Information Processing Systems*, 31, 2018.

Ayman Chaouki, Jesse Read, and Albert Bifet. Branches: Efficiently seeking optimal sparse decision trees with AO*. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, 2025.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*, pp. 785–794, New York, NY, USA, 2016. ACM.

Tianqi Chen, Carlos Guestrin, Tong He, Michael Benesty, Vadim Khotilovich, and Yuan Tang. Xgboost: Extreme gradient boosting. https://xgboost.readthedocs.io/, 2016. Software package.

Vinícius G Costa and Carlos E Pedreira. Recent advances in decision trees: An updated survey. *Artificial Intelligence Review*, 56(5):4765–4800, 2023.

Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J Stuckey. Murtree: Optimal decision trees via dynamic programming and search. *The Journal of Machine Learning Research*, 23(1):1169–1215, 2022.

Emir Demirović, Emmanuel Hebrard, and Louis Jean. Blossom: an anytime algorithm for computing optimal decision trees. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 7533–7562. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/demirovic23a.html.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7(1):1–30, 2006. URL https://www.jmlr.org/papers/volume7/demsar06a/demsar06a.pdf.

Jack Dunn. *Optimal trees for prediction and prescription*. PhD thesis, Massachusetts Institute of Technology, 2018.

Federico D'Onofrio, Giorgio Grani, Marta Monaci, and Laura Palagi. Margin optimal classification trees. *Computers & Operations Research*, 161:106441, 2024.

Murat Firat, Guillaume Crognier, Adriana F Gabor, CAJ Hurkens, and Yingqian Zhang. Column generation based heuristic for learning classification trees. *Computers & Operations Research*, 116:104866, 2020. ISSN 0305-0548.

Nicholas Frosst and Geoffrey E. Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017. doi: 10.48550/arXiv.1711.09784.

Oktay Günlük, Jayant Kalagnanam, Minhan Li, Matt Menickelly, and Katya Scheinberg. Optimal decision trees for categorical data via integer programming. *Journal of global optimization*, 81(1): 233–260, 2021.

Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. The tree ensemble layer: Differentiability meets conditional computation. In *International Conference on Machine Learning*, pp. 4138–4148. PMLR, 2020.

Thomas M Hehn and Fred A Hamprecht. End-to-end learning of deterministic decision trees. In *Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings 40*, pp. 612–627. Springer, 2019.

Thomas M. Hehn, Julian F. P. Kooij, and Fred A. Hamprecht. End-to-end learning of decision trees and forests. *International Journal of Computer Vision*, 128(4):997–1011, 2020. doi: 10.1007/s11263-019-01237-6.

Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Advances in Neural Information Processing Systems*, 32, 2019.

Kaixun Hua, Jiayang Ren, and Yankai Cao. A scalable deterministic global optimization algorithm for training optimal decision tree. In *Advances in Neural Information Processing Systems*, volume 35. NeurIPS, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/37771cc0be272368102a37f202bb88d8-Paper-Conference.pdf.

Laurent Hyafil and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

Ozan Irsoy, Olcay Taner Yıldız, and Ethem Alpaydın. Soft decision trees. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pp. 1819–1822. IEEE, 2012.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Michael I Jordan and Robert A Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2):181–214, 1994.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 6150–6160. PMLR, 2020. URL http://proceedings.mlr.press/v119/lin20g.html.

Yi Lin. A note on margin-based loss functions in classification. *Statistics & Probability Letters*, 68 (1):73–82, 2004.

Qiangqiang Mao, Jiayang Ren, Yixiu Wang, Chenxuanyin Zou, Jingjing Zheng, and Yankai Cao. Differentiable decision tree via "ReLU+Argmin" reformulation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. Spotlight.

Rahul Mazumder, Xiang Meng, and Haoyue Wang. Quant-bnb: A scalable branch-and-bound method for optimal decision trees with continuous features. In *International Conference on Machine Learning*, pp. 15255–15277. PMLR, 2022.

Sreerama K Murthy and Steven Salzberg. Decision tree induction: How effective is the greedy heuristic? In *KDD*, pp. 222–227, 1995.

Subrat Prasad Panda, Blaise Genest, Arvind Easwaran, and Ponnuthurai Nagaratnam Suganthan. Vanilla gradient descent for oblique decision trees. In *European Conference on Artificial Intelligence*, 2024.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Fortunato Pesarin and Luigi Salmaso. *Permutation tests for complex data: theory, applications and software*. John Wiley & Sons, 2010.

Philipp Probst, Marvin N Wright, and Anne-Laure Boulesteix. Hyperparameters and tuning strategies for random forest. *Wiley Interdisciplinary Reviews: data mining and knowledge discovery*, 9(3): e1301, 2019. doi: 10.1002/widm.1301.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

William M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

Robert L Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd. OSQP: an operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020. doi: 10.1007/s12532-020-00179-2.

Alberto Suárez and James F Lutsko. Globally optimal fuzzy decision trees for classification and regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12):1297–1311, 1999.

Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. In *International Conference on Machine Learning*, pp. 6166–6175. PMLR, 2019.

Robert Tibshirani and Trevor Hastie. Margin trees for high-dimensional classification. *Journal of Machine Learning Research*, 8(3), 2007.

Eleonora Vercesi, Stefano Gualandi, Monaldo Mastrolilli, and Luca Maria Gambardella. On the generation of metric tsp instances with a large integrality gap by branch-and-cut. *Mathematical Programming Computation*, 15(2):389–416, 2023.

Sicco Verwer and Yingqian Zhang. Learning optimal classification trees using a binary linear program formulation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 1625–1632, 2019.

Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

Valentina Zantedeschi, Matt Kusner, and Vlad Niculae. Learning binary decision trees by argmin differentiation. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12298–12309. PMLR, 18–24 Jul 2021.

A Related works

Since training DTs is NP-hard (Hyafil & Rivest, 1976) and challenging in practice, early methods (e.g., (Breiman et al., 1984; Quinlan, 1986; 1993)) rely on greedy tree construction. Over the years, numerous greedy DT models and training strategies have been proposed; see Costa & Pedreira (2023) for a survey. More recently, increasing attention has focused on methods that globally optimize deterministic and soft DTs by jointly tuning all model parameters, with local or global optimality guarantees.

For DCTs, most approaches rely on mixed-integer programming (MIP), dynamic programming (DP), or branch-and-bound (BnB) techniques, and almost all focus on univariate splits, with only few exceptions considering multivariate ones. Among MIP-based approaches for univariate classification trees, Verwer & Zhang (2019) use big-M constraints to model split thresholds, Hu et al. (2019) optimize a regularized objective balancing accuracy and tree complexity via BnB, and Firat et al. (2020) propose a column-generation formulation with heuristic pricing to improve scalability. Focusing on univariate trees with binary-feature datasets, Günlük et al. (2021) propose several techniques to improve computational performance, though limited to binary classification, while Aghaei et al. (2025) introduce a flow-based formulation combined with Benders decomposition specifically tailored to binary variables. Among DP-based univariate classification trees that handle only binary features, Demirović et al. (2022) address scalability under size and depth constraints through specialized search and pruning strategies, while Demirović et al. (2023) propose an anytime algorithm for depth-bounded trees that combines search with progressively improving bounds. Several works adopt BnB strategies to control the combinatorial search space, including an itemset-based exact search with aggressive caching (Aglin et al., 2020) and a DP approach with sparsity penalties and tight pruning (Lin et al., 2020). For MIP-based methods on univariate trees with continuous features, Mazumder et al. (2022) leverage quantile-based thresholding, while Hua et al. (2022) propose scalable BnB algorithms that build on the OCT formulation introduced by Bertsimas & Dunn (2019); Dunn (2018). Babbar et al. (2025) propose a near-optimal approach that combines limited exact search with effective pruning to rapidly construct high-quality univariate trees. Recently, Chaouki et al. (2025) propose a best-first heuristic search algorithm for learning optimal sparse univariate classification trees that reformulates tree optimization as a search problem over a specialized AND/OR graph. The method also supports non-binary trees on ordinally encoded data, but does not directly handle continuous variables.

Concerning multivariate DCTs, the seminal works of Bertsimas & Dunn (2019); Dunn (2018) propose MIP formulations and local search procedures for learning optimal classification trees with hyperplane splits (OCT-H). Carreira-Perpiñán & Tavallali (2018) introduce a level-wise alternating optimization scheme for fixed-depth trees with univariate and multivariate splits, where each node is optimized independently via SVM-based hyperplane selection with ℓ_1 regularization to promote sparsity. Zantedeschi et al. (2021) propose a method for learning multivariate binary trees via argmin differentiation, by sparsely relaxing a MIP formulation over tree traversals and node pruning and jointly optimizing split and prediction functions. In contrast to standard DTs, where each leaf is associated with a class label or regression value, the method relies on a learned predictor that can be either a linear model or a multilayer perceptron. Their empirical study considers mainly regression tasks, while classification is restricted to the binary case. Boutilier et al. (2022) derive valid inequalities based on VC-dimension theory to characterize subsets of points shatterable by hyperplane splits, while other MIP formulations model each branch node as a maximum-margin linear classifier, following earlier margin-based approaches (Bennett et al., 2000; Tibshirani & Hastie, 2007), including D'Onofrio et al. (2024); Blanco et al. (2023). More recently, Panda et al. (2024) propose a semantically equivalent and invertible encoding of hard multivariate decision trees as neural architectures, where trainable tree decisions correspond one-to-one to network weights. The model uses ReLU activations and linear operations, and can be trained with standard gradient descent method. Finally, Mao et al. (2025) introduce a differentiable multivariate tree training framework based on a "ReLU+Argmin" reformulation of hard routing. The method combines several ingredients, including a softmin-based approximation, a multi-run warm-start annealing strategy, and tailored adjustments of the initial solution, and is primarily developed and evaluated for regression, with more limited evidence on classification.

B CLASSIFICATION CALIBRATION

For this appendix we follow the notation in Bartlett et al. (2006). In binary classification, let (X, Y) denote a random input-label pair with $Y \in \{-1, 1\}$, $f : \mathcal{X} \to \mathbb{R}$ be a real-valued scoring function whose prediction is based on the sign of $f(X)$, and $\phi(Y, f(X))$ be a loss function depending on the margin $Y \cdot f(X)$. For margin-based loss functions $\phi : \mathbb{R} \to [0, +\infty)$, classification calibration guarantees Fisher consistency, namely, any minimizer of the ϕ-risk $R_\phi(f) = \mathbb{E}[\phi(Y, f(X))]$ induces a classifier whose sign coincides with the Bayes optimal rule minimizing the 0–1 risk $R(f) = \mathbb{P}(Y \neq \text{sign}(f(X)))$, with $\text{sign}(\alpha) > 0$ if $\alpha > 0$ and -1 otherwise. Furthermore, classification calibration provides explicit bounds on $R_\phi(f) - \inf_f R_\phi(f)$ with respect to the excess 0–1 risk, and supports learning-rate guarantees under realistic noise conditions using regularization and complexity control (Bartlett et al., 2006).

To define the notion of classification calibration, we recall notions of conditional risk associated with a loss function ϕ. For a detailed discussion, the reader is referred to Lin (2004); Bartlett et al. (2006).

Let $\eta(x) = P(Y = 1 \mid X = x)$ denote the posterior probability, and define the Bayes risk as $R^* = \inf_f R(f)$, where the infimum is taken over all measurable functions f. Any function f satisfying $\text{sign}(f(X)) = \text{sign}(\eta(X) - 1/2)$ achieves the Bayes risk, that is, $R(f) = R^*$.

Define $C_\eta(\alpha) = \eta\phi(\alpha) + (1 - \eta)\phi(-\alpha)$, so that the conditional ϕ-risk can be expressed as $\mathbb{E}(\phi(Y \cdot f(X)) \mid X = x) = C_{\eta(x)}(f(x)) = \eta(x)\phi(f(x)) + (1 - \eta(x))\phi(-f(x))$. For any $\eta \in [0, 1]$, the optimal conditional ϕ-risk is given by $H(\eta) = \inf_{\alpha \in \mathbb{R}} C_\eta(\alpha) = \inf_{\alpha \in \mathbb{R}} \big(\eta\phi(\alpha) + (1 - \eta)\phi(-\alpha)\big)$. Consequently, the optimal ϕ-risk satisfies $R^*_\phi := \inf_f R_\phi(f) = \mathbb{E}[H(\eta(X))]$.

To state the classification-calibrated condition, we introduce $H^-(\eta)$ as the optimal conditional ϕ-risk under the constraint that the sign of α disagrees with that of $2\eta - 1$ for $\eta \in [0, 1]$. Specifically, $H^-(\eta) = \inf_{\alpha : \alpha(2\eta - 1) \leq 0} C_\eta(\alpha) = \inf_{\alpha : \alpha(2\eta - 1) \leq 0} \big(\eta\phi(\alpha) + (1 - \eta)\phi(-\alpha)\big)$. This allows us to state the definition of a classification-calibrated loss function as follows:

Definition B.1. A function ϕ is *classification-calibrated* if, for any $\eta \neq 1/2$, $H^-(\eta) > H(\eta)$.

Although sNLL is not convex, we show that it is classification-calibrated and therefore preserves statistical consistency guarantees. This result is stated in the following proposition.

Proposition B.2. *The sNLL loss defined by* $\phi(x) = \frac{\log(1+e^{-x})}{2\log 2 + |x|}$ *is classification-calibrated.*

Proof. First, we observe that, for the function $\phi(\alpha) = \frac{\log(1+e^{-\alpha})}{2\log 2+|\alpha|}$, the inequality $\phi(\alpha) < \phi(-\alpha)$ holds for all $\alpha > 0$. This follows from the fact that the numerator is strictly decreasing, since $\frac{d}{d\alpha}\log\big(1+e^{-\alpha}\big) = -\frac{1}{1+e^{\alpha}} < 0$ for all $\alpha \in \mathbb{R}$, while the denominator of $\phi(\alpha)$ and $\phi(-\alpha)$ is identical and strictly positive.

For $\alpha > 0$, define $\Delta_\eta(\alpha) := C_\eta(\alpha) - C_\eta(-\alpha)$. A straightforward algebraic manipulation yields

$$\Delta_\eta(\alpha) = (2\eta - 1)\big(\phi(\alpha) - \phi(-\alpha)\big).$$

By the definition of classification-calibrated loss and of $H^-(\eta)$, it suffices to show that, when $\eta > \frac{1}{2}$ (respectively, $\eta < \frac{1}{2}$), the infimum of $C_\eta(\alpha)$ over $\alpha \le 0$ (respectively, $\alpha \ge 0$) is strictly larger than the infimum over the entire real line.

Recalling that for $\alpha > 0$ one has $\phi(\alpha) - \phi(-\alpha) < 0$, we distinguish the two cases. If $\eta > \frac{1}{2}$, then $2\eta - 1 > 0$, and for any $\alpha > 0$ it follows that $\Delta_\eta(\alpha) < 0$, that is, $C_\eta(\alpha) < C_\eta(-\alpha)$. Thus we have proved the strict pointwise inequality $f(x) < f(-x)$ for $x > 0$, and now we simply need to guarantee that this behaviour is satisfied also asymptotically, that is, the two infima (i.e., the infimum of $C_\eta(\alpha)$ over $\alpha < 0$ and the infimum over the whole domain) do not approach the same asymptotic limit. To prove it, we can show that the constrained infimum (i.e., the one related to $C_\eta(\alpha)$ over $\alpha \le 0$) is separated and attained at a finite point. We observe that $\lim_{\alpha\to-\infty} C_\eta(\alpha) = \eta > 1/2$. Because the global infimum is strictly less than $1/2$ (as shown via Taylor expansion equation 5 below $C_\eta(\alpha)$ is bounded above by $\frac{\log 2+\frac{\alpha}{2}+\frac{\alpha^2}{8}-\alpha\eta}{2\log 2}$), the infimum over $(-\infty, 0]$ is strictly bounded away from the asymptotic limit and must be attained at some finite point $\alpha^* \le 0$. At this finite point, the strict pointwise inequality $C_\eta(-|\alpha^*|) < C_\eta(|\alpha^*|)$ securely guarantees the strict separation of the infima. Consequently, the infimum of $C_\eta(\alpha)$ over $\alpha < 0$ must be strictly larger than the infimum over the whole domain.

An analogous argument applies when $\eta < \frac{1}{2}$. In this case, $2\eta - 1 < 0$, and for any $\alpha > 0$ we have $C_\eta(\alpha) < C_\eta(-\alpha)$. Therefore, the infimum of $C_\eta(\alpha)$ over $\alpha > 0$ is strictly larger than the infimum over the entire domain.

We now show that, in both cases, there exist values arbitrarily close to zero for which $C_\eta(\alpha)$ is strictly smaller than $C_\eta(0) = \frac{1}{2}$. To this end, we consider the Taylor expansion of $g(x) = \log(1 + e^x)$ around 0. In particular,

$$g(0) = \log 2, \qquad g'(x) = \frac{e^x}{1+e^x}, \qquad g'(0) = \frac{1}{2},$$

and

$$g''(x) = \frac{e^x}{(1+e^x)^2} \le \frac{1}{4} \qquad \forall x \in \mathbb{R}.$$

By Taylor's theorem, for every x there exists ξ between 0 and x such that

$$g(x) = g(0) + g'(0)\,x + \frac{g''(\xi)}{2}\,x^2.$$

Using the bound on g'', we obtain

$$g(x) \le \log 2 + \frac{x}{2} + \frac{x^2}{8}. \tag{5}$$

Applying this inequality to $x = \alpha$ and $x = -\alpha$ with $\alpha > 0$, we obtain

$$\log(1+e^{\alpha}) \le \log 2 + \frac{\alpha}{2} + \frac{\alpha^2}{8}, \qquad \log(1+e^{-\alpha}) \le \log 2 - \frac{\alpha}{2} + \frac{\alpha^2}{8}. \tag{6}$$

Imposing $C_\eta(0) > C_\eta(\alpha)$, we distinguish two cases, namely $\alpha > 0$ and $\alpha < 0$. For $\alpha > 0$, a straightforward algebraic manipulation (expanding $1/2 > C_\eta(\alpha)$ and substituting $\log(1+e^{-\alpha}) = \log(1+e^{\alpha}) - \alpha$) yields

$$\log 2 + \frac{\alpha}{2} > \log(1+e^{\alpha}) - \eta\alpha. \tag{7}$$

Using equation 6 from equation 7, we have:

$$\log 2 + \frac{\alpha}{2} > \log 2 + \frac{\alpha}{2} + \frac{\alpha^2}{8} - \eta\alpha. \tag{8}$$

Therefore, for any $0 < \alpha < 8\eta$ we can guarantee that $C_\eta(0) > C_\eta(\alpha)$. By an analogous argument, for $\alpha < 0$ we find $-8(1-\eta) < \alpha < 0$. Thus, for any $0 < \eta < 1$, letting $\delta = \min\{8\eta, 8(1-\eta)\}$, it follows that

$$0 < |\alpha| < \delta \quad \Longrightarrow \quad C_\eta(\alpha) < \frac{1}{2}.$$

Finally, $\eta = 0$ or $\eta = 1$, the monotonicity of ϕ directly implies that $H^-(\eta) > H(\eta)$. □

C Experimental setup details

This appendix provides details on the datasets considered in this work and on some implementation aspects related to the methods considered in the experiments.

C.1 Datasets

We consider 61 real-world datasets. Table 5 lists the names and the main characteristics of the datasets, namely, the number of features p, the number of samples N, and the number of classes K.

Table 5: Names and characteristics of the 61 datasets.

