Machine Learning (2025) 114:253
https://doi.org/10.1007/s10994-025-06906-x

Adaptive differentiable trees for transparent learning on data streams

Kirsten Köbschall[1] · **Lisa Hartung**[1] · **Stefan Kramer**[1]

Received: 22 April 2025 / Revised: 7 August 2025 / Accepted: 23 September 2025 /
Published online: 19 October 2025

Abstract
Maintaining learning models in dynamic environments requires transparency for trust and compliance, particularly under regulatory frameworks like the Artificial Intelligence (AI) Act by the European Union. Data stream models must balance adaptability with interpretability, and to keep AI models effective in evolving contexts, maintaining transparency is essential. To address this, we introduce Soft Hoeffding Trees (SoHoT) as transparent, differentiable decision trees for data streams. SoHoTs use a novel routing function, leveraging the Hoeffding inequality for tree expansion, while gradient descent updates tree weights to adapt to drifting data distributions. Transparency is further enhanced with decision-rule-based feature importance and a sparse activation function, enabling selective subtree consideration for final predictions. We also provide a visualization of the model's decision-making process for user interpretability. Evaluated on 20 data streams, SoHoT outperforms Hoeffding trees and competes with Hoeffding adaptive trees and soft trees under AUROC. We also demonstrate how to balance transparency and performance, by looking at the trade-off and measuring prediction performance per complexity, which showcases SoHoT's benefits compared to existing data stream algorithms.

Keywords Data streams · Hoeffding bound · Concept drift · Soft decision tree

1 Introduction

The European Union proposed the AI Act (Parliament and of the European Union, 2024) to regulate artificial intelligence (AI) by promoting transparency, accountability, and efficiency in AI-driven systems. Since AI systems increasingly influence critical aspects of society, from health care to finance, it becomes vital to trust the models driving these decisions. Transparency allows users to build trust and ensure ethical and responsible behavior in

Editors: Riccardo Guidotti, Anna Monreale, Dino Pedreschi.

✉ Kirsten Köbschall
koebschall@uni-mainz.de

[1] Johannes Gutenberg University Mainz, Mainz, Germany

machine learning supported decision processes even for high-risk applications (Panigutti et al., 2023). Furthermore, as AI systems often operate in complex environments, the integration of data streams enhances the system's ability to respond efficiently and responsibly, ensuring that resource limitations are taken into account. This balance of transparency and resource awareness is key to developing AI models that are both effective and ethically aligned.

In this work, we propose Soft Hoeffding Trees (SoHoT) as transparent and differentiable decision trees designed for data streams, providing adaptive data processing as visualized in Fig. 1. First, we introduce SoHoTs and conduct a detailed complexity analysis to provide deeper insights into its computational efficiency. Moreover, we enhance the transparency of the tree algorithm by introducing new methods, such as a customized feature importance, that improve interpretability and accountability in its decision-making process. Last, we perform extensive experiments to evaluate the robustness of our approach across a broader range of state-of-the-art methods for data streams. SoHoTs demonstrate superior efficiency and adaptability across diverse datasets, especially those with various types of drift. It consistently outperforms soft trees and Hoeffding adaptive tree in balancing AUROC and model size while maintaining competitive AUROC performance. Integrating backpropagation with decision trees introduces challenges due to increased complexity. To mitigate this, we employ a sparse activation function (Hazimeh et al., 2020) and show in our complexity analysis that it adds only minor overhead—limited to worst-case scenarios—while remaining asymptotically comparable to recently proposed efficient soft trees. The main contributions of this paper are: (i) We propose a new model for data streams that allows tuning the trade-off between prediction performance and transparency by a hyperparameter (called α in the paper), we do this by (ii) a new transparent routing function and (iii) show how to split nodes in a data stream setting by applying Hoeffding's inequality in a gradient-based tree. We (iv) introduce a metric to measure feature importance in a soft tree and soft Hoeffding tree. Extending our previous conference paper (Köbschall et al., 2025), we additionally provide (v) a complexity analysis of SoHoTs, (vi) a comparison with state-of-the-art stream learning methods, especially with respect to the performance relative to the model complexity, (vii) a statistical analysis of the results, (viii) a runtime analysis, and (ix) a brief qualitative analysis of the results.

This paper is structured as follows: Related work is discussed in Sect. 2. Section 3 presents the new method, Soft Hoeffding Trees (SoHoT). An explanation of the transparency property is given in Sect. 4. The results are presented in Sect. 5, and Sect. 6 concludes our work.

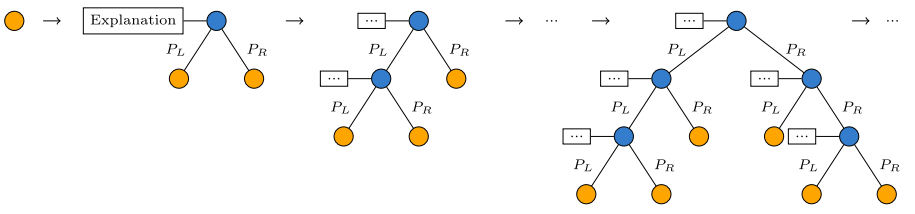


Fig. 1 SoHoT adaptation to an evolving data stream over time, denoted by routing probabilities to the left child (P_L) or right child (P_R)

2 Related work

Hoeffding trees (HT) (Domingos & Hulten, 2000), as introduced by Domingos and Hulten in 2000, are an algorithm for mining decision trees from continuously changing data streams, which is still the basis for many state-of-the-art learners for data streams (Gouk et al., 2019; Gunasekara et al., 2023). We briefly review the principle of HT in order to apply some individual techniques in Sect. 3. In contrast to standard decision trees, HT uses the Hoeffding bound to determine whether there is sufficient evidence to select the best split test or if more samples are needed to extend the tree. More precisely, let $G(x_i)$ be the heuristic measure (e.g., information gain) for an attribute x_i to evaluate for a split test and $\bar{G}(x_i)$ be the observed value after n samples. A Hoeffding tree guarantees with probability $1-\delta$ that the attribute chosen after seeing n samples is the same as if an infinite number of samples had been seen. If $\bar{G}(x_a) - \bar{G}(x_b) > \epsilon$, where x_a is the attribute with the highest observed $\bar{G}$ after n samples, x_b the second-best attribute, then x_a is the best attribute to perform a split on a leaf node with probability $1-\delta$, where δ is the significance level, $\epsilon = \sqrt{(R^2 \ln 1/\delta)/(2n)}$ and $R = \log k$ and k the number of classes. Hulten et al. proposed the Concept-adapting Very Fast Decision Tree learner (CVFDT) (Hulten et al., 2001) to adapt to changing data streams by building an alternative subtree and replacing the old with the new as soon as the old subtree becomes less accurate. Bifet and Gavaldà (2009) proposed the Hoeffding window tree (HWT) and the Hoeffding adaptive tree (HAT) as a sliding window approach and an adaptive approach, respectively, to deal with distribution and concept drift based on change detectors and estimator modules. The HWT maintains a sliding window of instances, and the HAT overcomes the issue of having to choose a window size by storing instances of estimators of frequency statistics at each node. Naive Bayes Hoeffding trees are a hybrid adaptive method having trees with naive Bayes models in the leaf nodes (Holmes et al., 2005). For each training sample, a naive Bayes prediction is made and compared to the majority class voting. Another variant of Hoeffding trees is to replace the model in the leaf nodes with a perceptron classifier (Bifet et al., 2010). Manapragada et al. introduced in 2018 in incremental tree denoted as Extremely Fast Decision Tree (EFDT) (or Hoeffding Anytime Tree) (Manapragada et al., 2018). EFDT reevaluates split decisions and updates them when a superior split emerges, ensuring faster convergence compared to HT. Nevertheless, decision trees (esp. Hoeffding trees) are non-differentiable, since hard routing (i.e., a sample can only be routed as a whole to the left or the right) causes discontinuities in the loss function (Hazimeh et al., 2020),[1] making them incompatible with end-to-end learning and, therefore unsuitable for integration into neural networks. In deep learning pipelines, where a feature representation is learned, it is desirable to use differentiable models to enable end-to-end learning.

Ensemble methods for data streams (Oza & Russell, 2001) are widely used and include algorithms like adaptive random forests (ARF) (Gomes et al., 2017), which feature effective resampling and adaptive operators such as drift detection and recovery strategies. More recent work on ensemble methods proposed Streaming Random (SRP) (Gomes et al., 2019), combining random subspaces and online bagging. In this study, we focus on base learners to analyze their individual capabilities without adding aggregation complexity.

