SPECTRA: SPARSE PROTOTYPE-ENCODED CLASSIFICATION TREES WITH ROUTING ATTRIBUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Differentiable decision trees combine the structure of trees with joint optimization, while forming predictions from multiple paths. Interpreting these models requires connecting a displayed path to the computation that selects it and the prediction it represents. We introduce SPECTRA, a sparse prototype-encoded classification tree with routing attribution. Complement-coded prototypes express branch preferences through additive feature evidence, and learned structural gates shape a compact tree. SPECTRA derives an executable dominant-path explanation from the model's routing computation: a complete-path score combines prototype evidence, gate offsets, and routing normalizers to identify the path carrying the greatest native flow. Comparing competing paths then reveals how feature evidence and structural preferences determine their relative flow, including the effect of normalization. To connect this explanation to the full soft prediction, SPECTRA supports reconstruction with a chosen number of paths and calibrated single-path display, making explanation size and predictive agreement jointly measurable. Experiments across 69 OpenML-CC18 tasks verify exact dominant-path extraction, characterize reconstruction as the path budget grows, and provide functional evidence through feature interventions and retraining on a fixed subset. SPECTRA thus links compact tree structure, explicit routing decisions, and feature evidence within a common account of how a soft tree computes and how its paths can explain its behavior.

1 INTRODUCTION

Decision trees make predictions inspectable through a sequence of branch decisions. Differentiable trees extend this structure with continuous routing and joint optimization of branch parameters and leaf values (Kontschieder et al., 2015; Frosst & Hinton, 2017; Popov et al., 2020; Hazimeh et al., 2020). Their predictions aggregate contributions from multiple leaves. Turning that computation into a displayed path therefore requires a choice of explanatory target: which component does the path identify, what determines its selection, and how much of the full prediction does it represent?

Structural sparsity makes this question concrete. Learned gates can suppress branches whose routing scores remain high. A path selected from routing scores alone can then differ from the path carrying the most mass in the fitted model. Even after accounting for gates, choosing the largest branch locally need not recover the largest complete path, because downstream routing changes the accumulated flow. Tree-shaped parameters and a small topology leave these computational distinctions unresolved.

We develop SPECTRA (Sparse Prototype-Encoded Classification Trees with Routing Attribution) around the native flow of a soft tree. Complement-coded prototypes yield smooth, additive feature scores; learned routing scales determine branch competition; and continuous gates select the active topology. The key observation is that each branch's log flow equals its scaled prototype score plus a log-gate offset minus a routing normalizer. Summing these quantities over a complete path gives its native log flow. This defines an executable dominant-path program directly from the predictor and exposes the terms responsible for its selection.

A second question concerns representation. The dominant path identifies one component of the soft computation, while classification uses the weighted contributions of all leaves. SPECTRA makes this relationship measurable through flow concentration, reconstruction with the top-k paths, and independently calibrated single-path display. These operations connect the size of an explanation

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.

to its agreement with the soft prediction. Feature-margin decompositions provide a complementary view of which inputs distinguish competing branches and paths.

Our evaluation follows this computational chain. We first check path extraction on 69 OpenML-CC18 tasks and 199 fitted trees. We then measure reconstruction as the path budget grows and evaluate calibration on disjoint data. On a fixed 12-task subset, local interventions, forward simulation, and remove-and-retrain test the information carried by feature evidence. Predictive comparisons, matched leaf budgets, mechanism ablations, and CPU measurements establish the accuracy and cost of the resulting trees.

The contributions are:

- A sparse prototype tree whose native computation supports a gate-aware dominant-path program and additive routing attribution.
- An exact path-flow identity that separates feature evidence, structural gates, and routing normalizers, together with path-budget reconstruction and calibrated single-path display.
- An evaluation linking executable path agreement, soft-prediction agreement, feature functionality, and the predictive and computational cost of compact trees.

2 Related Work

Differentiable trees and structured prediction. Neural decision forests and soft decision trees jointly learn probabilistic routing and leaf predictions (Kontschieder et al., 2015; Frosst & Hinton, 2017). NODE uses ensembles of oblivious trees (Popov et al., 2020); Tree Ensemble Layers support sparse conditional computation (Hazimeh et al., 2020). ProtoTree uses image-part prototypes and compares maximum-probability, greedy, and soft prediction (Nauta et al., 2021). GradTree learns axis-aligned trees by gradient descent (Marton et al., 2024). SPECTRA combines tabular prototypes, structural gates, and a feature-level decomposition of complete-path flow.

Rules, bounded regions, and fidelity. Rule extraction approximates a predictor with a decision program (Craven & Shavlik, 1996); local explanations approximate behavior around an input (Ribeiro et al., 2016). Complement coding provides paired coordinates for bounded prototype responses (Carpenter et al., 1991). SPECTRA uses a comparative path score: prototype intervals visualize parameters, while the gate-aware program determines the dominant path.

Feature evidence and tabular evaluation. Additive attribution methods assign feature contributions to a specified output (Lundberg & Lee, 2017); EBM exposes additive terms in its predictor (Nori et al., 2019). Remove-and-retrain tests feature rankings through their predictive consequences (Hooker et al., 2019), and simulation tests whether selected information helps reconstruct model behavior (Chen et al., 2022). We apply these tests to routing-margin evidence. Tabular benchmarks motivate explicit split, tuning, and aggregation protocols (Gorishniy et al., 2021; Grinsztajn et al., 2022; McElfresh et al., 2023). Our main roster uses OpenML-CC18 tasks (Bischl et al., 2021), with fold-aligned TabZilla references and separate matched-capacity studies.

3 Sparse Prototype Trees

Let $x \in [0,1]^D$ denote a preprocessed input and C the number of classes. Preprocessing is fitted on training data. We use L for the number of tree levels, including leaves, so each complete path contains $L-1$ routing decisions. Each internal node has at most M candidate children. Figure 1 shows the predictor and its explanation readouts.

3.1 Prototype evidence and soft routing

Complement coding pairs each coordinate with its complement, $x' = [x, 1-x] \in [0,1]^{2D}$ (Carpenter et al., 1991). Branch k at node n has prototype $w_{n,k} \in (0,1)^{2D}$. Its smooth intersection and

normalized score are

$$q_{n,k,r}(x) = -\delta \log\left(e^{-x'_r/\delta} + e^{-w_{n,k,r}/\delta}\right), \tag{1}$$

$$T_{n,k}(x) = \frac{\sum_{r=1}^{2D} q_{n,k,r}(x)}{\epsilon + \sum_{r=1}^{2D} w_{n,k,r}}. \tag{2}$$

The temperature $\delta > 0$ smooths the coordinatewise minimum. The denominator makes the score comparable across prototypes, and $\epsilon = 10^{-3}$ stabilizes small prototype norms. The paired coordinates let a prototype respond to a bounded part of a feature's domain: $w_{n,k,j}$ controls the upper-side match and $w_{n,k,D+j}$ the complementary match. Their two contributions can be regrouped into one term per original feature, as used below for attribution.

At node n, a learned scale $a_n \geq 1$ turns prototype scores into soft routing probabilities:

$$\mu_{n,k}(x) = \frac{\exp(a_n T_{n,k}(x))}{C_n(x)}, \qquad C_n(x) = \sum_m \exp(a_n T_{n,m}(x)). \tag{3}$$

Increasing a_n concentrates routing on higher-scoring candidates while preserving their prototype-score order. The normalizer C_n is shared among branches at this node, but differs along paths through different nodes.

3.2 Structural gates and native prediction

A learned gate $z_{n,k} \in [0, 1]$ multiplies each branch probability. Gates use a clipped sigmoid relaxation with an expected L_0 penalty (Louizos et al., 2018); Appendix A gives the transform and optimization details. Starting with unit root mass, the tree propagates

$$p_{\text{child}(n,k)}(x) = p_n(x)\mu_{n,k}(x)z_{n,k}, \qquad B_\pi(x) = \prod_{(n,k)\in\pi} \mu_{n,k}(x)z_{n,k}. \tag{4}$$

Here B_π is the native flow of a complete root-to-leaf path π. Each leaf has a class-logit vector $V_\pi \in \mathbb{R}^C$. The native soft predictor is

$$f_{\text{soft}}(x) = \sum_\pi B_\pi(x)V_\pi, \qquad p_\theta(y \mid x) = \text{softmax}(f_{\text{soft}}(x))_y. \tag{5}$$

Training jointly optimizes prototypes, routing scales, gates, and leaf values using cross-entropy plus $\lambda_0\mathcal{R}_0$. A connectivity update preserves a viable child at each internal node. Active leaves are those reachable through positive gates. Gates suppress flow rather than renormalize it: surviving leaf flows need not sum to one, and the native logits retain their total mass in Eq. (5). Thus the displayed topology, its leaf flows, and the soft prediction arise from one fitted computation.

4 Executable Paths and Routing Attribution

4.1 An exact dominant-path program

The effective score of a branch combines its scaled prototype evidence and gate:

$$A_{n,k}(x) = a_n T_{n,k}(x) + \log z_{n,k}. \tag{6}$$

This score orders branches at the same node because their routing normalizer is shared. For a complete root-to-leaf path π, accumulate the terms encountered at every visited node:

$$S_\pi(x) = \sum_{(n,k)\in\pi} \big(A_{n,k}(x) - \log C_n(x)\big), \qquad \pi^*(x) = \arg\overset{\text{det}}{\max}_\pi S_\pi(x). \tag{7}$$

We set $\log 0 = -\infty$ and resolve ties by the smallest branch or leaf index. Since $\mu_{n,k}z_{n,k} = \exp(A_{n,k})/C_n$, the product of branch flows along π gives $\exp(S_\pi) = B_\pi$. This identity gives the program its computational target.

Proposition 1 (Executable dominant path)**.** *For every input with positive surviving flow, $S_\pi(x) = \log B_\pi(x)$. Consequently, π^* is exactly the native maximum-flow path. At each node, maximizing $A_{n,k}$ likewise maximizes the local branch flow $\mu_{n,k}z_{n,k}$.*

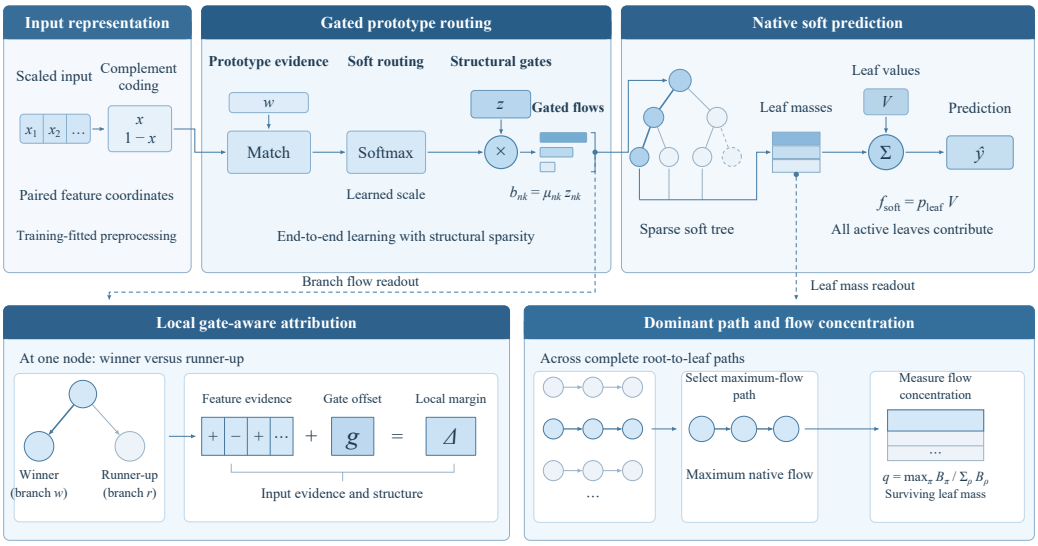


Figure 1: SPECTRA computation and explanation. Complement-coded inputs produce prototype scores, scaled soft routing probabilities, and gated branch flows. All active leaves contribute to the native soft prediction. The explanation reads the same computation: local margins separate feature evidence from gate offsets, and complete-path flow determines the dominant path and its concentration. Paths and mass bars are schematic.

The dominant path can be recovered by a bottom-up maximization. Set $H(n) = 0$ at every leaf; at an internal node, compute

$$H(n) = \max_k \left\{ A_{n,k}(x) - \log C_n(x) + H\big(\text{child}(n,k)\big) \right\}. \tag{8}$$

Recording the maximizing child at each node and following those choices from the root yields π^*. This recurrence accounts for every downstream decision before choosing a branch, using the same tie rule as Eq. (7). The detailed path-flow proof is in Appendix A.4. Complete paths include their downstream normalizers, so their ordering can differ from repeated local maximization: a branch with the largest current flow can lead to leaves with smaller complete-path flow than another branch. The score-only control uses the same fitted parameters and removes just $\log z_{n,k}$ from Eq. (6); the learned scales and normalizers remain. It isolates how structural gates change path selection.

A fitted six-leaf diabetes example illustrates the distinction (Figure 2). The largest routing probability, 0.303, points to L2, whose gate is closed. L1 instead has the largest surviving flow, 0.232, and becomes the dominant path. Its share of surviving mass is 49.5%; L3 and L5 carry the remainder. The path program therefore displays L1, while the native predictor combines all three active leaves. Their different predictions in this record make clear what each readout answers: the path identifies the largest component, and the soft prediction summarizes the complete fitted tree. Appendix C.4 gives the selection protocol and numerical replay.