Name	N	p	K
Acute-inflammations-1	120	6	2
Acute-inflammations-2	120	6	2
Balance-scale	625	4	3
Banknote-authentication	1372	4	2
Blood-transfusion	748	4	2
Breast-cancer-diagnostic	569	30	2
Breast-cancer-prognostic	194	32	2
Breast-cancer	683	9	2
Car-evaluation	1728	15	4
Chess-king-rook-versus-king-pawn	3196	37	2
Climate-model-crashes	540	18	2
Congressional-voting-records	232	16	2
Connectionist-bench-sonar	208	60	2
Connectionist-bench	990	10	11
Contraceptive-method-choice	1473	11	3
Credit-approval	653	37	2
Cylinder-bands	277	484	2
Dermatology	358	34	6
Echocardiogram	61	6	2
Fertility	100	12	2
Haberman-survival	306	3	2
Hayes–roth	132	4	3
Heart-disease-Cleveland	297	18	5
Hepatitis	80	19	2
Image-segmentation	210	19	7
Indian-liver-patient	579	10	2
Ionosphere	351	34	2
Iris	150	4	3
Lymphography	148	19	4
Mammographic-mass	830	10	2
Monks-problems-1	124	11	2
Monks-problems-2	169	11	2
Monks-problems-3	122	11	2
Optical-recognition	3823	64	10
Ozone-level-detection-eight	1847	72	2
Ozone-level-detection-one	1848	72	2

Name	N	p	K
Parkinsons	195	21	2
Pen-based-recognition	7494	16	10
Pima-Indians-diabetes	768	8	2
Planning-relax	182	12	2
Qsar-biodegradation	1055	41	2
Seeds	210	7	3
Segment	2310	20	7
Seismic-bumps	2584	20	2
Skin-segmentation	245057	3	2
Soybean-small	47	37	4
Spambase	4601	57	2
Spect-heart	80	22	2
Spectf-heart	80	44	2
Statlog-project-German-credit	1000	48	2
Statlog-project-landsat-satellite	4435	36	6
Teaching-assistant-evaluation	151	52	3
Texture	5500	41	11
Thoracic-surgery	470	24	2
Thyroid-disease-ann-thyroid	3772	21	3
Thyroid-disease-new-thyroid	215	5	3
Tic-tac-toe-endgame	958	18	2
Vowel	990	11	11
Wall-following-robot-2	5456	2	4
Wine	178	13	3
Zoo	101	16	7

C.2 SYNTHETIC DATASET GENERATION

In this Appendix, details are given on the way synthetic datasets are generated. The procedure is slightly different from the previous approaches in Murthy & Salzberg (1995); Bertsimas & Dunn (2017; 2019), as here we are interested in multivariate hyperplane splits

First, a maximal DT of a specified depth is constructed by determining random splits in the p-dimensional space. To this aim, for a given current node of the tree, $p + 1$ samples are randomly generated by a hit-and-run algorithm (see Vercesi et al., 2023; Smith, 1984) within the subregion corresponding to the node. Then, the hyperplane split associated to the current node is obtained as the regression hyperplane passing through the random samples. This split will determine the two child nodes with the associated complementary subregions. The node split is recursively iterated from the root node to the leaves until the desired depth is obtained.

Then, the leaves are labeled such that no two adjacent regions induced by the tree have the same label, so that the tree results to be the smallest representation of the ground truth. Excluding the leaf nodes sharing the same parent which are clearly adjacent, two leaf nodes are considered not adjacent if the intersection of the systems of linear inequalities (splits) defining the corresponding subregions is the empty set. Once obtained the adjacency information for any pair of leaf nodes, the class labels at all leaf nodes are initialized to $k =$ number of leaves, then the label of each leaf node is modified to be the smallest number in $\{1, 2, \ldots, k\}$ not yet assigned to any adjacent leaf node.

Finally, the datasets can be generated by determining each feature vector $\mathbf{x}_i \in \mathbb{R}^p$ uniformly random over its domain, and assigning y_i according to the label of the leaf node where $\mathbf{x}_i$ falls.

C.3 TRAINING DETAILS

The experiments were carried out on a PC equipped with an Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz, 64 GB of RAM, and an NVIDIA GeForce RTX 2080 GPU with 8 GB of memory.

For all datasets, continuous features are normalized to the $[0, 1]$ range, except for those methods requiring a specific normalization according to the reference work, in which case the original normalization is used. Categorical features are one-hot encoded.

We use 20% of the training set as a validation set for model selection during training and for hyperparameter tuning whenever applicable.

For the OCT-H model used in this work, we relied on the `OptimalTreeClassifier` function from the Interpretable AI package, version 2.0.2, available at `https://docs.interpretable.ai/stable/`. In OCT-H, the `cp` hyperparameter governs cost-complexity pruning and thus has a direct impact on the effective tree size, such as the number of leaves. To make the comparison as complete as possible, we considered both the OCT-H tree at a fixed maximum depth (OCT-H single), so as to isolate the role of the splitting and routing mechanism, and a second setting in which the pruning-induced accuracy/complexity trade-off can be explored by letting `OptimalTreeClassifier` choose the pruning level on a validation set (OCT-H grid). For this reason, in OCT-H single we disable pruning by setting `cp`=0, which places OCT-H in the same maximal-tree regime as our model. In OCT-H grid, instead, we tune `cp`, which can lead to a different accuracy–complexity trade-off.

For the ORCT model in (Blanquero et al., 2021), no public implementation was available, so we developed our own implementation following the same settings reported in Blanquero et al. (2020). Specifically, we used the same Python and IPOPT versions adopted in that work. For each fold, we employed a multi-start strategy with 10 different random starting points.

For DTSemNet from Panda et al. (2024), we relied on the official code provided at `https://github.com/CPS-research-group/dtsemnet`. In their work, the authors adopt a different training configuration for each dataset. More specifically, for each dataset considered in Panda et al. (2024), they report different choices of optimization method (e.g., momentum, RMSprop, Adam), epochs, batch size, learning rate, and learning-rate scheduler settings. Since we consider 61 datasets, we did not follow this dataset-specific tuning procedure. To maintain a fair and reproducible comparison across a broad and heterogeneous set of datasets, we instead kept the core training choices of the original method, namely feature standardization, validation-based model selection, and gradient-based optimization with learning-rate scheduling, while using one global configuration for each method on all datasets. This choice prevents per-dataset over-specialization and ensures that no method receives a tuning effort that is not equally available to the others. The adopted configuration is reported in Table 6.

Table 6: Global training configuration used in all experiments for DTSemNet.

Parameter	**Value**	**Notes**
Learning rate	0.1	initial learning rate
Batch size	128	mini-batch size
Epochs	400	total number of training epochs
`over_param`	`[]`	no additional over-parameterization
`wt_init`	`True`	weight initialization enabled
Optimizer	Adam	`opt = "adam"`
`momentum`	0.0	not used with Adam
`lr_scheduler`	`True`	learning rate scheduler enabled
`lr_scheduler_type`	`linear`	linear schedule
`lr_scheduler_gamma`	0.95	scheduler factor
`use_L1`	`True`	L1 regularization enabled
L1 coefficient	10^{-4}	`lamda_L1 = 1e-4`
`grad_clip`	`False`	gradient clipping disabled

For LatentTrees from Zantedeschi et al. (2021), we used the official implementation available at `https://github.com/vzantedeschi/LatentTrees`. We considered multivariate splits and, to ensure a fair comparison across all models, we used a linear model rather than a multilayer perceptron at the leaf nodes[4]. Following Zantedeschi et al. (2021), we optimized the hyperparameters with the Optuna Bayesian optimizer and set the number of trials to 100. We used the same training setup, namely batch size equal to 512, a scheduler with factor 10 after 2 epochs with no improvement,

[4]In LatentTrees, the final prediction is not determined solely by the leaf node reached by the input vector $\mathbf{x}$. Instead, the model takes $\mathbf{x}$, constructs a latent tree representation $\mathbf{z}$, concatenates the two, and applies either a linear model (logistic regression) or a multilayer perceptron at the corresponding leaf node.

max epochs equal to 100, and early stopping patience equal to 10. On tabular datasets, their method tunes the regularization parameter λ, and we adopted the same grid of values. The only difference was the learning rate, which we set to 0.1, since we found that this choice led to better average test accuracy.

For RADDT from (Mao et al., 2025), we used the official implementation available at `https://github.com/YankaiGroup/RADDT/tree/classification`. We did not modify the model implementation, and only adapted the experimental script to run RADDT on the same datasets, folds, and depth settings used for the other methods. Training followed the default structure of the official code: CART warm start, deterministic tree-path routing through `readTreePath`, and the multi-start gradient-based routine `multiStartTreeOptbyGRAD_withC`. We used 3000 epochs, 10 starting points, and 5 scale factors in the softmin annealing procedure.

Concerning the ensemble methods, RF was trained using the `Scikit-learn` library in Python (Pedregosa et al., 2011) while XGBoost was implemented using `XGBClassifier` from the `xgboost` package (Chen et al., 2016). For both ensemble methods, the maximum tree depth and the number of trees were tuned using a validation set (20% of the training set). After selecting these two hyperparameters, the final model was retrained on the full training set. In particular, the number of trees was tuned over $\{50, 100, 200, 300, 400, 500\}$, while the maximum tree depth was tuned over a wider range, from 2 to 50 with step 2, including 1 (i.e., 1, 2, 4, ..., 50), in order to potentially select good depth values. This choice was motivated by the fact that, especially for RF, deeper trees were found empirically to improve test accuracy (Probst et al., 2019). All other hyperparameters, such as the number of features per split and the number of samples per tree for RF, and the learning rate for XGBoost, were kept at their default values.

The sNLL-CT was implemented in `Python 3.11.11`, and the Projected Adam Algorithm 1 through `pytorch 2.5.1` (Paszke et al., 2019). The projection operator $\mathcal{P}_{\mathcal{C}}$ is computed as the Euclidean projection onto the feasible set defined by constraints equation 3b-equation 3c. This projection is formulated as a convex quadratic program with linear constraints and solved using OSQP version 1.0.4 (Stellato et al., 2020). In the training of sNLL-CT and Sig (both using Algorithm 1) we included in the loss function equation 3a an ℓ_2 regularization term on $\mathbf{w}$, with hyperparameter equal to $\frac{1e^{-6}}{p}$. In the optimization of variables $\mathbf{w}$ and $\mathbf{c}$, a 20% of the training set was used as validation set to iteratively check the progress of the training and arrest the algorithm when the stopping conditions were met. For the hyperparameters of Algorithm 1, two slightly different settings were considered depending on whether the problem to be solved was binary or multi-class classification. Specifically, for multiclass datasets, we set a maximum number of iterations $T = 30000$, the batch size $B = 512$, the initial learning rate $\eta = 2$, and a plateau patience $P = 200$. For binary classification datasets, we set a maximum number of iterations $T = 10000$, a batch size $B = 256$, an initial learning rate $\eta = 1.75$, and a plateau patience $P = 50$ [5]. For both settings, the decay rates were set $\beta_1 = 0.9$ and $\beta_2 = 0.999$, the decay factor $\rho = 0.85$, the tolerance $\epsilon = 1e^{-8}$, and the evaluation frequency $E = 10$. In order to prevent the algorithm from stopping either too early or too late, both settings adopted an initial warm-up period, during which stopping was not permitted, and a maximum iteration threshold, after which the algorithm stops if the validation loss had not improved. The warm-up threshold was set to 50 iterations for binary datasets and 100 iterations for multiclass datasets, whereas the maximum number of iterations without improvement was fixed at 250 and 1000, respectively. For binary classification datasets, to simplify the training problem, variables c_{kt} were fixed so that two leaf nodes sharing the same parent (branch node) were always assigned to different classes. This avoids the projection onto the convex set defined by equation 3b–equation 3c in Algorithm 1. Preliminary experiments on binary datasets showed that fixing the c_{kt} values did not result in appreciable change. Concerning the parameter $e_{y_i k}$ in formulation equation 3, following Blanquero et al. (2021; 2020), we set $e_{y_i k} = 0.5$ when $y_i \neq k$ and 0 otherwise, except for the Balance-scale, Thyroid-disease-new-thyroid, and Thyroid-disease-ann-thyroid datasets. For these three datasets, preliminary results indicated that better performance is obtained by defining $e_{y_i k}$ for $y_i \neq k$ using a balanced class-weighting scheme, in which class weights are inversely proportional to the observed class frequencies in the data. This approach reflects a standard practice in the machine learning literature and is commonly adopted in widely used implementations such as scikit-learn.

[5] Due to the faster gradient decay for the sigmoid function, in preliminary experiments, smaller η values have been tested for Sig yielding no specific benefits.

For the synthetic dataset, the same validation procedure and the same experimental setting considered for the real-world dataset in the case of multi-class datasets were applied. The only difference is that, in this case, we used a full-batch version (i.e. the batch size coincides with the size of the training set), since our goal here is only to assess how well sNLL-CT can reconstruct the original tree, rather than on balancing accuracy and runtime. For the same reason we report in Table 17 of Appendix D the results on the synthetic datasets where the median is used to aggregate the performance over the multiple runs. Indeed, the median tends to be less sensitive to potential anomalous occurrences which may arise more likely for randomly generated ground truths.

Our implementation of the Projected Adam algorithm relies on vectorized tensor operations based on broadcasting, with support for GPU acceleration. This design is intended to amortize the fixed costs of tensor-based computation over increasingly large batches of operations, and is therefore most effective on medium and large scale datasets. On very small datasets, these fixed costs, including kernel dispatch, memory allocation, and framework overhead, may represent a non-negligible fraction of the total running time. As a result, the implementation is less advantageous in the small-data regime, where more specialized CPU-oriented implementations could be competitive. To ensure consistency and reproducibility, all experiments reported in the paper are conducted with the same vectorized implementation.

The sNLL-CT code used for this paper can be found at the following link.

D Detailed experimental results

In this appendix, we report the complete results for all four settings corresponding to the different maximum depths, including the mean 5-fold accuracy and training time. We also include box plots of the training times and a table reporting the results of the paired two-sided Wilcoxon signed-rank test across the 61 datasets. Finally, we present several figures showing pairwise accuracy comparisons between sNLL-CT on the y-axis and each alternative method on the x-axis.

Overall, the results are highly similar and consistent across the four settings. As shown in Tables 10, 8, 12, and 14, which report the paired two-sided Wilcoxon signed-rank tests, sNLL-CT and OCT-H grid consistently emerge as the best single-tree models, tying for first place. It is also worth noting that sNLL-CT is always significantly better than OCT-H single, that is, OCT-H without complexity tuning and therefore with the same tree topology as sNLL-CT. Among ensemble methods, XGBoost significantly outperforms all single-tree methods in all four settings. By contrast, sNLL-CT and OCT-H grid remain broadly competitive with RF.

The scatterplots of Figures 3-10 depict pairwise comparison in terms of test accuracy between sNLL-CT and all other methods for all datasets (markers). Markers above the bisector corresponds to dataset for which sNLL-CT obtains better performance. Three relevant observations emerge from the comparisons between sNLL-CT and the OCT-H variants in Figures 3, 5, 7, and 9. First, compared to OCT-H grid, sNLL-CT yields better performance on more datasets (32 versus 22) in the two shallower settings, i.e., i) and ii). In settings iii) and iv), OCT-H grid achieves more wins than sNLL-CT, namely, 32 versus 24 and 28 versus 26, respectively. This pattern is consistent with the fact that OCT-H grid explicitly tunes tree depth, whereas sNLL-CT is restricted to maximal trees. Second, sNLL-CT consistently outperforms OCT-H single across all four settings, with the number of favorable datasets ranging from 41 to 48. Finally, in terms of training time, sNLL-CT is on average substantially faster than all single-tree alternatives and even than RF, while also scaling well with both tree depth and sample size.

Hardware related discussions. The training procedure of sNLL-CT is based on an unconstrained optimization method, which makes the method naturally amenable to scalable implementations. In particular, as mentioned, our implementation relies on PyTorch and can exploit vectorized operations and GPU acceleration. This computational setting should be taken into account when interpreting the training-time results reported in the experiments, since not all competing methods are implemented under the same hardware and software assumptions. For DTSemNet and LatentTrees, the comparison is fully aligned: all methods are trained on the same hardware and within the same PyTorch-based framework. For ORCT, the reference implementation relies on IPOPT and it is therefore CPU only and does not exploit GPU acceleration. To provide a more informative point of comparison with sigmoid-based soft trees, we also consider Sig, which can be viewed as a GPU-compatible analogue

of ORCT since it is trained within the same PyTorch framework as sNLL-CT. Across different tree depths, Sig achieves training times that are comparable to those of sNLL-CT and, in some cases, lower. However, this apparent advantage is largely explained by the fact that Sig often converges prematurely at lower-quality solutions, as reflected by the large accuracy gap in favor of sNLL-CT. Moreover, applying to Sig the same multi-start strategy used for ORCT would substantially increase its overall training time, making it no longer competitive from a cpu time perspective. Finally, the OCT-H implementation provided by the authors is based on proprietary code and does not support GPU execution. We therefore do not interpret the OCT-H comparison as an isolated measure of hardware-independent algorithmic efficiency, but rather as the training time observed for the available implementations under their respective computational backends. It is worth pointing out that, for large datasets, the local search algorithm used to solve the OCT-H training formulation is generally very time consuming.

D.1 Results for depth setting i)

In this subsection, we report the results obtained by setting the maximum depth of the single-tree methods to the minimum depth required to allow at least one leaf node per class, and then increasing it by one to improve representational capacity, that is, $D = \lceil \log_2 K \rceil + 1$. In particular, Table 8 reports the results of the paired two-sided Wilcoxon signed-rank test across the 61 datasets, while Table 7 presents the test accuracy and average training time of all models for each dataset.

Table 7: Average classification accuracy and, in parentheses, training time in seconds for each method on each dataset for depth setting i). Bold indicates, for each dataset, the best test accuracy among the single-tree methods.