Since hard-routing decision trees lack a good mechanism for representation learning (Hazimeh et al., 2020), soft trees (ST) were introduced as differentiable decision trees. Ini-

[1] Hard routing is based on hard thresholds, creating a piecewise constant function that maps each region in feature space to a discrete output.

tially proposed by Jordan and Jacobs (1993), they were further developed in various directions by other researchers (Kontschieder et al., 2016; Frosst & Hinton, 2017; Hehn et al., 2020). In the following section, we briefly discuss soft trees to establish a stronger foundation for explaining individual mechanisms. Soft trees perform soft routing, i.e., an internal node distributes a sample simultaneously to both the left and the right child, allowing for different proportions in each direction. A common choice for the gating function is the sigmoid function. Hazimeh et al. (2020) introduced the smooth-step function S as a continuously differentiable gating function, i.e.,

$$S(t) = \begin{cases} 0 & t \leq -\gamma/2 \\ -\frac{2}{\gamma^3}t^3 + \frac{3}{2\gamma}t + \frac{1}{2} & -\gamma/2 \leq t \leq \gamma/2 \\ 1 & t \geq \gamma/2, \end{cases} \tag{1}$$

where γ is a non-negative scalar. A property of S is the ability to output exact zeros and ones, which provides a balance between soft and hard routed samples and enables a conditional computation. Along with the smooth-step function, the authors proposed the tree ensemble layer (TEL) for neural networks as an additive model of soft trees utilizing S and concluded that TEL has a ten-fold speed-up compared to differentiable trees and a twenty-fold reduction in the number of parameters compared to gradient boosted trees. İrsoy et al. proposed an incremental architecture with soft decision trees (İrsoy et al., 2012), where the tree grows incrementally as long as it improves. Hehn et al. (2020) proposed a greedy algorithm to grow a tree level-by-level by splitting nodes, optimizing it by maximizing the log-likelihood on subsets of the training data. Stochastic gradient trees (Gouk et al., 2019) are an incremental learning algorithm using stochastic gradient information to evaluate splits and compute leaf node predictions. The tree applies t-tests instead of the Hoeffding inequality to decide whether a node is to be split. Generally, differentiable decision trees commonly lack transparency and the facility to adjust to concept drift. Finally, they do not have the ability to learn and dynamically build the model's architecture from a data stream.

Recent work has proposed various approaches to instance-level explanation. Mignone et al. (2024) introduce an instance-based feature ranking using self-organizing maps, where the importance of each feature is derived from its contribution to the Euclidean distance between an input instance and its best-matching neuron. Swamy et al. (2025) propose InterpretCC, a class of interpretable neural networks that leverage sparse gating mechanisms to activate only a subset of features, making the decision process explainable. Lastly, Zhong and Mattijs (Li & Leeuwen, 2023) focus on explaining anomaly detection by ranking features based on their influence on each prediction, particularly in temporal and spatial contexts. In contrast to these approaches, our method employs a novel transparent routing mechanism that ranks features by their contribution to each branch and the final prediction, while filtering out less relevant features from the explanation. The key differences of soft Hoeffding trees in comparison to Hoeffding trees and soft trees are summarized in Table 1.

Table 1 Key differences of our method, soft Hoeffding trees (SoHoT), to Hoeffding trees (HT) and soft trees (ST)

	Our method (SoHoT)	HT	ST
Data streams	✓	✓	Must be adjusted
Differentiable	✓	×	✓
Routing mechanism	Soft and hard routing	Hard routing	Soft routing
Adaptation to drifts	Weight optimization (gradient descent) and growth	Growth	Weight optimization (gradient descent)
Transparency	Feature importance, split tests, tree structure	Split tests, tree structure	Tree structure

3 Soft Hoeffding tree

We propose soft Hoeffding trees as a transparent and differentiable decision tree over data streams. Fundamentally, a SoHoT is an adaptive, transparent tree that intuitively reveals the inference process for incoming examples. The tree's ability to process data streams eliminates the need for data storage, reducing memory and computational overhead (Gama, 2012). SoHoTs apply a new routing function and enlarge the tree dynamically using the Hoeffding inequality, which is explained in detail next. Its transparency is detailed in Sect. 4, while Sect. 5 analyzes its performance against state-of-the-art methods and execution times.

3.1 Problem formulation

We consider a supervised learning setting where $\mathfrak{S} = \{(x_t, y_t) | t \in \{0, \dots\}\}$ is an open-ended sequence of feature vectors $x_t \in \mathcal{X}$ and labels $y_t \in \mathcal{Y}$. Let $\mathcal{X} \subseteq \mathbb{R}^p$ be the input space and $\mathcal{Y} \subseteq \mathbb{R}^k$ be the output space, where k equals the number of classes. While we focus on classification, our method trivially extends to regression with $k = 1$. A soft Hoeffding tree $T : \mathcal{X} \to \mathbb{R}^k$ is a full binary tree of maximum depth $d_{\max}$ that iteratively observes data from $\mathfrak{S}$. T has a set of internal nodes $\mathcal{I}$ and leaf nodes $\mathcal{L}$. Each internal node $i \in \mathcal{I}$ has a weight $w \in \mathbb{R}^p$ and holds a split decision, and each leaf node $l \in \mathcal{L}$ has a weight $o \in \mathbb{R}^k$ and holds sufficient statistics to measure impurity of each feature using a function G such as information gain. Moreover, let $\mathcal{W} := \{w_i \mid i \in \mathcal{I}\}$ and $\mathcal{O} := \{o_l \mid l \in \mathcal{L}\}$. For any observed input (x_t, y_t), the tree updates its node weights via gradient descent and determines whether to expand its structure to adapt to potential data drift. A new routing function R (Eq. 2) is used to guide a sample through T. We begin by introducing the forward pass to determine a prediction for an input, followed by the backward pass to show how the tree adapts to drifts. Table 2 lists the notation used throughout the paper.

At the beginning, T consists of a single leaf node.

3.2 Prediction

The input x,[2] starting at the root node i, is routed to the left child (if it exists), which is denoted by $\{x \swarrow i\}$, with probability $P(\{x \swarrow i\})$ and to the right child (if it exists) with

[2] For simplicity, we omit the subindex t when the time frame is irrelevant.

Table 2 Summary of notation used

Notation	Space or type	Explanation
$\mathcal{X}$	$\mathbb{R}^p$	Input feature space
$\mathcal{Y}$	$\mathbb{R}^k$	Output label space
$T(x)$	Function	The output of SoHoT. Formally $T : \mathcal{X} \to \mathbb{R}^k$
$\mathcal{I}$	Set	Set of internal nodes
$\mathcal{L}$	Set	Set of leaf nodes
w_i	$\mathbb{R}^p$	Weight vector of internal node i
o_l	$\mathbb{R}^k$	Weight vector of leaf node l
t	$\mathbb{N}$	Time point
$\mathcal{W}$	Set	Set of internal node weights
$\mathcal{O}$	Set	Set of leaf node weights
R	Function	The routing function of SoHoT. Formally $R : \mathcal{X} \times \mathcal{I} \to \mathbb{R}$
$d_{\max}$	$\mathbb{N}$	Maximum tree depth

probability $P(\{x \searrow i\}) = 1 - P(\{x \swarrow i\})$. To achieve both a transparent model and a transparent prediction, a new routing mechanism is proposed as follows.

Definition 1 Let $R : \mathcal{X} \times \mathcal{I} \to \mathbb{R}$ be a function, $\alpha \in [0, 1]$ be a parameter, x_a the split feature, θ the split value and "$(x_a < \theta)$?" the split test computed for the internal node i. Further, $\mathbb{1}(\cdot)$ is the indicator function, which is 1 if the condition in $(\cdot)$ is true and 0 otherwise. Then, our new routing function R is defined as

$$R(x, i) := P(\{x \swarrow i\}) = \alpha \cdot S(\langle w_i, x \rangle) + (1 - \alpha) \cdot \mathbb{1}(x_a < \theta), \tag{2}$$

where $w_i \in \mathbb{R}^p$ is the weight vector of i.
However, R is a convex combination of the split test on i and the routing probability by applying the continuously differentiable smooth-step function S (Eq. 1). In other words, the probability of routing x to the left child is determined by the combined and weighted impact of the multivariate split by $S(\langle w_i, x \rangle)$ and the univariate split test by $\mathbb{1}(x_a < \theta)$. The parameter α regulates the transparency of the model. Specifically, when α becomes smaller, the model's transparency increases, as the split tests provide users with explicit information about the criterion used for decision-making. However, performing more multivariate splits may also lead to higher performance, as we evaluate in Sect. 5.4. Note that with $\alpha = 0$, a SoHoT acts similarly to a Hoeffding tree, and with $\alpha = 1$, a forward pass similar to soft trees is performed, and the performance should resemble the performance of ST once the tree is fully grown.