4.2 Feature Evidence, Gates, and Normalization

Pair the two coded coordinates of feature j:

$$\phi_{n,k,j}(x) = \frac{q_{n,k,j}(x) + q_{n,k,D+j}(x)}{\epsilon + \sum_r w_{n,k,r}}, \qquad T_{n,k}(x) = \sum_{j=1}^{D} \phi_{n,k,j}(x). \tag{9}$$

For the local winner u and runner-up v under $A_{n,k}$, the margin decomposes as

$$\Delta_n = A_{n,u} - A_{n,v} = \sum_j \underbrace{a_n(\phi_{n,u,j} - \phi_{n,v,j})}_{m_{n,j}} + \underbrace{\log z_{n,u} - \log z_{n,v}}_{g_n}. \tag{10}$$

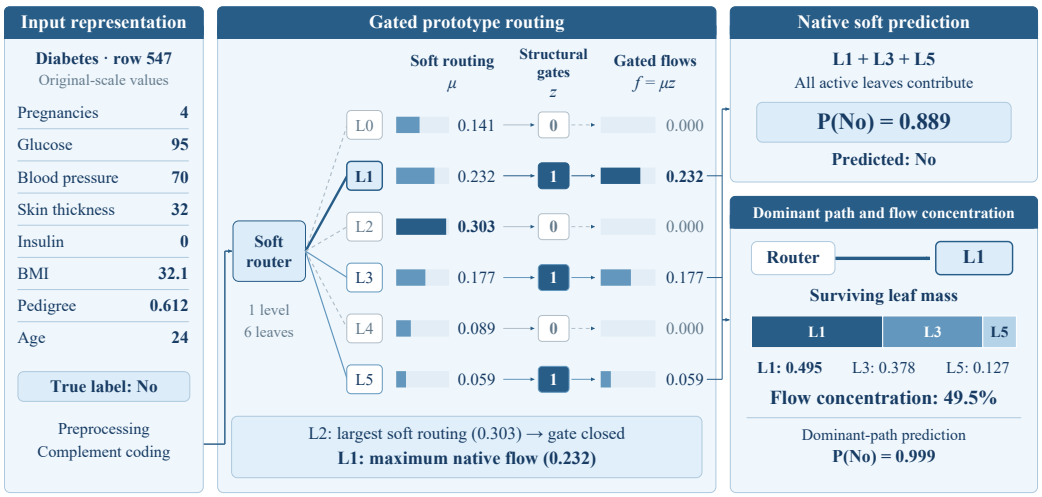


Figure 2: A gate-induced route change for one diabetes test record. L2 has the largest ungated routing probability but zero gated flow; L1 has the largest surviving flow and accounts for 49.5% of it. The native soft prediction combines L1, L3, and L5. This one-level example isolates the effect of gates; Eq. (7) scores complete paths at every tree depth.

A positive $m_{n,j}$ favors the local winner; g_n measures structural preference. This local comparison explains branch competition at a visited node. The dominant path itself is explained by comparing complete paths.

For the largest and second-largest positive-flow paths π^* and ρ^*, collect the terms of each complete-path score as

$$\begin{aligned} F_{\pi,j}(x) &= \sum_{(n,k)\in\pi} a_n \phi_{n,k,j}(x), \\ G_\pi &= \sum_{(n,k)\in\pi} \log z_{n,k}, \qquad N_\pi(x) = -\sum_{(n,k)\in\pi} \log C_n(x), \\ S_\pi(x) &= \sum_j F_{\pi,j}(x) + G_\pi + N_\pi(x). \end{aligned} \tag{11}$$

Subtracting the runner-up score from the dominant score yields

$$\log \frac{B_{\pi^*}}{B_{\rho^*}} = \sum_j (F_{\pi^*,j} - F_{\rho^*,j}) + (G_{\pi^*} - G_{\rho^*}) + (N_{\pi^*} - N_{\rho^*}). \tag{12}$$

The three terms expose feature evidence, gate offsets, and input-dependent normalization. Normalization cancels for two branches at one node; paths through different nodes retain it. The signed feature difference indicates which path a feature favors, while its magnitude supplies a ranking for functional tests. Gates can change the chosen path without changing the input-dependent feature terms; keeping these sources of evidence separate is therefore necessary to interpret a route change.

Feature tests use two specified rankings. Local interventions rank the absolute path feature terms in Eq. (12). Remove-and-retrain ranks baseline-relative prototype-margin evidence: the winner and runner-up are selected by $T_{n,k}$, and their paired feature contributions are centered at training medians. The latter tests the information encoded by the prototypes. Appendix A.4 gives baseline-relative evidence and zero-flow conventions, and Appendix D connects each ranking to its experimental setting.

4.3 From one path to the soft prediction

Flow concentration is the dominant path's share of surviving mass:

$$q(x) = \frac{B_{\pi^*}(x)}{\sum_{\pi} B_{\pi}(x)}. \tag{13}$$

It quantifies how much of the active tree's flow the displayed path carries. For a budget of k paths, let $\mathcal{P}_k$ contain the top-k paths by native flow and reconstruct logits as

$$f_k(x) = \frac{\sum_{\pi} B_{\pi}(x)}{\sum_{\pi \in \mathcal{P}_k} B_{\pi}(x)} \sum_{\pi \in \mathcal{P}_k} B_{\pi}(x) V_{\pi}. \tag{14}$$

The prefactor preserves total surviving mass while allocating it to the selected paths. At $k = 1$, the dominant leaf supplies the class-logit vector scaled by total surviving mass; at the full leaf count, $f_k = f_{\text{soft}}$. We measure soft-prediction agreement between their predicted classes and Jensen–Shannon (JS) divergence between their class probabilities. Concentration supplies a ranking score for single-path display; a threshold calibrated on disjoint examples determines when to display that path or abstain (Geifman & El-Yaniv, 2017).

The distinction between flow and class evidence matters here. A path can carry the largest share of mass while the remaining leaves collectively favor another class. Consequently, q measures the dominance of one route, whereas agreement between f_1 and f_{soft} measures its adequacy as a class explanation. The top-k construction retains the same leaf values and routing flows; changing k changes only how many components of the fitted prediction are displayed. Calibration addresses the separate decision of when a single displayed path is sufficiently representative for a given input.

5 Experiments

5.1 Evaluation design

We use a fixed 69-task OpenML-CC18 roster and official repeat-0/fold-0 splits throughout the predictive and path studies (Bischl et al., 2021). The predictive benchmark selects SPECTRA profiles by inner-validation accuracy and averages final class probabilities across three seeds, or two on eight large tasks. Each constituent fit is a single tree. The path studies evaluate these trees individually, with 199 fits per study; their metrics first average seeds within task and then tasks equally. Table 2 therefore reports the probability-averaged prediction protocol, while path agreement refers to one fitted tree. Appendix B specifies preprocessing, profile selection, dataset exclusions, and comparator coverage.

Feature, matched-capacity, and efficiency studies share a fixed 12-task numeric subset selected by dataset metadata. Local feature interventions use the main benchmark's selected profiles; remove-and-retrain uses an eight-leaf ceiling. Capacity and efficiency studies also evaluate individual trees, averaging metrics across seeds. Accuracy evaluates the 69-task benchmark and ablations; balanced accuracy (BAcc) evaluates the 12-task predictive comparisons. Unless specified otherwise, 95% confidence intervals resample tasks after seed averaging. Appendix B.1 maps each experiment to its fitting and aggregation protocol.

SPECTRA's predictive search varies tree levels, candidate branches, smooth temperature, sparsity strength, class weighting, and scaling. Six profiles are evaluated on the 80/20 inner split, with four profiles on large tasks; the validation-selected training horizon is reused for the final fits. The structural comparisons instead fix 500 training epochs and the leaf ceiling. CART uses a maximum-leaf constraint, GradTree a single tree at matching depth, and MurTree bounded-time search over training-derived binary conditions (Breiman et al., 1984; Marton et al., 2024; Demirovic et al., 2022). This separation lets the predictive benchmark select operating points while the capacity study measures realized structure under a common ceiling.

5.2 Path agreement and explanation budget

The gate-aware program selects the native dominant path on every audited example across 69 tasks and 199 fitted trees. The score-only control has task-macro path agreement 0.9956, with route changes

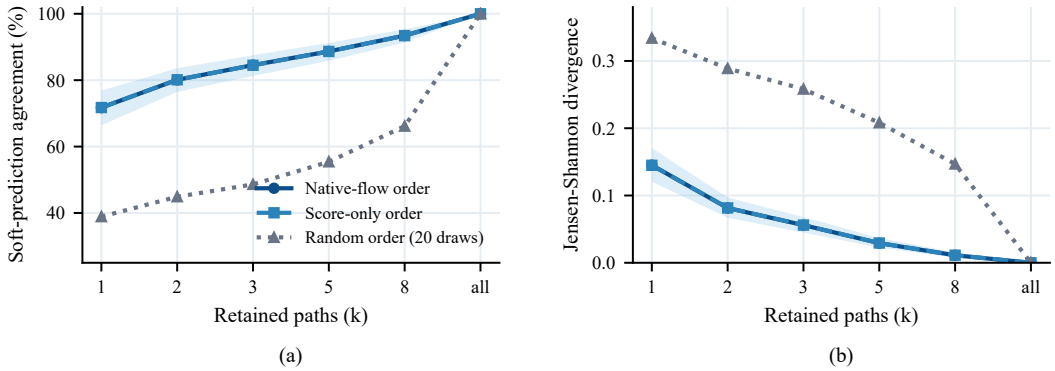


Figure 3: Path budget and soft-prediction reconstruction on 69 CC18 tasks (199 fitted trees). (a) Soft-prediction agreement. (b) JS divergence between reconstructed and native class probabilities. Native-flow, score-only, and 20-draw random rankings use the same fits. Points average seeds within tasks and tasks equally; the pale band is a 95% task-bootstrap interval for native ranking. Budgets are capped at the available leaf count; "all" checks exact reconstruction.

on six tasks. The largest effect is on `diabetes`, where removing gate offsets changes 28.57% of selected paths. Native-logit replay and margin reconstruction have task-macro maximum absolute errors of 4.14×10^{-6} and 1.70×10^{-6}, respectively. Appendix C gives numerical diagnostics and task-level differences; Appendix C.4 traces a gate-induced path change through all six candidate leaves.

Figure 3 shows how the path budget controls soft-prediction agreement in a separate 199-tree refit. Agreement increases from 0.7174 at one path to 0.8454 at three and approximately 0.934 at eight, while JS divergence falls from 0.145 to 0.056 and 0.011. Three native-ranked paths exceed random selection by 35.9 percentage points (95% CI $[30.9, 40.9]$). Score-only and native ordering have similar aggregate reconstruction curves, consistent with gate-induced route changes being concentrated in six tasks. Full-path reconstruction reproduces the native output.

The retained share of surviving flow increases from 0.239 with one path to 0.519 with three and 0.836 with eight (Appendix C). The agreement curve therefore has a concrete interpretation: a short explanation captures a limited part of the weighted logit sum, and additional paths recover more of that same computation. The random-order comparison tests whether native-flow ranking identifies useful components at a fixed explanation size; it does not change the predictor or retrain its leaves.

Calibrated single-path display. A separate experiment splits each outer training partition into 70% fitting, 15% profile selection, and 15% calibration. The selected profile is refit on the first 85%. For 21 thresholds fixed from refit inputs, a Bonferroni-corrected one-sided Clopper–Pearson bound (Clopper & Pearson, 1934) selects the highest-coverage threshold meeting the calibration error target. The threshold is frozen before test evaluation. At a 5% target, concentration yields 98.8% conditional soft-prediction agreement, 20.5% task-macro coverage, and nonempty selections on 28/69 tasks. At 1%, agreement is 99.8% with 5.6% coverage on 7/69 tasks. Conditional agreement averages nonempty seeds within nonempty tasks; coverage includes all tasks. Appendix C.3 reports uncertainty, alternative scores, and the exchangeability assumption for the calibration bound.

5.3 Functional value of feature evidence

We test feature evidence through two complementary operations on the fixed 12-task subset: intervention in a fitted predictor and removal followed by retraining. The first measures changes in model output; the second measures how well a refitted model recovers after losing the ranked information.

These operations also distinguish the two attribution targets. The path-margin ranking asks which input features distinguish the dominant complete path from its closest positive-flow competitor. The prototype-margin ranking asks which original features distinguish a fixed pair of prototype responses relative to the training-median input. Both are derived from the fitted tree, but the local

Table 1: Functional feature evidence on the fixed 12-task subset. Differences are SPECTRA's specified ranking minus its random-ranking control, in percentage points, with paired 95% task-bootstrap intervals. Local interventions and simulation use selected profiles; retraining uses fixed eight-leaf trees.

Feature evidence	Evaluation	Difference	95% CI
Path margin	30% retention: class agreement	$+3.16$	$[-0.27, 6.85]$
Path margin	30% removal: probability drop	$+5.04$	$[1.17, 10.30]$
Path margin	Top-3 simulator agreement	$+10.12$	$[5.04, 15.93]$
Prototype margin	10% removal and refit: BAcc drop	$+1.47$	$[0.22, 2.92]$
Prototype margin	30% removal and refit: BAcc drop	$+2.79$	$[0.80, 5.33]$
Prototype margin	50% removal and refit: BAcc drop	$+1.85$	$[-0.01, 3.86]$

path comparison includes gate and normalization terms in its complete score, whereas the retraining study ranks the feature-dependent prototype terms. This protocol distinction is carried through the summaries below.

With the main benchmark profiles, replacing the top 30% of features ranked by the path margin reduces the native predicted-class probability by 9.29 percentage points, compared with 4.25 for random ranking. The paired difference is 5.04 points (95% CI $[1.17, 10.30]$). A fixed logistic simulator (Chen et al., 2022) given the top three feature values and their indices predicts SPECTRA's class with 92.96% agreement, compared with 82.84% from three random features (difference 10.12 points, CI $[5.04, 15.93]$). These results connect path evidence to information used by the soft predictor. Retaining the top 30% yields 86.7% soft-prediction agreement versus 83.5% for random retention; their paired difference has interval $[-0.27, 6.85]$. Thus removal resolves a difference at this feature budget, while retention gives a less precise contrast.

Under an eight-leaf ceiling, removing the top 30% of features ranked by SPECTRA's baseline-relative prototype margin and retraining reduces BAcc by 3.49 points, 2.79 points more than random removal (CI $[0.80, 5.33]$). The difference from training-permutation ranking is 0.39 points (CI $[-1.29, 2.37]$). Prototype-margin evidence places 90% of its absolute mass on 6.94 features on average, compared with 2.26 for CART and 13.86 for EBM. At Gaussian noise standard deviation 0.05 in scaled feature space, SPECTRA retains 94.90% prediction agreement and 56.56% Top-3 feature Jaccard. Appendix D brings together intervention, simulation, retraining, and stability results with their respective feature rankings.

Table 1 separates the information tested by the two rankings. At small removal budgets, the prototype-margin order identifies information that remains consequential after refitting; the 50% contrast is less precise. The selected-profile stability study adds a path-level view: at 5% input noise, soft predictions agree on 93.7% of examples and dominant paths on 89.6%; Top-3 path-feature Jaccard is 77.8% among examples with unchanged predicted class. Keeping prediction, path, and feature stability separate reveals which part of an explanation changes under perturbation.