Dataset	sNLL-CT	OCT-H single	OCT-H grid	ORCT	DTSemNet	RADDT	Sigmoid	RF	XGBoost
Acute-inflammations-1	**100.0 (29)**	**100.0 (0.1)**	99.17 (1)	**100.0 (1)**	**100.0 (2)**	**100.0 (672)**	41.67 (6)	98.33 (35)	100.0 (4)
Acute-inflammations-2	**100.0 (28)**	99.17 (0.1)	99.17 (0.2)	**100.0 (1)**	**100.0 (2)**	**100.0 (241)**	49.17 (6)	96.67 (35)	100.0 (5)
Balance-scale	**91.36 (70)**	88.64 (3)	89.6 (5)	88.64 (19)	89.12 (8)	90.08 (290)	49.76 (39)	85.6 (41)	92.64 (22)
Banknote-authentication	**100.0 (26)**	99.85 (5)	99.85 (9)	99.2 (12)	99.2 (12)	99.71 (243)	44.46 (6)	99.13 (55)	99.71 (6)
Blood-transfusion	77.15 (13)	**79.02 (2)**	78.48 (2)	76.48 (6)	**79.02 (7)**	78.35 (264)	23.65 (6)	76.74 (43)	78.35 (14)
Breast-cancer	73.26 (9)	69.32 (2)	70.76 (3)	**73.64 (3)**	68.59 (4)	**73.64 (245)**	70.76 (6)	73.63 (38)	68.95 (9)
Breast-cancer-diagnostic	**97.89 (10)**	95.26 (7)	95.96 (15)	97.36 (16)	96.66 (5)	95.6 (237)	62.74 (10)	95.25 (52)	96.66 (7)
Breast-cancer-prognostic	77.29 (11)	**78.33 (3)**	76.29 (5)	73.14 (6)	75.25 (2)	69.53 (241)	76.28 (6)	76.79 (41)	79.34 (7)
Car-evaluation	**97.69 (64)**	96.87 (54)	96.82 (95)	88.31 (66)	94.1 (21)	96.24 (309)	70.02 (47)	95.48 (48)	98.38 (30)
Chess-king-rook-versus-king-pawn	99.19 (28)	99.19 (197)	**99.28 (382)**	96.15 (62)	98.03 (26)	99.09 (229)	52.22 (6)	98.94 (60)	99.12 (15)
Climate-model-crashes	90.19 (12)	88.7 (4)	**91.48 (6)**	90.19 (11)	90.19 (5)	87.22 (400)	8.52 (6)	91.3 (50)	91.3 (8)
Congressional-voting-records	93.52 (14)	92.67 (1)	**96.98 (3)**	93.11 (3)	93.52 (4)	93.54 (235)	74.15 (12)	95.24 (36)	96.11 (5)
Connectionist-bench	**81.41 (85)**	74.65 (46)	73.84 (81)	38.69 (936)	60.2 (22)	80.91 (840)	6.87 (46)	95.56 (77)	91.41 (59)
Connectionist-bench-sonar	77.39 (15)	73.07 (5)	74.05 (13)	74.99 (10)	75.96 (4)	**80.33 (249)**	53.34 (6)	79.78 (44)	84.18 (7)
Contraceptive-method-choice	45.22 (81)	52.55 (62)	**55.67 (93)**	46.5 (73)	53.29 (16)	49.56 (389)	44.94 (44)	55.53 (56)	55.39 (59)
Credit-approval	**87.14 (9)**	83.31 (19)	86.83 (29)	86.37 (11)	83.3 (7)	84.99 (246)	54.67 (6)	86.22 (43)	87.44 (9)
Cylinder-bands	74.36 (9)	69.29 (31)	**77.95 (76)**	68.89 (14)	67.5 (4)	73.28 (238)	35.77 (6)	77.63 (41)	74.77 (9)
Dermatology	**97.26 (58)**	93.71 (8)	94.53 (20)	69.39 (93)	96.45 (7)	96.72 (348)	30.57 (36)	97.26 (39)	97.26 (19)
Echocardiogram	68.95 (8)	67.71 (0.2)	**70.19 (0.4)**	66.19 (1)	**70.19 (2)**	62.19 (285)	67.33 (6)	68.86 (36)	72.95 (5)
Fertility	**88.0 (9)**	81.0 (0.3)	**88.0 (0.4)**	84.0 (1)	86.0 (2)	81.0 (237)	12.0 (6)	88.0 (36)	87.0 (5)
Haberman-survival	**74.21 (9)**	71.25 (0.5)	73.54 (0.7)	72.89 (3)	73.21 (4)	71.26 (372)	73.54 (6)	70.26 (38)	69.29 (9)
Hayes–roth	80.62 (68)	81.25 (0.9)	**81.88 (2)**	78.75 (6)	81.25 (3)	78.12 (292)	58.75 (36)	84.38 (37)	84.38 (17)
Heart-disease-Cleveland	**59.07 (62)**	52.46 (6)	55.45 (9)	58.07 (46)	54.13 (6)	52.87 (419)	11.89 (34)	56.75 (40)	56.44 (24)
Hepatitis	86.25 (8)	86.25 (0.4)	**91.25 (0.6)**	87.5 (1)	88.75 (2)	85.0 (323)	**91.25 (20)**	91.25 (36)	86.25 (5)
Image-segmentation	86.67 (58)	**88.57 (5)**	88.1 (11)	46.19 (56)	81.9 (7)	88.1 (338)	26.67 (38)	91.43 (41)	88.57 (21)
Indian-liver-patient	70.66 (11)	69.99 (4)	**71.87 (5)**	71.35 (7)	71.18 (5)	70.32 (418)	71.35 (19)	69.81 (46)	70.32 (9)

Continued on next page

Dataset	sNLL-CT	OCT-H single	OCT-H grid	ORCT	DTSemNet	RADDT	Sigmoid	RF	XGBoost
Ionosphere	**90.33 (12)**	88.89 (5)	89.73 (12)	85.19 (10)	89.73 (4)	86.6 (245)	35.89 (6)	92.59 (47)	92.02 (7)
Iris	**96.67 (49)**	95.33 (0.3)	94.67 (0.5)	96.0 (5)	88.67 (3)	95.33 (512)	66.67 (38)	95.33 (36)	95.33 (10)
Lymphography	**83.08 (44)**	78.32 (4)	81.03 (9)	80.41 (12)	80.41 (3)	80.41 (280)	58.92 (34)	85.08 (38)	84.44 (16)
Mammographic-mass	82.05 (12)	83.13 (2)	**83.86 (4)**	80.6 (7)	82.41 (8)	83.01 (627)	80.0 (11)	84.34 (43)	82.53 (14)
Monks-problems-1	**100.0 (12)**	99.64 (2)	**100.0 (5)**	**100.0 (6)**	97.84 (5)	99.46 (251)	49.99 (6)	96.94 (40)	99.82 (7)
Monks-problems-2	**100.0 (17)**	99.83 (3)	98.83 (6)	80.38 (6)	76.17 (6)	87.68 (251)	34.27 (7)	63.06 (42)	72.05 (12)
Monks-problems-3	98.74 (12)	98.02 (2)	**98.92 (3)**	**98.92 (6)**	97.83 (5)	98.56 (254)	52.0 (6)	98.56 (38)	98.92 (6)
Optical-recognition	**95.74 (88)**	90.35 (2528)	92.49 (4836)	30.71 (12246)	94.45 (81)	95.5 (704)	26.44 (67)	98.09 (131)	97.7 (70)
Ozone-level-detection-eight	92.91 (12)	93.02 (52)	92.96 (103)	**93.07 (116)**	**93.07 (17)**	92.58 (560)	**93.07 (6)**	93.5 (118)	93.67 (19)
Ozone-level-detection-one	**96.91 (10)**	96.37 (41)	**96.91 (78)**	**96.91 (117)**	96.37 (17)	96.59 (558)	**96.91 (6)**	96.91 (101)	96.75 (15)
Parkinsons	90.26 (9)	86.15 (2)	87.18 (4)	85.64 (4)	**93.33 (2)**	**93.33 (265)**	75.38 (6)	89.23 (39)	92.82 (5)
Pen-based_recognition	97.89 (79)	96.74 (1542)	96.92 (2699)	69.87 (16005)	95.18 (152)	**98.4 (866)**	16.09 (53)	99.05 (208)	99.25 (86)
Pima-Indians-diabetes	**76.04 (9)**	74.87 (5)	74.73 (7)	75.65 (8)	75.65 (7)	74.87 (377)	65.11 (6)	76.3 (47)	76.43 (10)
Planning-relax	69.79 (14)	58.8 (0.8)	68.68 (1)	**71.46 (3)**	69.83 (2)	53.36 (321)	**71.46 (24)**	70.92 (40)	67.6 (6)
Qsar-biodegradation	85.31 (12)	84.74 (34)	84.64 (64)	75.17 (23)	**85.59 (10)**	85.5 (251)	33.74 (6)	86.64 (62)	84.93 (14)
Seeds	**94.76 (46)**	92.86 (0.9)	93.81 (2)	94.29 (8)	92.38 (5)	**94.76 (364)**	66.19 (37)	89.52 (38)	92.86 (10)
Segment	95.02 (98)	95.37 (145)	**95.76 (286)**	50.17 (1069)	91.69 (38)	93.38 (321)	26.1 (60)	97.92 (97)	98.48 (47)
Seismic-bumps	93.34 (12)	93.15 (27)	**93.42 (39)**	**93.42 (32)**	93.11 (22)	92.57 (712)	**93.42 (6)**	93.19 (67)	93.07 (16)
Skin-segmentation	97.93 (31)	99.11 (2410)	98.91 (3250)	99.12 (4248)	98.95 (2003)	**99.37 (571)**	57.26 (12)	99.95 (1625)	99.96 (96)
Soybean-small	98.0 (74)	98.0 (0.1)	98.0 (0.4)	98.0 (3)	95.78 (2)	**100.0 (665)**	98.0 (76)	98.0 (36)	98.0 (4)
Spambase	92.5 (23)	93.89 (370)	**93.98 (683)**	83.6 (77)	92.39 (39)	92.46 (259)	60.6 (6)	95.33 (141)	95.35 (50)
Spect-heart	65.0 (9)	68.75 (0.5)	**76.25 (1)**	67.5 (1)	67.5 (2)	71.25 (249)	50.0 (6)	73.75 (36)	75.0 (6)
Spectf-heart	68.75 (10)	**75.0 (1)**	72.5 (3)	67.5 (3)	61.25 (2)	70.0 (263)	50.0 (6)	76.25 (37)	72.5 (4)
Statlog-project-German-credit	**74.0 (8)**	71.0 (57)	72.3 (102)	71.9 (18)	69.2 (9)	71.1 (241)	30.0 (6)	75.1 (51)	74.1 (13)
Statlog-project-landsat-satellite	86.1 (77)	**87.71 (930)**	87.47 (1828)	73.08 (6226)	87.09 (94)	85.71 (315)	43.09 (58)	91.96 (253)	91.98 (99)
Teaching-assistant-evaluation	56.92 (50)	60.22 (5)	62.9 (11)	60.19 (8)	45.08 (3)	**64.24 (296)**	33.1 (32)	57.66 (38)	62.88 (24)
Texture	98.89 (107)	95.56 (1533)	96.07 (2935)	32.85 (22608)	98.69 (121)	**99.55 (1045)**	23.8 (74)	97.6 (411)	98.58 (146)
Thoracic-surgery	84.04 (8)	78.94 (6)	**85.11 (7)**	83.62 (5)	79.79 (5)	81.7 (267)	**85.11 (6)**	84.68 (41)	84.47 (9)
Thyroid-disease-ann-thyroid	96.66 (70)	99.28 (75)	**99.58 (138)**	92.71 (184)	97.11 (37)	98.54 (912)	3.92 (35)	99.68 (60)	99.52 (20)
Thyroid-disease-new-thyroid	**96.28 (66)**	95.81 (0.6)	93.02 (1)	95.35 (7)	93.95 (4)	95.81 (294)	61.4 (37)	94.42 (38)	95.35 (10)
Tic-tac-toe-endgame	98.02 (15)	97.08 (26)	97.39 (42)	**98.33 (12)**	97.08 (9)	98.02 (246)	65.33 (6)	98.43 (45)	99.37 (8)
Vowel	81.11 (109)	71.31 (45)	69.8 (81)	34.55 (899)	70.3 (22)	**82.93 (836)**	8.59 (50)	95.76 (78)	91.41 (60)
Wall-following-robot-2	97.91 (61)	**100.0 (4)**	**100.0 (8)**	84.77 (191)	89.97 (63)	**100.0 (851)**	40.41 (42)	99.96 (66)	99.78 (22)
Wine	**98.32 (63)**	94.92 (0.9)	93.83 (3)	96.6 (11)	96.6 (3)	96.59 (434)	55.57 (50)	95.51 (38)	97.75 (9)
Zoo	**96.05 (69)**	95.05 (0.9)	92.05 (2)	82.05 (17)	**96.05 (4)**	93.1 (308)	58.71 (47)	94.05 (36)	97.05 (17)
Avg.	**86.89 (36)**	85.56 (169)	86.73 (297)	78.78 (1077)	84.55 (49)	85.9 (400)	51.29 (23)	87.39 (89)	87.83 (22)

Figures 3 and 4 show pairwise test-accuracy comparisons between sNLL-CT and the alternative methods for setting i). In all plots, sNLL-CT is reported on the y-axis, while the competing method is reported on the x-axis. Figure 3 compares sNLL-CT with OCT-H single, OCT-H grid, DTSemNet, and RADDT. Figure 4 compares sNLL-CT with ORCT, RF, and XGBoost.

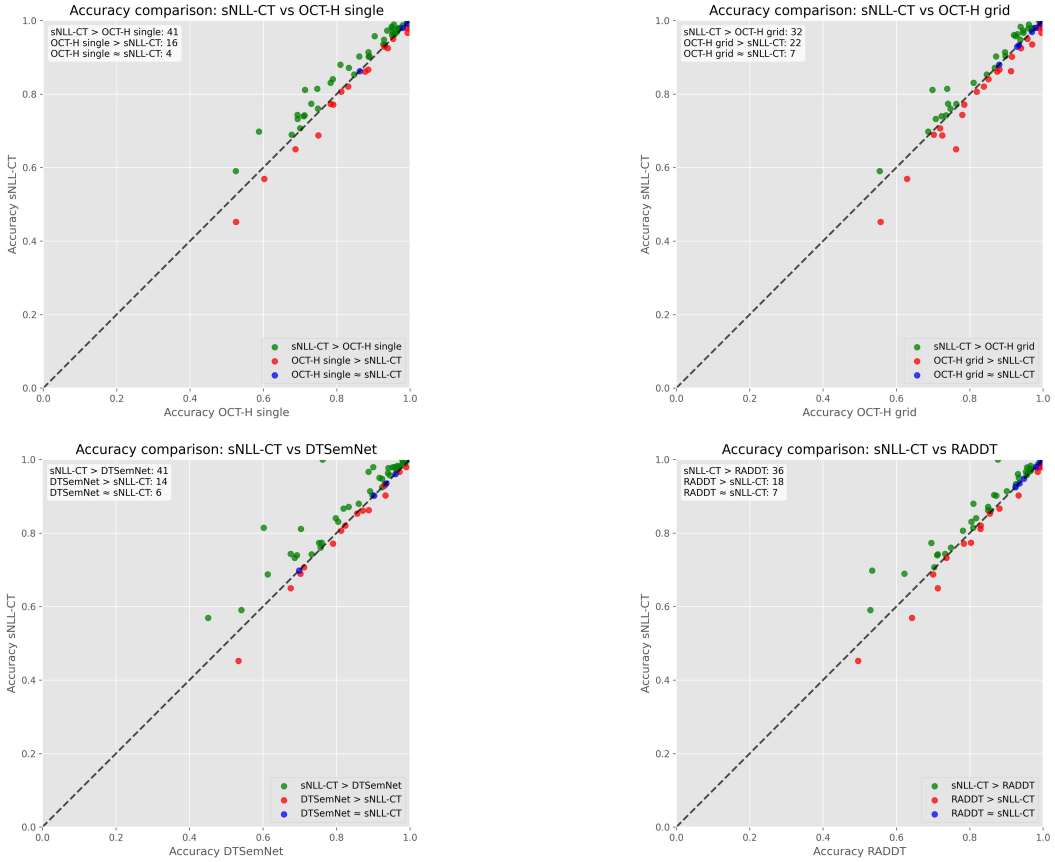


Figure 3: Comparison of test accuracy between sNLL-CT and the single-tree methods for depth setting i). From left to right and top to bottom, the plots compare sNLL-CT with OCT-H single, OCT-H grid, DTSemNet, and RADDT. Each point corresponds to one dataset. Numbers in parentheses in the legend report the count of datasets where sNLL-CT performs better (green), worse (red), or achieves the same accuracy (blue).

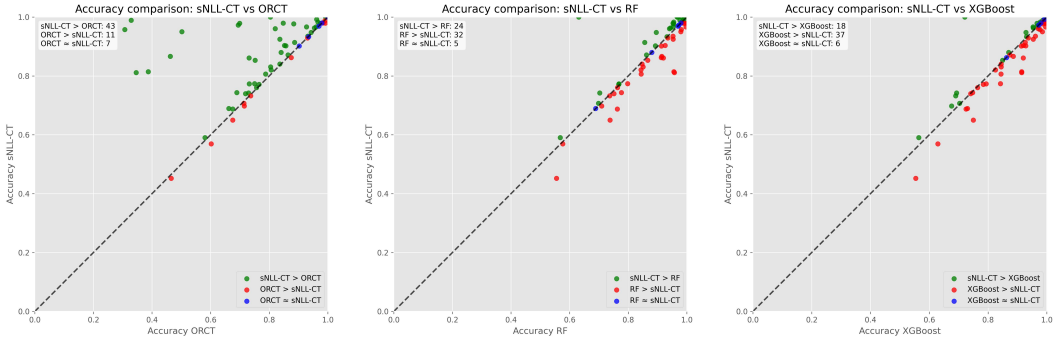


Figure 4: Comparison of test accuracy between sNLL-CT and ORCT, RF, and XGBoost for depth setting i). From left to right, the plots compare sNLL-CT with ORCT, RF, and XGBoost. Each point corresponds to one dataset. Numbers in parentheses in the legend report the count of datasets where sNLL-CT performs better (green), worse (red), or achieves the same accuracy (blue).

Table 8: Paired two-sided Wilcoxon signed-rank tests on test accuracy across 61 datasets for depth setting i). For each comparison, the better-performing method is shown in bold. Raw and Holm-adjusted p-values are reported, with comparisons ordered by statistical significance. Results above the dividing line are significant at the 95% confidence level. Sig is excluded due to its very poor performance.

Comparison setting i)	**p-value**	**Adjusted p-value**
XGBoost vs. DTSemNet	1.76e-7	4.92e-6
XGBoost vs. OCT-H single	3.50e-7	9.46e-6
sNLL-CT vs. ORCT	6.10e-7	1.59e-5
XGBoost vs. ORCT	1.26e-6	3.16e-5
OCT-H grid vs. ORCT	3.07e-6	7.38e-5
XGBoost vs. RADDT	1.20e-5	2.75e-4
sNLL-CT vs. DTSemNet	5.05e-5	0.001
RF vs. ORCT	6.12e-5	0.001
RF vs. OCT-H single	2.19e-4	0.004
RF vs. DTSemNet	2.26e-4	0.004
OCT-H grid vs. DTSemNet	5.40e-4	0.010
OCT-H grid vs. OCT-H single	8.12e-4	0.014
sNLL-CT vs. OCT-H single	0.002	0.031
XGBoost vs. sNLL-CT	0.002	0.031
RADDT vs. ORCT	0.003	0.043
XGBoost vs. OCT-H grid	0.006	0.075
RF vs. RADDT	0.006	0.077
OCT-H single vs. ORCT	0.022	0.237
DTSemNet vs. ORCT	0.025	0.247
sNLL-CT vs. RADDT	0.030	0.269
RADDT vs. DTSemNet	0.048	0.383
RF vs. OCT-H grid	0.084	0.588
OCT-H grid vs. RADDT	0.111	0.667
RF vs. sNLL-CT	0.126	0.667
XGBoost vs. RF	0.195	0.779
RADDT vs. OCT-H single	0.289	0.866
sNLL-CT vs. OCT-H grid	0.467	0.934
OCT-H single vs. DTSemNet	0.576	0.934

D.2 RESULT FOR DEPTH SETTING II)

In this subsection, we report the results obtained by setting the maximum depth of the single-tree methods to $D = 3$ for all datasets with $K \leq 8$ and to $D = 4$ otherwise. In particular, Table 10 reports the results of the paired two-sided Wilcoxon signed-rank test across the 61 datasets, while Table 9 presents the test accuracy and average training time of all models for each dataset.

Table 9: Average classification accuracy and, in parentheses, training time in seconds for each method on each dataset for depth setting ii). Bold indicates, for each dataset, the best test accuracy among the single-tree methods.