We denote $P(\{x \to l\})$ as the probability that x reaches the leaf $l \in \mathcal{L}$ determined by the product of the probabilities on the path from root towards l. The final prediction for x is then defined as

$$T(x) = \sum_{l \in \mathcal{L}} P(\{x \to l\}) o_l, \tag{3}$$

where $o_l \in \mathbb{R}^k$ is the weight vector of a leaf $l \in \mathcal{L}$. For classification tasks, the softmax function can be applied to $T(x)$ to obtain class probabilities. Figure 2 shows an example

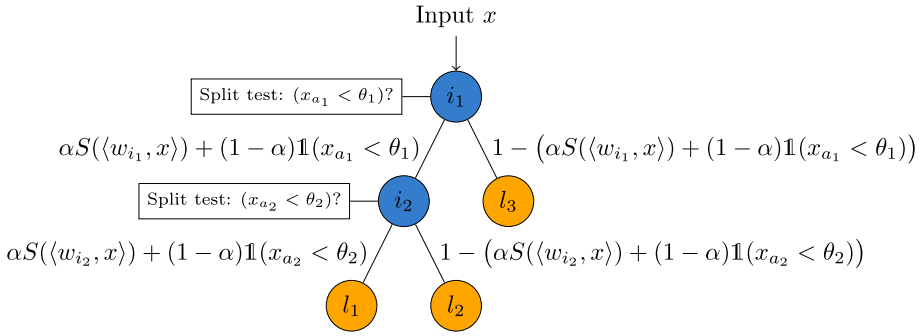


Fig. 2 Soft Hoeffding tree T at time $t > 0$ with routing probabilities at the edges

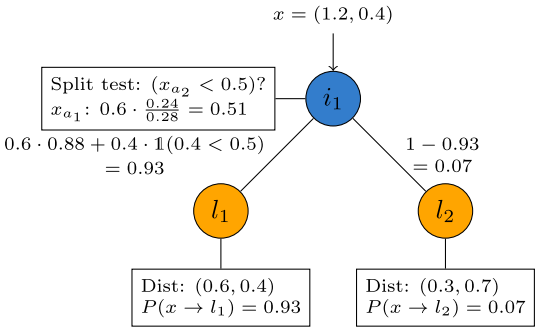


Fig. 3 Example routing to a SoHoT with α=0.6, p=2 input features, weights w_i=(0.2, 0.1), $o_{l_1} = (0.5, 0.1)$, $o_{l_2} = (0.1, 0.9)$. The output $T(x) = P(x \to l_1) \cdot o_{l_1} + P(x \to l_2) \cdot o_{l_2} = (0.47, 0.16)$, and the class probabilities after softmax function are (0.58, 0.42) and the tree predicts class 1 with probability 0.58. To compute the feature importance (introduced later in Sect. 4) as explanation, we determine $\sigma = |0.2 \cdot 1.2| + |0.1 \cdot 0.4| = 0.24 + 0.04 = 0.28$ and show the feature a_1 next to node i as the feature importance of x_{a_1} is $0.51 > \frac{1}{2}$. The second feature x_{a_2} is not important according to Definition 3, but is used inside the split test. A user interprets the output as follows: The input reaches leaf node l_1 with probability 0.93, where class 1 is predicted with probability 0.6. This decision primarily relies on feature x_{a_1}, with a split on x_{a_2} at 0.5. With probability 0.07, x is routed to the right subtree, where class 2 is predicted with probability 0.7

of a SoHoT T at a time step $t > 0$, including the routing probabilities on each edge. We exploit the properties of S and compute each root-to-leaf probability only for reachable leaves, which means $P(\{x \to i\}) > 0$ for any node i on the path from the root to a leaf. The conditional computation implies that eventually subtrees might not be visited during the forward and backward pass. This makes the online framework particularly resource-efficient, as discussed for S (Hazimeh et al., 2020). An example of transparent routing in a SoHoT is given in Fig. 3.

3.2.1 Time complexity

A single forward pass for x ensures that x encounters each node at most once. Computing the routing probability (Eq. 2) requires $\mathcal{O}(k)$ operations, while computing each output variable

in the leaves (Eq. 3) requires $\mathcal{O}(p)$ operations, leading to a worst-case time complexity of $\mathcal{O}(|\mathcal{I}|k + |\mathcal{L}|p))$.

3.2.2 Space complexity

The space required for the forward pass varies based on whether it is used for inference or training. In general, traversing a full binary tree in a depth-first manner has a complexity of $\mathcal{O}(d_{\max})$, as the recursion stack stores at most one node per level. If the forward pass is used for subsequent training, routing probabilities for an internal node $i \in \mathcal{I}$ and a leaf node $l \in \mathcal{L}$, i.e., $\langle w_i, x \rangle$ and $P(\{x \to l\})$ respectively, need to be stored. This requires a worst-case space complexity of $\mathcal{O}(d_{\max} + |\mathcal{I}| + |\mathcal{L}|)$.

3.3 Adaptation to drifting data

The training of T to adapt to the data stream can be captured in two steps: the gradient computation and the growth of the tree structure; and is summarized in Algorithm 1. Let L be a loss function. Given T, the input $x \in \mathbb{R}^p$ and $\frac{\partial L}{\partial T}$, which should be available from the backpropagation algorithm as an input to the backward algorithm. T is traversed in post-order to compute the gradients and to update the statistics in the leaf nodes. The gradients of each weight in $\mathcal{W}$ and $\mathcal{O}$ and the input x are calculated as proposed for TEL (Hazimeh et al., 2020). To compute gradients, the forward operations must be differentiable. For drift adaptation, we leverage the continuously differentiable component of S in Eq. 2 to enable gradient calculation. Note that the split tests are non-differentiable and are therefore excluded from this process. Beyond the gradient computation of the weights, we also focus on the selection of split tests within the tree. In order to determine univariate splits on a data stream, we use the Hoeffding tree methodology here. First, we collect the statistics of the incoming instances in the leaf node l to compute potential split tests later, but only if the maximum depth $d_{\max}$ on this path has not yet been reached and $P(\{x \to l\}) > \epsilon_s$, where $0 < \epsilon_s \leq 1$[3]. A cutoff ϵ_s is chosen since certain examples have a limited likelihood of reaching leaf nodes, resulting in their inclusion in predictions with a relatively small proportion; therefore, these statistics are not relevant for any new univariate split in l. As soon as the tree traversal is finished, the algorithm iterates over all leaves and attempts to split (see Algorithm 2).[4] Splitting follows the Hoeffding tree approach using the Hoeffding inequality (Sect. 2), where a leaf node becomes an internal node based on an impurity measure G (e.g., information gain) to select the split attribute x_a and apply the test "$(x_a < \theta)$?". When extended, a leaf node l becomes an internal node i with two child nodes l_1 and l_2, i.e., $\mathcal{I} = \mathcal{I} \cup \{i\}$, and $\mathcal{L} = (\mathcal{L} \cup \{l_1, l_2\}) \setminus \{l\}$, and $\mathcal{W} = \mathcal{W} \cup \{w_i\}$, and $\mathcal{O} = (\mathcal{O} \cup \{o_{l_1}, o_{l_2}\}) \setminus \{o_l\}$. An example of node splitting is given in Fig. 11 in the appendix. However, a sample may reach multiple leaf nodes due to the soft-routing mechanism, and thus, the splits may differ from those of Hoeffding trees. After the tree has been traversed, gradient descent can be performed based on the computed gradients.

[3] A reasonable choice is, e.g., $\epsilon_s = 0.25$.

[4] Since all features are numerical, split candidates are determined by discretizing feature values using, e.g., a binning approach.

1: **Input:** T, $x \in \mathbb{R}^p$, $\frac{\partial L}{\partial T}$, ϵ_s
2: **Output:** $\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial W}$, $\frac{\partial L}{\partial O}$
3: Initialize $\frac{\partial L}{\partial x} = 0$
4: Traverse T in post-order:
5: i is the current node
6: **if** i is a leaf **then**
7: Compute $\frac{\partial L}{\partial o_i}$
8: **if** i.depth $\leq d_{\max}$ **and** $P(\{x \to i\}) > \epsilon_s$ **then**
9: Update leaf node statistics
10: **end if**
11: **else**
12: Compute $\frac{\partial L}{\partial x_i}$ and $\frac{\partial L}{\partial w_i}$
13: **end if**
14: Iterate over all leaves and attempt to split
15: **Return:** $\frac{\partial L}{\partial x}$, $\frac{\partial L}{\partial W}$, $\frac{\partial L}{\partial O}$

Algorithm 1 Backward pass

3.3.1 Time complexity

Computing the gradient of $o_i \in \mathcal{O}$ during the backward pass in Algorithm 1 in Line 7 requires $\mathcal{O}(k)$ operations per leaf (see Hazimeh et al., 2020). Line 9 requires $\mathcal{O}(p)$ operations, as it maintains frequency counts for each of the p attributes, each with v possible values. Since attributes are numerical, their values are grouped into fixed discrete intervals (bins) for efficient frequency counting, keeping the number of possible values v constant. Line 12 requires $\mathcal{O}(p)$ operations for vector-scalar multiplication (see Hazimeh et al., 2020) per internal node and input sample. The attempt to split (Algorithm 2) finds a possible split candidate for each attribute, which takes $\mathcal{O}(pv) = \mathcal{O}(p)$. Therefore, the worst-case time complexity is

$$\begin{aligned}\mathcal{O}(|\mathcal{L}|(k + pv + pv) + |\mathcal{I}|p) &= \mathcal{O}(|\mathcal{L}|(k + p) + |\mathcal{I}|p) \\ &= \mathcal{O}(|\mathcal{L}|(k + p)),\end{aligned}$$

as $|\mathcal{I}| = |\mathcal{L}| - 1$. The best-case time complexity is $\mathcal{O}(k + d_{\max}p)$ with $|\mathcal{L}| = 1$ when only one leaf node is visited.