5.4 Predictive performance and compact structure

SPECTRA's probability-averaged benchmark reaches task-macro accuracy 0.8300. Table 2 compares shared tasks with locally fitted EBM, GradTree, and MurTree and public TabZilla fold-0 references (McElfresh et al., 2023). Against single-tree comparator families, its paired accuracy differences are $+1.62$, $+7.16$, and $+10.12$ percentage points for DecisionTree, MurTree, and GradTree. EBM and the ensemble references have higher average accuracy. The public references retain their original tuning pipelines; local comparator configurations and failures are documented in Appendix B.

The matched-capacity experiment evaluates individual trees at ceilings of 4, 8, and 16 leaves on the 12-task subset. At four leaves, SPECTRA reaches 71.11% BAcc with 2.94 active leaves. Its paired advantages over GradTree and MurTree are 4.69 and 3.83 points, with intervals excluding zero; the 2.19-point difference from CART has interval $[-0.76, 5.18]$. At eight and 16 leaves, CART reaches 73.16% and 73.36% BAcc, while SPECTRA reaches 70.46% and 70.10% using 4.53 and 7.42 active leaves. The capacity curves in Appendix F.1 show the operating points available under each leaf budget.

Table 2: Predictive accuracy on the 69-task CC18 roster. SPECTRA averages final class probabilities over seeds. Means and differences use shared successful tasks; W/T/L counts SPECTRA outcomes. Local comparators use our split-aligned pipeline; public rows use TabZilla fold-0 results.

Comparator	Source	Tasks	Comparator	SPECTRA	Δ	W/T/L
GradTree	Local	68/69	0.7340	0.8353	+0.1012	47/5/16
MurTree	Local	64/69	0.7705	0.8421	+0.0716	45/3/16
DecisionTree	Public	68/69	0.8191	0.8353	+0.0162	35/4/29
EBM	Local	68/69	0.8587	0.8353	−0.0235	16/5/47
NODE	Public	58/69	0.8624	0.8397	−0.0227	14/4/40
Random Forest	Public	68/69	0.8515	0.8353	−0.0162	24/5/39
XGBoost	Public	68/69	0.8745	0.8353	−0.0392	15/3/50

5.5 Mechanisms and computational cost

A fixed sparse configuration isolates three mechanisms on all 69 tasks, with three seeds and 828 fits (Appendix E.1). Fixing the routing scale at its initialization lowers accuracy from 0.8317 to 0.8067, a paired change of -2.50 points (CI $[-3.67, -1.54]$). Removing the L_0 penalty increases active branches from 4.80 to 6.00 and changes accuracy by $+0.24$ points (CI $[-0.05, 0.53]$). Removing complement coding changes accuracy by -0.37 points (CI $[-0.95, 0.20]$). The scale has the clearest predictive effect, while the L_0 penalty reduces realized topology. Parameter sweeps further connect branch budget to representation: increasing M from 3 to 12 reduces mean one-path soft-prediction agreement from 92.28% to 83.06% across three tasks, alongside accuracy of 90.34% and 90.88% (Appendix E.2).

Together, these results separate three roles of the design. The routing scale changes how decisively a node distributes mass among candidate children. The L_0 term controls how many branches remain active after fitting. Complement coding supplies bounded prototype responses and their paired feature decomposition. The branch-budget sweep shows why predictive accuracy alone cannot characterize the explanation: a configuration can retain similar accuracy while distributing flow over more paths, so its dominant path represents less of the soft output.

The CPU study refits all four single-tree methods on one core under a four-leaf ceiling (Appendix F.2). SPECTRA reaches 70.30% BAcc with mean training time 6.072 s and batch inference time 4.90 ms per 1,000 examples. CART reaches 68.92% with 0.011 s and 1.07 ms; GradTree and MurTree reach 66.42% and 67.27%. SPECTRA's soft routing evaluates candidate branches, while CART follows hard predicates. Reducing SPECTRA training from 500 to 200 epochs lowers mean time from 6.11 to 3.28 s and changes BAcc by -0.55 points (CI $[-1.40, 0.11]$), with active leaves increasing from 2.97 to 3.47. Later optimization contributes to both fitting and pruning.

6 Conclusion

SPECTRA connects sparse prototype trees to an executable dominant-path explanation. The path-flow identity determines which route is displayed, and margin decompositions expose its feature evidence, structural preference, and normalization. Path-budget reconstruction and disjoint calibration make the relation to the native soft prediction operational. Across the CC18 studies, feature interventions and retraining establish functional evidence, while capacity, ablation, and efficiency measurements characterize the cost of compact soft trees. Together, these results provide a unified account of what a displayed path computes, how its evidence can be tested, and how much of the prediction it represents.

Reproducibility Statement

Appendix A specifies optimization, path extraction, and the supporting identities. Appendix B records task selection, preprocessing, model selection, aggregation, and task-level predictive results. Appendices C–F give experimental protocols, uncertainty estimates, and complete result tables. The accompanying experiment files contain task rosters, seeds, raw results, aggregation scripts, and plotting sources.

AI USE STATEMENT

Generative AI assistants helped organize and revise manuscript text and LaTeX, interpret supplied experimental results, and check mathematical exposition and consistency across sections. The authors are responsible for the final mathematics, implementation, experiments, references, and conclusions.

REFERENCES

Bernd Bischl, Giuseppe Casalicchio, Matthias Feurer, Pieter Gijsbers, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. OpenML benchmarking suites. In *Advances in Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/hash/c7e1249ffc03eb9ded908c236bd1996d-Abstract-round2.html.

Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. doi:10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, 1984. URL https://www.routledge.com/Classification-and-Regression-Trees/Breiman-Friedman-Olshen-Stone/p/book/9781315139470.

Gail A. Carpenter, Stephen Grossberg, and David B. Rosen. Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, 4(6):759–771, 1991. doi:10.1016/0893-6080(91)90056-B.

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*, pp. 785–794, 2016. doi:10.1145/2939672.2939785.

Valerie Chen, Nari Johnson, Nicholay Topin, Gregory Plumb, and Ameet Talwalkar. Use-case-grounded simulations for explanation evaluation. In *Advances in Neural Information Processing Systems*, volume 35, 2022. doi:10.52202/068431-0129. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/0b9536e186a77feff516893a5f393f7a-Abstract-Conference.html.

C. J. Clopper and E. S. Pearson. The use of confidence or fiducial limits illustrated in the case of the binomial. *Biometrika*, 26(4):404–413, 1934. doi:10.1093/biomet/26.4.404. URL https://doi.org/10.1093/biomet/26.4.404.

Mark Craven and Jude W. Shavlik. Extracting tree-structured representations of trained networks. In *Advances in Neural Information Processing Systems*, volume 8, 1996. URL https://proceedings.neurips.cc/paper/1995/hash/45f31d16b1058d586fc3be7207b58053-Abstract.html.

Sanjoy Dasgupta, Nave Frost, and Michal Moshkovitz. Framework for evaluating faithfulness of local explanations. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4794–4815. PMLR, 2022. URL https://proceedings.mlr.press/v162/dasgupta22a.html.

Emir Demirovic, 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. *Journal of Machine Learning Research*, 23(26):1–47, 2022. URL https://jmlr.org/papers/v23/20-520.html.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4443–4458, 2020. doi:10.18653/v1/2020.acl-main.408. URL https://aclanthology.org/2020.acl-main.408/.

Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017. URL https://arxiv.org/abs/1711.09784.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.neurips.cc/paper_files/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/9d86d83f925f2149e9edb0ac3b49229c-Abstract.html.

Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/0378c7692da36807bdec87ab043cdadc-Abstract-Datasets_and_Benchmarks.html.

Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. The tree ensemble layer: Differentiability meets conditional computation. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *PMLR*, pp. 4138–4148, 2020. URL https://proceedings.mlr.press/v119/hazimeh20a.html.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/fe4b8556000d0f0cae99daa5c5c5a410-Abstract.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1412.6980.

Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *ICCV*, 2015. URL https://openaccess.thecvf.com/content_iccv_2015/html/Kontschieder_Deep_Neural_Decision_ICCV_2015_paper.html.

Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1Y8hhg0b.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

Sascha Marton, Stefan Luedtke, Christian Bartelt, and Heiner Stuckenschmidt. GradTree: Learning axis-aligned decision trees with gradient descent. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 14323–14331, 2024. doi:10.1609/aaai.v38i13.29345. URL https://ojs.aaai.org/index.php/AAAI/article/view/29345.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/f06d5ebd4ff40b40dd97e30cee632123-Abstract.html.

Meike Nauta, Ron van Bree, and Christin Seifert. Neural prototype trees for interpretable fine-grained image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. doi:10.1109/CVPR46437.2021.01469. URL https://openaccess.thecvf.com/content/CVPR2021/html/Nauta_Neural_Prototype_Trees_for_Interpretable_Fine-Grained_Image_Recognition_CVPR_2021_paper.html.

Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. InterpretML: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019. URL https://arxiv.org/abs/1909.09223.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=r1eiu2VtwH.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016. URL https://arxiv.org/abs/1602.04938.

Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Sai Suggala, David I. Inouye, and Pradeep Ravikumar. On the (in)fidelity and sensitivity of explanations. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/a7471fdc77b3435276507cc8f2dc2569-Abstract.html.

SUPPLEMENTARY MATERIAL

A MODEL SPECIFICATION AND EXECUTABLE ROUTING

A.1 GATES AND OPTIMIZATION

For learned gate parameters $\alpha_{n,k}$, the deterministic transform is

$$z_{n,k} = \mathrm{clip}_{[0,1]}[(\zeta - \gamma)\sigma(\alpha_{n,k}/\beta) + \gamma], \qquad \mathcal{R}_0 = \sum_{n,k} \sigma\left(\alpha_{n,k} - \beta \log \frac{-\gamma}{\zeta}\right). \tag{15}$$

We use $\beta = 0.66$, $\gamma = -0.1$, and $\zeta = 1.1$. The deterministic gate and expected L_0 surrogate adapt the sparsity relaxation of Louizos et al. (2018); the forward pass here does not sample gate noise. The deterministic forward pass does not sample gate noise. Prototype coordinates are projected to $[10^{-4}, 1 - 10^{-4}]$, and routing scales are initialized at 3 and constrained to $a_n \geq 1$. Thus the routing scale changes branch concentration without reversing the ordering of prototype scores. The experiments use full-batch updates; the loss below is written for a general batch.

For a mini-batch $\mathcal{B}$, training minimizes

$$\mathcal{L}(\theta) = -\frac{1}{\sum_{(x_i,y_i)\in\mathcal{B}} \omega_{y_i}} \sum_{(x_i,y_i)\in\mathcal{B}} \omega_{y_i} \log p_\theta(y_i \mid x_i) + \lambda_0 \mathcal{R}_0. \tag{16}$$

Here $\omega_y = 1$ for unweighted profiles; class-weighted profiles use inverse training-class frequency. Training uses Adam (Kingma & Ba, 2015) with parameter-group learning rates for prototypes, routing scales, gates, and leaf values. Branch prototypes are initialized from training inputs. The connectivity operation preserves at least one viable child at each internal node during optimization.

A.2 PREDICTION AND EXPLANATION ALGORITHM

Algorithm 1 summarizes one fitted tree from training to prediction and explanation. It retains the native soft leaf mixture for classification; the exported path is computed afterward from the same routing probabilities and gates. The local feature margin describes the best branch at each visited node, which can differ from the branch on the globally dominant path.

Algorithm 1 SPECTRA training, prediction, and path extraction

Require: Training data $\mathcal{D}_{\mathrm{tr}}$; L, M, δ, λ_0; query x
Ensure: Fitted tree, soft prediction, dominant path, concentration, and routing evidence

```
Fit preprocessing on D_tr; complement-code inputs.
Initialize prototypes from training inputs, scales at 3, gates, and leaf values.
for each full-batch training update do
    Compute T, μ, z using Eqs. (1), (2), and (15).
    Propagate all path flows B_π and form f_soft by Eqs. (4)–(5).
    Update parameters by Adam on Eq. (16).
    Project prototypes and scales; preserve a viable child at every internal node.
end for
Apply the fitted preprocessor and forward pass to x to obtain all B_π and f_soft.
Accumulate S_π = Σ_{(n,k)∈π} (A_{n,k} − log C_n); select π* = arg max^det_π S_π.
Compute q = B_{π*} / Σ_ρ B_ρ and local margins by Eq. (10).
If a positive-flow runner-up exists, decompose its comparison with π* by Eq. (12).
return fitted tree, arg max_c f_{soft,c}, π*, q, and the specified margin terms.
```

A.3 STORED STRUCTURE AND COMPUTATION

For a complete M-ary tree with $L - 1$ internal routing levels, the dense counts are

$$N_{\mathrm{int}} = \sum_{\ell=0}^{L-2} M^\ell, \qquad N_{\mathrm{leaf}} = M^{L-1}. \tag{17}$$

Prototype parameters contribute $2DMN_{\text{int}}$, gates contribute MN_{int}, routing scales contribute N_{int}, and leaf values contribute CN_{leaf}. Thus

$$P_{\text{dense}} = N_{\text{int}}(2DM + M + 1) + CN_{\text{leaf}}. \tag{18}$$

After training, structural reporting distinguishes stored, active, and reachable parameters. An edge is active when its deterministic gate is positive, and a parameter is reachable when every ancestral edge is active. The displayed topology contains exactly these reachable components.

Computing all branch scores costs $O(N_{\text{int}}MD)$, and aggregating leaf logits costs $O(N_{\text{leaf}}C)$. The dominant path is obtained from complete-path flows computed by this forward pass; greedy traversal is not an equivalent extraction algorithm. Once the branch scores are available, accumulating log flow and selecting a leaf takes time linear in the stored tree size. Feature contributions reuse the pairwise smooth-intersection terms. A sparse displayed topology need not reduce dense implementation cost because gated-off branches retain stored parameters.

A.4 Path identity and attribution details

For local flow $b_{n,k} = \mu_{n,k}z_{n,k}$, the routing definition gives $b_{n,k} = \exp(A_{n,k})/C_n$. Since C_n is positive and shared at a node, $A_{n,k}$ and $b_{n,k}$ have the same ordering. Taking products along a positive-flow path gives

$$\log B_\pi = \sum_{(n,k)\in\pi} \log b_{n,k} = \sum_{(n,k)\in\pi} (a_n T_{n,k} + \log z_{n,k} - \log C_n) = S_\pi. \tag{19}$$

A zero gate gives zero path flow and score $-\infty$. The same smallest-index tie rule makes the maximizers identical, proving Proposition 1. This is a real-arithmetic identity; numerical replay checks finite-precision evaluation separately.