Dataset	sNLL-CT	OCT-H single	OCT-H grid	ORCT	DTSemNet	RADDT	Sigmoid	RF	XGBoost
Acute-inflammations-1	**100.0 (29)**	**100.0 (0.1)**	99.17 (0.2)	**100.0 (3)**	**100.0 (2)**	**100.0 (749)**	58.33 (6)	98.33 (35)	100.0 (4)
Acute-inflammations-2	**100.0 (28)**	99.17 (0.1)	99.17 (0.2)	**100.0 (3)**	**100.0 (2)**	**100.0 (725)**	50.83 (6)	96.67 (35)	100.0 (5)
Balance-scale	**91.36 (70)**	88.64 (3)	89.6 (5)	88.64 (19)	89.12 (8)	90.08 (290)	49.76 (39)	85.6 (41)	92.64 (22)
Banknote-authentication	99.78 (26)	99.71 (5)	**100.0 (9)**	99.2 (38)	99.12 (12)	99.85 (273)	55.54 (6)	99.13 (55)	99.71 (6)
Blood-transfusion	77.15 (14)	77.42 (3)	78.21 (3)	77.69 (18)	**79.02 (7)**	78.49 (440)	76.21 (14)	76.74 (43)	78.35 (14)
Breast-cancer	72.56 (8)	69.33 (3)	71.45 (4)	**75.08 (9)**	70.42 (4)	69.29 (314)	70.76 (22)	73.63 (38)	68.95 (9)
Breast-cancer-diagnostic	97.01 (11)	95.25 (9)	95.25 (22)	**97.36 (49)**	97.01 (5)	96.48 (286)	62.74 (6)	95.25 (52)	96.66 (7)
Breast-cancer-prognostic	75.22 (9)	72.67 (4)	74.17 (9)	72.12 (16)	68.52 (2)	71.08 (283)	**76.28 (6)**	76.79 (41)	79.34 (7)
Car-evaluation	**97.69 (64)**	96.87 (54)	96.82 (95)	88.31 (66)	94.1 (21)	96.24 (309)	70.02 (47)	95.48 (48)	98.38 (30)
Chess-king-rook-versus-king-pawn	**99.22 (25)**	98.81 (213)	**99.22 (472)**	96.18 (369)	98.69 (26)	99.09 (269)	47.78 (6)	98.94 (60)	99.12 (15)
Climate-model-crashes	90.56 (8)	88.52 (6)	90.93 (9)	90.74 (33)	90.37 (5)	89.81 (551)	**91.48 (6)**	91.3 (50)	91.3 (8)
Congressional-voting-records	93.52 (14)	91.8 (1)	**96.98 (3)**	91.8 (11)	91.79 (4)	92.23 (267)	53.48 (6)	95.24 (36)	96.11 (5)
Connectionist-bench	69.49 (77)	67.17 (50)	64.04 (89)	22.83 (221)	52.53 (22)	**71.52 (534)**	6.57 (53)	95.56 (77)	91.41 (59)
Connectionist-bench-sonar	78.39 (12)	76.47 (8)	75.95 (22)	74.53 (37)	79.85 (4)	**83.64 (279)**	74.53 (8)	79.78 (44)	84.18 (7)
Contraceptive-method-choice	45.22 (81)	52.55 (62)	**55.67 (93)**	46.5 (73)	53.29 (16)	49.56 (389)	44.94 (44)	55.53 (56)	55.39 (59)
Credit-approval	**86.83 (8)**	81.46 (27)	86.52 (42)	86.37 (28)	82.08 (7)	83.46 (298)	54.67 (6)	86.22 (43)	87.44 (9)
Cylinder-bands	72.16 (8)	74.36 (28)	74.71 (89)	64.23 (40)	69.34 (4)	**76.89 (284)**	64.23 (6)	77.63 (41)	74.77 (9)
Dermatology	95.63 (53)	93.7 (7)	**96.99 (20)**	71.0 (31)	96.17 (7)	96.72 (300)	14.21 (36)	97.26 (39)	97.26 (19)
Echocardiogram	70.38 (8)	70.29 (0.3)	**72.95 (0.5)**	64.76 (2)	67.71 (2)	52.48 (387)	70.19 (7)	68.86 (36)	72.95 (5)
Fertility	86.0 (11)	83.0 (0.4)	**88.0 (0.5)**	82.0 (3)	82.0 (2)	81.0 (273)	**88.0 (21)**	88.0 (36)	87.0 (5)
Haberman-survival	73.88 (17)	68.98 (0.8)	**74.52 (0.9)**	74.19 (7)	73.87 (4)	68.63 (532)	26.46 (6)	70.26 (38)	69.29 (9)
Hayes–roth	80.62 (68)	81.25 (0.9)	**81.88 (2)**	78.75 (6)	81.25 (3)	78.12 (292)	58.75 (36)	84.38 (37)	84.38 (17)
Heart-disease-Cleveland	**59.09 (74)**	54.44 (5)	56.77 (8)	55.78 (16)	56.45 (6)	49.19 (305)	54.13 (48)	56.75 (40)	56.44 (24)
Hepatitis	90.0 (10)	87.5 (0.4)	**91.25 (0.6)**	88.75 (3)	87.5 (2)	81.25 (368)	**91.25 (6)**	91.25 (36)	86.25 (5)
Image-segmentation	82.86 (55)	83.33 (3)	**89.05 (8)**	53.33 (21)	68.1 (7)	79.05 (331)	14.29 (43)	91.43 (41)	88.57 (21)
Indian-liver-patient	71.35 (14)	68.27 (7)	69.99 (7)	**71.52 (20)**	70.14 (5)	68.77 (487)	71.35 (6)	69.81 (46)	70.32 (9)
Ionosphere	91.18 (10)	88.32 (8)	89.46 (20)	**92.31 (38)**	88.88 (4)	91.74 (285)	35.89 (6)	92.59 (47)	92.02 (7)
Iris	**96.67 (49)**	95.33 (0.3)	94.67 (0.5)	96.0 (5)	88.67 (3)	95.33 (512)	66.67 (38)	95.33 (36)	95.33 (10)
Lymphography	**83.08 (44)**	78.32 (4)	81.03 (9)	80.41 (12)	80.41 (3)	80.41 (280)	58.92 (34)	85.08 (38)	84.44 (16)
Mammographic-mass	83.01 (16)	82.05 (4)	**83.25 (4)**	80.0 (20)	83.13 (8)	81.33 (783)	79.64 (7)	84.34 (43)	82.53 (14)
Monks-problems-1	**100.0 (23)**	99.46 (2)	**100.0 (5)**	**100.0 (18)**	98.02 (5)	**100.0 (285)**	92.09 (14)	96.94 (40)	99.82 (7)
Monks-problems-2	**100.0 (26)**	99.0 (4)	99.83 (8)	84.36 (17)	92.0 (6)	99.17 (292)	68.88 (8)	63.06 (42)	72.05 (12)
Monks-problems-3	98.38 (13)	98.38 (2)	97.84 (4)	**98.92 (16)**	**98.92 (5)**	98.56 (281)	**98.92 (19)**	98.56 (38)	98.92 (6)
Optical-recognition	**94.93 (69)**	90.22 (2603)	91.21 (4770)	42.3 (4518)	93.33 (81)	94.48 (356)	10.91 (39)	98.09 (131)	97.7 (70)
Ozone-level-detection-eight	**93.23 (12)**	92.48 (80)	**93.23 (144)**	93.07 (427)	92.47 (17)	93.12 (585)	93.07 (7)	93.5 (118)	93.67 (19)
Ozone-level-detection-one	**96.91 (14)**	95.94 (59)	**96.91 (97)**	**96.91 (436)**	95.83 (17)	96.32 (634)	**96.91 (8)**	96.91 (101)	96.75 (15)
Parkinsons	**92.31 (16)**	87.69 (3)	89.74 (6)	83.08 (11)	87.18 (2)	**92.31 (462)**	24.62 (6)	89.23 (39)	92.82 (5)
Pen-based_recognition	**96.37 (77)**	95.13 (1780)	94.7 (3033)	47.41 (4677)	90.34 (152)	88.52 (548)	11.78 (53)	99.05 (208)	99.25 (86)
Pima-Indians-diabetes	74.35 (10)	72.27 (8)	73.57 (11)	**75.91 (23)**	75.52 (7)	75.52 (327)	65.11 (6)	76.3 (47)	76.43 (10)
Planning-relax	72.03 (15)	57.13 (1)	71.46 (2)	**73.65 (7)**	64.85 (2)	59.94 (319)	71.46 (24)	70.92 (40)	67.6 (6)

Continued on next page

Dataset	sNLL-CT	OCT-H single	OCT-H grid	ORCT	DTSemNet	RADDT	Sigmoid	RF	XGBoost
Qsar-biodegradation	**86.45 (11)**	82.65 (50)	84.08 (102)	74.22 (71)	85.4 (10)	84.45 (286)	33.74 (6)	86.64 (62)	84.93 (14)
Seeds	**94.76 (46)**	92.86 (0.9)	93.81 (2)	94.29 (8)	92.38 (5)	**94.76 (364)**	66.19 (37)	89.52 (38)	92.86 (10)
Segment	93.42 (85)	**93.68 (89)**	**93.68 (164)**	55.89 (251)	82.12 (38)	91.26 (275)	12.55 (47)	97.92 (97)	98.48 (47)
Seismic-bumps	93.34 (17)	92.11 (43)	**93.42 (52)**	93.03 (108)	92.92 (22)	91.8 (766)	**93.42 (6)**	93.19 (67)	93.07 (16)
Skin-segmentation	99.77 (34)	99.74 (3796)	99.68 (5907)	99.51 (14788)	99.72 (2004)	**99.83 (974)**	79.25 (7)	99.95 (1625)	99.96 (96)
Soybean-small	98.0 (74)	98.0 (0.1)	98.0 (0.4)	98.0 (3)	95.78 (2)	**100.0 (665)**	98.0 (76)	98.0 (36)	98.0 (4)
Spambase	92.5 (25)	93.35 (503)	**93.96 (1019)**	90.05 (285)	92.33 (39)	93.2 (298)	60.6 (6)	95.33 (141)	95.35 (50)
Spect-heart	72.5 (9)	66.25 (0.6)	**75.0 (1)**	63.75 (3)	67.5 (2)	71.25 (298)	50.0 (6)	73.75 (36)	75.0 (6)
Spectf-heart	73.75 (10)	75.0 (1)	68.75 (4)	75.0 (10)	68.75 (2)	**77.5 (424)**	45.0 (7)	76.25 (37)	72.5 (4)
Statlog-project-German-credit	**73.7 (8)**	69.3 (91)	72.5 (145)	72.3 (59)	68.1 (9)	69.8 (286)	70.0 (7)	75.1 (51)	74.1 (13)
Statlog-project-landsat-satellite	**85.94 (62)**	85.66 (806)	85.35 (1504)	65.01 (1522)	85.07 (95)	85.46 (327)	13.28 (36)	91.96 (253)	91.98 (99)
Teaching-assistant-evaluation	56.92 (50)	60.22 (5)	62.9 (11)	60.19 (8)	45.08 (3)	**64.24 (296)**	33.1 (32)	57.66 (38)	62.88 (24)
Texture	**98.36 (70)**	95.02 (1295)	94.93 (2659)	31.33 (7434)	97.27 (121)	90.47 (389)	8.22 (50)	97.6 (411)	98.58 (146)
Thoracic-surgery	82.13 (8)	77.87 (9)	**85.11 (9)**	82.98 (16)	80.85 (5)	79.79 (302)	**85.11 (6)**	84.68 (41)	84.47 (9)
Thyroid-disease-ann-thyroid	96.66 (70)	99.28 (75)	**99.58 (138)**	92.71 (184)	97.11 (37)	98.54 (912)	3.92 (35)	99.68 (60)	99.52 (20)
Thyroid-disease-new-thyroid	**96.28 (66)**	95.81 (0.6)	93.02 (1)	95.35 (7)	93.95 (4)	95.81 (294)	61.4 (37)	94.42 (38)	95.35 (10)
Tic-tac-toe-endgame	97.6 (12)	96.56 (36)	97.7 (70)	**98.33 (31)**	87.37 (9)	96.87 (292)	65.33 (6)	98.43 (45)	99.37 (8)
Vowel	**75.35 (84)**	64.85 (50)	67.78 (90)	25.56 (222)	64.14 (22)	70.71 (480)	9.19 (47)	95.76 (78)	91.41 (60)
Wall-following-robot-2	97.91 (61)	**100.0 (4)**	**100.0 (8)**	84.77 (191)	89.97 (63)	**100.0 (851)**	40.41 (42)	99.96 (66)	99.78 (22)
Wine	**98.32 (63)**	94.92 (0.9)	93.83 (3)	96.6 (11)	96.6 (3)	96.59 (434)	55.57 (50)	95.51 (38)	97.75 (9)
Zoo	95.05 (55)	82.19 (0.6)	89.05 (1)	84.19 (7)	92.1 (4)	**96.05 (307)**	27.9 (43)	94.05 (36)	97.05 (17)
Avg.	**86.67 (35)**	84.72 (195)	86.37 (345)	78.51 (600)	83.65 (49)	85.21 (414)	55.47 (22)	87.39 (89)	87.83 (22)

Figures 5 and 6 show pairwise test-accuracy comparisons between sNLL-CT and the alternative methods for setting ii). In all plots, sNLL-CT is reported on the y-axis, while the competing method is reported on the x-axis. Figure 5 compares sNLL-CT with OCT-H single, OCT-H grid, DTSemNet, and RADDT. Figure 6 compares sNLL-CT with ORCT, RF, and XGBoost.

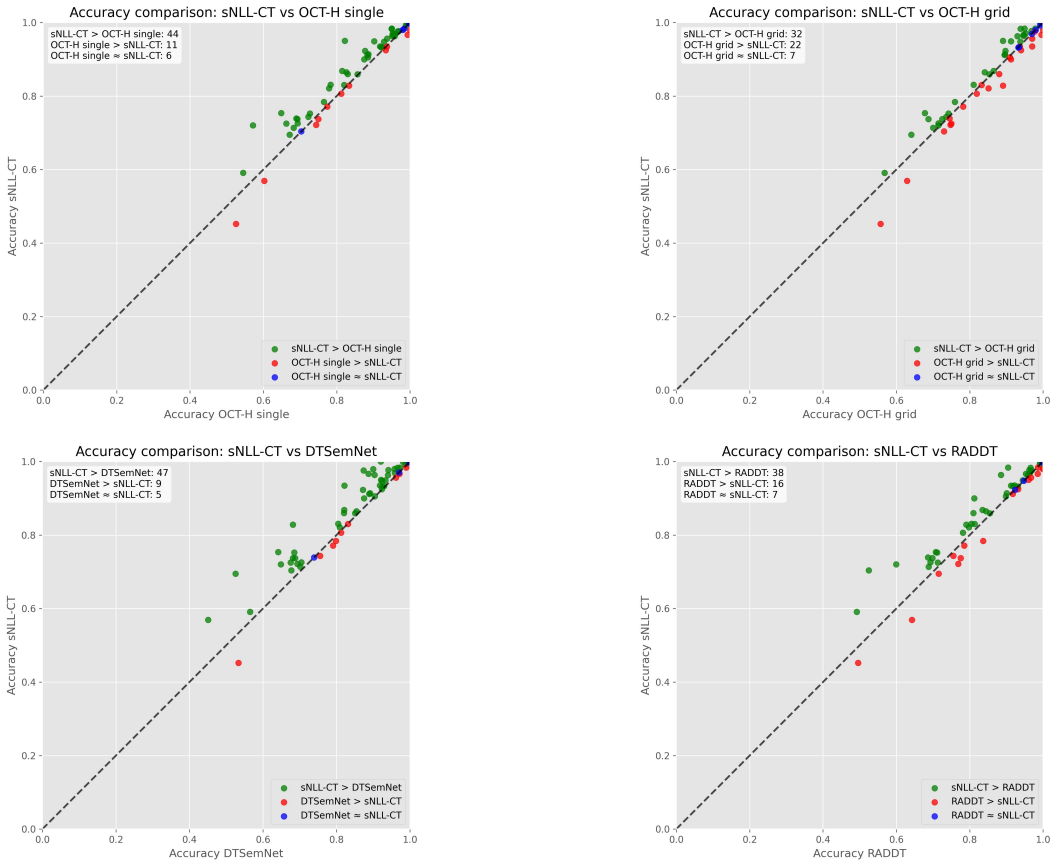


Figure 5: Comparison of test accuracy between sNLL-CT and the single-tree methods for depth setting ii). From left to right and top to bottom, the plots compare sNLL-CT with OCT-H single, OCT-H grid, DTSemNet, and RADDT. Each point corresponds to one dataset. Numbers in parentheses in the legend report the count of datasets where sNLL-CT performs better (green), worse (red), or achieves the same accuracy (blue).

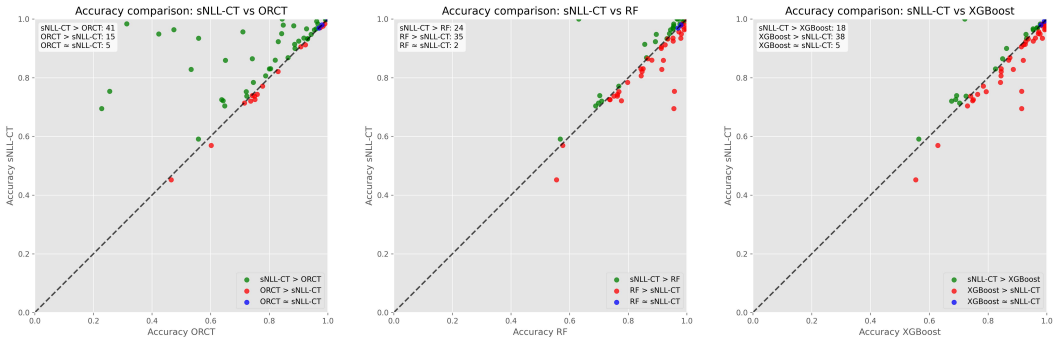


Figure 6: Comparison of test accuracy between sNLL-CT and ORCT, RF, and XGBoost for depth setting ii). From left to right, the plots compare sNLL-CT with ORCT, RF, and XGBoost. Each point corresponds to one dataset. Numbers in parentheses in the legend report the count of datasets where sNLL-CT performs better (green), worse (red), or achieves the same accuracy (blue).

D.3 Result for depth setting iii)

In this subsection, we report the results obtained by setting the maximum depth of the single-tree methods to $D = 4$ for all datasets. In particular, Table 12 reports the results of the paired two-sided Wilcoxon signed-rank test across the 61 datasets, while Table 11 presents the test accuracy and average training time of all models for each dataset.

Table 11: Average classification accuracy and, in parentheses, training time in seconds for each method on each dataset for depth setting iii). Bold indicates, for each dataset, the best test accuracy among the single-tree methods.