1: **Input:** Leaf l, $d_{\max}$, tie breaking threshold τ, impurity measure G
2: **if** samples seen so far are not from the same class **and** $d_{\max}$ not reached **then**
3: Find split candidates[4] by computing $\bar{G}(x_j)$ for each attribute x_j
4: Select the top two x_a, x_b
5: Compute Hoeffding bound ϵ
6: **if** $\left((\bar{G}(x_a) - \bar{G}(x_b) = \Delta\bar{G}) > \epsilon$ **and** $x_a \neq x_\emptyset\right)$ **or** $\epsilon < \tau$ **then**
7: Split l
8: **end if**
9: **end if**

Algorithm 2 Attempt to split

3.3.2 Space complexity

During the backward pass, the tree is traversed in post-order, requiring at most one node per level to be tracked, resulting in a space complexity of $\mathcal{O}(d_{\max})$. Additionally, each leaf stores a frequency counter for every attribute-value pair (line 9), requiring $\mathcal{O}(pv) = \mathcal{O}(p)$ space. Since this statistic is no longer needed once the maximum depth limit is reached, it improves space efficiency, especially for large datasets. However, computing the gradients in line 12 requires storing one scalar per internal node (Hazimeh et al., 2020). Thus, the space complexity is $\mathcal{O}(d_{\max} + |\mathcal{L}|p)$. If the maximum depth is reached and γ is chosen very small, leading to many zero gradients, Algorithm 1 requires constant space.

4 Transparency

Post-hoc interpretations often fail to explain clearly how a model works. Therefore, we focus on transparency, aiming to elucidate the model's functioning rather than merely describing what else the model can tell the user (Lipton, 2016). We examine the transparency of a SoHoT and how to measure it compared to soft trees at the level of the entire model (simulatability). To make the model and the prediction transparent and explainable, we follow two principles: growth and inclusion of split tests. To enhance tree transparency, we introduce a feature importance measure for internal nodes to quantify each attribute's relevance to the prediction, which we also evaluate in Sect. 5.4.

4.1 Measure transparency

We aim to compare the interpretability of the decision rules by the number of important features. The fewer important features, the easier and shorter the explanation and, therefore, the more transparent the model. First, we define an important feature for soft trees and then adapt it to SoHoTs.

Definition 2 Let $T : \mathcal{X} \to \mathcal{Y}$ be a full binary tree with internal nodes $\mathcal{I}$ and leaf nodes $\mathcal{L}$, associated weights $\mathcal{W}$ and $\mathcal{O}$ as previously defined, and a soft routing decision $P(\{x \swarrow i\}) = S(\langle w_i, x_t \rangle)$, where $i \in \mathcal{I}$, $w_i \in \mathcal{W}$ and $x_t \in \mathcal{X}$. We define a feature a as important if its weighted proportion of x_t exceeds a uniform distribution relative to the number of features p, i.e., if the following holds

$$\frac{|w_{i,a} x_{t,a}|}{\sigma} \geq \frac{1}{p}, \tag{4}$$

where $w_{i,a}$ ($x_{t,a}$) denotes the ath entry of w_i (x, respectively) and $\sigma := |w_{i,1}x_1| + |w_{i,2}x_2| + \cdots + |w_{i,p}x_p|$.
However, to assess an attribute's importance, we measure the extent to which each summand in $w_{i,1}x_1 + w_{i,2}x_2 + \cdots + w_{i,p}x_p$ is weighted more strongly (positively or negatively) by w_i at decision node i.

Definition 2 holds for soft trees and SoHoTs with routing function R and $\alpha = 1$. For $\alpha \neq 1$, the splitting attribute in the splitting test may also emerge as one of the important

attributes. Hence, we apply the convex weighting on the impact rule to define important features under the routing function R. More precisely, we weight the percentage impact of a feature on the soft decision rule (left side of Eq. 4) and its weighted impact when used as a splitting attribute in the split test.

Definition 3 Let $T : \mathcal{X} \to \mathcal{Y}$ be a full binary tree with internal nodes $\mathcal{I}$ and leaf nodes $\mathcal{L}$, associated weights $\mathcal{W}$ and $\mathcal{O}$ as previously defined, and a soft routing decision R. We define a feature a as important if its weighted proportion of x_t exceeds a uniform distribution relative to the number of features p, or if a is chosen as the split attribute in i and $1-\alpha$ surpasses this threshold, i.e., if the following holds

$$\alpha \cdot \frac{|w_{i,a} x_{t,a}|}{\sigma} \geq \frac{1}{p} \text{ or } (1-\alpha) \cdot \mathbb{1}(\text{split attribute on } i \text{ is } a) \geq \frac{1}{p}. \tag{5}$$

We use these definitions to assess both the length and content of the explanation in the following section.

5 Experiments

We study the performance of SoHoTs in terms of prediction, drift adaptation, and transparency. We evaluate SoHoTs in comparison to Hoeffding tree (HT), Hoeffding adaptive tree (HAT), extremely fast decision tree (EFDT), streaming stochastic gradient descent classifier (SGDClassifier) from library CapyMOA (Gomes et al., 2025), and soft trees (ST) (Hazimeh et al., 2020). SGDClassifier is a linear classifier that utilizes stochastic gradient descent for learning, specifically wrapped to facilitate use in streaming contexts.

5.1 Experimental setting

5.1.1 Model implementation

SoHoT is implemented in PyTorch 2.1 (Paszke et al., 2019), and we provide an open-source Python implementation.[5] SoHoT and ST perform online one-hot encoding and normalization while using the Adam optimizer (Kingma & Ba, 2015) and cross-entropy loss. We assume a test-then-train setting, with all measurements averaged over five random repetitions. To obtain the benefits of the models, we employ an efficient per-instance training of a pool of models to enable hyperparameter tuning on streams (Gunasekara et al., 2022). The model with the lowest estimated predictive performance is chosen for every prediction, and half of the models in the pool are selected for training. The pool consists of models with all different hyperparameter combinations. Additional details are in the appendix. To ensure a fair evaluation, we performed hyperparameter tuning for data streams, aiming to showcase the best possible results. However, the improvements were generally modest, as all methods produced outcomes that remained close to those obtained with default parameters.

[5] https://github.com/kramerlab/SoHoT.

5.1.2 Datasets

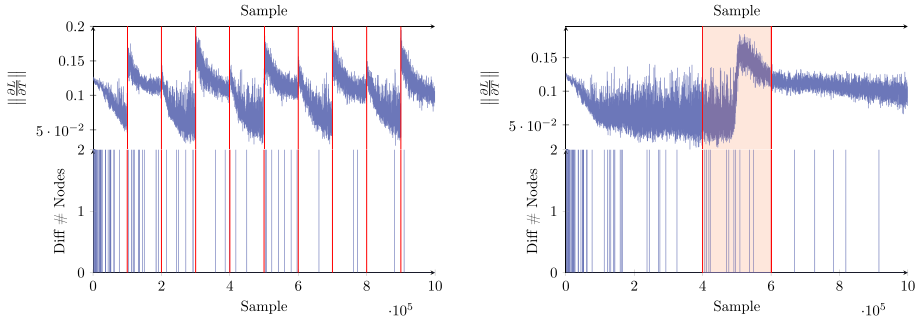
We employ 20 classification data streams (binary and multiclass; Table 5), 8 are purely synthetic, 4 are large real-world data streams [both are used in various literature (Bifet et al., 2013; Gunasekara et al., 2022)], and the remaining 8 are synthetic streams derived from real-world data streams [from the PMLB repository (Olson et al., 2017)] and generated as follows. We employ a Conditional Tabular Generative Adversarial Network (CTGAN) (Xu et al., 2019) to emulate a realistic distribution within a substantial data stream of 10^6 samples and injected abrupt drifts by oversampling a randomly selected class. Ten drifts are induced, and each context contains samples in which around 75% belong to the randomly selected class.

5.2 Drift adaptation

Soft Hoeffding trees adapt to drifting data streams by updating the weights in the tree by gradient descent and growing the tree structure by splitting leaf nodes. Figure 4 illustrates the extent to which a SoHoT adapts to a drifting data stream by visualizing $\left\| \frac{\partial L}{\partial T} \right\|$ and the number of added nodes over time. In the beginning, the tree grows to establish a foundation to accommodate the data stream. Following the abrupt drift in Fig. 4, the gradient rises, prompting updates to the node weights $\mathcal{W}$ and $\mathcal{O}$ as the model adapts to the new classification function. A similar pattern occurs with gradual drift, where adaptation happens later.

5.3 Comparison with state-of-the-art algorithms

Since this paper primarily investigates the trade-off between transparency and predictive performance, several relevant metrics are considered. We assess predictive performance using AUROC; however, it does not capture the reasoning behind the prediction. Comparison models tend to have more nodes, making their explanation larger and more complex. To account for this, we introduce a fair comparison metric to better illustrate the trade-off. The transparency property from Sect. 4 does not apply to HT, HAT, or EFDT decision rules and



(a) Agrawal with abrupt drift. (b) Agrawal with gradual drift.

Fig. 4 Drift adaptation for a single SoHoT on Agrawal stream with **a** abrupt and **b** gradual drift (indicated by red lines). The difference in number of nodes in a SoHoT and $\left\| \frac{\partial L}{\partial T} \right\|$, where T is the output of the tree and L the loss function, shows how a SoHoT adapts to different types of drifts over time (Color figure online)

will be examined in Subsection 5.4. To standardize comparisons, we measure explanation length by the average number of nodes, aligning with the transparency motivation in Sect. 4. *AUROC per logarithm of complexity* (APLC) is then defined as

$$\text{APLC} = \frac{\text{AUROC}}{\log_2(\text{average number of nodes})}. \tag{6}$$

The AUROC in the numerator is a measure of the sorting or ranking capacity of a classifier, being the probability that an instance of the positive class is sorted before an instance of the negative class according to the class probability estimates. The logarithm of the number of nodes in the denominator is a measure of the complexity of a tree, being the depth of a tree with the same number of nodes if that tree was balanced. Thus, APLC is a measure of the sorting capacity of the tree per depth, if the tree was balanced with the same number of nodes.