The score-only control is $S_\pi^{\text{score}} = \sum_{(n,k)\in\pi}(a_n T_{n,k} - \log C_n)$. It retains the learned scale and the routing normalizers while omitting the gate offsets. Paths must be scored in full in both programs. Local winners can differ from the branches on π^*, so local margins describe branch competition rather than the complete-path decision by themselves.

Feature completeness follows by regrouping the $2D$ smooth-intersection terms into D coordinate pairs. Substitution into $A_{n,u} - A_{n,v}$ proves Eq. (10); summing along two paths and retaining their distinct normalizers proves Eq. (12). The normalizer difference is input dependent and remains explicit. Ranking $|F_{\pi^*,j} - F_{\rho^*,j}|$ evaluates the feature component of that decomposition, rather than attributing the entire path margin to features.

For a training-median reference b, fix the branch pair (u, v) chosen at the query x, either by the gate-aware score A or by the prototype score T for the specified analysis. Define $\Delta_n(b; u, v)$ and $m_{n,j}(b; u, v)$ using that same pair. Then

$$\Delta_n(x; u, v) = \Delta_n(b; u, v) + \sum_j \big[m_{n,j}(x; u, v) - m_{n,j}(b; u, v)\big]. \tag{20}$$

The input-independent gate offset cancels. Re-selecting the winner and runner-up at b would change the comparison and invalidate this identity. The eight-leaf feature study chooses (u, v) by T and computes

$$e_j(x; b) = \big[\phi_{n,u,j}(x) - \phi_{n,v,j}(x)\big] - \big[\phi_{n,u,j}(b) - \phi_{n,v,j}(b)\big]. \tag{21}$$

Its local ranking is $|e_j|$, and its global removal ranking is the training average of $|e_j|$. These values reconstruct changes in the chosen prototype-score margin. They do not include gate offsets or switch to a gate-aware winner. Local interventions on selected profiles use the distinct complete-path feature terms in Eq. (12).

Exact path selection retains $\log 0 = -\infty$. Finite ranking statistics use a 10^{-12} gate floor, and path-margin reconstruction is evaluated on examples with positive runner-up flow. With one positive-flow branch or path, its identity remains defined, but a finite positive-flow runner-up comparison is unavailable. These conventions distinguish numerical ranking statistics from exact path selection.

Prototype coordinates can be visualized as intervals $[1 - w_{n,k,D+j}, w_{n,k,j}]$. Their endpoints summarize parameters. Membership in these intervals does not execute the comparative score in Eq. (7); branch competition also uses normalization, routing scale, and gates.

B OpenML-CC18 predictive benchmark

B.1 Experimental units and aggregation

The 69-task roster uses the official OpenML-CC18 repeat-0/fold-0 split (Bischl et al., 2021). We exclude MNIST and Fashion-MNIST from the 71 tasks available to the pipeline because their high-dimensional pixel inputs carry spatial relationships that SPECTRA's coordinate-wise tabular prototypes do not encode. Under our fixed top-256 feature rule, the flattened inputs would also undergo univariate pixel selection, making the result depend on an image representation outside the model studied here. CIFAR-10 was unavailable after a download-integrity failure. Some retained feature datasets have image-derived origins and are processed as tabular coordinates. The studies share task identities and outer splits, but their fitting protocols answer different questions:

- The predictive benchmark selects profiles by validation accuracy, refits on the full outer training data, and averages final class probabilities across seeds before computing task accuracy. Final seeds are 2027–2029 for 61 tasks and 2027–2028 for eight tasks with at least 30,000 outer-training examples.
- The executable audit evaluates one tree at a time. The path-budget study separately refits the same selected profiles, and calibration selects and refits profiles using its own disjoint split. Each study has 199 fitted trees. Their reported metrics average seed-level results within tasks before averaging tasks.
- The 12-task feature and capacity studies use seeds 42–44. Local interventions and simulation use selected benchmark profiles; remove-and-retrain fixes an eight-leaf ceiling. Capacity comparisons use 4/8/16 leaves; efficiency refits four-leaf trees on one CPU core.
- Mechanism ablation fixes one sparse configuration on all 69 tasks and averages metrics over seeds 42–44. Parameter sweeps and training-budget experiments vary the stated factor without profile reselection.

Consequently, the 0.8300 predictive result describes probability-averaged trees, not the accuracy of an individual explained tree. Single-tree capacity studies supply the direct per-tree structural comparison. The fixed-configuration ablation accuracy and the calibrated-model accuracy belong to their respective fitting protocols.

B.2 Preprocessing and profile selection

All imputation, encoding, scaling, and feature selection are fitted on the relevant training partition. Categorical variables are encoded before scaling. When the processed dimension exceeds 256, a fixed ANOVA F top-256 rule selects coordinates using training labels. The main predictive protocol uses a stratified 80/20 inner split and validation accuracy. The six candidate profiles appear in Table 3; tasks with at least 30,000 outer-training examples search the first four profiles. Tuning uses at most 260 epochs (160 for large tasks), with seed 2027. The selected validation epoch determines the final horizon, clipped to 20–500 epochs. Ties prefer smaller depth and branch budget. Selected profiles are refit before outer-test evaluation.

B.3 Comparator protocols and predictive results

Public DecisionTree (Breiman et al., 1984), Random Forest (Breiman, 2001), XGBoost (Chen & Guestrin, 2016), and NODE (Popov et al., 2020) results are TabZilla accuracy-tuned fold-0 references matched by task ID (McElfresh et al., 2023). Their original preprocessing and tuning pipelines are retained. Local EBM (Nori et al., 2019) is additive and tunes learning rates 0.03 and 0.05; GradTree (Marton et al., 2024) tunes depths 3–5; MurTree (Demirovic et al., 2022) searches depths 2–4 over at most 64 training-selected binary conditions. EBM and GradTree have a 1,800-second task limit, and MurTree uses bounded solver time. Shared-task means exclude failures only for the affected comparator. The comparison aligns outer splits and selection objectives, not tuning budgets or seed aggregation across all methods.

Table 3: Fixed SPECTRA validation-search space. L is depth, M the branch budget, δ the smooth-intersection temperature, and λ_0 the sparsity coefficient.

Profile	L	M	δ	λ_0	Class wt.	Scaling
d2-m6-sparse	2	6	0.03	10^{-3}	yes	min–max
d2-m12-sparse	2	12	0.03	10^{-4}	yes	min–max
d2-m18-dense	2	18	0.08	0	no	min–max
d3-m3-sparse	3	3	0.03	10^{-4}	yes	min–max
d3-m5-dense	3	5	0.06	0	no	min–max
d2-m12-quantile	2	12	0.03	10^{-4}	yes	quantile

Table 4: Local baseline summary. Δ is SPECTRA minus comparator on common successful tasks.

Model	Tasks	Acc.	Δ	W/T/L	Pred. (μs)	Size (KiB)
EBM	68	0.8587	-0.0235	16/5/47	6.7	411.8
GradTree	68	0.7340	$+0.1012$	47/5/16	840.1	20.4
MurTree	64	0.7705	$+0.0716$	45/3/16	8.0	–

Table 5 aligns all eight methods by OpenML task ID. SPECTRA, EBM, GradTree, and MurTree use our evaluation pipeline; DecisionTree (DT), RandomForest (RF), XGBoost (XGB), and NODE use the public TabZilla Accuracy-tuned fold-0 results. Missing runs are left blank rather than imputed.

Table 5: Task-level test accuracy on the registered 69-task OpenML-CC18 subset. The first four method columns are from our runs; DT, RF, XGB, and NODE are public TabZilla fold-0 reference results matched by task ID. A dash indicates an unavailable result or a failed local fit. Coverage and means are computed separately over each method's available tasks; paired differences in Table 2 use only tasks shared with SPECTRA.

Task	Dataset	Spec.	EBM	Grad.	Mur.	DT	RF	XGB	NODE
3	kr-vs-kp	0.991	0.972	0.900	0.956	0.994	0.984	0.994	0.994
6	letter	0.750	0.881	0.303	–	0.757	0.894	0.968	–
11	balance-scale	0.984	1.000	0.794	0.825	0.857	0.905	0.952	1.000
12	mfeat-factors	0.950	0.965	0.805	0.740	0.865	0.950	0.960	0.930
14	mfeat-fourier	0.830	0.830	0.645	0.710	0.750	0.780	0.845	0.835
15	breast-w	0.957	0.986	0.957	0.971	0.957	0.971	0.957	0.971
16	mfeat-karhunen	0.955	0.965	0.615	0.695	0.790	0.950	0.960	0.970
18	mfeat-morphological	0.710	0.695	0.500	0.620	0.650	0.665	0.715	0.640
22	mfeat-zernike	0.765	0.780	0.540	0.625	0.650	0.740	0.790	0.815
23	cmc	0.608	0.601	0.574	0.574	0.514	0.595	0.595	0.581
28	optdigits	0.979	0.977	0.710	0.765	0.886	0.980	0.977	–
29	credit-approval	0.870	0.870	0.870	0.884	0.884	0.870	0.884	0.884
31	credit-g	0.810	0.810	0.630	0.670	0.750	0.710	0.830	0.700
32	pendigits	0.980	0.985	0.785	0.820	0.965	0.990	0.988	–
37	diabetes	0.792	0.818	0.779	0.844	0.805	0.792	0.831	0.818
43	spambase	0.941	0.941	0.870	0.842	0.898	0.924	0.931	0.946
45	splice	0.969	0.962	0.850	0.890	0.915	0.953	0.947	0.934
49	tic-tac-toe	0.979	0.958	0.812	0.781	0.875	0.917	0.990	0.958
53	vehicle	0.729	0.788	0.682	0.741	0.671	0.812	0.776	0.800
219	electricity	0.693	0.865	0.759	0.773	0.840	0.847	0.933	0.825
2074	satimage	0.857	0.902	0.722	0.837	0.863	0.897	0.911	–
2079	eucalyptus	0.676	0.649	0.581	0.541	0.689	0.676	0.689	0.730
3021	sick	0.820	0.984	0.976	0.942	0.995	0.995	0.995	0.992
3022	vowel	0.929	0.939	0.354	0.535	0.798	0.899	0.919	1.000
3481	isolet	0.903	0.950	0.341	0.403	0.800	0.919	0.944	–
3549	analcatdata_authorship	0.976	0.988	0.776	0.941	0.929	0.965	0.988	0.965
3560	analcatdata_dmft	0.175	0.225	0.300	0.237	0.250	0.237	0.200	0.263
3902	pc4	0.829	0.925	0.918	0.884	0.918	0.884	0.911	0.877
3903	pc3	0.898	0.879	0.898	0.892	0.873	0.879	0.866	0.898
3904	jm1	0.691	0.814	0.802	0.805	0.794	0.811	0.823	0.811
3913	kc2	0.811	0.755	0.774	0.774	0.811	0.792	0.755	0.830
3917	kc1	0.858	0.867	0.848	0.848	0.858	0.848	0.858	0.872

Task	Dataset	Spec.	EBM	Grad.	Mur.	DT	RF	XGB	NODE
3918	pc1	0.937	0.946	0.937	0.928	0.946	0.955	0.955	0.937
7592	adult	0.837	0.875	0.797	0.831	0.858	0.856	0.874	0.853
9910	Bioresponse	0.747	0.771	0.707	0.726	0.777	0.779	0.822	0.726
9946	wdbc	0.982	1.000	0.982	0.912	0.947	0.947	0.947	0.965
9952	phoneme	0.800	0.835	0.817	0.802	0.856	0.885	0.898	0.893
9957	qsar-biodeg	0.849	0.896	0.792	0.811	0.849	0.868	0.887	0.887
9960	wall-robot-navigation	0.934	0.995	0.890	0.914	0.989	0.989	0.998	0.954
9964	semeion	0.906	0.912	0.356	0.644	0.700	0.925	0.894	0.925
9971	ilpd	0.695	0.661	0.712	0.695	0.610	0.712	0.678	0.712
9976	madelon	0.608	0.631	0.650	0.688	0.773	0.688	0.850	0.812
9977	nomao	0.940	0.967	0.923	0.929	0.952	0.959	0.975	0.954
9978	ozone-level-8hr	0.937	0.941	0.941	0.937	0.933	0.957	0.945	0.937
9981	cnae-9	0.898	0.963	0.343	0.574	0.713	0.907	0.935	0.963
9985	first-order-theorem-proving	0.477	0.577	0.502	0.480	0.554	0.582	0.600	–
10093	banknote-authentication	1.000	1.000	0.978	0.964	0.978	0.993	0.993	1.000
10101	blood-transfusion-service-center	0.733	0.747	0.800	0.720	0.787	0.773	0.787	0.760
14952	PhishingWebsites	0.944	0.935	0.898	0.917	0.939	0.950	0.964	0.958
14954	cylinder-bands	0.778	0.889	0.741	0.759	0.796	0.796	0.852	0.815
14965	bank-marketing	0.879	0.903	0.893	–	0.899	0.903	0.908	0.906
14969	GesturePhase Segmentation	0.488	0.540	0.472	0.501	0.509	0.619	0.683	–
14970	har	0.934	0.989	0.612	0.636	0.943	0.977	0.995	–
125920	dresses-sales	0.560	0.580	0.580	0.620	0.600	0.480	0.640	0.660
125922	texture	0.947	0.984	0.705	0.760	0.916	0.969	0.985	–
146195	connect-4	0.754	0.759	0.659	–	0.730	0.730	0.854	0.840
146800	MiceProtein	0.972	1.000	0.676	0.685	0.843	0.981	0.981	0.935
146817	steel-plates-fault	0.749	0.764	0.605	0.677	0.728	0.774	0.810	0.574
146819	climate-model-simulation-crashes	0.870	0.926	0.926	0.963	0.852	0.926	0.963	0.963
146820	wilt	0.963	0.971	0.971	0.977	0.971	0.977	0.977	0.946
146821	car	0.994	0.919	0.827	0.850	0.954	0.960	0.994	1.000
146822	segment	0.948	0.974	0.892	0.918	0.918	0.935	0.926	0.961
146824	mfeat-pixel	0.965	0.975	0.715	0.685	0.885	0.975	0.955	0.975
167119	jungle-chess-2pcs	0.784	0.755	0.704	0.728	0.814	0.825	0.864	–
167120	numerai28.6	0.521	0.517	0.508	–	0.511	0.510	0.514	0.515
167121	Devnagari-Script	0.472	–	–	–	–	–	–	–
167125	Internet-Advertisements	0.945	0.954	0.933	0.936	0.960	0.945	0.966	0.966
167140	dna	0.953	0.947	0.893	0.878	0.922	0.925	0.962	0.925
167141	churn	0.902	0.868	0.906	0.902	0.934	0.934	0.948	0.964
Coverage		69/69	68/69	68/69	64/69	68/69	68/69	68/69	58/69
Mean		0.8300	0.8587	0.7340	0.7705	0.8191	0.8515	0.8745	0.8624

EBM and GradTree each time out on Devnagari-Script under the fixed 1,800-second task limit. MurTree's native solver exits on letter, bank-marketing, connect-4, numerai28.6, and Devnagari-Script. The public baselines were not retrained in our pipeline.