Dataset	sNLL-CT	OCT-H single	OCT-H grid	ORCT	DTSemNet	RADDT	Sigmoid	RF	XGBoost
Acute-inflammations-1	**100.0 (29)**	**100.0 (0.1)**	99.17 (0.2)	**100.0 (7)**	**100.0 (2)**	**100.0 (810)**	**100.0 (21)**	98.33 (35)	100.0 (4)
Acute-inflammations-2	**100.0 (29)**	99.17 (0.1)	**100.0 (0.2)**	**100.0 (8)**	**100.0 (2)**	**100.0 (795)**	49.17 (6)	96.67 (35)	100.0 (5)
Balance-scale	**91.68 (82)**	89.12 (4)	90.88 (6)	90.4 (67)	89.92 (8)	91.52 (332)	**91.68 (38)**	85.6 (41)	92.64 (22)
Banknote-authentication	99.71 (20)	99.64 (5)	**100.0 (9)**	99.64 (139)	99.71 (12)	99.93 (311)	98.91 (18)	99.13 (55)	99.71 (6)
Blood-transfusion	77.69 (15)	76.48 (3)	79.02 (4)	77.02 (59)	**79.15 (7)**	78.89 (579)	76.21 (6)	76.74 (43)	78.35 (14)
Breast-cancer	70.41 (9)	66.42 (4)	71.81 (6)	**72.91 (27)**	68.58 (4)	66.04 (369)	70.76 (6)	73.63 (38)	68.95 (9)
Breast-cancer-diagnostic	97.01 (10)	95.95 (10)	95.43 (23)	97.36 (166)	95.96 (5)	96.66 (649)	**98.07 (12)**	95.25 (52)	96.66 (7)
Breast-cancer-prognostic	75.75 (8)	67.53 (4)	73.74 (12)	72.12 (44)	69.08 (2)	67.52 (337)	**76.28 (24)**	76.79 (41)	79.34 (7)
Car-evaluation	96.24 (44)	96.7 (56)	**98.67 (110)**	89.35 (266)	96.53 (21)	96.76 (346)	88.43 (49)	95.48 (48)	98.38 (30)
Chess-king-rook-versus-king-pawn	99.09 (25)	98.78 (195)	**99.53 (485)**	96.65 (1370)	98.9 (26)	99.0 (305)	77.67 (17)	98.94 (60)	99.12 (15)
Climate-model-crashes	91.11 (8)	88.89 (8)	**91.48 (10)**	89.44 (113)	89.07 (5)	88.33 (574)	**91.48 (18)**	91.3 (50)	91.3 (8)
Congressional-voting-records	93.52 (12)	93.52 (1)	**96.54 (4)**	90.07 (33)	90.93 (4)	94.83 (415)	77.63 (7)	95.24 (36)	96.11 (5)
Connectionist-bench	69.49 (77)	67.17 (50)	64.04 (89)	22.83 (221)	52.53 (22)	**71.52 (534)**	6.57 (53)	95.56 (77)	91.41 (59)
Connectionist-bench-sonar	80.75 (13)	69.72 (9)	72.1 (30)	75.99 (129)	78.87 (4)	**83.19 (331)**	51.84 (10)	79.78 (44)	84.18 (7)
Contraceptive-method-choice	45.96 (69)	52.68 (79)	**55.33 (130)**	45.28 (325)	49.97 (16)	49.15 (463)	43.58 (36)	55.53 (56)	55.39 (59)
Credit-approval	85.15 (8)	82.85 (31)	**86.21 (44)**	85.3 (127)	81.62 (7)	80.86 (370)	45.33 (6)	86.22 (43)	87.44 (9)
Cylinder-bands	67.82 (8)	74.0 (25)	74.33 (105)	**75.08 (143)**	66.45 (4)	74.39 (331)	64.23 (7)	77.63 (41)	74.77 (9)
Dermatology	**97.26 (58)**	93.71 (8)	94.53 (20)	69.39 (93)	96.45 (7)	96.72 (348)	30.57 (36)	97.26 (39)	97.26 (19)
Echocardiogram	65.05 (8)	62.19 (0.3)	71.52 (0.5)	64.86 (7)	**72.95 (2)**	62.19 (613)	67.33 (6)	68.86 (36)	72.95 (5)
Fertility	85.0 (8)	80.0 (0.4)	**88.0 (0.6)**	82.0 (9)	86.0 (2)	81.0 (314)	24.0 (10)	88.0 (36)	87.0 (5)
Haberman-survival	71.27 (21)	69.61 (0.9)	**73.54 (1)**	71.25 (21)	72.23 (4)	68.97 (726)	72.55 (14)	70.26 (38)	69.29 (9)
Hayes–roth	79.38 (56)	81.25 (1)	**83.75 (2)**	78.75 (16)	77.5 (3)	76.88 (346)	40.0 (32)	84.38 (37)	84.38 (17)
Heart-disease-Cleveland	**59.07 (62)**	52.46 (6)	55.45 (9)	58.07 (46)	54.13 (6)	52.87 (419)	11.89 (34)	56.75 (40)	56.44 (24)

Continued on next page

Dataset	sNLL-CT	OCT-H single	OCT-H grid	ORCT	DTSemNet	RADDT	Sigmoid	RF	XGBoost
Hepatitis	86.25 (8)	81.25 (0.4)	**91.25 (0.7)**	88.75 (8)	82.5 (2)	85.0 (412)	**91.25 (6)**	91.25 (36)	86.25 (5)
Image-segmentation	86.67 (58)	**88.57 (5)**	88.1 (11)	46.19 (56)	81.9 (7)	88.1 (338)	26.67 (38)	91.43 (41)	88.57 (21)
Indian-liver-patient	**71.7 (12)**	68.09 (8)	71.35 (8)	70.5 (55)	69.98 (5)	70.15 (599)	71.35 (6)	69.81 (46)	70.32 (9)
Ionosphere	88.9 (11)	88.6 (10)	**90.02 (27)**	88.61 (130)	86.89 (4)	87.47 (325)	64.11 (6)	92.59 (47)	92.02 (7)
Iris	**96.67 (55)**	95.33 (0.2)	94.67 (0.5)	**96.67 (13)**	92.0 (3)	95.33 (748)	90.0 (38)	95.33 (36)	95.33 (10)
Lymphography	83.79 (48)	74.3 (3)	79.01 (9)	77.72 (45)	79.7 (3)	**85.1 (324)**	62.76 (36)	85.08 (38)	84.44 (16)
Mammographic-mass	83.13 (15)	80.48 (4)	**83.37 (6)**	80.96 (81)	**83.37 (8)**	80.72 (922)	48.55 (6)	84.34 (43)	82.53 (14)
Monks-problems-1	**100.0 (28)**	99.46 (2)	**100.0 (5)**	**100.0 (54)**	**100.0 (5)**	**100.0 (328)**	49.99 (6)	96.94 (40)	99.82 (7)
Monks-problems-2	**100.0 (28)**	98.33 (4)	99.5 (9)	79.69 (49)	85.84 (6)	**100.0 (340)**	34.27 (6)	63.06 (42)	72.05 (12)
Monks-problems-3	98.74 (13)	98.38 (2)	**98.92 (4)**	**98.92 (53)**	98.19 (5)	98.38 (758)	52.0 (6)	98.56 (38)	98.92 (6)
Optical-recognition	**94.93 (69)**	90.22 (2603)	91.21 (4770)	42.3 (4518)	93.33 (81)	94.48 (356)	10.91 (39)	98.09 (131)	97.7 (70)
Ozone-level-detection-eight	**93.07 (14)**	91.77 (110)	92.8 (239)	92.91 (1724)	92.09 (17)	92.31 (687)	**93.07 (7)**	93.5 (118)	93.67 (19)
Ozone-level-detection-one	**96.91 (12)**	96.05 (75)	**96.91 (116)**	**96.91 (1713)**	95.72 (17)	96.54 (701)	**96.91 (6)**	96.91 (101)	96.75 (15)
Parkinsons	90.26 (10)	89.23 (3)	90.77 (7)	89.23 (32)	87.69 (2)	**91.28 (732)**	24.62 (6)	89.23 (39)	92.82 (5)
Pen-based_recognition	**96.37 (77)**	95.13 (1780)	94.7 (3033)	47.41 (4677)	90.34 (152)	88.52 (548)	11.78 (53)	99.05 (208)	99.25 (86)
Pima-Indians-diabetes	75.65 (10)	69.8 (10)	74.48 (14)	**76.17 (81)**	74.09 (7)	74.35 (550)	34.89 (6)	76.3 (47)	76.43 (10)
Planning-relax	**72.01 (13)**	60.0 (2)	71.46 (2)	69.26 (19)	66.47 (2)	49.52 (406)	71.46 (6)	70.92 (40)	67.6 (6)
Qsar-biodegradation	86.45 (12)	82.18 (62)	83.79 (129)	**87.2 (269)**	82.94 (10)	84.08 (335)	66.26 (6)	86.64 (62)	84.93 (14)
Seeds	93.33 (44)	90.48 (0.9)	93.81 (2)	**94.76 (24)**	93.33 (5)	**94.76 (484)**	33.33 (33)	89.52 (38)	92.86 (10)
Segment	95.02 (98)	95.37 (145)	**95.76 (286)**	50.17 (1069)	91.69 (38)	93.38 (321)	26.1 (60)	97.92 (97)	98.48 (47)
Seismic-bumps	93.34 (13)	90.67 (57)	**93.42 (67)**	**93.42 (391)**	92.72 (22)	91.8 (836)	**93.42 (6)**	93.19 (67)	93.07 (16)
Skin-segmentation	99.78 (45)	99.82 (4579)	99.83 (7450)	99.43 (44325)	99.65 (2007)	**99.9 (1070)**	79.25 (8)	99.95 (1625)	99.96 (96)
Soybean-small	95.78 (86)	98.0 (0.1)	98.0 (0.4)	98.0 (6)	96.0 (2)	**100.0 (744)**	98.0 (67)	98.0 (36)	98.0 (4)
Spambase	92.11 (17)	93.33 (601)	**93.94 (1242)**	90.05 (824)	92.76 (39)	93.07 (368)	60.6 (6)	95.33 (141)	95.35 (50)
Spect-heart	61.25 (11)	67.5 (0.6)	**75.0 (1)**	70.0 (7)	66.25 (2)	72.5 (353)	50.0 (6)	73.75 (36)	75.0 (6)
Spectf-heart	65.0 (10)	61.25 (1)	67.5 (5)	76.25 (28)	**80.0 (2)**	63.75 (685)	50.0 (6)	76.25 (37)	72.5 (4)
Statlog-project-German-credit	**74.2 (8)**	71.3 (118)	71.6 (181)	72.0 (358)	68.0 (9)	67.0 (338)	30.0 (6)	75.1 (51)	74.1 (13)
Statlog-project-landsat-satellite	86.1 (77)	**87.71 (930)**	87.47 (1828)	73.08 (6226)	87.09 (94)	85.71 (315)	43.09 (58)	91.96 (253)	91.98 (99)
Teaching-assistant-evaluation	56.26 (55)	**64.22 (6)**	**64.22 (12)**	53.66 (20)	45.03 (3)	60.88 (375)	52.84 (51)	57.66 (38)	62.88 (24)
Texture	**98.36 (70)**	95.02 (1295)	94.93 (2659)	31.33 (7434)	97.27 (121)	90.47 (389)	8.22 (50)	97.6 (411)	98.58 (146)
Thoracic-surgery	81.91 (8)	77.02 (10)	**85.11 (11)**	81.91 (49)	80.43 (5)	79.57 (373)	**85.11 (11)**	84.68 (41)	84.47 (9)
Thyroid-disease-ann-thyroid	96.37 (83)	99.44 (74)	**99.66 (144)**	92.76 (545)	97.45 (38)	98.73 (984)	93.35 (46)	99.68 (60)	99.52 (20)
Thyroid-disease-new-thyroid	**96.28 (64)**	92.56 (0.6)	94.88 (1)	95.81 (21)	95.81 (4)	94.88 (479)	39.07 (36)	94.42 (38)	95.35 (10)
Tic-tac-toe-endgame	91.34 (14)	94.05 (40)	97.29 (83)	**98.33 (136)**	84.45 (9)	96.66 (373)	65.33 (8)	98.43 (45)	99.37 (8)
Vowel	**75.35 (84)**	64.85 (50)	67.78 (90)	25.56 (222)	64.14 (22)	70.71 (480)	9.19 (47)	95.76 (78)	91.41 (60)
Wall-following-robot-2	97.82 (65)	**100.0 (3)**	**100.0 (8)**	87.28 (635)	95.33 (66)	**100.0 (988)**	87.13 (70)	99.96 (66)	99.78 (22)
Wine	**97.71 (67)**	94.92 (0.8)	92.7 (3)	96.05 (32)	97.21 (3)	93.84 (665)	26.98 (34)	95.51 (38)	97.75 (9)
Zoo	**96.05 (69)**	95.05 (0.9)	92.05 (2)	82.05 (17)	**96.05 (4)**	93.1 (308)	58.71 (47)	94.05 (36)	97.05 (17)
Avg.	85.95 (35)	84.22 (215)	**86.43 (386)**	78.61 (1301)	84.1 (49)	84.91 (507)	58.11 (23)	87.39 (89)	87.83 (22)

Table 10: Paired two-sided Wilcoxon signed-rank tests on test accuracy across 61 datasets for depth setting ii). For each comparison, the better-performing method is shown in bold. Raw and Holm-adjusted p-values are reported, with comparisons ordered by statistical significance. Results above the dividing line are significant at the 95% confidence level. Sig is excluded due to its very poor performance.

Comparison setting ii)	**p-value**	**Adjusted p-value**
XGBoost vs. OCT-H single	2.08e-8	5.82e-7
sNLL-CT vs. DTSemNet	5.49e-8	1.48e-6
XGBoost vs. DTSemNet	9.38e-8	2.44e-6
XGBoost vs. RADDT	3.15e-6	7.87e-5
sNLL-CT vs. ORCT	3.60e-6	8.64e-5
sNLL-CT vs. OCT-H single	4.33e-6	9.95e-5
OCT-H grid vs. OCT-H single	4.38e-6	9.95e-5
RF vs. OCT-H single	6.40e-6	1.34e-4
RF vs. DTSemNet	6.80e-6	1.36e-4
XGBoost vs. ORCT	8.76e-6	1.67e-4
OCT-H grid vs. DTSemNet	3.51e-5	6.33e-4
OCT-H grid vs. ORCT	4.73e-5	8.04e-4
RF vs. ORCT	1.61e-4	0.003
XGBoost vs. OCT-H grid	0.002	0.032
RF vs. RADDT	0.003	0.046
XGBoost vs. sNLL-CT	0.004	0.047
sNLL-CT vs. RADDT	0.005	0.059
RADDT vs. ORCT	0.009	0.104
RADDT vs. DTSemNet	0.023	0.226
RADDT vs. OCT-H single	0.042	0.381
DTSemNet vs. ORCT	0.045	0.381
OCT-H single vs. ORCT	0.064	0.445
RF vs. OCT-H grid	0.071	0.445
OCT-H grid vs. RADDT	0.123	0.613
RF vs. sNLL-CT	0.145	0.613
XGBoost vs. RF	0.195	0.613
sNLL-CT vs. OCT-H grid	0.217	0.613
OCT-H single vs. DTSemNet	0.533	0.613

Table 12: Paired two-sided Wilcoxon signed-rank tests on test accuracy across 61 datasets for depth setting iii). For each comparison, the better-performing method is shown in bold. Raw and Holm-adjusted p-values are reported, with comparisons ordered by statistical significance. Results above the dividing line are significant at the 95% confidence level. Sig is excluded due to its very poor performance.

Comparison setting iii)	**p-value**	**Adjusted p-value**
XGBoost vs. OCT-H single	2.73e-9	7.65e-8
XGBoost vs. RADDT	8.80e-8	2.37e-6
OCT-H grid vs. OCT-H single	1.73e-7	4.50e-6
XGBoost vs. DTSemNet	1.77e-7	4.50e-6
RF vs. OCT-H single	2.57e-6	6.17e-5
XGBoost vs. ORCT	1.60e-5	3.68e-4
OCT-H grid vs. DTSemNet	3.89e-5	8.57e-4
OCT-H grid vs. ORCT	4.34e-5	9.11e-4
sNLL-CT vs. DTSemNet	5.03e-5	0.001
RF vs. ORCT	6.54e-5	0.001
RF vs. DTSemNet	9.47e-5	0.002
XGBoost vs. sNLL-CT	1.39e-4	0.002
sNLL-CT vs. ORCT	1.87e-4	0.003
sNLL-CT vs. OCT-H single	4.36e-4	0.007
RF vs. RADDT	4.62e-4	0.007
OCT-H grid vs. RADDT	0.003	0.044
XGBoost vs. OCT-H grid	0.005	0.055
RF vs. sNLL-CT	0.018	0.195
sNLL-CT vs. RADDT	0.035	0.352
RADDT vs. OCT-H single	0.058	0.523
RADDT vs. ORCT	0.077	0.617
RF vs. OCT-H grid	0.132	0.925
DTSemNet vs. ORCT	0.149	0.925
XGBoost vs. RF	0.195	0.973
RADDT vs. DTSemNet	0.253	1
sNLL-CT vs. OCT-H grid	0.432	1
OCT-H single vs. DTSemNet	0.713	1
OCT-H single vs. ORCT	0.730	1

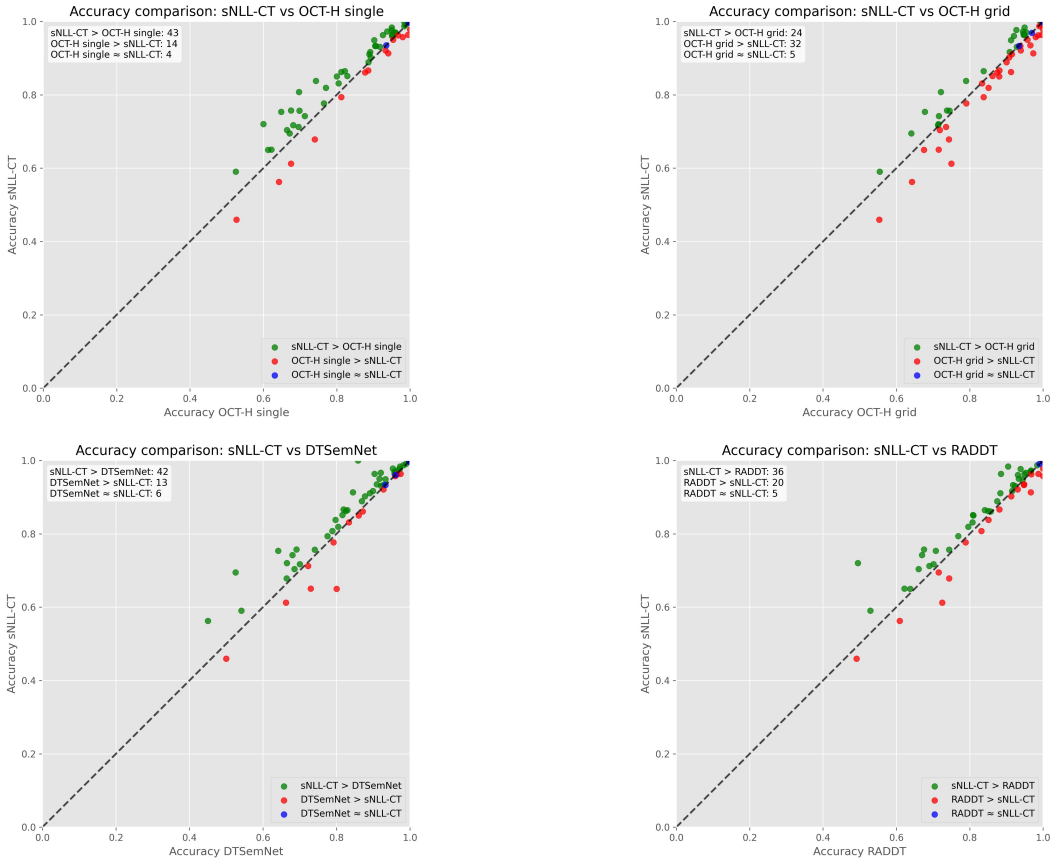


Figure 7: Comparison of test accuracy between sNLL-CT and the single-tree methods for depth setting iii). From left to right and top to bottom, the plots compare sNLL-CT with OCT-H single, OCT-H grid, DTSemNet, and RADDT. Each point corresponds to one dataset. Numbers in parentheses in the legend report the count of datasets where sNLL-CT performs better (green), worse (red), or achieves the same accuracy (blue).