Larger values for APLC are preferable as they lead to a higher AUROC and a more compact model. We compare AUROC and APLC (Eq. 6) of SoHoT and several state-of-the-art online algorithms on 20 datasets in Table 3. SoHoTs outperform the comparison methods in APLC and demonstrates superior performance on datasets with different types of drift, achieving higher AUROC than HAT in cases of abrupt (SEA_f, SEA_m) and gradual (HYP_f, HYP_m). More details on the average number of nodes for each method and dataset are in the appendix (Fig. 12). While ST achieves the best AUROC overall, its fixed tree size results in lower APLC and longer explanations despite good predictive performance. HAT, on the other hand, adapts to concept drift by replacing subtrees, leading to smaller model sizes but a lower mean rank in terms of AUROC. Overall, SoHoTs strikes the best balance, excelling in both APLC and adaptability. We evaluated four regression datasets in Sect. B.2, and we observe that SoHoTs perform competitively with state-of-the-art streaming regression methods. However, we emphasize that SoHoTs were primarily developed for classification tasks. The win-count matrix in Fig. 5 summarizes the number of datasets in which each method (row) statistically outperformed another (column) based on a paired t-test with a significance level of 0.05. The results highlight the competitive advantage of SoHoTs. SoHoTs achieve significantly better APLC scores than HT, EFDT, and ST. A paired t-test confirms that SoHoTs statistically outperforms ST across all datasets. While SoHoTs also outperforms HAT on 10 datasets, HAT surpasses SoHoT in only two. Critical difference diagrams using Wilcoxon signed rank test in Fig. 6 show that SoHoT outperforms HAT in terms of APLC, with no critical difference between the two. As shown in Table 3, the diagram for AUROC indicates that ST outperforms the other methods. For additional details, refer to the Appendix (Fig. 13). Overall, SoHoT is competitive with state-of-the-art online algorithms and especially with HAT, but when transparency and, therefore tree size are prioritized, SoHoTs deliver better results. To confirm SoHoTs' generality, we performed experiments on larger stationary data in Table 4. We generated data streams containing 10^7 instances using the procedure described in Sect. 5.1.2 with CTGAN, without introducing any drifts. We observe similar trends as proved in Table 3. The APLC score improved for SoHoTs, as they adapt to the data distribution by learning the weights, not just by increasing the tree depth. In the next section, we explore the advantage SoHoTs provide in terms of transparency.

Table 3 Results in terms of AUROC and APLC (see Eq. 6) on 20 datasets

Data	Metric	SoHoT	HT	HAT	EFDT	SGDClass	ST
SEA_f	AUROC	0.878	0.876	0.877	0.876	0.871	0.881
	APLC	**0.146**	0.108	0.108	0.120	–	0.122
SEA_m	AUROC	0.879	0.876	0.877	0.876	0.870	0.881
	APLC	**0.146**	0.108	0.108	0.120	–	0.122
HYP_f	AUROC	0.942	0.892	0.911	0.908	0.943	0.948
	APLC	**0.157**	0.108	0.108	0.115	–	0.131
HYP_m	AUROC	0.947	0.926	0.920	0.922	0.946	0.950
	APLC	**0.158**	0.115	0.108	0.114	–	0.131
RBF_f	AUROC	0.537	0.584	0.698	0.591	0.673	0.653
	APLC	0.115	0.099	**0.154**	0.095	–	0.090
RBF_m	AUROC	0.603	0.748	0.860	0.781	0.780	0.878
	APLC	0.127	0.118	**0.219**	0.093	–	0.122
AGR_a	AUROC	0.932	0.891	0.949	0.903	0.669	0.980
	APLC	**0.164**	0.104	0.114	0.110	–	0.136
AGR_g	AUROC	0.918	0.849	0.925	0.876	0.659	0.970
	APLC	**0.162**	0.099	0.105	0.106	–	0.134
Sleep	AUROC	0.837	0.842	0.859	0.854	0.893	0.914
	APLC	0.177	0.134	**0.303**	0.116	–	0.126
Nursery	AUROC	0.844	0.837	0.883	0.849	0.873	0.894
	APLC	**0.161**	0.135	0.156	0.133	–	0.124
Twonorm	AUROC	0.992	0.987	0.984	0.986	0.996	0.996
	APLC	**0.165**	0.122	0.117	0.119	–	0.138
Ann-Thyroid	AUROC	0.802	0.746	0.864	0.837	0.876	0.867
	APLC	**0.163**	0.105	0.161	0.119	–	0.120
Satimage	AUROC	0.786	0.743	0.793	0.757	0.652	0.842
	APLC	0.178	0.133	**0.285**	0.118	–	0.117
Optdigits	AUROC	0.695	0.674	0.722	0.660	0.657	0.720
	APLC	0.173	0.132	**0.184**	0.117	–	0.100
Texture	AUROC	0.741	0.750	0.748	0.770	0.853	0.859
	APLC	0.199	0.161	**0.331**	0.128	–	0.119
Churn	AUROC	0.791	0.661	0.783	0.703	0.679	0.814
	APLC	**0.153**	0.079	0.096	0.087	–	0.113
Poker	AUROC	0.484	0.617	0.622	0.626	0.460	0.667
	APLC	0.120	0.122	0.113	**0.125**	–	0.092
Kdd99	AUROC	0.794	0.810	0.789	0.796	0.736	0.949
	APLC	**0.151**	0.121	0.103	0.110	–	0.131
Covtype	AUROC	0.784	0.950	0.963	0.963	0.878	0.946
	APLC	**0.149**	0.146	0.143	0.130	–	0.131
Epsilon	AUROC	0.654	0.643	0.651	0.633	0.888	0.784
	APLC	0.170	0.128	0.121	**0.177**	–	0.109
Mean Rank	Mean rank AUROC	3.900	4.700	3.050	3.950	3.950	**1.450**
	Mean rank APLC	**1.450**	3.600	2.950	3.650	–	3.350

All methods are hyperparameter-tuned with ranges described in Appendix B.1. Best results regarding APLC are in **bold**. SGDClassifier has no depth since it is based on SVM

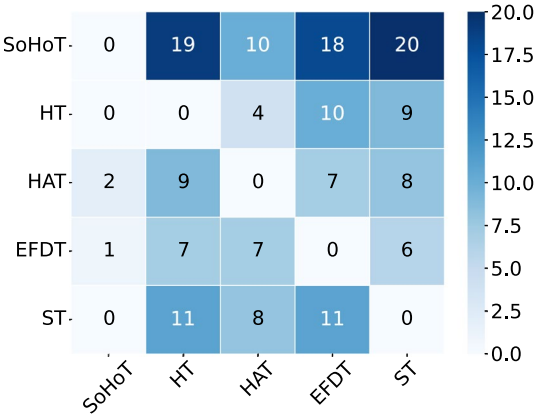


Fig. 5 Number of datasets on which a model (row) has statistically outperformed another (column) based on a paired t-test with significance level 0.05 under metric APLC

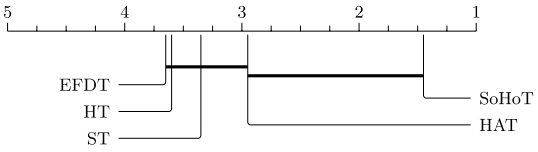


Fig. 6 Critical difference diagram in terms of APLC determined using the Wilcoxon signed-rank test with 95% confidence level and adjusted with Holm's method

Table 4 State-of-the-art comparison with stationary data without injected drifts

Data	Metric	SoHoT	HT	HAT	EFDT	SGDClass	ST
$\text{Sleep}_{\text{stationary}}$	AUROC	0.761	0.815	0.824	**0.830**	0.698	0.825
	APLC	**0.162**	0.129	0.123	0.111	–	0.114
$\text{Twonorm}_{\text{stationary}}$	AUROC	**0.725**	0.682	0.682	0.708	0.622	0.708
	APLC	**0.120**	0.085	0.080	0.089	–	0.098
Mean Rank	Mean Rank AUROC	3.000	4.500	3.500	2.000	6.000	**2.000**
	Mean Rank APLC	**1.000**	3.000	4.000	4.000	–	3.000

Best results regarding AUROC and APLC in bold

5.4 Transparency

First, we examine the length of the explanation of SoHoTs compared to ST, followed by the trade-off between transparency and predictive performance for a SoHoT, and then analyze a visualization of a SoHoT, including drift reaction.

5.4.1 Explanation length

We start by analyzing the average number of important features per decision rule for a soft Hoeffding tree and a soft tree in Fig. 7 as proposed in Sect. 4. It can be observed that, on average, the number of important features for a decision rule is higher for soft trees than for SoHoTs. This suggests that the coding length of an explanation for SoHoT is shorter and, therefore, easier than for ST. To regulate transparency, α can be employed by assigning a higher weight to a single feature from the split test. However, this adjustment may lead to a potential loss of performance, as examined next.