B.4 Selection objective and uncertainty

Validation accuracy is the declared primary selection objective for Table 2. The earlier balanced-accuracy selection run reached 0.8145 accuracy and 0.8125 BAcc; accuracy selection reaches 0.8300 and 0.7784, respectively. The selected profile changes on 25/69 tasks. This trade-off is relevant on imbalanced datasets, where majority-class accuracy can increase while minority-class recall decreases. The two selection runs are separate protocols and are not pooled.

Task-macro results weight each task equally. Seed-level metrics are averaged before task bootstrap, except that the main predictive benchmark first averages class probabilities as stated above. Paired intervals resample paired task differences. One outer fold measures performance on that split and does not estimate variation across outer folds. Public references and local methods also have different

search budgets; the table supports the reported pipeline comparison rather than an equal-budget ranking of model families.

C Path Agreement, Reconstruction, and Calibration

This appendix tests three distinct properties of the explanation: whether the exported gate-aware program reproduces the model's dominant flow, how much of the full soft prediction is captured by a limited path budget, and when an independently calibrated rule can display a single path. Agreement with the dominant leaf and agreement with the soft prediction have different targets and are reported separately.

C.1 Executable path agreement

The executable audit uses the same repeat-0/fold-0 CC18 roster as the predictive comparison, with 61 tasks evaluated under three final seeds and eight under two (199 fitted models). Each task is aggregated over its registered seeds before the 69 task means are averaged. Table 6 checks numerical replay, route selection, and soft-output agreement. The native replay and margin tolerances are 10^{-4} in float32, adopted after numerical diagnosis of an initial 10^{-6} check.

Table 6: Executable-semantics audit across 69 OpenML-CC18 tasks and 199 fitted models. Values are task-macro means; intervals are 95% task-bootstrap confidence intervals. Gate-aware and score-only agreement are path agreement measures.

Replay / flow metric	Result	Agreement / risk metric	Result
Native replay max error	4.14×10^{-6}	Gate-aware path agreement	1.0000
Replay 95% CI	$[3.30, 5.06] \times 10^{-6}$	Score-only path agreement	0.9956
Margin reconstruction error	1.70×10^{-6}	One-path agreement	0.7174
Margin error 95% CI	$[1.51, 1.90] \times 10^{-6}$	One-path JS divergence	0.1451
Flow concentration	0.2385	Concentration AURC	0.1976
Flow-gap AURC	0.2070	Log-margin AURC	0.2130

The gate-aware program reproduces the native dominant path on every audited example, with replay and margin errors below the stated float32 tolerance. Its single-path class agrees with the native soft class on 0.7174 of examples. Thus the exported path exactly identifies the largest flow component, while the complete predictor still aggregates multiple paths.

Figure 4 tests where gates affect that route and whether path-level scores rank examples by single-path disagreement. Removing structural gate offsets changes the selected leaf on six tasks: 28.57% of examples on `diabetes`, followed by 0.83% on `analcatdata_dmft`, 0.32% on `electricity`, 0.30% on `GesturePhaseSegmentationProcessed`, 0.26% on `sick`, and 0.07% on `wilt`. The other 63 tasks show no observed route change. The fixed-coverage curves assess ranking only; they do not calibrate an abstention rule. The risk–coverage areas in Table 6 summarize the same ranking behavior across coverage levels, with lower values indicating less disagreement.

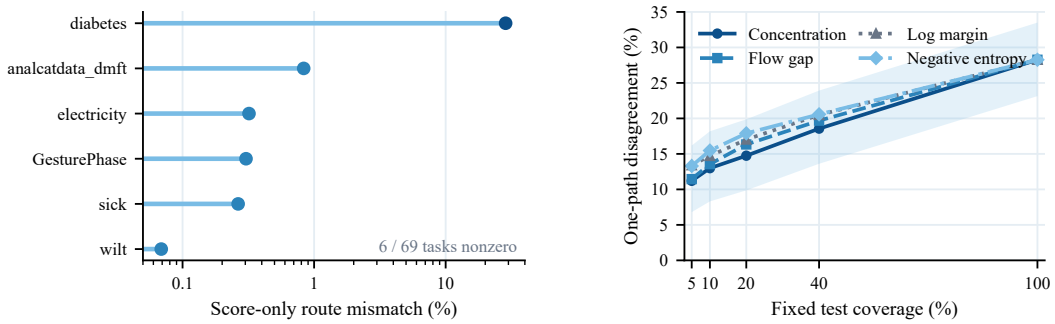


Figure 4: Gate heterogeneity and selective-ranking diagnostic on the same 69 OpenML-CC18 tasks and 199 fitted models. (a) Score-only versus native dominant-path mismatch on the six tasks with nonzero differences; the horizontal axis is logarithmic. (b) Held-out class disagreement after selecting a fixed fraction of test inputs by each score. Means average seeds within tasks and then tasks equally; the blue band is a 95% task-bootstrap interval for concentration.

C.2 Path-budget reconstruction

The 0.7174 single-path agreement motivates the path-budget comparison in Table 7. Paths are ranked by native flow, and their selected logit contributions are rescaled to preserve total surviving mass as in Eq. (14). Adding paths raises class agreement from 0.717 at $k = 1$ to 0.845 at $k = 3$ and 0.934 at $k = 8$; random path order is consistently worse. These are within-model reconstruction results, not test-label accuracy.

Table 7: Top-k soft-output reconstruction on the registered 69-task CC18 subset (199 models). Class agreement and JS compare reconstructed predictions with the same fitted model's native soft output. Means average seeds within tasks and tasks equally. Brackets are 95% task-bootstrap intervals. Random ordering averages 20 fixed draws per model; "all" checks replay.

k	Native agreement [CI]	Score-only agreement	Random agreement	Native JS	Retained flow
1	0.717 [0.664, 0.769]	0.718	0.389	0.145	0.239
2	0.801 [0.765, 0.836]	0.801	0.449	0.081	0.399
3	0.845 [0.813, 0.875]	0.845	0.486	0.056	0.519
5	0.887 [0.858, 0.912]	0.887	0.554	0.029	0.689
8	0.934 [0.913, 0.953]	0.934	0.661	0.011	0.836
all	1.000 [1.000, 1.000]	1.000	1.000	$< 10^{-7}$	1.000

C.3 Independent within-task calibration

Selective classification motivates calibrating a display threshold using data disjoint from testing (Geifman & El-Yaniv, 2017). A ranking score is useful for displaying one path only if its threshold can be chosen without test outcomes. For each task, the outer training split is stratified into 70% fitting, 15% model selection, and 15% explanation calibration. Six predefined profiles are compared by validation accuracy on the model-selection partition (four profiles for large tasks). The selected profile is refit on the first 85%, separately for each registered seed. The calibration examples are unseen by both fitting and profile selection.

For each score, 21 candidate thresholds are fixed as quantiles of the refit inputs without consulting calibration outcomes. At each candidate threshold, the number of dominant-path disagreements with the native soft class among selected calibration examples gives a one-sided Clopper–Pearson error bound (Clopper & Pearson, 1934) at error probability $0.05/21$. The highest-coverage candidate whose bound is at most the target error is frozen; if none qualifies, the model abstains. This Bonferroni correction covers selection among candidate thresholds. Applying the frozen threshold to the official test split produces coverage and conditional fidelity; empty test selections contribute zero coverage

and an undefined fidelity. The risk-bound interpretation requires calibration and test examples to be exchangeable for the fitted model and does not assert transfer between tasks. All 69 tasks and 199 fits completed without training failures; the task-macro base accuracy of these 85%-training models is 0.8242.

Table 8 shows the resulting fidelity–coverage trade-off. At the 5% target, concentration selects nonempty test sets on 28 of 69 tasks, with 20.5% task-macro coverage and 98.8% fidelity conditional on nonempty tasks. Tightening the target to 1% leaves seven tasks nonempty and reduces coverage to 5.6%. The "target met" count requires every registered seed on a task to make a nonempty selection and satisfy the empirical target; it does not follow from the conditional mean fidelity.

Table 8: Within-task disjoint calibration on the registered 69-task CC18 subset, 199 refitted models (zero failures). Coverage includes empty selections; fidelity is averaged over nonempty seeds within each nonempty task and is undefined for empty tasks. Brackets are 95% task-bootstrap intervals. The last column counts tasks for which every registered seed makes a nonempty selection and meets the empirical target. Median calibration size is 270; model test accuracy is 0.8242.

Score	Target error	Nonempty	Coverage [CI] %	Fidelity [CI] %	Target met
Concentration	0.05	28/69	20.5 [12.9, 28.5]	98.8 [98.3, 99.2]	15/69
Concentration	0.01	7/69	5.6 [1.6, 10.7]	99.8 [99.6, 99.9]	6/69
Flow gap	0.05	26/69	19.9 [12.3, 28.3]	98.6 [98.1, 99.1]	15/69
Flow gap	0.01	7/69	5.2 [1.3, 10.2]	99.8 [99.7, 100.0]	5/69
Log margin	0.05	25/69	18.7 [11.5, 26.7]	98.5 [97.9, 99.0]	13/69
Log margin	0.01	6/69	4.8 [0.9, 9.8]	99.8 [99.6, 100.0]	5/69
Negative entropy	0.05	22/69	18.4 [10.9, 26.9]	98.6 [98.2, 99.1]	15/69
Negative entropy	0.01	7/69	4.4 [0.8, 9.2]	99.8 [99.7, 100.0]	5/69

Figure 5 makes abstention visible: stricter error targets reduce coverage for all four scores, and the concentration distribution places many tasks at zero test coverage. This is a selective single-path interface rather than a high-coverage guarantee across the CC18 roster.

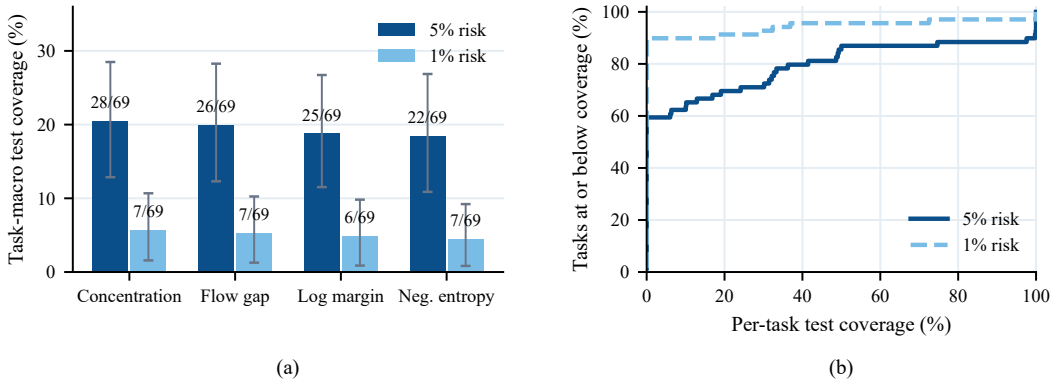


Figure 5: Coverage and abstention after disjoint within-task calibration. (a) Macro test coverage with 95% task-bootstrap intervals; labels give the number of tasks with any nonempty test selection out of 69. Dark and light blue distinguish 5% and 1% target error. (b) Empirical cumulative distribution of per-task test coverage for concentration; the mass at zero represents abstaining tasks. Thresholds were frozen before outer-test evaluation.

The earlier leave-one-task-out threshold diagnostic produced a nonempty selection on only 1/69 held-out tasks. Its conditional agreement of 1.0000 therefore does not establish useful cross-task calibration. The disjoint within-task protocol above instead calibrates each fitted model using examples from its own task. Its 5% error target has nonempty test selections on 28 tasks, while 15 tasks have nonempty selections meeting the empirical target for every registered seed. Neither count is a simultaneous guarantee over all tasks and scores.

C.4 A GATE-INDUCED PATH CHANGE

Figure 2 traces a diabetes test record through a fitted one-routing-level, six-leaf SPECTRA router (official repeat-0/fold-0 split; seed 42; OpenML row 547). Among the 19 of 256 test records with correct soft and dominant-path predictions but different score-only and gated leaves, we selected the record nearest the median flow concentration, breaking ties by OpenML row ID. This seed-42 fit is a separate illustrative model from the 199-tree audit. The figure shows its original-scale inputs and all six candidate flows.

The largest soft-routing probability alone would select L2, a leaf with zero gated flow. The native predictor instead combines L1, L3, and L5 and assigns probability 0.889 to the correct class (No); the L1-only dominant-path prediction is 0.999. Replaying all six gated flows reconstructs the native logits to maximum absolute error 4.8×10^{-7}. This selected case illustrates executable route semantics, not a population-level accuracy benefit of gating.

D FEATURE EVIDENCE: INTERVENTIONS, SIMULATION, AND RETRAINING

D.1 SHARED TASKS AND FEATURE-RANKING TARGETS

Before observing the 12-task study outcomes, we scanned the 69 registered IDs in ascending order and selected the first 12 with numeric-only inputs, 5–64 features, at most 30,000 outer-training rows, and no image-derived origin among eight excluded IDs. The resulting tasks are breast-w (15), diabetes (37), spambase (43), pc4 (3902), pc3 (3903), jm1 (3904), kc2 (3913), kc1 (3917), pc1 (3918), wdbc (9946), phoneme (9952), and qsar-biodeg (9957). They use official repeat-0/fold-0 splits and seeds 42–44. All feature manipulations refer to original numeric features, whose dimensions match the processed input dimensions.

Two fitting regimes test different uses of routing evidence. Local interventions, forward simulation, and the selected-profile stability study use the main benchmark's selected SPECTRA profiles and final training epochs (36 fitted trees). They rank $|F_{\pi^*,j} - F_{\rho^*,j}|$ from Eq. (12), evaluate at most 1,000 held-out examples per fit, and use training medians for replacements. Remove-and-retrain and cross-model attribution comparisons use fixed eight-leaf trees. SPECTRA then has one routing level. The implemented ranking selects the winner and runner-up by the ungated prototype score $T_{n,k}$, subtracts the same pair's contributions at the training-median reference, and averages their absolute values over training examples for global removal. We call this *baseline-relative prototype-margin evidence*; it differs from the gate-aware path-margin ranking used in local interventions. A shared positive routing scale would multiply every feature by the same constant and therefore leaves these rankings and F_{90} unchanged. Test labels are used for outcome evaluation, not ranking or simulator training.