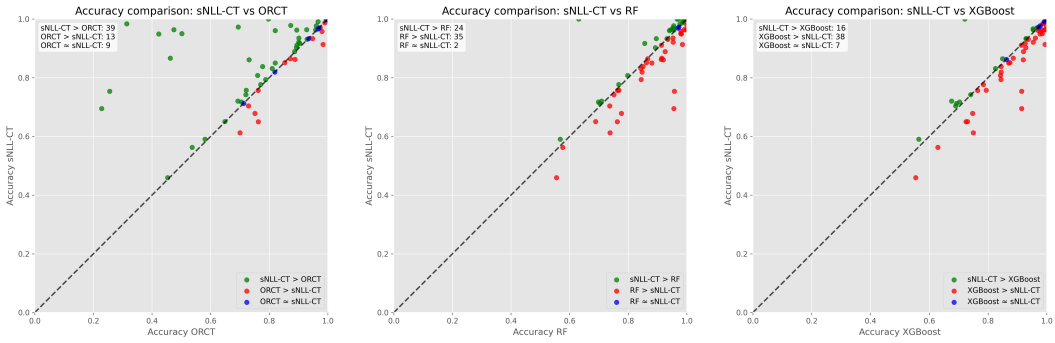


Figure 8: Comparison of test accuracy between sNLL-CT and ORCT, RF, and XGBoost for depth setting iii). From left to right, the plots compare sNLL-CT with ORCT, RF, and XGBoost. Each point corresponds to one dataset. Numbers in parentheses in the legend report the count of datasets where sNLL-CT performs better (green), worse (red), or achieves the same accuracy (blue).

Figures 7 and 8 show pairwise test-accuracy comparisons between sNLL-CT and the alternative methods for setting iii). In all plots, sNLL-CT is reported on the y-axis, while the competing method is reported on the x-axis. Figure 7 compares sNLL-CT with OCT-H single, OCT-H grid, DTSemNet, and RADDT. Figure 8 compares sNLL-CT with ORCT, RF, and XGBoost.

D.4 RESULT FOR DEPTH SETTING IV)

In this subsection, we report the results obtained by setting the maximum depth of the single-tree methods to $D = 5$ for all datasets. In particular, Table 14 reports the results of the paired two-sided Wilcoxon signed-rank test across the 61 datasets, while Table 13 presents the test accuracy and average training time of all models for each dataset.

Table 13: Average classification accuracy and, in parentheses, training time in seconds for each method on each dataset for depth setting iv). Bold indicates, for each dataset, the best test accuracy among the single-tree methods.

Dataset	sNLL-CT	OCT-H single	OCT-H grid	ORCT	DTSemNet	RADDT	Sigmoid	RF	XGBoost
Acute-inflammations-1	**100.0 (28)**	**100.0 (0.1)**	99.17 (1)	**100.0 (24)**	**100.0 (2)**	**100.0 (866)**	**100.0 (21)**	98.33 (35)	100.0 (4)
Acute-inflammations-2	**100.0 (28)**	99.17 (0.1)	**100.0 (0.2)**	**100.0 (24)**	**100.0 (2)**	**100.0 (863)**	**100.0 (23)**	96.67 (35)	100.0 (5)
Balance-scale	91.68 (70)	89.6 (4)	91.2 (8)	90.08 (173)	89.6 (8)	**92.32 (339)**	91.04 (40)	85.6 (41)	92.64 (22)
Banknote-authentication	**100.0 (23)**	99.64 (4)	**100.0 (9)**	99.78 (379)	99.71 (12)	99.85 (350)	99.05 (20)	99.13 (55)	99.71 (6)
Blood-transfusion	77.42 (17)	75.54 (3)	77.95 (5)	77.15 (154)	**78.89 (7)**	76.75 (821)	76.75 (7)	76.74 (43)	78.35 (14)
Breast-cancer	71.47 (9)	68.93 (4)	72.58 (6)	**73.64 (102)**	68.97 (4)	67.49 (736)	72.57 (8)	73.63 (38)	68.95 (9)
Breast-cancer-diagnostic	97.36 (10)	94.2 (10)	96.13 (24)	97.36 (572)	96.13 (5)	96.13 (997)	**97.89 (11)**	95.25 (52)	96.66 (7)
Breast-cancer-prognostic	**79.37 (9)**	69.03 (5)	72.66 (11)	73.67 (147)	70.08 (2)	74.71 (847)	76.28 (16)	76.79 (41)	79.34 (7)
Car-evaluation	**99.02 (59)**	97.34 (58)	98.38 (113)	90.92 (803)	96.76 (21)	98.21 (686)	70.02 (40)	95.48 (48)	98.38 (30)
Chess-king-rook-versus-king-pawn	99.37 (23)	98.78 (186)	99.5 (485)	96.15 (3648)	98.9 (26)	**99.53 (358)**	97.9 (24)	98.94 (60)	99.12 (15)
Climate-model-crashes	90.74 (8)	89.07 (10)	**91.48 (11)**	91.11 (383)	90.56 (5)	89.07 (648)	**91.48 (6)**	91.3 (50)	91.3 (8)
Congressional-voting-records	93.06 (12)	93.52 (1)	**96.54 (4)**	91.38 (74)	92.21 (4)	95.7 (794)	86.56 (8)	95.24 (36)	96.11 (5)
Connectionist-bench	**81.41 (85)**	74.65 (46)	73.84 (81)	38.69 (936)	60.2 (22)	80.91 (840)	6.87 (46)	95.56 (77)	91.41 (59)
Connectionist-bench-sonar	74.95 (9)	76.47 (9)	69.71 (34)	75.51 (376)	76.93 (4)	**80.73 (742)**	46.66 (6)	79.78 (44)	84.18 (7)
Contraceptive-method-choice	45.42 (93)	51.12 (88)	**55.33 (145)**	48.47 (1425)	48.41 (16)	48.4 (1122)	44.13 (35)	55.53 (56)	55.39 (59)
Credit-approval	**87.6 (8)**	81.47 (33)	84.38 (55)	85.9 (663)	81.77 (7)	81.78 (795)	60.9 (9)	86.22 (43)	87.44 (9)
Cylinder-bands	74.72 (8)	74.0 (24)	**76.51 (106)**	74.71 (547)	68.99 (4)	76.1 (374)	64.23 (6)	77.63 (41)	74.77 (9)
Dermatology	96.71 (61)	93.16 (7)	95.35 (21)	76.52 (305)	96.17 (7)	**98.08 (557)**	13.4 (34)	97.26 (39)	97.26 (19)
Echocardiogram	70.19 (8)	67.52 (0.3)	74.29 (0.6)	66.19 (18)	**77.05 (2)**	68.95 (698)	67.33 (6)	68.86 (36)	72.95 (5)
Fertility	79.0 (12)	74.0 (0.4)	88.0 (0.6)	87.0 (20)	**89.0 (2)**	81.0 (537)	87.0 (10)	88.0 (36)	87.0 (5)
Haberman-survival	73.54 (20)	65.7 (1)	73.54 (1)	71.25 (68)	**74.2 (4)**	69.94 (874)	73.54 (6)	70.26 (38)	69.29 (9)
Hayes–roth	81.88 (57)	81.25 (1)	**82.5 (2)**	77.5 (54)	80.0 (3)	78.12 (388)	81.88 (36)	84.38 (37)	84.38 (17)
Heart-disease-Cleveland	56.1 (63)	52.46 (7)	55.77 (11)	56.75 (168)	56.1 (6)	54.13 (869)	**58.4 (37)**	56.75 (40)	56.44 (24)
Hepatitis	87.5 (9)	81.25 (0.4)	**91.25 (0.7)**	87.5 (23)	86.25 (2)	82.5 (700)	**91.25 (22)**	91.25 (36)	86.25 (5)
Image-segmentation	**86.67 (58)**	79.05 (6)	86.19 (12)	63.33 (191)	81.9 (7)	85.24 (732)	34.76 (57)	91.43 (41)	88.57 (21)
Indian-liver-patient	71.18 (17)	69.47 (10)	**71.35 (9)**	70.66 (232)	67.58 (5)	67.06 (918)	28.65 (6)	69.81 (46)	70.32 (9)
Ionosphere	88.32 (9)	88.04 (12)	**91.17 (33)**	84.62 (292)	84.33 (4)	89.74 (638)	64.11 (6)	92.59 (47)	92.02 (7)
Iris	**97.33 (69)**	95.33 (0.2)	94.67 (0.5)	96.0 (34)	91.33 (3)	95.33 (833)	78.0 (35)	95.33 (36)	95.33 (10)
Lymphography	**83.75 (58)**	78.97 (3)	80.37 (10)	74.39 (100)	81.72 (3)	75.7 (533)	54.78 (40)	85.08 (38)	84.44 (16)
Mammographic-mass	83.13 (14)	81.08 (5)	**83.25 (6)**	81.57 (224)	81.69 (8)	80.84 (981)	79.88 (7)	84.34 (43)	82.53 (14)
Monks-problems-1	**100.0 (28)**	99.46 (2)	**100.0 (5)**	**100.0 (252)**	99.46 (6)	99.82 (389)	49.99 (6)	96.94 (40)	99.82 (7)
Monks-problems-2	**100.0 (29)**	98.33 (4)	99.83 (9)	94.85 (294)	87.02 (6)	99.83 (355)	65.4 (9)	63.06 (42)	72.05 (12)
Monks-problems-3	98.56 (19)	98.38 (3)	**98.92 (5)**	**98.92 (196)**	98.38 (5)	97.48 (992)	52.0 (6)	98.56 (38)	98.92 (6)
Optical-recognition	**95.74 (88)**	90.35 (2528)	92.49 (4836)	30.71 (12246)	94.45 (81)	95.5 (704)	26.44 (67)	98.09 (131)	97.7 (70)
Ozone-level-detection-eight	**93.07 (13)**	91.93 (128)	92.91 (235)	92.8 (6096)	92.58 (18)	91.72 (958)	**93.07 (8)**	93.5 (118)	93.67 (19)
Ozone-level-detection-one	96.75 (19)	95.83 (85)	**96.91 (130)**	**96.91 (5814)**	96.16 (17)	95.45 (993)	**96.91 (10)**	96.91 (101)	96.75 (15)
Parkinsons	89.23 (10)	87.18 (3)	**91.79 (7)**	88.21 (116)	88.72 (2)	88.72 (990)	84.1 (10)	89.23 (39)	92.82 (5)
Pen-based_recognition	97.89 (79)	96.74 (1542)	96.92 (2699)	69.87 (16005)	95.18 (152)	**98.4 (866)**	16.09 (53)	99.05 (208)	99.25 (86)
Pima-Indians-diabetes	74.74 (10)	72.4 (11)	74.99 (15)	74.74 (394)	**77.34 (7)**	70.44 (928)	64.84 (6)	76.3 (47)	76.43 (10)
Planning-relax	**72.57 (12)**	57.72 (2)	67.57 (3)	61.58 (56)	69.29 (2)	58.84 (930)	71.46 (6)	70.92 (40)	67.6 (6)

Continued on next page

Dataset	sNLL-CT	OCT-H single	OCT-H grid	ORCT	DTSemNet	RADDT	Sigmoid	RF	XGBoost
Qsar-biodegradation	86.73 (12)	82.84 (67)	83.51 (156)	**87.49 (1222)**	85.31 (10)	84.64 (1092)	66.26 (6)	86.64 (62)	84.93 (14)
Seeds	93.33 (79)	92.38 (0.9)	**94.76 (2)**	93.81 (73)	92.86 (5)	93.81 (863)	37.62 (33)	89.52 (38)	92.86 (10)
Segment	95.54 (80)	94.98 (150)	96.06 (301)	66.02 (3514)	93.16 (38)	**96.32 (844)**	26.1 (61)	97.92 (97)	98.48 (47)
Seismic-bumps	93.34 (23)	90.79 (66)	**93.42 (79)**	93.34 (1191)	92.42 (22)	92.03 (1117)	**93.42 (7)**	93.19 (67)	93.07 (16)
Skin-segmentation	99.78 (95)	99.87 (4329)	99.87 (7187)	99.43 (44325)	99.74 (2012)	**99.88 (2701)**	79.25 (13)	99.95 (1625)	99.96 (96)
Soybean-small	**98.0 (64)**	**98.0 (0.1)**	**98.0 (0.4)**	**98.0 (19)**	93.56 (2)	**98.0 (852)**	68.22 (41)	98.0 (36)	98.0 (4)
Spambase	91.78 (15)	93.57 (634)	**94.28 (1332)**	90.05 (2097)	92.72 (39)	92.59 (657)	60.6 (6)	95.33 (141)	95.35 (50)
Spect-heart	66.25 (9)	67.5 (0.6)	70.0 (1)	71.25 (29)	67.5 (2)	**73.75 (396)**	66.25 (9)	73.75 (36)	75.0 (6)
Spectf-heart	72.5 (9)	60.0 (1)	67.5 (5)	73.75 (88)	**77.5 (2)**	65.0 (863)	50.0 (6)	76.25 (37)	72.5 (4)
Statlog-project-German-credit	**72.9 (8)**	69.9 (131)	71.4 (192)	72.0 (1192)	65.3 (9)	70.4 (526)	62.0 (11)	75.1 (51)	74.1 (13)
Statlog-project-landsat-satellite	85.91 (92)	**87.79 (901)**	87.68 (1829)	76.66 (19433)	87.65 (95)	87.28 (1002)	32.12 (57)	91.96 (253)	91.98 (99)
Teaching-assistant-evaluation	56.22 (45)	**66.19 (6)**	63.51 (13)	58.19 (61)	39.74 (3)	60.84 (490)	43.66 (50)	57.66 (38)	62.88 (24)
Texture	98.89 (107)	95.56 (1533)	96.07 (2935)	32.85 (22608)	98.69 (121)	**99.55 (1045)**	23.8 (74)	97.6 (411)	98.58 (146)
Thoracic-surgery	81.91 (9)	75.96 (12)	**85.11 (13)**	81.91 (242)	79.15 (5)	77.45 (808)	**85.11 (6)**	84.68 (41)	84.47 (9)
Thyroid-disease-ann-thyroid	96.13 (89)	99.39 (68)	99.55 (151)	93.8 (1573)	97.67 (38)	**99.73 (1211)**	96.05 (70)	99.68 (60)	99.52 (20)
Thyroid-disease-new-thyroid	**96.74 (54)**	93.49 (0.6)	94.88 (1)	94.42 (58)	94.42 (4)	94.42 (776)	75.35 (37)	94.42 (38)	95.35 (10)
Tic-tac-toe-endgame	94.67 (13)	94.26 (42)	97.39 (90)	**97.91 (699)**	86.94 (9)	97.18 (355)	65.33 (6)	98.43 (45)	99.37 (8)
Vowel	81.11 (109)	71.31 (45)	69.8 (81)	34.55 (899)	70.3 (22)	**82.93 (836)**	8.59 (50)	95.76 (78)	91.41 (60)
Wall-following-robot-2	97.67 (77)	**100.0 (3)**	**100.0 (7)**	91.68 (1554)	95.99 (63)	**100.0 (925)**	78.35 (93)	99.96 (66)	99.78 (22)
Wine	94.94 (67)	94.92 (0.8)	93.25 (3)	**96.05 (90)**	**96.05 (3)**	89.33 (842)	67.44 (62)	95.51 (38)	97.75 (9)
Zoo	**98.05 (77)**	93.05 (0.9)	97.05 (2)	88.19 (49)	90.05 (4)	93.05 (359)	40.43 (35)	94.05 (36)	97.05 (17)
Avg.	**86.72 (39)**	84.42 (210)	86.7 (386)	80.29 (2535)	84.6 (49)	85.72 (789)	64.62 (25)	87.39 (89)	87.83 (22)

Figures 9 and 10 show pairwise test-accuracy comparisons between sNLL-CT and the alternative methods for setting iv). In all plots, sNLL-CT is reported on the y-axis, while the competing method is reported on the x-axis. Figure 9 compares sNLL-CT with OCT-H single, OCT-H grid, DTSemNet, and RADDT. Figure 10 compares sNLL-CT with ORCT, RF, and XGBoost.

Table 14: Paired two-sided Wilcoxon signed-rank tests on test accuracy across 61 datasets for depth setting iv). For each comparison, the better-performing method is shown in bold. Raw and Holm-adjusted p-values are reported, with comparisons ordered by statistical significance. Results above the dividing line are significant at the 95% confidence level. Sig is excluded due to its very poor performance.

Comparison setting iv)	**p-value**	**Adjusted p-value**
XGBoost vs. OCT-H single	3.56e-9	9.96e-8
OCT-H grid vs. OCT-H single	1.14e-7	3.08e-6
XGBoost vs. RADDT	1.19e-7	3.09e-6
RF vs. OCT-H single	2.46e-6	6.15e-5
sNLL-CT vs. OCT-H single	3.45e-6	8.28e-5
XGBoost vs. ORCT	3.60e-6	8.28e-5
XGBoost vs. DTSemNet	5.01e-6	1.10e-4
OCT-H grid vs. ORCT	3.94e-5	8.27e-4
RF vs. ORCT	7.99e-5	0.002
sNLL-CT vs. ORCT	1.96e-4	0.004
RF vs. DTSemNet	2.01e-4	0.004
sNLL-CT vs. DTSemNet	3.09e-4	0.005
RADDT vs. OCT-H single	5.24e-4	0.008
OCT-H grid vs. DTSemNet	5.26e-4	0.008
RF vs. RADDT	7.02e-4	0.010
XGBoost vs. sNLL-CT	0.004	0.050
OCT-H grid vs. RADDT	0.004	0.050
XGBoost vs. OCT-H grid	0.006	0.069
sNLL-CT vs. RADDT	0.035	0.349
sNLL-CT vs. RF	0.063	0.568
RADDT vs. ORCT	0.098	0.782
RF vs. OCT-H grid	0.130	0.911
ORCT vs. DTSemNet	0.177	1
RF vs. XGBoost	0.195	1
RADDT vs. DTSemNet	0.197	1
OCT-H single vs. ORCT	0.480	1
OCT-H single vs. DTSemNet	0.493	1
sNLL-CT vs. OCT-H grid	0.967	1

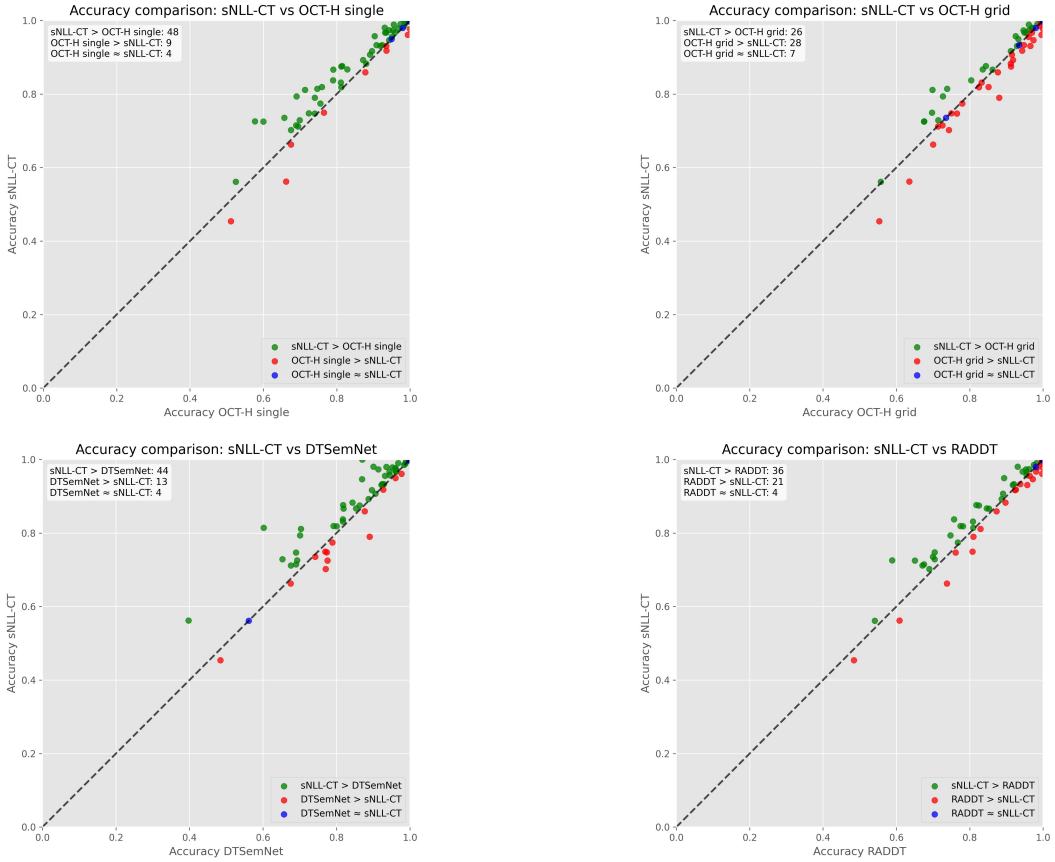


Figure 9: Comparison of test accuracy between sNLL-CT and the single-tree methods for depth setting iv). From left to right and top to bottom, the plots compare sNLL-CT with OCT-H single, OCT-H grid, DTSemNet, and RADDT. Each point corresponds to one dataset. Numbers in parentheses in the legend report the count of datasets where sNLL-CT performs better (green), worse (red), or achieves the same accuracy (blue).