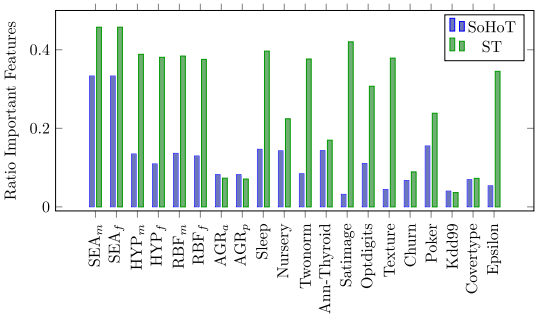


Fig. 7 The average proportion of important features per decision rule in a single SoHoT with $\alpha = 0.3$ and a soft tree (ST) for $n = 10^5$ samples of each data stream

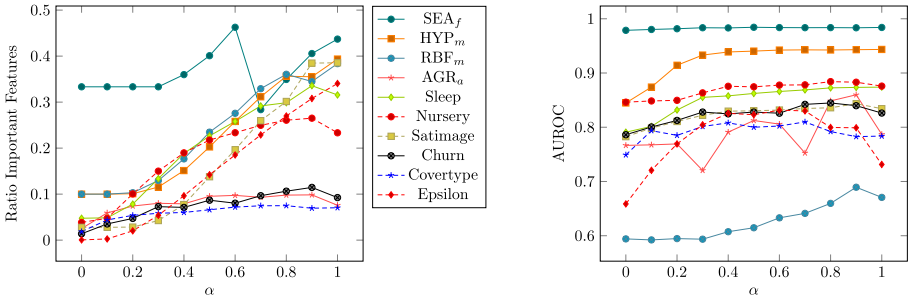


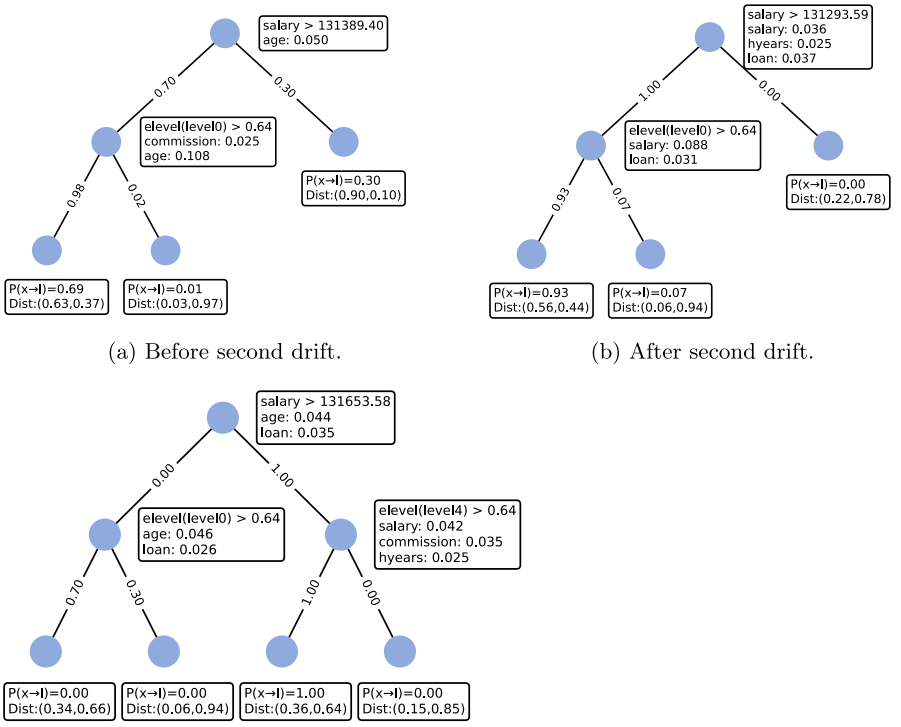
Fig. 8 Trade-off between transparency and performance for a single SoHoT with test-then-train of a tree of depth 7. The results for the other data streams have very similar trends and are omitted due to space constraints

5.4.2 Trade-off analysis

As α regulates the impact of the univariate split criterion, varying α results in a trade-off between transparency and predictive performance. Small values for α yield more univariate splits, while values close to 1 yield more multivariate splits. We performed an ablation study varying only α, while keeping all other components fixed, to illustrate its effect on the transparency-performance trade-off in Fig. 8. It shows the average proportion of important features to the total number of features for each respective data stream, and the AUROC for varying α on each data stream. The transparency ratio peak for the SEA data stream occurs when the univariate split criterion weighting is deemed unimportant (Eq. 2) due to the small number of features (3). The drop is also observed in other streams at different α values.

5.4.3 Qualitative analysis

We examine and analyze the output presented to a user in Fig. 9 on the example of the synthetic stream generator by Agrawal et al. (1993). A stream of 10,000 instances with three abrupt drifts, i.e., changing the classification function, after each 2,500 samples is used for evaluation. The explanation for the split and routing is labeled to the left of each internal node. The representation is based on an input instance x and includes the split test determined using Hoeffding's inequality, along with the feature importance calculated according to Definition 3. At each branch, the routing probability is noted on the edges of the tree. If



(a) Before second drift.

(b) After second drift.

(c) Long time after second drift.

Fig. 9 Transparent tree evolving over time on the Agrawal data stream, which includes a nominal feature *elevel* with values *level0* to *level4*. In (**a**), the tree is trained on classification function [Function 2 in Agrawal et al. (1993)] that utilizes *age* and *salary*, and after an abrupt concept drift, it adapted to a new classification function [Function 3 in Agrawal et al. (1993)] based on *age* and *elevel*. The split tests and feature importances align with the relevant features in the active classification function. After a second abrupt drift, shown in (**b**), the tree adapts to Function 5, which incorporates *age*, *salary*, and *loan*. A shift in feature importance—most notably the emergence of *loan*—reflects this change. For long-term adaptation, the tree expands by growing new subtrees and introducing additional split tests, as illustrated in (**c**)

an attribute a is nominal, it is internally one-hot encoded. When a specific feature value v of a is labeled, we denote it as $a(v)$. Since each instance undergoes internal normalization, it is necessary to revert the transformed values to their original scale. As the mean and standard deviation of each feature are updated with the inclusion of new samples of the stream, this inverse transformation undergoes slight variations. At a leaf node l, the probability of an input reaching the node $P(\{x \to l\})$ is displayed, and the class distribution is obtained by applying a softmax function to $o_l \in \mathcal{O}$. We train a SoHoT on the synthetic data stream Agrawal (Fig. 9) and observe drift adaptation shortly after the abrupt concept change. The adaptation is reflected in the shifting feature importances, which align with the relevant features in the new classification function. Long-term adaptation is achieved by growing new subtrees and incorporating new split tests.

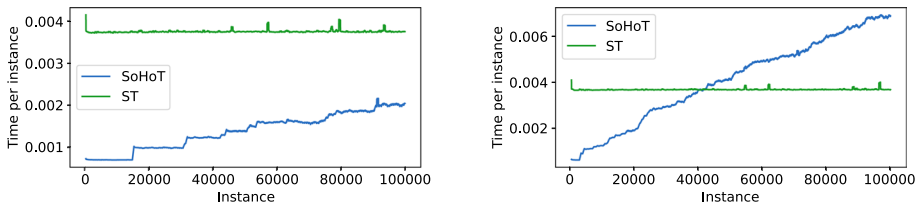
5.5 Runtime analysis

Efficient resource utilization is crucial in online learning, making training costs a key consideration. However, comparing execution times across models with different implementations poses challenges, as they may be built using different programming languages. In this study, SoHoTs are implemented in PyTorch, while ST are in TensorFlow with a C++ backend. Despite these differences, to gain practical insights into runtime performance—beyond the theoretical analysis in Sect. 3—we examine empirical results for 10^5 instances. In the asymptotic analysis, ST and SoHoTs differ in tree depth and the additional computational effort required to search for split attributes.[6] Over time, the training time per instance in the backward pass increases as the tree depth grows (see Fig. 10). Although the average number of nodes for data stream RBF_f is lower than for SEA_f (see also Fig. 12), both have shallower tree structures compared to ST. While SoHoTs exhibits asymptotically longer training times at the same depth, it takes time to reach that depth, and in some cases—depending on the dataset—SoHoT can be faster. A similar trend is observed in the forward pass, as shown also in Fig. 14 in the Appendix. Overall, similar trends for both forward and backward passes were observed across all other datasets.

6 Conclusion

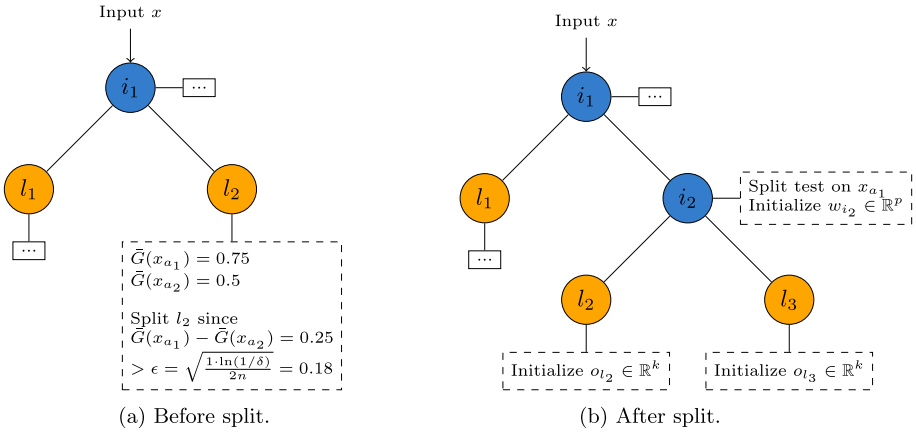
In this work, we tackle transparency concerns in infinite data streams by effectively conveying decision-making rules in a concise manner, thereby enhancing user confidence. Soft Hoeffding trees are a novel, transparent and differentiable model on data streams applying a new routing function. This tree-based model dynamically adapts to drifting data streams by growing new subtrees using the Hoeffding inequality and adapting node weights using gradient descent. The way this is done allows tuning the trade-off between prediction performance and transparency in a data stream setting explicitly by a hyperparameter (called α in the paper). We extended our previous work (Köbschall et al., 2025) with a detailed complexity analysis showing that the overall time complexity increases over time due to growing tree depth, with SoHoTs having an asymptotically longer runtime compared to soft

[6] They also differ in the set of reachable leaves—i.e., leaves l for which $P(\{x \to l\}) > 0$ holds—due to variations in their routing functions.