The studies use functional explanation tests (Dasgupta et al., 2022), remove-and-retrain (Hooker et al., 2019), and forward simulation (Chen et al., 2022). Perturbation tests examine stability separately from fidelity (Yeh et al., 2019). Seeds and perturbation draws are averaged within task; 95% intervals use 10,000 task-bootstrap resamples. The 12-task subset is not a representative estimate for all 69 tasks, and simulator agreement does not measure human understanding.

D.2 LOCAL RETENTION AND REMOVAL

For each held-out example, we retain or replace the top 10%, 30%, or 50% original numeric features in the *same fitted* soft predictor. Inspired by rationale retention and removal tests (DeYoung et al., 2020), we define sufficiency as native soft-class agreement after retention and comprehensiveness as native predicted-class probability loss after replacement. The retention measure is class agreement rather than the probability-difference sufficiency metric used in ERASER. Random and reversed ranks use identical budgets and medians. These model-output interventions are neither causal effects nor remove-and-retrain estimates (Fig. 6).

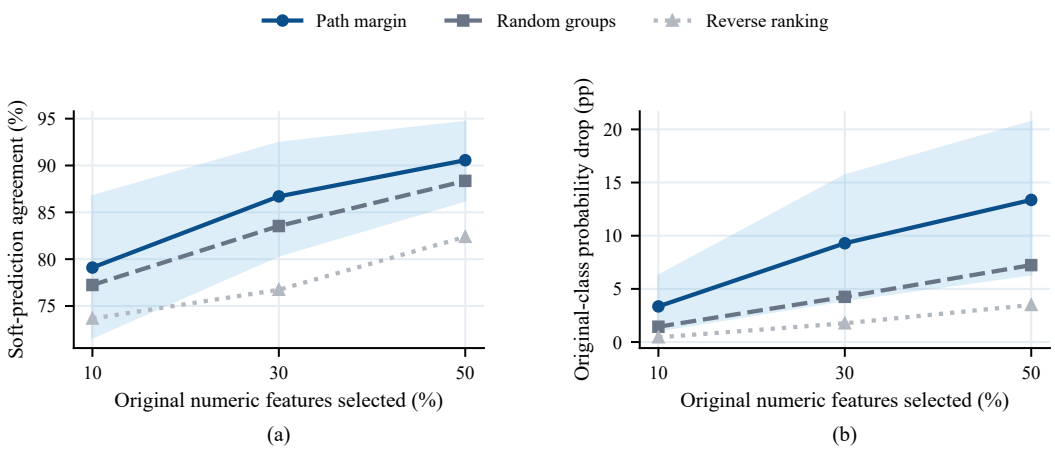


Figure 6: Feature-budget interventions on 12 metadata-selected CC18 tasks (36 fitted models). (a) Agreement with the unmodified native soft class after retaining selected original numeric features. (b) Mean decrease in the native predicted-class probability after replacing selected features by training medians. Native complete-path margin ranking is dark blue; random and reversed rankings use the same per-example budget. Pale blue bands are 95% task-bootstrap intervals for native ranking. Neither intervention retrains the model.

At 30% retention, path-margin sufficiency is 86.7%, versus 83.5% for random, but the paired path-margin–random difference is 3.16 points with interval $[-0.27, 6.85]$. Replacing path-margin-ranked features reduces native predicted-class probability by 9.29 points versus 4.25 for random: a paired difference of 5.04 points $[1.17, 10.30]$. The deletion comparison supports functional relevance on this subset; the retention comparison does not resolve a gain over random.

D.3 Explanation-assisted forward simulation

Following simulation-based explanation evaluation (Chen et al., 2022), we train the same logistic-regression simulator to predict SPECTRA's *native soft class* from the indices and values of either its top three path-margin features or three random features. Input to each simulator is a length-D vector of median-centered selected values (zero elsewhere), concatenated with a length-D selection mask. Training uses up to 3,000 outer-training examples per fitted model; evaluation uses the same held-out test sample as the interventions. A majority-class simulator is the floor and a logistic simulator with all features is a budget ceiling. Neither student sees test model outputs during fitting, and simulator agreement is not human-subject evidence.

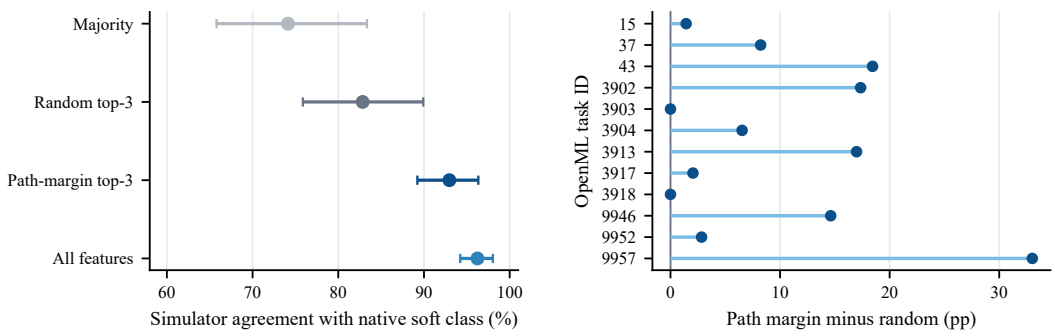


Figure 7: Forward simulation on the same 12 CC18 tasks and 36 fitted SPECTRA models. (a) Task-macro agreement of a fixed logistic student with the teacher's native soft class; bars are 95% task-bootstrap intervals. Native and random students each receive three feature values plus their positions. Majority and all-feature students define information floors/ceilings rather than explanation methods. (b) Native-minus-random paired accuracy by frozen OpenML task ID, after averaging seeds within task.

The top-three path-feature simulator reaches 92.96% task-macro agreement, versus 82.84% for random features, a paired 10.12-point gain $[5.04, 15.93]$. It remains 3.27 points $[1.22, 5.71]$ below the all-feature student's 96.24%. The gain measures the information supplied to this fixed algorithmic student, not the effectiveness of an explanation for people or a faithful reconstruction of all soft probabilities.

D.4 Remove-and-retrain and attribution compactness

The retraining comparison uses the shared 12 tasks, official split, and seeds defined above, with training-fitted min–max preprocessing. Ranking and replacement use training information only.

The single-tree controls follow the eight-leaf setting of Appendix F.1. SPECTRA uses depth two, branch budget eight, and 500 epochs. CART has at most eight leaves. GradTree has one depth-three tree and 500 maximum epochs, with its early-stop monitor drawn from training inputs. MurTree has depth three, at most seven internal nodes, training-derived binary conditions, and a 60-second per-fit solver limit. Its result is a bounded-time search result. Additive EBM has no interactions, eight outer bags, at most 3,000 rounds, and a 15% internal validation fraction. The models share splits and processed inputs, while their optimizers and fitted structures remain method-specific.

SPECTRA ranks training-average absolute baseline-relative prototype-margin evidence, CART uses impurity reductions along each realized path, and EBM uses native additive terms (Nori et al., 2019). The available GradTree and MurTree predictors do not expose a directly comparable signed feature-attribution object. Their rankings therefore use training-only single-permutation importance and are labelled post-hoc throughout. We compare SPECTRA's prototype-margin order against its own permutation, random, and reversed orders; between-model ROAR curves describe each model with its declared ranking source rather than a common attribution mechanism.

Following ROAR (Hooker et al., 2019), we replace the top-ranked 10%, 30%, or 50% of original features in both training and test data with outer-training medians, retrain the same model family from scratch, and measure the balanced-accuracy decrease from its unmodified fit. A larger decrease indicates that the refitted predictor cannot readily replace the removed features. Data-dependent rankings use the training split only; random order uses a fixed seed, and reverse order inverts SPECTRA's prototype-margin ranking.

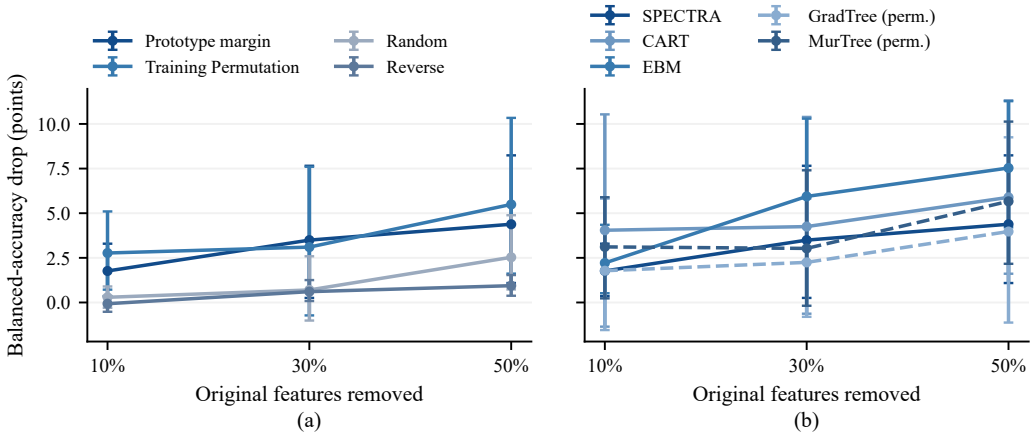


Figure 8: Remove-and-retrain evaluation on 12 OpenML-CC18 tasks and three seeds. (a) SPECTRA's prototype-margin order versus training-permutation, random, and reversed orders. (b) Each model's declared ranking: prototype margin for SPECTRA, path impurity for CART, and additive terms for EBM; training-permutation for GradTree and MurTree (dashed). Points are task-macro balanced-accuracy decreases; error bars are task-bootstrap 95% intervals.

Table 9: Predictive and explanation characteristics on the 12-task CC18 feature suite. Brackets give task-bootstrap 95% intervals for balanced accuracy. F_{90} counts original features needed for 90% of absolute attribution mass; it is inapplicable to the post-hoc GradTree and MurTree rankings. Explanation time is measured in microseconds per test example for native methods.

Method	BAcc [95% CI]	Macro-F1	Accuracy	F_{90}	μs/example
CART	73.16 [64.83, 81.68]	73.46	86.11	2.26	26.72
SPECTRA	70.46 [60.52, 80.98]	70.20	87.44	6.94	113.72
EBM	74.31 [65.00, 83.78]	75.46	88.65	13.86	6.79
GradTree	67.23 [57.92, 76.89]	66.17	85.55	–	–
MurTree	69.76 [61.02, 78.81]	69.81	85.86	–	–

Table 9 shows that SPECTRA's prototype-margin evidence is more concentrated than EBM's but less concentrated than CART's. Its mean balanced accuracy is 70.46%, below CART's 73.16% and EBM's 74.31% in this fixed eight-leaf setting; GradTree and MurTree score 67.23% and 69.76%. The SPECTRA prototype-margin calculation takes 113.72 μs per test example in this implementation, compared with 26.72 μs for CART and 6.79 μs for EBM. These timing values describe the native routines, not model-independent explanation costs.

Table 10: Task-macro balanced-accuracy decrease (percentage points) after removing ranked features and retraining. SPECTRA uses prototype-margin evidence, CART path impurity, and EBM additive terms; GradTree and MurTree use training-only permutation importance. Values describe ranking consequences within each fitted family, not a ranking of explanation quality across unequal base predictors.

Method	Remove 10%	Remove 30%	Remove 50%
CART (native)	4.05	4.25	5.89
SPECTRA (prototype margin)	1.76	3.49	4.38
EBM (native)	2.21	5.94	7.53
GradTree (permutation)	1.78	2.25	3.97
MurTree (permutation)	3.11	3.03	5.67

Table 11: SPECTRA remove-and-retrain controls on the same 12 tasks. Prototype denotes baseline-relative prototype-margin evidence. The first four columns are task-macro balanced-accuracy decreases in percentage points; the final columns give paired prototype-minus-control differences and task-bootstrap 95% intervals.

Removed	Prototype	Perm.	Random	Reverse	Prototype−random	Prototype−reverse
10%	1.76	2.77	0.29	−0.07	1.47 [0.22, 2.92]	1.83 [0.58, 3.15]
30%	3.49	3.10	0.70	0.61	2.79 [0.80, 5.33]	2.89 [−0.11, 6.43]
50%	4.38	5.49	2.53	0.94	1.85 [−0.01, 3.86]	3.44 [0.01, 7.40]

At 30% removal, SPECTRA's prototype-margin order reduces refit balanced accuracy by 3.49 points, 2.79 points more than random order (95% task-bootstrap interval $[0.80, 5.33]$). The reverse-order difference at this fraction is 2.89 points, but its interval crosses zero. At 10% removal, training-permutation importance is 1.01 points more consequential than the prototype-margin ranking (paired interval $[0.03, 2.12]$ for permutation minus prototype margin); at 30% and 50%, their paired intervals overlap zero. The evidence therefore supports functional relevance beyond random ranking at 10% and 30% removal, while a trained perturbation ranking can be stronger for the smallest removal budget. Three tasks have baseline balanced accuracy 0.50 and zero ROAR-30 drop, which limits the task-macro effect.

D.5 Prediction, path, and feature stability

The two stability regimes share a task roster but use different fitted profiles and conditioning. Their curves should therefore be interpreted within each regime rather than pooled.

D.5.1 Selected benchmark profiles

We add independent Gaussian noise at standard deviations of 0.01, 0.02, 0.03, 0.04, 0.05, 0.075, and 0.10 in the normalized numeric feature space, clip to $[0, 1]$, and repeat three draws per example at each level. We separately measure soft-prediction agreement, dominant-path agreement, and Top-3 feature-rank Jaccard conditioned on an unchanged soft class, following the distinction between explanation sensitivity and fidelity (Yeh et al., 2019). This separates instability of an explanation from instability of the predictor being explained; it does not certify robustness to realistic distribution shift or adversarial attacks.