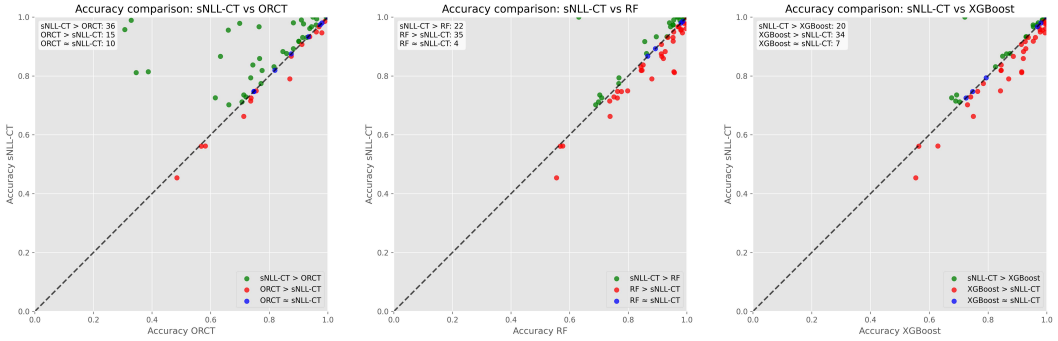
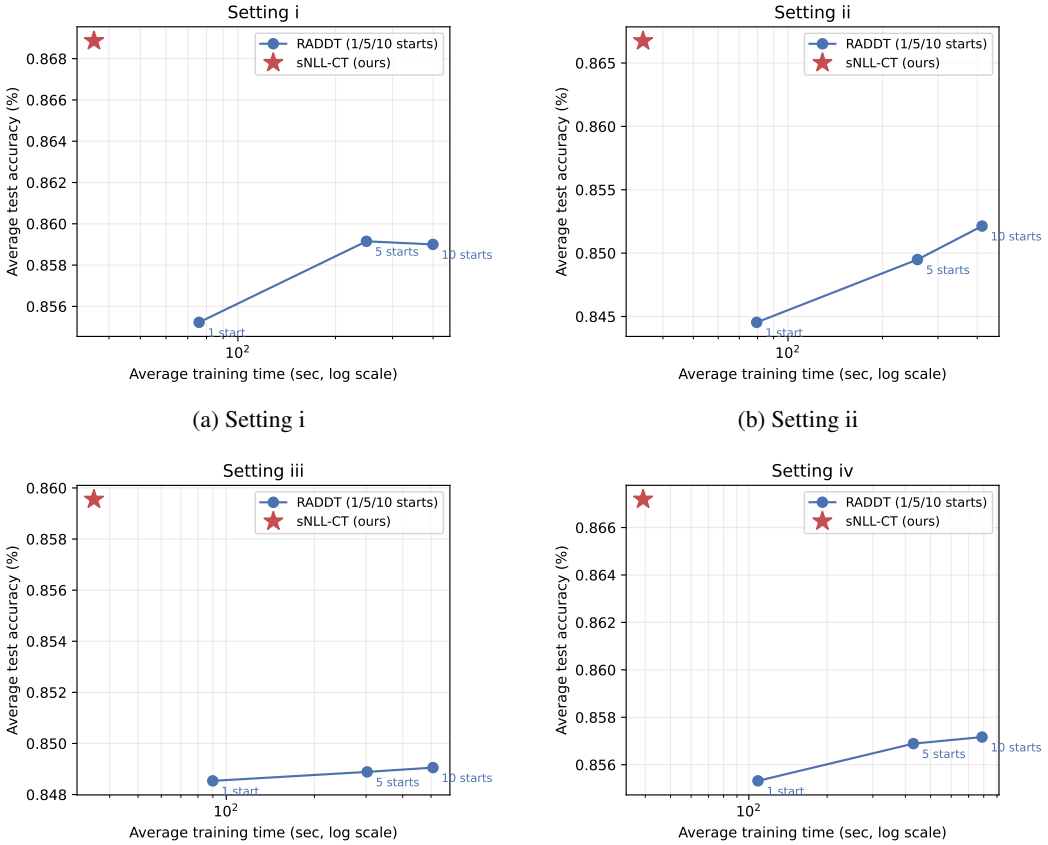


Figure 10: Comparison of test accuracy between sNLL-CT and ORCT, RF, and XGBoost for depth setting iv). From left to right, the plots compare sNLL-CT with ORCT, RF, and XGBoost. Each point corresponds to one dataset. Numbers in parentheses in the legend report the count of datasets where sNLL-CT performs better (green), worse (red), or achieves the same accuracy (blue).

E Robustness with respect to RADDT's optimization budget

RADDT relies on a multi-start strategy, so a natural concern is whether sNLL-CT's efficiency advantage merely reflects a comparison against RADDT at its most expensive setting. To address this, we re-run RADDT with 1, 5, and 10 starting points on the same 61 datasets, folds, and depth settings, keeping everything else fixed. As Figure 11 shows, in every setting sNLL-CT lies above and to the left of the entire RADDT budget curve: it is more accurate and substantially faster than RADDT at each of its budgets, confirming that the advantage does not stem from a favorable single choice of RADDT's number of starts.



(a) Setting i

(b) Setting ii

(c) Setting iii

(d) Setting iv

Figure 11: Accuracy–training-time trade-off between sNLL-CT and RADDT across its number of starting points (1, 5, 10), for the four depth settings. Each panel reports average test accuracy against average training time (log scale) over the 61 datasets; the blue curve traces RADDT as its budget grows, the red star is sNLL-CT.

F Comparison with the LatentTree model

In this appendix, we compare sNLL-CT and LatentTree on binary classification datasets. We restrict the analysis to the 37 binary datasets because LatentTree from Zantedeschi et al. (2021) is designed only for binary classification tasks. Table 16 reports the test accuracy and average training time of all models on each dataset. Table 15 additionally shows, for each depth $D \in \{2, 3, 4, 5\}$, the pairwise statistical comparison between our sNLL-CT and **LatentTree**. Significance is assessed using the Wilcoxon signed-rank test, and the corresponding p-values are reported. Since the comparison is performed only against **LatentTree**, no multiple-comparison adjustment is applied. For each depth, the table also summarizes how many times, out of 37 comparisons, sNLL-CT performs better, worse, or ties. Overall, sNLL-CT significantly outperforms **LatentTree**, achieving higher accuracy across all depths on at least 28 datasets.

Table 15: Paired two-sided Wilcoxon signed-rank tests on test accuracy across 37 datasets for depth $D \in \{2, 3, 4, 5\}$ where together with the p-values, are also reported the number of times when sNLL-CT performed better (Wins), worse (Losses), or tied (Ties).

sNLL-CT vs. LatentTree				
Depth	Wins	Losses	Ties	p-value
2	28	8	1	0.0014
3	32	5	0	7.32e-07
4	30	6	1	0.0087
5	28	9	0	0.0027

Table 16: Average classification accuracy and training time (in seconds, in parentheses) for sNLL-CT and LatentTree on each binary classification dataset at depths $D \in \{2, 3, 4, 5\}$. Bold indicates, for each dataset and depth, the higher test accuracy between the two methods.

Dataset	$D = 2$		$D = 3$		$D = 4$		$D = 5$	
	sNLL-CT	LatentTree	sNLL-CT	LatentTree	sNLL-CT	LatentTree	sNLL-CT	LatentTree
Acute-inflammations-1	**100.0 (29.3)**	95.83 (0.2)	**100.0 (28.5)**	98.33 (0.2)	**100.0 (28.8)**	99.17 (0.3)	**100.0 (28.1)**	98.33 (0.3)
Acute-inflammations-2	**100.0 (27.5)**	99.17 (0.2)	**100.0 (28.0)**	96.67 (0.2)	**100.0 (28.8)**	93.33 (0.2)	**100.0 (28.0)**	95.0 (0.3)
Banknote-authentication	**100.0 (25.8)**	97.23 (0.6)	99.78 (25.9)	95.77 (0.7)	99.71 (20.2)	89.59 (0.8)	100.0 (22.5)	90.9 (1.0)
Blood-transfusion	77.15 (12.9)	**78.22 (0.2)**	77.15 (14.2)	**77.82 (0.3)**	**77.69 (14.7)**	77.42 (0.4)	77.42 (16.6)	**77.96 (0.5)**
Breast-cancer	**73.26 (9.3)**	68.23 (0.2)	**72.56 (8.0)**	70.04 (0.2)	**70.41 (8.7)**	67.49 (0.2)	**71.47 (8.5)**	70.75 (0.3)
Breast-cancer-diagnostic	**97.89 (9.8)**	95.08 (0.3)	**97.01 (10.5)**	95.08 (0.3)	**97.01 (9.6)**	94.38 (0.4)	**97.36 (9.7)**	94.55 (0.5)
Breast-cancer-prognostic	**77.29 (10.8)**	77.26 (0.2)	75.22 (8.7)	**77.27 (0.3)**	**75.75 (8.2)**	75.72 (0.3)	**79.37 (8.6)**	78.83 (0.4)
Chess-king-rook-versus-king-pawn	**99.19 (28.0)**	97.09 (1.5)	**99.22 (24.9)**	94.81 (1.6)	**99.09 (25.3)**	93.37 (1.9)	**99.37 (22.8)**	95.09 (2.5)
Climate-model-crashes	**90.19 (11.6)**	89.07 (0.3)	**90.56 (7.9)**	87.04 (0.3)	**91.11 (8.0)**	89.26 (0.3)	**90.74 (8.1)**	88.89 (0.4)
Congressional-voting-records	**93.52 (13.7)**	89.18 (0.2)	**93.52 (13.8)**	89.21 (0.2)	**93.52 (12.2)**	89.61 (0.2)	**93.06 (12.2)**	90.92 (0.4)
Connectionist-bench-sonar	**77.39 (14.9)**	76.92 (0.2)	**78.39 (12.0)**	75.96 (0.3)	**80.75 (12.6)**	75.97 (0.3)	74.95 (9.4)	**76.91 (0.4)**
Credit-approval	**87.14 (9.0)**	82.38 (0.2)	**86.83 (8.1)**	84.83 (0.3)	**85.15 (8.2)**	84.37 (0.4)	**87.6 (8.3)**	83.31 (0.5)
Cylinder-bands	74.36 (9.3)	**74.75 (0.2)**	72.16 (8.0)	**73.27 (0.2)**	67.82 (7.6)	**77.25 (0.3)**	**74.72 (8.0)**	73.64 (0.4)
Echocardiogram	68.95 (7.8)	**78.38 (0.1)**	70.38 (7.9)	**74.38 (0.2)**	65.05 (7.6)	**81.14 (0.2)**	70.19 (8.0)	**79.71 (0.3)**
Fertility	**88.0 (8.6)**	81.0 (0.2)	**86.0 (10.7)**	84.0 (0.2)	**85.0 (7.6)**	83.0 (0.3)	79.0 (12.0)	**87.0 (0.3)**
Haberman-survival	74.21 (9.4)	**74.85 (0.2)**	**73.88 (17.4)**	**73.88 (0.2)**	71.27 (20.7)	**74.2 (0.2)**	**73.54 (19.7)**	**73.54 (0.4)**
Hepatitis	**86.25 (7.7)**	**86.25 (0.2)**	**90.0 (10.2)**	81.25 (0.2)	**86.25 (8.4)**	82.5 (0.3)	**87.5 (9.0)**	83.75 (0.3)
Indian-liver-patient	**70.66 (10.5)**	69.81 (0.2)	**71.35 (13.8)**	69.46 (0.3)	**71.7 (12.0)**	69.81 (0.3)	**71.18 (17.0)**	67.24 (0.5)
Ionosphere	**90.33 (11.7)**	86.04 (0.2)	**91.18 (10.3)**	88.04 (0.2)	**88.9 (11.2)**	86.91 (0.3)	**88.32 (8.9)**	85.48 (0.5)
Mammographic-mass	**82.05 (11.5)**	81.57 (0.4)	**83.01 (15.7)**	80.48 (0.5)	**83.13 (15.0)**	80.36 (0.6)	**83.13 (13.9)**	81.2 (0.8)
Monks-problems-1	**100.0 (12.4)**	84.36 (0.4)	**100.0 (23.2)**	85.43 (0.4)	**100.0 (28.0)**	82.22 (0.6)	**100.0 (28.1)**	84.54 (0.6)

Continued on next page

Dataset	$D = 2$		$D = 3$		$D = 4$		$D = 5$	
	sNLL-CT	LatentTree	sNLL-CT	LatentTree	sNLL-CT	LatentTree	sNLL-CT	LatentTree
Monks-problems-2	**100.0 (17.2)**	61.56 (0.2)	**100.0 (25.6)**	69.57 (0.4)	**100.0 (27.8)**	66.4 (0.5)	**100.0 (28.7)**	59.41 (0.5)
Monks-problems-3	**98.74 (11.5)**	93.67 (0.4)	**98.38 (13.2)**	93.68 (0.3)	**98.74 (13.1)**	92.6 (0.4)	**98.56 (18.7)**	93.51 (0.7)
Ozone-level-detection-eight	92.91 (11.7)	**93.12 (0.8)**	**93.23 (11.8)**	91.77 (0.8)	**93.07 (14.1)**	92.8 (0.9)	**93.07 (13.1)**	89.01 (1.2)
Ozone-level-detection-one	**96.91 (10.2)**	96.86 (0.6)	**96.91 (13.5)**	96.81 (0.7)	**96.91 (12.2)**	**96.91 (0.8)**	**96.75 (18.8)**	96.64 (1.0)
Parkinsons	**90.26 (8.5)**	82.05 (0.2)	**92.31 (15.6)**	78.46 (0.2)	**90.26 (10.3)**	80.51 (0.3)	**89.23 (10.0)**	79.49 (0.4)
Pima-Indians-diabetes	**76.04 (9.0)**	73.18 (0.2)	**74.35 (10.0)**	73.04 (0.3)	**75.65 (10.3)**	75.26 (0.4)	74.74 (9.9)	**75.26 (0.5)**
Planning-relax	69.79 (14.4)	**69.85 (0.2)**	**72.03 (14.9)**	68.71 (0.2)	**72.01 (12.6)**	69.8 (0.2)	**72.57 (12.3)**	67.58 (0.3)
Qsar-biodegradation	**85.31 (12.0)**	84.17 (0.5)	**86.45 (11.3)**	85.21 (0.6)	**86.45 (11.8)**	85.4 (0.8)	**86.73 (12.4)**	85.02 (1.0)
Seismic-bumps	**93.34 (11.7)**	93.11 (0.8)	**93.34 (16.7)**	92.61 (0.9)	**93.34 (12.5)**	92.88 (1.0)	**93.34 (23.3)**	92.92 (1.2)
Skin-segmentation	**97.93 (30.7)**	82.29 (122.8)	**99.77 (34.3)**	77.62 (164.9)	**99.78 (44.9)**	82.12 (185.7)	**99.78 (95.4)**	80.9 (227.5)
Spambase	**92.5 (23.1)**	91.33 (1.6)	**92.5 (24.5)**	90.94 (1.7)	**92.11 (17.1)**	91.05 (2.0)	**91.78 (15.2)**	89.87 (2.9)
Spect-heart	65.0 (8.6)	**73.75 (0.2)**	**72.5 (8.6)**	68.75 (0.2)	61.25 (11.0)	**70.0 (0.2)**	66.25 (9.1)	**73.75 (0.3)**
Spectf-heart	68.75 (9.7)	**77.5 (0.2)**	**73.75 (9.6)**	71.25 (0.2)	65.0 (10.4)	**77.5 (0.2)**	72.5 (9.0)	**75.0 (0.3)**
Statlog-project-German-credit	**74.0 (8.2)**	72.8 (0.4)	**73.7 (8.0)**	72.7 (0.4)	**74.2 (8.1)**	74.1 (0.6)	**72.9 (7.9)**	71.8 (0.7)
Thoracic-surgery	**84.04 (7.5)**	79.79 (0.2)	**82.13 (7.9)**	80.85 (0.2)	**81.91 (7.8)**	81.7 (0.3)	**81.91 (9.4)**	80.85 (0.4)
Tic-tac-toe-endgame	**98.02 (15.1)**	91.87 (0.6)	**97.6 (12.4)**	90.3 (0.8)	91.34 (14.4)	**96.66 (1.2)**	94.67 (12.9)	**96.87 (1.4)**
Avg.	**86.25 (13.5)**	83.23 (3.7)	**86.68 (14.7)**	82.58 (4.9)	**85.44 (14.6)**	83.05 (5.5)	**86.05 (16.6)**	82.85 (6.8)

G SYNTHETIC DATASET EXPERIMENTS

In this appendix, we report the results for the individual configurations of each of the five settings considered in the synthetic dataset experiments. Table 17 summarizes these five experimental settings, each corresponding to a different parameter configuration. Specifically, the table reports the median train and test accuracy, as well as RI, over 100 runs for sNLL-CT and OCT-H.

Overall, sNLL-CT outperforms OCT-H in the majority of experimental settings. In **Exp 1**, sNLL-CT provides superior general performance. Similarly, in **Exp 2**, sNLL-CT achieves in terms of median better overall results. In **Exp 4**, sNLL-CT performs in terms of test accuracy better at low noise levels but degrades at higher noise values. By contrast, **Exp 3**, and **Exp 5**, sNLL-CT achieves consistently superior and statistically significant performance over OCT-H across almost all setting configurations (i.e., feature noise and feature dimensionality) in both accuracy and RI.