(a) RBF_f (b) SEA_f

Fig. 10 Runtime (in seconds) per instance for the backward pass, averaged over a window of 500 instances



(a) Before split. (b) After split.

Fig. 11 Example node splitting in a SoHoT with $p = 2$ features and confidence $\delta = 10^{-6}$. **a** The heuristic measure (e.g., information gain) $\bar{G}$ is computed for each attribute (here: a_1, a_2) inside the leaf nodes. Assume l_2 has seen $n = 200$ samples. **b** After a split, two new leaves with new weights were initialized randomly. Empirically, we found that the newly initialized weight vector exhibits rapid adaptation when w is uniformly sampled from $[-0.01, 0.01]$

trees but sometimes achieving faster execution depending on the dataset. Moreover, a transparent visualization based on feature importance during stream processing is presented. We performed extensive experiments to evaluate SoHoTs in comparison to state-of-the-art streaming methods. Our model statistically outperforms state-of-the-art online algorithms in terms of AUROC. Compared to soft trees, they offer greater transparency due to their shorter explanation length and achieve the best efficiency by effectively balancing predictive performance and model size.

It should be noted that Rutkowski et al. criticized the application of Hoeffding's inequality in mining data streams and suggested the use of McDiarmid's bound instead (Rutkowski et al., 2013). In this work, the bound was applied heuristically, as in many other machine learning papers using stochastic bounds, and McDiarmid's bound could similarly be employed as an alternative.

Table 5 Characteristics of the data streams, with e.g., abrupt (a) or gradual (g) drift, or injected perturbation (p)

Data stream	Instances	Features		Classes	Drift	Source
		Before one-hot	After one-hot			
SEA_m	10^6	3	3	2	a	s
SEA_f	10^6	3	3	2	a	s
HYP_m	10^6	10	10	2	g	s
HYP_f	10^6	10	10	2	g	s
RBF_m	10^6	10	10	5	g	s
RBF_f	10^6	10	10	5	g	s
AGR_a	10^6	9	24	2	a	s
AGR_g	10^6	9	24	2	p	s
Sleep	10^6	13	22	5	a	r,s
Nursery	10^6	8	8	4	a	r,s
Twonorm	10^6	20	20	2	a	r,s
Ann-Thyroid	10^6	21	21	3	a	r,s
Satimage	10^6	36	36	6	a	r,s
Optdigits	10^6	64	64	10	a	r,s
Texture	10^6	40	40	11	a	r,s
Churn	10^6	20	72	2	a	r,s
Poker	1,025,010	10	10	10	–	r
KDD99	4,898,430	41	122	23	–	r
Covtype	581,010	54	54	7	–	r
Epsilon	10^5	2,000	2,000	2	–	r
Fried	40,768	10	10	1	–	s
$\mathrm{HYP}_{\mathrm{reg}}$	10^6	10	10	1	g	s
House8L	22,784	8	8	1	–	r
Bikes	182,470	8	15	1	–	r

The streams are synthetic (s) generated or from real-world (r) sources. Subindices denote moderate (m) change, fast (f) change. The regression datasets are characterized by a class number of 1

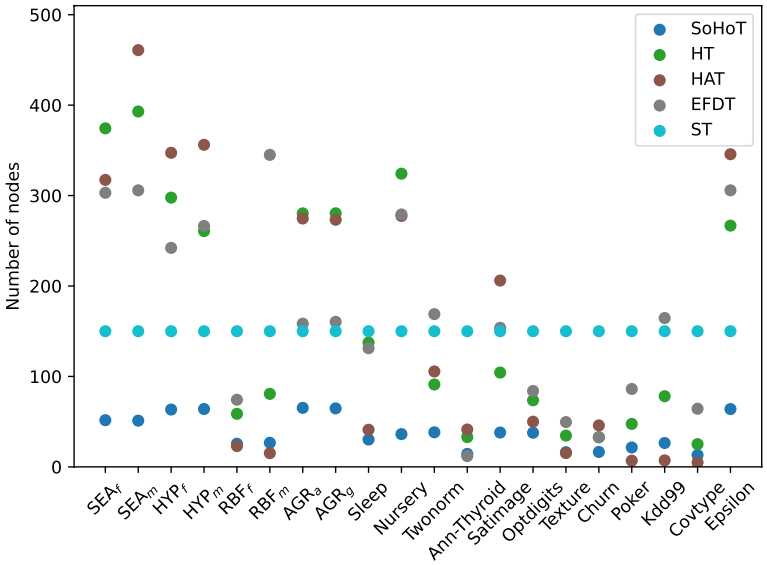


Fig. 12 Average number of nodes of experiments in Table 3

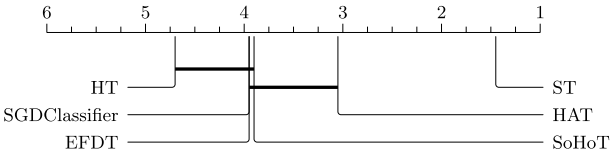
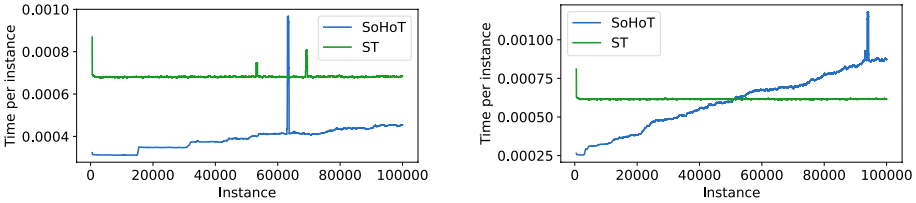


Fig. 13 Critical difference diagram in terms of AUROC determined using the Wilcoxon signed-rank test with 95% confidence level and adjusted with Holm's method

Table 6 Regression under metric rooted mean squared error (RMSE)

	SoHoT	HT	HAT	KNN	SGDReg
Fried	2.535	**2.053**	2.323	2.323	15.026
HYP^{reg}	23.420	**12.779**	13.199	22.821	297.159
House8L	39316.742	40544.527	39833.276	**37,611.863**	72800.309
Bikes	4.155	9.098	7.312	**3.133**	13.329
Mean Rank	3.000	2.500	2.750	**1.750**	5.000

Best results regarding RMSE are inbold



(a) RBF_f (b) SEA_f

Fig. 14 Runtime (in seconds) per instance for the forward pass, averaged over a window of 500 instances

Appendix: Methodology

A SoHoT splits a leaf node by selecting the best attribute for splitting based on sufficient statistical evidence using the Hoeffding bound that the chosen attribute is significantly better than the second-best. In Sect. 3.3, we described how new weights are added, and Fig. 11 illustrates an example of a depth-2 tree where a leaf node is split, extending the tree along a single path to depth 3.

Evaluation

Experimental Details

All datasets and their properties are listed in Table 5.

In Sect. 5 is explained how hyperparameter tuning is applied in the data stream setting. A list of all the tuning parameters and their value ranges is given below.

- Soft Hoeffding tree:
  - $d_{\max}$: $\{5, 6, 7\}$
  - γ: $\{1, 0.1\}$
  - α: $\{0.3, 0.4\}$
- Hoeffding tree (limited), Hoeffding adaptive tree:
  - Leaf prediction: $\{\text{majority class}, \text{naive Bayes adaptive}\}$
  - δ: $\{10^{-3}, 10^{-4}\}$
  - Grace period: $\{200, 400, 600\}$
- Extremely fast decision tree:
  - Leaf prediction: $\{\text{majority class}, \text{naive Bayes adaptive}\}$
  - Minimum samples to reevaluate: $\{200, 600\}$
  - Grace period: $\{200, 400, 600\}$
- Streaming stochastic gradient descent classifier:
  - Loss: $\{\text{log loss}, \text{modified Huber}\}$
  - Penalty: {L1,L2}
  - Learning rate: $\{10^{-2}, 10^{-3}, \text{optimal}\}$
- Soft tree:
  - $d_{\max}$: $\{5, 6, 7\}$
  - γ: $\{1, 0.1, 0.01\}$
  - Learning rate: $\{10^{-2}, 10^{-3}\}$The metric efficiency introduced in Sect. 5 in formular 6 balances predictive performance and tree size. We observed that the number of nodes in a tree of comparable state-of-the-art methods is higher than for SoHoTs as shown in Fig. 12. These values were used for efficiency in Table 3.