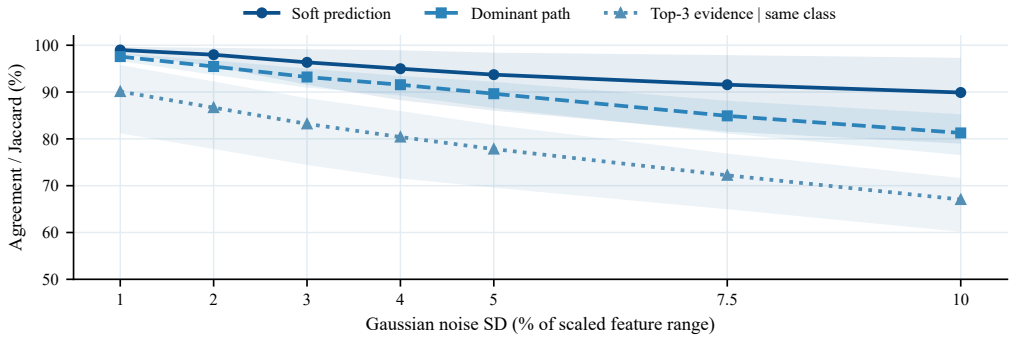


Figure 9: Three distinct stability targets across seven Gaussian noise levels on the same 12 CC18 tasks and 36 fits. Each example receives three bounded numeric perturbations per level. Points are task-macro means and pale bands are 95% task-bootstrap intervals. The dotted Top-3 Jaccard curve is computed only on perturbations for which the soft model retains its class; the solid and dashed curves show unconditional soft-prediction and dominant-path agreement.

Across 1% to 10% noise, soft-prediction agreement declines from 99.0% to 89.9%, dominant-path agreement from 97.6% to 81.3%, and class-conditioned Top-3 feature Jaccard from 90.1% to 67.0%. At the original 2% and 5% levels, the respective values remain 98.0%/93.7%, 95.5%/89.6%, and

86.7%/77.8%. The wider noise range makes the faster decline in route and evidence stability visible even when most soft predictions remain unchanged. The last metric conditions on model-class stability and does not measure invariance across retrained models.

D.5.2 FIXED EIGHT-LEAF COMPARISON

For SPECTRA, the local ranking uses the same baseline-relative prototype margins as the removal study; the feature pair is re-selected at each perturbed input. For each test point, we add Gaussian perturbations at levels 0.01, 0.05, and 0.10 after min–max scaling and clip the result to $[0, 1]$. Feature-evidence stability reports Top-3 feature Jaccard and prediction agreement for SPECTRA, CART, and EBM. GradTree and MurTree are excluded from this local stability panel because their permutation rankings are global. Seeds are averaged within task; reported means and bootstrap intervals then treat the 12 tasks as the sampling unit. All five models complete all 36 task–seed units.

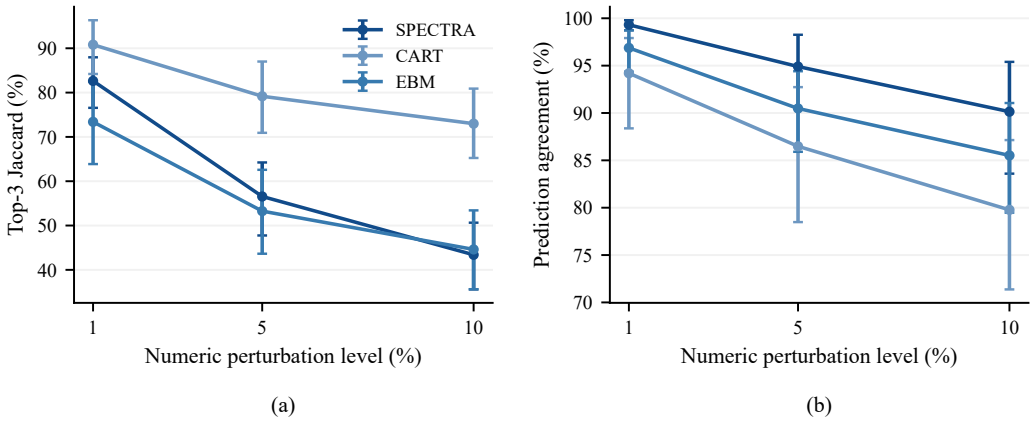


Figure 10: Feature-evidence stability on 12 CC18 tasks and three seeds. (a) Top-3 Jaccard; (b) prediction agreement under 1%, 5%, and 10% numeric noise. Curves are task-macro means with task-bootstrap 95% intervals. GradTree and MurTree have global post-hoc rankings and do not enter this local test.

Table 12: Top-3 explanation Jaccard / prediction agreement under valid perturbations. Values are task-macro percentages.

Method	0.01	0.05	0.10
CART	90.81 / 94.20	79.17 / 86.48	72.99 / 79.77
SPECTRA	82.64 / 99.31	56.56 / 94.90	43.40 / 90.14
EBM	73.40 / 96.87	53.26 / 90.48	44.61 / 85.52

At perturbation level 0.05, SPECTRA retains 56.56% Top-3 Jaccard, versus 53.26% for EBM and 79.17% for CART. Its prediction agreement is 94.90%, above EBM (90.48%) and CART (86.48%). Thus, feature order varies more than predicted labels; CART has the most stable Top-3 sets in this comparison.

Table 13: SPECTRA task-level results (three-seed means) on 12 CC18 tasks. ROAR-30 is balanced-accuracy decrease; Top-3 and prediction agreement use perturbation level 0.05.

Task	BAcc	Macro-F1	F_{90}	ROAR-30	Top-3 / Pred.
breast-w	96.14	96.29	2.95	−1.69	68.33 / 100.00
diabetes	81.70	82.14	4.45	18.43	67.97 / 93.07
spambase	88.06	88.73	5.61	16.44	32.83 / 74.77
pc4	50.00	46.72	6.81	0.00	69.95 / 100.00
pc3	50.00	47.32	7.28	0.00	67.47 / 100.00
jm1	55.00	54.76	6.97	5.00	25.38 / 97.64
kc2	56.64	57.17	7.20	−3.35	46.23 / 92.45
kc1	61.34	64.00	7.21	2.14	55.67 / 97.95
pc1	50.00	48.37	6.84	0.00	52.61 / 100.00
wdbc	99.54	99.38	12.79	1.26	55.73 / 98.25
phoneme	72.29	72.63	3.10	0.91	74.89 / 96.98
qsar-biodeg	84.81	84.89	12.07	2.78	61.67 / 87.74

The artifact includes all 36 task–seed records, 180 base-model rows, 324 stability rows, and 1,152 remove-and-retrain rows. Breast-w and kc2 have negative ROAR-30 changes, and three tasks with balanced accuracy 0.50 have zero change; all remain in the task-macro estimate.

E Mechanism Ablation and Parameter Sensitivity

E.1 Fixed-configuration ablation on 69 tasks

The component study covers the same registered 69-task OpenML-CC18 subset used in the predictive benchmark. We use the official repeat-0/fold-0 splits, fit preprocessing on the outer training data, and apply the same fixed ANOVA F top-256 rule when the processed dimension exceeds 256. Accuracy is the primary endpoint, matching the model-selection and comparison metric in Section 5. Balanced accuracy, macro-F1, and log loss provide secondary views of classwise and probabilistic behavior. Every configuration is trained with seeds 42–44, giving 828 completed fits with no task failures. Reported means first average the three seeds within a task and then average across tasks. Paired 95% intervals resample the 69 task differences 20,000 times.

Mechanism isolation requires the component to be active on every task, so this study fixes the canonical sparse configuration rather than reselecting a profile separately for each ablation. The full model uses depth two, six candidate branches, smooth-intersection temperature 0.05, $\lambda_0 = 10^{-3}$, complement coding, and learned routing scales. It is optimized for 500 full-batch epochs with the same optimizer and initialization protocol across configurations. The dense-routing control sets $\lambda_0 = 0$ while retaining trainable gate parameters. Because the all-open initialization lies in the clipped region of the hard-concrete transform, this control realizes the intended six-branch dense topology without duplicating it as a separate fixed-gate row. The fixed-scale control holds every routing scale at its initialization of 3. The no-complement control supplies normalized features directly instead of concatenating $(x, 1 - x)$. Each control therefore changes one mechanism while preserving the remaining training specification.

Table 14: Mechanism ablation on all 69 OpenML-CC18 tasks with three seeds. Δ is the task-paired accuracy difference of each control relative to full SPECTRA. W/T/L counts control-versus-full outcomes over tasks. Active is the mean number of nonzero structural gates out of six.

Configuration	Accuracy	Δ	95% CI of Δ	W/T/L
Full SPECTRA	0.8317	0.0000	[0, 0]	0/69/0
w/o L_0 penalty	0.8341	+0.0024	[-0.0005, 0.0053]	28/20/21
Fixed routing scale	0.8067	-0.0250	[-0.0367, -0.0154]	8/13/48
w/o complement coding	0.8281	-0.0037	[-0.0095, 0.0020]	27/14/28

Configuration	Balanced acc.	Macro-F1	Log loss	Active
Full SPECTRA	0.7475	0.7457	0.4374	4.80
w/o L_0 penalty	0.7575	0.7610	0.4359	6.00
Fixed routing scale	0.7037	0.7000	0.5243	4.95
w/o complement coding	0.7482	0.7498	0.4560	5.13

The learned routing scale is the only component whose removal produces a stable accuracy decrease. Fixing the scale lowers mean accuracy by 2.50 percentage points, with a paired interval of $[-3.67, -1.54]$ points, and loses on 48 of 69 tasks. Balanced accuracy falls by 4.38 points, macro-F1 by 4.57 points, and log loss increases from 0.4374 to 0.5243. Learning the scale improves the predictive operating point of these prototype scores across the evaluated tasks.

Complement coding has a smaller and task-dependent predictive effect. Removing it lowers mean accuracy by 0.37 points, with an interval spanning zero, and produces 27 wins, 14 ties, and 28 losses. The corresponding log loss increases by 0.0186. These results support complement coding as the representation used for bounded prototype evidence, but do not attribute a uniform accuracy gain to it in isolation.

Removing the L_0 term opens all six branches, whereas full SPECTRA retains 4.80 on average, a 20.0% reduction in realized routing structure. The dense control gains 0.24 accuracy points on average, but its interval $[-0.05, 0.53]$ points includes zero and its two-sided Wilcoxon signed-rank test gives $p = 0.141$. The measured role of L_0 is therefore structural: it removes approximately one fifth of the candidate branches without a detectable task-macro accuracy change. This result complements the predictive effect of the learned scale. The scale determines how strongly prototype evidence separates routes, while the L_0 term determines how much of the candidate topology remains executable.

The main OpenML-CC18 result selects among six profiles by validation accuracy and selects a predictive operating point for each task. The fixed configuration here instead holds the branch budget and sparsity pressure constant to identify component roles. This separation prevents validation selection from disabling the component under study and keeps the ablation comparison paired across all 69 tasks. The full fixed configuration reaches 0.8317 mean accuracy, close to the 0.8300 validation-tuned benchmark result, while exposing the accuracy–topology tradeoff directly.

E.2 Parameter sensitivity on three tasks

We vary three parameters of the fixed SPECTRA configuration used in Appendix E.1: candidate branches M, smooth-intersection temperature δ, and the L_0 coefficient λ_0. The depth-two model otherwise retains 500 full-batch training epochs, complement coding, learned routing scales, and the same optimizer. Each grid changes one parameter at a time around $(M, \delta, \lambda_0) = (6, 0.05, 10^{-3})$. We use the official OpenML-CC18 repeat-0/fold-0 split and training-only min–max preprocessing. The three tasks come from the frozen 12-task subset in Appendix D.1: breast-w (9 features, 629 outer-training rows), kc1 (21, 1,898), and spambase (57, 4,140). Breast-w and spambase retain their original measurements. After the original middle-stratum task pc1 showed a large classwise performance loss under sparse settings, we selected kc1 from the same frozen roster by its matching 21-feature dimension before running the replacement sweep. Seeds 42–44 give 135 task–setting–seed records across three five-point grids. The three tasks span feature dimension and training size within this subset, but do not estimate sensitivity over all 69 CC18 tasks. Test labels measure each completed setting and do not select a sweep point.

Figures 11 and 12 show predictive and one-path behavior separately. Each point is the mean of three independent initializations on the same task, and whiskers span their observed range; no shaded uncertainty region is used. The one-path measure is agreement between the dominant-path prediction and the native soft prediction, rather than a score of feature-attribution quality.

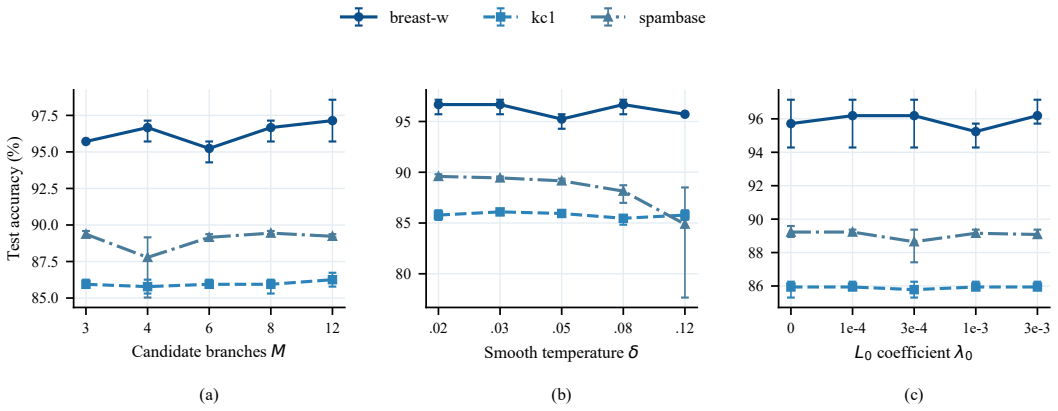


Figure 11: Test-accuracy sensitivity on three CC18 sensitivity tasks. (a) Candidate branch budget $M \in \{3, 4, 6, 8, 12\}$. (b) Smooth-intersection temperature $\delta \in \{0.02, 0.03, 0.05, 0.08, 0.12\}$. (c) Sparsity coefficient $\lambda_0 \in \{0, 10^{-4}, 3 \times 10^{-4}, 10^{-3}, 3 \times 10^{-3}\}$. All other settings are fixed at the stated baseline; points average three seeds and whiskers give their minimum–maximum range. Higher accuracy is better.