Table 17: Results on synthetic datasets for the five settings. Bold denotes the best median over 100 runs, and * indicates statistical significance according to an exact paired permutation test at $\alpha = 0.05$ with ground truths as blocks.

	sNLL-CT				OCT-H			
Exp1	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
2	**0.993**	**0.984**	**0.9827**	**0.9622**	0.983	0.958	0.9577	0.9004
3	**0.978***	**0.9555***	**0.9537**	**0.9148***	0.946	0.8848	0.8926	0.7988
4	**0.938**	**0.8886***	**0.9018**	**0.8367***	0.917	0.8105	0.8748	0.7359
Exp2	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
1000	**0.978***	**0.9555***	**0.9537**	**0.9148***	0.946	0.8848	0.8926	0.7988

Table 17: Results on synthetic datasets for the five settings (continued).

	sNLL-CT				OCT-H			
Exp1	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
5000	**0.9817**	**0.9757**	**0.9793**	**0.9724**	0.9684	0.9584	0.9512	0.9322
10000	**0.9808**	**0.9786**	**0.9736**	**0.9687**	0.9652	0.9574	0.9547	0.9402
20000	**0.9927**	**0.9913**	**0.9902**	**0.9886**	0.9793	0.9761	0.9746	0.9683
Exp3	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
0	**0.978***	**0.9555***	**0.9537**	**0.9148***	0.946	0.8848	0.8926	0.7988
5	**0.9615***	**0.9338***	**0.9133**	**0.8912***	0.9315	0.8758	0.8704	0.7865
10	**0.959***	**0.948***	**0.9083***	**0.8975***	0.9255	0.8671	0.8602	0.7771
15	**0.948***	**0.9389***	**0.8938**	**0.8933***	0.922	0.8651	0.8496	0.7807
20	**0.943***	**0.9356***	**0.8736**	**0.8775***	0.9205	0.8706	0.8375	0.7683
Exp4	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
0	**0.978***	**0.9555***	**0.9537**	**0.9148***	0.946	0.8848	0.8926	0.7988
5	**0.911**	**0.9264***	**0.8925**	**0.8637***	0.901	0.8781	0.8855	0.7874
10	0.848	**0.9061***	0.8688	**0.8177**	**0.851**	0.8737	**0.8793**	0.7926
15	0.7755	0.8637	0.8111	0.7579	**0.809***	**0.8676**	**0.8696***	**0.7758**
20	0.7055	0.8105	0.7303	0.6782	**0.7635***	**0.8579***	**0.8549***	**0.7771***
Exp5	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
5	**0.978***	**0.9555***	**0.9537**	**0.9148***	0.946	0.8848	0.8926	0.7988
10	**0.9615***	**0.8608***	**0.9416***	**0.7905***	0.918	0.7507	0.8171	0.5491
15	**0.961***	**0.7831***	**0.9208***	**0.6357***	0.934	0.659	0.7933	0.3359
20	**0.966***	**0.6708***	**0.9127***	**0.3657***	0.936	0.519	0.7838	0.1536

H EXPERIMENTS WITH NLL ACTIVATION

In this appendix, we compare sNLL-CT with a SCT that uses the NLL activation function and is trained with Algorithm 1. We refer to this baseline as NLL-CT. We consider both real-world and synthetic experiments.

Table 19 reports the test accuracy of sNLL-CT and NLL-CT on the 61 real-world datasets under the four settings. In setting iii), which gives the best average test accuracy for NLL-CT, NLL-CT outperforms sNLL-CT on 18 datasets, whereas sNLL-CT achieves higher accuracy on 38 datasets. The difference is even more pronounced in the other settings, where sNLL-CT outperforms NLL-CT on more than two-thirds of the datasets. Table 18 reports the results for each configuration of the five settings used in the synthetic dataset experiments. In particular, it shows the median train and test accuracy, together with the Rand Index (RI), over 100 runs for both sNLL-CT and NLL-CT. Across all settings and configurations, sNLL-CT consistently achieves substantially higher accuracy and better reconstruction of the ground-truth tree structure than NLL-CT.

Table 18: Results on synthetic datasets for the five settings comparing sNLL-CT and NLL-CT. Bold indicates, for each setting and metric, the higher value between the two methods.

	sNLL-CT				NLL-CT			
Exp1	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
2	**0.993**	**0.984**	**0.9827**	**0.9622**	0.934	0.9219	0.8335	0.8168
3	**0.978**	**0.9555**	**0.9537**	**0.9148**	0.7525	0.7274	0.4629	0.4352
4	**0.938**	**0.8886**	**0.9018**	**0.8367**	0.5755	0.5499	0.3053	0.2767
Exp2	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
1000	**0.978**	**0.9555**	**0.9537**	**0.9148**	0.7525	0.7274	0.4629	0.4352
5000	**0.9817**	**0.9757**	**0.9793**	**0.9724**	0.7207	0.7098	0.4028	0.3929

Table 18: Results on synthetic datasets for the five settings comparing sNLL-CT and NLL-CT (continued).

	sNLL-CT				NLL-CT			
Exp1	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
10000	**0.9808**	**0.9786**	**0.9736**	**0.9687**	0.6989	0.7009	0.4053	0.4018
20000	**0.9927**	**0.9913**	**0.9902**	**0.9886**	0.7796	0.7816	0.4701	0.4659
Exp3	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
0	**0.978**	**0.9555**	**0.9537**	**0.9148**	0.7525	0.7274	0.4629	0.4352
5	**0.9615**	**0.9338**	**0.9133**	**0.8912**	0.764	0.7296	0.4873	0.4611
10	**0.959**	**0.948**	**0.9083**	**0.8975**	0.7565	0.7273	0.4728	0.4559
15	**0.948**	**0.9389**	**0.8938**	**0.8933**	0.7545	0.7348	0.4673	0.4569
20	**0.943**	**0.9356**	**0.8736**	**0.8775**	0.7605	0.7325	0.4576	0.4479
Exp4	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
0	**0.978**	**0.9555**	**0.9537**	**0.9148**	0.7525	0.7274	0.4629	0.4352
5	**0.911**	**0.9264**	**0.8925**	**0.8637**	0.71	0.7174	0.4038	0.3705
10	**0.848**	**0.9061**	**0.8688**	**0.8177**	0.6665	0.7027	0.3972	0.3728
15	**0.7755**	**0.8637**	**0.8111**	**0.7579**	0.628	0.6911	0.3891	0.3628
20	**0.7055**	**0.8105**	**0.7303**	**0.6782**	0.598	0.6891	0.375	0.3621
Exp5	Train Acc.	Test Acc.	Train RI	Test RI	Train Acc.	Test Acc.	Train RI	Test RI
5	**0.978**	**0.9555**	**0.9537**	**0.9148**	0.7525	0.7274	0.4629	0.4352
10	**0.9615**	**0.8608**	**0.9416**	**0.7905**	0.784	0.6951	0.6035	0.4903
15	**0.961**	**0.7831**	**0.9208**	**0.6357**	0.855	0.6973	0.6979	0.4371
20	**0.966**	**0.6708**	**0.9127**	**0.3657**	0.8715	0.556	0.7158	0.2318

Table 19: Average classification accuracy for sNLL-CT and NLL-CT across the 61 datasets under the four settings. Bold indicates, for each dataset and setting, the higher test accuracy between the two methods.

Dataset	setting i)		setting ii)		setting iii)		setting iv)	
	sNLL-CT	NLL-CT	sNLL-CT	NLL-CT	sNLL-CT	NLL-CT	sNLL-CT	NLL-CT
Acute-inflammations-1	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Acute-inflammations-2	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Balance-scale	91.36	**91.68**	91.36	**91.68**	**91.68**	90.56	**91.68**	88.0
Banknote-authentication	**100.0**	99.85	99.78	**99.93**	99.71	**100.0**	**100.0**	99.85
Blood-transfusion	77.15	**77.42**	77.15	**77.82**	**77.69**	77.29	**77.42**	76.35
Breast-cancer	**73.26**	70.4	**72.56**	71.49	**70.41**	70.03	**71.47**	70.75
Breast-cancer-diagnostic	**97.89**	97.01	**97.01**	96.84	**97.01**	96.65	**97.36**	96.84
Breast-cancer-prognostic	77.29	**81.42**	75.22	**77.29**	75.75	**76.23**	**79.37**	77.31
Car-evaluation	**97.69**	91.55	**97.69**	91.55	**96.24**	91.9	**99.02**	93.23
Chess-king-rook-versus-king-pawn	**99.19**	97.84	**99.22**	98.28	**99.09**	98.5	**99.37**	98.56
Climate-model-crashes	**90.19**	89.81	**90.56**	90.0	91.11	**91.67**	**90.74**	89.63
Congressional-voting-records	**93.52**	93.51	**93.52**	88.32	93.52	**94.82**	93.06	**93.52**
Connectionist-bench	**81.41**	31.82	**69.49**	30.61	**69.49**	30.61	**81.41**	31.82
Connectionist-bench-sonar	**77.39**	71.65	**78.39**	77.9	**80.75**	76.45	**74.95**	73.55
Contraceptive-method-choice	45.22	**48.74**	45.22	**48.74**	45.96	**48.74**	45.42	**49.56**

Continued on next page

Dataset	setting i)		setting ii)		setting iii)		setting iv)	
	sNLL-CT	NLL-CT	sNLL-CT	NLL-CT	sNLL-CT	NLL-CT	sNLL-CT	NLL-CT
Credit-approval	**87.14**	84.53	**86.83**	84.08	**85.15**	85.14	**87.6**	84.68
Cylinder-bands	**74.36**	72.52	**72.16**	71.49	67.82	**72.53**	74.72	**75.43**
Dermatology	**97.26**	95.08	**95.63**	89.87	**97.26**	95.08	**96.71**	95.07
Echocardiogram	68.95	**70.48**	**70.38**	66.29	65.05	**70.48**	**70.19**	58.0
Fertility	**88.0**	81.0	**86.0**	80.0	**85.0**	82.0	79.0	**80.0**
Haberman-survival	74.21	**74.53**	73.88	**74.52**	71.27	**74.84**	**73.54**	71.59
Hayes-roth	**80.63**	79.38	**80.63**	79.38	79.38	**80.0**	**81.88**	**81.88**
Heart-disease-Cleveland	**59.07**	58.42	**59.09**	59.07	**59.07**	58.42	56.1	**60.04**
Hepatitis	**86.25**	85.0	**90.0**	82.5	**86.25**	**86.25**	**87.5**	85.0
Image-segmentation	**86.67**	68.57	**82.86**	77.62	**86.67**	68.57	**86.67**	81.9
Indian-liver-patient	70.66	**72.56**	71.35	**72.38**	**71.7**	70.67	71.18	**73.41**
Ionosphere	**90.33**	85.77	**91.18**	87.46	88.9	**89.18**	**88.32**	88.06
Iris	**96.67**	94.0	**96.67**	94.0	**96.67**	96.0	**97.33**	94.67
Lymphography	**83.08**	79.75	**83.08**	79.75	**83.79**	81.75	83.75	**84.37**
Mammographic-mass	82.05	**83.86**	83.01	**84.1**	**83.13**	82.65	**83.13**	82.17
Monks-problems-1	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Monks-problems-2	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Monks-problems-3	**98.74**	98.02	**98.38**	97.48	**98.74**	98.19	**98.56**	94.41
Optical-recognition	**95.74**	78.27	**94.93**	83.94	**94.93**	83.94	**95.74**	78.27
Ozone-level-detection-eight	92.91	**93.72**	93.23	**93.56**	**93.07**	**93.07**	93.07	**93.45**
Ozone-level-detection-one	**96.91**	96.86	**96.91**	96.81	**96.91**	96.59	**96.75**	96.64
Parkinsons	**90.26**	84.62	**92.31**	86.15	**90.26**	87.18	**89.23**	87.18
Pen-based_recognition	**97.89**	81.81	**96.37**	83.21	**96.37**	83.21	**97.89**	81.81
Pima-Indians-diabetes	76.04	**77.34**	74.35	**77.87**	75.65	**76.69**	74.74	**76.17**
Planning-relax	69.79	**70.38**	**72.03**	69.83	**72.01**	65.41	**72.57**	67.66
Qsar-biodegradation	**85.31**	84.93	**86.45**	85.02	**86.45**	84.74	**86.73**	86.35
Seeds	94.76	**95.24**	94.76	**95.24**	93.33	**95.24**	**93.33**	92.86
Segment	**95.02**	87.88	**93.42**	86.8	**95.02**	87.88	**95.54**	87.71
Seismic-bumps	**93.34**	93.03	**93.34**	93.07	**93.34**	93.11	**93.34**	93.19
Skin-segmentation	97.93	**99.2**	**99.77**	99.33	**99.78**	99.35	**99.78**	99.73
Soybean-small	**98.0**	95.78	**98.0**	95.78	95.78	**98.0**	**98.0**	**98.0**
Spambase	**92.5**	92.22	**92.5**	92.37	**92.11**	91.94	91.78	**92.41**
Spect-heart	**65.0**	62.5	**72.5**	60.0	**61.25**	**61.25**	**66.25**	60.0
Spectf-heart	**68.75**	67.5	**73.75**	**73.75**	65.0	**67.5**	72.5	**77.5**
Statlog-project-German-credit	74.0	**74.7**	73.7	**74.2**	**74.2**	73.4	**72.9**	69.0
Statlog-project-landsat-satellite	**86.1**	77.82	**85.94**	78.35	**86.1**	77.82	**85.91**	75.46
Teaching-assistant-evaluation	**56.92**	45.83	**56.92**	45.83	**56.26**	54.24	56.22	**64.15**
Texture	**98.89**	84.95	**98.36**	87.91	**98.36**	87.91	**98.89**	84.95
Thoracic-surgery	**84.04**	82.34	82.13	**82.55**	81.91	**82.77**	**81.91**	78.09
Thyroid-disease-ann-thyroid	**96.66**	91.57	**96.66**	91.57	**96.37**	90.03	**96.13**	89.95
Thyroid-disease-new-thyroid	**96.28**	93.95	**96.28**	93.95	96.28	**96.74**	**96.74**	96.28
Tic-tac-toe-endgame	**98.02**	97.91	97.6	**98.02**	91.34	**97.18**	94.67	**96.77**
Vowel	**81.11**	31.52	**75.35**	30.0	**75.35**	30.0	**81.11**	31.52
Wall-following-robot-2	**97.91**	74.61	**97.91**	74.61	**97.82**	87.54	**97.67**	67.3
Wine	**98.32**	96.62	**98.32**	96.62	**97.71**	95.51	**94.94**	92.71
Zoo	**96.05**	90.29	**95.05**	89.05	**96.05**	90.29	**98.05**	95.05
Avg.	**86.89**	82.45	**86.67**	82.55	**85.95**	83.01	**86.72**	82.62

I SINGLE-PATH VERSUS SOFT PREDICTION

In this appendix, we assess for sNLL-CT how much the deterministic prediction rule in equation 2 deviates from soft aggregate prediction ($\arg\max_k \sum_{t\in\tau_L} M_t(x)\, c_{kt}$). For each of the 61 datasets, we calculate the average over the 5 folds of the fraction of test samples on which the two prediction rules agree and the difference in test accuracy $\Delta = \mathrm{Acc}_{\text{single}} - \mathrm{Acc}_{\text{soft}}$. To explain the agreement between

the two prediction rules, we also report two measures: the routing margin $r_t(x) = \frac{1}{p}\mathbf{w}_t^\top \mathbf{x} + w_{0t}$, specifically the 5-th percentile over the test samples of the smallest absolute margin along each sample's path, $r_{\min}(x) = \min_{t \in \text{path}(x)} \left| r_t(x) \right|$; and the average over the test samples of the mass concentration $M_{t^\star}(x) / \sum_t M_t(x)$ (explicitly normalized, since with sNLL the activations are not complementary), where $t^\star$ is the leaf selected by the single-path rule in equation 2. Table 20 aggregates these metrics across the 61 datasets, for each of the four depth settings.

Table 20: Single-path versus soft prediction over the 61 datasets, per depth setting. "Agreement" is the fraction of test samples on which the two rules predict the same class (mean over datasets, and minimum over datasets in parentheses). "Acc. gap" is $\text{Acc}_{\text{single}} - \text{Acc}_{\text{soft}}$ (mean). "Mass" is the average fraction of routing mass on the selected leaf; "Margin ($p5$)" is the average over datasets of the 5th-percentile path margin.

Setting	Agreement (min)	Acc. gap	Mass	Margin ($p5$)
i)	99.60% (96.97%)	−0.11%	90.93%	1.164
ii)	99.44% (96.67%)	−0.05%	87.38%	0.821
iii)	99.28% (96.25%)	−0.20%	82.87%	0.617
iv)	99.06% (94.02%)	−0.22%	77.84%	0.496

The two prediction rules agree on more than 99% of test samples on average, and on at least 94% even on the least-aligned dataset. The accuracy gap is small and slightly in favour of soft aggregation, but its magnitude stays below 0.25% on average across all settings. Single-path inference costs a negligible amount of accuracy while recovering conditional computation and path-level intepretability. The agreement is explained by the two additional measures: most of the routing mass concentrates on the selected leaf, and the path margin stays well away from zero. Both quantities, together with the agreement decrease mildly and monotonically with depth (deeper trees have longer paths, hence more branch nodes at which a decision can be less sharp), which confirms that mass concentration and margin sharpness are indeed the mechanism underlying the equivalence between the two rules.

J INTEGRALITY OF THE CLASS ASSIGNMENTS VARIABLES

In this appendix, we present an analysis of the level of completeness of the variables c_{kt} related to class assignment in leaf nodes. It is worth pointing out that even though the projected Adam need not return integral c_{kt}, this has no practical effect on the model. Indeed, the prediction rule equation 2 is based on $\arg\max_k c_{kt}$ at the selected leaf as in equation 2), so the output is integral even when the learned c_{kt} are not. On the 23 multiclass datasets (on binary datasets the c_{kt} are fixed by construction), we therefore report, per depth setting: the mean fraction of c_{kt} that are integral up to 10^{-6}; the largest deviation from integrality over all datasets, i.e., $\max_{k,t} \min(c_{kt}, 1 - c_{kt})$; and the fraction of test set predictions that change when the c_{kt} are rounded, taken as the maximum over all datasets.

Table 21: Integrality of the c_{kt} variables on the 23 multiclass datasets, per depth setting. "Frac. integral" is the mean fraction of c_{kt} within 10^{-6} of $\{0, 1\}$; "Worst-case" is the maximum of $\min(c_{kt}, 1 - c_{kt})$ over all variables and datasets; "Pred. change" is the maximum over datasets of the fraction of test set predictions that differ when the c_{kt} are rounded.

Setting	Frac. integral ($\leq 10^{-6}$)	Worst-case	Pred. change
i)	86.46%	4.37e−1	0
ii)	90.92%	3.26e−1	0
iii)	83.40%	4.80e−1	0
iv)	76.12%	4.99e−1	0

Although a significant portion of the c_{kt} values are not integer, and the worst-case deviation is close to 0.5, across all 23 multiclass datasets and for all four depth settings, rounding the c_{kt} values never changes a single test prediction (as shown in the "Pred. change" column). This is consistent with the total-unimodularity argument: fractional values are confined to variables that do not affect the $\arg\max$ at the reached leaf (typically leaf nodes that receive no samples), and thus leave the model's output unchanged. The guarantee that matters is therefore on the output, not on the solution: the prediction is always integral, even in the worst case.