In Sect. 5.3, we analyzed statistical differences in the mean rank, and Fig. 13 shows these additionally under the metric AUROC.

Regression

SoHoTs can also be applied to regression tasks by setting $k = 1$. In this case, split decisions are based on heuristic measures appropriate for continuous targets, such as variance reduction, rather than information gain or Gini index. Since class probabilities are not computed in regression settings, there is no need to apply a softmax function to the output. Additionally, because target values are not constrained to the interval [0, 1], target standardization is recommended. Our implementation incorporates all of these adaptations and includes automatic detection of whether the task is classification or regression. We performed experiments on four datasets used a related paper (Verma et al., 2025). Our experiments against

state-of-the-art regressors on data streams are summarized in Table 6. Note we omitted ST from the regression experiments, as they were neither implemented nor evaluated for regression tasks in the original publication (Hazimeh et al., 2020).

Runtime analysis

Figure 14 shows the runtime measurements for the forward pass on two datasets, namely RBF_f and SEA_f.

Author Contributions K.K. contributed to the conceptualization, coding, figure preparation, and writing of the manuscript. S.K. and L.H. supervised this work and contributed to the conceptualization and provided critical feedback throughout the project. All authors reviewed and approved the final version of the manuscript.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability All datasets used in this study are publicly available, with details provided in the manuscript. •Synthetic data streams (SEA, HYP, RBF, and AGR) were generated, and the Covtype dataset was downloaded using the Python library CapyMOA. •The datasets Sleep, Nursery, Twonorm, Ann-Thyroid, Satimage, Optdigits, Texture, and Churn are based on datasets of the same names and are available at the Penn Machine Learning Benchmarks (Olson, R., La Cava, W., Orzechowski, P., Urbanowicz, R., Moore, J.: Pmlb: A large benchmark suite for machine learning evaluation and comparison. BioData Mining 10 (2017): https://epistasislab.github.io/pmlb/). •Poker Hand: Robert Cattral and Franz Oppacher, Poker Hand, UCI Repository, doi: 10.24432/C5KW38, 2006: https://archive.ics.uci.edu/ml/datasets/Poker+Hand. •KDD99: KDD Cup 1999, UCI KDD Archive, 1999: http://kdd.ics.uci.edu/databases/kddcup99/kddcup.data.gz. •Epsil on: PASCAL Challenge 2008: https://www.csie.ntu.edu.tw/cjlin/libsvmtools/datasets/

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

Agrawal, R., Imielinski, T., & Swami, A. (1993). Database mining: a performance perspective. *IEEE Transactions on Knowledge and Data Engineering, 5*(6), 914–925.

Bifet, A., & Gavaldà, R. (2009). Adaptive learning from evolving data streams. In *Proceedings of the 8th international symposium on intelligent data analysis: Advances in intelligent data analysis VIII. IDA* (pp. 249–260). Springer.

Bifet, A., Holmes, G., Pfahringer, B., & Frank, E. (2010). Fast perceptron decision tree learning from evolving data streams. In *Advances in knowledge discovery and data mining* (pp. 299–310). Springer

Bifet, A., Pfahringer, B., Read, J., & Holmes, G. (2013). Efficient data stream classification via probabilistic adaptive windows. In *Proceedings of the 28th annual ACM symposium on applied computing. SAC* (pp. 801–806). Association for Computing Machinery.

Domingos, P., & Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining. KDD* (pp. 71–80). Association for Computing Machinery.

Frosst, N., & Hinton, G. E. (2017). Distilling a neural network into a soft decision tree. In *Proceedings of the first international workshop on comprehensibility and explanation in AI and ML 2017 co-located with 16th international conference of the italian association for artificial intelligence (AI*IA). CEUR workshop proceedings* (Vol. 2071).

Gama, J. (2012). A survey on learning from data streams: Current and future trends. *Progress in Artificial Intelligence, 1*, 45–55.

Gomes, H. M., Lee, A., Gunasekara, N., Sun, Y., Cassales, G. W., Liu, J. J., Heyden, M., Cerqueira, V., Bahri, M., Koh, Y. S., Pfahringer, B., & Bifet, A. (2025). *CapyMOA: Efficient machine learning for data streams in Python*.

Gomes, H. M., Read, J., & Bifet, A. (2019). Streaming random patches for evolving data stream classification. In *2019 IEEE international conference on data mining (ICDM)* (pp. 240–249).

Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfahringer, B., Holmes, G., & Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning, 106*(9–10), 1469–1495.

Gouk, H., & Pfahringer, B. (2019). Stochastic gradient trees. In *Proceedings of The Eleventh Asian conference on machine learning research* (Vol. 101, pp. 1094–1109).

Gunasekara, N., Gomes, H. M., Pfahringer, B., & Bifet, A. (2022) Online hyperparameter optimization for streaming neural networks. In *2022 International joint conference on neural networks (IJCNN)* (pp. 1–9).

Gunasekara, N., Pfahringer, B., Gomes, H. M., & Bifet, A. (2023). Survey on online streaming continual learning. In *Proceedings of the thirty-second international joint conference on artificial intelligence, IJCAI* (pp. 6628–6637).

Hazimeh, H., Ponomareva, N., Mol, P., Tan, Z., & Mazumder, R. (2020). The tree ensemble layer: differentiability meets conditional computation. In *Proceedings of the 37th international conference on machine learning*. JMLR.org.

Hehn, T., Kooij, J., & Hamprecht, F. (2020). End-to-end learning of decision trees and forests. *International Journal of Computer Vision, 128*, 997–1011.

Holmes, G., Kirkby, R., & Pfahringer, B. (2005). Stress-testing hoeffding trees. In *Knowledge discovery in databases: PKDD* (pp. 495–502). Springer.

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the Seventh ACM SIGKDD international conference on knowledge discovery and data mining, KDD* (pp. 97–106). Association for Computing Machinery.

İrsoy, O., Yıldız, O. T., & Alpaydın, E. (2012). Soft decision trees. In *Proceedings of the 21st international conference on pattern recognition (ICPR)* (pp. 1819–1822).

Jordan, M. I., & Jacobs, R.A. (1993). Hierarchical mixtures of experts and the EM algorithm. In *Proceedings of 1993 international conference on neural networks (IJCNN)* (Vol. 2, pp. 1339–13442).

Kingma, D., & Ba, J. (2015). ADAM: A method for stochastic optimization. In *International conference on learning representations (ICLR)*.

Köbschall, K., Hartung, L., & Kramer, S. (2025). Soft hoeffding tree: A transparent and differentiable model on data streams. In *Discovery science* (pp. 167–182). Springer.

Kontschieder, P., Fiterau, M., Criminisi, A., & Bulò, S. R. (2016). Deep neural decision forests. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence* (pp. 4190–4194). AAAI Press.

Li, Z., & Leeuwen, M. (2023). Explainable contextual anomaly detection using quantile regression forests. *Data Mining and Knowledge Discovery, 37*, 1–47.

Lipton, Z. (2016). The mythos of model interpretability. Communications of the ACM. 61.

Manapragada, C., Webb, G. I., & Salehi, M. (2018). Extremely fast decision tree. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1953–1962). Association for Computing Machinery.

Mignone, P., Corizzo, R., & Ceci, M. (2024). Distributed and explainable GHSOM for anomaly detection in sensor networks. *Machine Learning, 113*, 1–42.

Olson, R., La Cava, W., Orzechowski, P., Urbanowicz, R., & Moore, J. (2017). PMLB: A large benchmark suite for machine learning evaluation and comparison. *BioData Mining, 10*(1), 36.

Oza, N. C. (2001). Online bagging and boosting. In *Proceedings of the eighth international workshop on artificial intelligence and statistics. PMLR, R3* (pp. 229–236).

Panigutti, C., Hamon, R., Hupont, I., Fernandez Llorca, D., Fano Yela, D., Junklewitz, H., Scalzo, S., Mazzini, G., Sanchez, I., Soler Garrido, J., & Gomez, E. (2023). The role of explainable ai in the context of the ai act. In *Proceedings of the 2023 ACM conference on fairness, accountability, and transparency. FAccT '23* (pp. 1139–1150). Association for Computing Machinery.

Parliament, E., European Union, C. (2024). Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence. Official Journal of the European Union.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). *PyTorch: An imperative style, high-performance deep learning library*. Curran Associates Inc.

Rutkowski, L., Pietruczuk, L., Duda, P., & Jaworski, M. (2013). Decision trees for mining data streams based on the Mcdiarmid's bound. *IEEE Transactions on Knowledge and Data Engineering, 25*(6), 1272–1279.

Swamy, V., Montariol, S., Blackwell, J., Frej, J., Jaggi, M., & Käser, T. (2025). Intrinsic user-centric interpretability through global mixture of experts.

Verma, N., Bifet, A., Pfahringer, B., & Bahri, M. (2025). ASML-REG: Automated machine learning for data stream regression. In *Advances in knowledge discovery and data mining - 29th Pacific-Asia conference on knowledge discovery and data mining, proceedings, Part IV* (pp. 440–447).

Xu, L., Skoularidou, M., Cuesta-Infante, A., & Veeramachaneni, K. (2019). *Modeling tabular data using conditional GAN*. Curran Associates Inc.