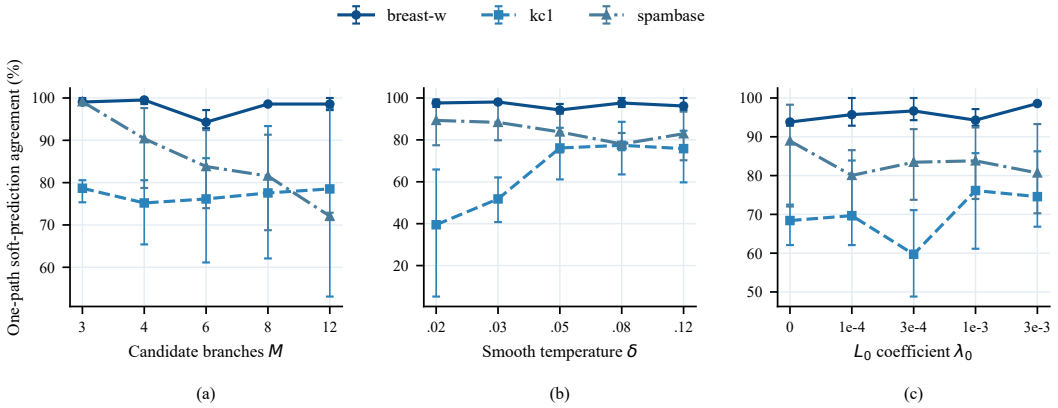


Figure 12: Sensitivity of one-path soft-prediction agreement under the same fits and grids. Panels (a), (b), and (c) vary M, δ, and λ_0, respectively; whiskers are the three-seed minimum–maximum range. Agreement falls as M increases on spambase, while kc1 is most sensitive to small δ. This diagnostic measures one-path prediction agreement rather than predictive correctness or feature-attribution quality.

Table 15 gives the task-macro means after averaging seeds within task. The repeated baseline points in the three scans agree for each task and seed. Active is the mean number of nonzero root gates; all runs have depth two, so it is also the number of active leaves.

Table 15: Three-task parameter sensitivity. Accuracy, balanced accuracy (BAcc), and one-path soft-prediction agreement are percentages; Active is the realized active-leaf count. Values average three seeds within each task and then the three tasks equally. Bold rows reproduce the shared baseline configuration.

Parameter	Value	Accuracy	BAcc	Active	One-path agreement
	3	90.34	81.21	2.67	92.28
	4	90.08	81.38	3.33	88.37
M	**6**	**90.11**	**81.03**	**4.89**	**84.74**
	8	90.68	81.85	6.00	85.90
	12	90.88	81.92	8.44	83.06
	0.02	90.68	81.60	5.00	75.47
	0.03	90.74	81.94	5.00	79.42
δ	**0.05**	**90.11**	**81.03**	**4.89**	**84.74**
	0.08	90.09	80.70	4.67	84.35
	0.12	88.79	79.18	5.44	84.99
	0	90.29	81.30	6.00	83.72
	10^{-4}	90.45	81.54	5.44	81.81
λ_0	3×10^{-4}	90.21	81.30	4.89	79.94
	10^{-3}	**90.11**	**81.03**	**4.89**	**84.74**
	3×10^{-3}	90.40	81.34	4.11	84.61

The task-macro accuracy spans 90.08–90.88% across branch budgets, while mean active leaves rise from 2.67 to 8.44. Temperatures 0.02–0.08 give 90.09–90.74% accuracy; 0.12 falls to 88.79%. Increasing λ_0 from zero to 3×10^{-3} reduces active leaves from 6.00 to 4.11 while accuracy changes from 90.29% to 90.40% and balanced accuracy from 81.30% to 81.34%. On kc1, balanced accuracy changes from 61.34% without the penalty to 60.92% at 3×10^{-3}; the 50% balanced-accuracy plateau observed on the original pc1 task does not recur in this replacement sweep.

Increasing M from 3 to 12 reduces task-macro one-path soft-prediction agreement from 92.28% to 83.06%. The decline is driven mainly by spambase; kc1 stays near 75–79% across branch budgets. On kc1, agreement rises from 39.49% at $\delta = 0.02$ to 76.15% at the default 0.05, while test accuracy stays near 86%. The temperature and sparsity curves thus change the representativeness of a single path independently of predictive accuracy. This diagnostic complements the 69-task path-budget analysis: a larger candidate tree can retain predictive accuracy while requiring more than one path to summarize its soft computation.

F Tree Capacity and Computational Cost

F.1 Matched leaf budgets

We reuse the frozen 12-task OpenML-CC18 subset of Appendix D.1, with the official repeat-0/fold-0 split and three seeds (42–44). The four models share training-fitted min–max preprocessing and maximum leaf budgets of 4, 8, and 16, giving 432 completed fits. SPECTRA has fixed depth two and 500 epochs, with only its branch budget changed. CART uses a maximum-leaf constraint; GradTree uses one tree at depths 2–4 and its final-fit training-input early-stop monitor; MurTree uses matching depth limits, at most 3, 7, or 15 internal nodes, training-derived binary conditions, and a 60-second solve limit. MurTree is consequently a bounded-time result, not a certified optimum. The shared splits and capacity ceilings permit a structural comparison, although optimization and MurTree's binary representation are method-specific.

Table 16 reports task-macro balanced accuracy and realized leaf counts. Figure 13 separates paired predictive differences from realized structure; both summarize the same 12 tasks after averaging seeds within each task.

Table 16: Matched-capacity single-tree comparison on 12 CC18 tasks (three-seed task-macro means). Δ is SPECTRA minus comparator (pp), with paired 95% task-bootstrap CIs (10,000 resamples). Bold marks the highest BAcc and fewest active leaves per budget, by column.

Max. leaves	Model	BAcc (%)	Active leaves	Δ BAcc pp [95% CI]
4	SPECTRA	**71.11**	**2.94**	–
	CART	68.92	4.00	+2.19 [−0.76, 5.18]
	GradTree	66.42	4.00	+4.69 [1.83, 7.69]
	MurTree	67.27	3.67	+3.83 [1.25, 6.26]
8	SPECTRA	70.46	**4.53**	–
	CART	**73.16**	8.00	−2.70 [−9.65, 2.12]
	GradTree	67.23	8.00	+3.23 [0.62, 6.00]
	MurTree	69.76	7.58	+0.70 [−2.93, 4.15]
16	SPECTRA	70.10	**7.42**	–
	CART	**73.36**	16.00	−3.26 [−9.90, 2.02]
	GradTree	69.24	16.00	+0.86 [−1.68, 3.47]
	MurTree	70.88	15.08	−0.77 [−3.76, 2.24]

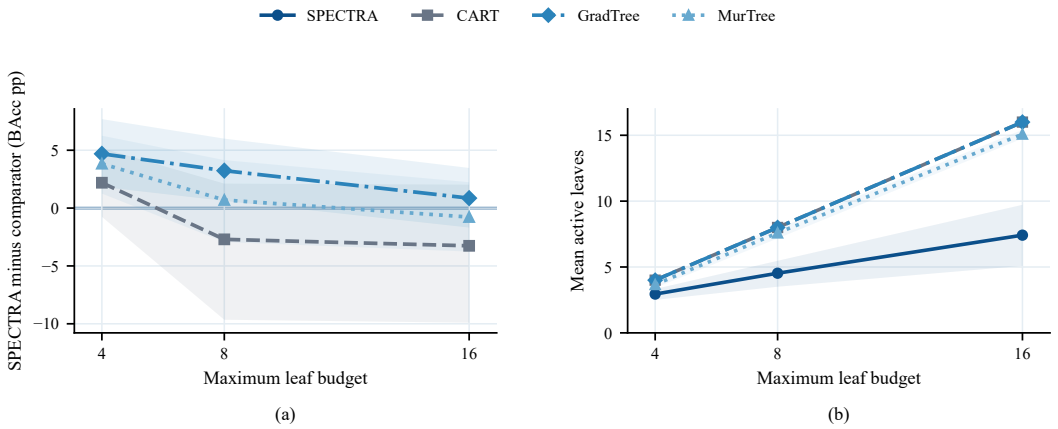


Figure 13: Matched-capacity single-tree comparison on the 12-task CC18 subset. (a) Task-macro SPECTRA-minus-comparator balanced-accuracy differences in percentage points; shaded bands are paired 95% task-bootstrap intervals. The blue zero line is the SPECTRA self-comparison, so no separate SPECTRA curve appears in this panel. (b) Mean realized active leaves at each maximum leaf budget; bands are 95% task-bootstrap intervals. Seeds are averaged within task before the 10,000 task-level resamples.

At four leaves, SPECTRA exceeds GradTree and MurTree by 4.69 and 3.83 balanced-accuracy points, with paired intervals above zero. Its 2.19-point advantage over CART has an interval crossing zero. At eight and 16 leaves, CART has higher mean balanced accuracy, while SPECTRA uses 4.53 and 7.42 active leaves on average instead of CART's 8 and 16. The evidence supports a compact-tree trade-off rather than uniform predictive dominance.

F.2 CPU EFFICIENCY AND STORED STRUCTURE

We compare the four single-tree methods used in the matched-capacity study: SPECTRA, CART, GradTree (Marton et al., 2024), and MurTree (Demirovic et al., 2022). The experiment reuses its frozen 12 numeric OpenML-CC18 tasks, official repeat-0/fold-0 splits, training-only min–max preprocessing, four-leaf ceiling, and seeds 42–44. SPECTRA retains its 500 training epochs, GradTree its 500-epoch ceiling with early stopping, and MurTree its depth-two, three-internal-node, 60-second search limit with training-derived binary conditions. All 144 fits completed. The resource audit refits every method on the same Intel Xeon Gold 5218 host with each fresh process pinned to one CPU

core and GPU access disabled. These CPU fits are separate from the matched-capacity study, which fitted SPECTRA on GPU; consequently, this section uses the accuracy of the same CPU fits whose costs it measures.

The two differentiable models admit a trainable-scalar count, whereas the hard trees are more naturally described by realized topology. For binary classification with D processed features and a four-leaf ceiling, Eq. (18) gives $8D + 13$ stored SPECTRA scalars. The released depth-two GradTree block stores two $3 \times D$ split arrays and four leaf values, giving $6D + 4$ trainable scalars. A gate can make a SPECTRA leaf unreachable without removing its stored prototype, so parameter count and active leaf count answer different questions. CART and MurTree store discrete feature choices, thresholds, and child links; assigning them a neural-network-style trainable-scalar or multiply–accumulate count would not give a common computational primitive. Soft SPECTRA inference evaluates all candidate branches, costing $O(KD + KC)$ for K leaves and C classes, while a hard tree follows at most its depth in branch tests. We therefore compare observed time, memory, predictive quality, and realized structure rather than a cross-family MAC count.

Table 17 reports task-macro means after averaging three seeds within each task. Training time excludes shared data loading and min–max preprocessing but includes MurTree's method-specific binary-condition construction, recorded separately in the raw results. Inference timing uses five warm-up calls followed by 30 complete-test-set calls, or 20 warm-ups followed by 200 single-record calls; each run contributes its median. Peak resident memory is the entire fresh process, including the common scientific runtime, fitting, and inference, rather than model-weight memory alone.

Table 17: Four-leaf single-tree efficiency on 12 CC18 tasks and three seeds. Balanced accuracy (BAcc) is higher-better; times and peak process memory are lower-better. Batch time is normalized to 1,000 test examples. All entries are task-macro means of the same CPU refits.

Model	BAcc (%)	Train (s)	Batch (ms/1k)	Single (μs)	Peak RSS (MiB)
SPECTRA	70.30	6.072	4.90	318.8	746.6
CART	68.92	0.011	1.07	113.5	619.3
GradTree	66.42	19.600	799.86	92,120.7	1,170.1
MurTree	67.27	0.036	6.69	12.0	620.8

F.3 Training budget and pruning

We varied SPECTRA's training budget to determine when predictive quality and gate-induced sparsity emerge. The 50-, 100-, 200-, and 500-epoch fits use the same 12 tasks, official splits, four-leaf ceiling, seeds, one-core CPU setting, and full model implementation as Table 17; only the epoch count changes. The 500-epoch predictions and active-leaf counts exactly reproduce all 36 corresponding resource-audit fits. Table 18 and Figure 14 report the complete sweep. At 200 epochs, mean training time falls from 6.11 to 3.28 s, while balanced accuracy changes by -0.55 percentage points (paired 95% task-bootstrap interval $[-1.40, 0.11]$). The mean active-leaf count rises from 2.97 to 3.47, showing that the later optimization also prunes the displayed tree. The 100- and 50-epoch variants incur larger accuracy losses. These are training-budget variants, not replacements for the 500-epoch main model.

Table 18: SPECTRA training-budget trade-off on the same 12 tasks and three seeds. Values are task-macro means after seed averaging; ΔBAcc is paired to the 500-epoch fit. Lower training and batch times are better, whereas fewer active leaves indicate a smaller displayed tree.

Epochs	BAcc (%)	ΔBAcc (pp)	Train (s)	Batch (ms/1k)	Active leaves
50	63.89	-6.41	1.90	4.92	4.00
100	68.04	-2.26	2.35	4.92	3.75
200	69.75	-0.55	3.28	4.88	3.47
500	70.30	0.00	6.11	4.94	2.97

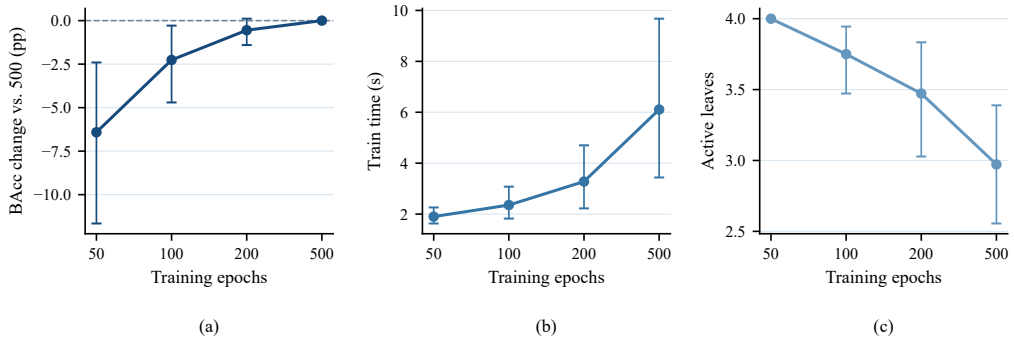


Figure 14: Training-budget sensitivity of SPECTRA on the frozen 12-task CC18 subset. (a) Paired balanced-accuracy difference from 500 epochs; (b) CPU training time; (c) active leaves under the four-leaf ceiling. Points are task-macro means over three seeds and capped bars are 95% task-bootstrap intervals. Only the epoch budget varies; the 500-epoch setting remains the model used in the four-method comparison.