TT-SPARSE: Learning Sparse Rule Models with Differentiable Truth Tables

Anonymous Authors[1]

Abstract

Interpretable machine learning is essential in high-stakes domains where decision-making requires accountability, transparency, and trust. While rule-based models offer global and exact interpretability, learning rule sets that simultaneously achieve high predictive performance and low, human-understandable complexity remains challenging. To address this, we introduce TT-SPARSE, a flexible neural building block that leverages differentiable truth tables as nodes to learn sparse, effective connections. A key contribution of our approach is a new soft TOPK operator with straight-through estimation for learning discrete, cardinality-constrained feature selection in an end-to-end differentiable manner. Crucially, the forward pass remains sparse, enabling efficient computation and exact symbolic rule extraction. As a result, each node (and the entire model) can be transformed exactly into compact, globally interpretable DNF/CNF Boolean formulas via Quine–McCluskey minimization. Extensive empirical results across 28 datasets spanning binary, multiclass, and regression tasks show that the learned sparse rules exhibit superior predictive performance with lower complexity compared to existing state-of-the-art methods.

1. Introduction

In high-stakes deployments including healthcare, finance, public policy, and safety-critical engineering, interpretability is often a functional requirement, enabling accountability and auditability (Doshi-Velez & Kim, 2017; Seshia et al., 2022). In these regimes, post-hoc explainability methods that attempt to rationalize a black-box predictor can be brittle: the explanation may be misaligned with the true decision process (Slack et al., 2020; Adebayo et al., 2018).

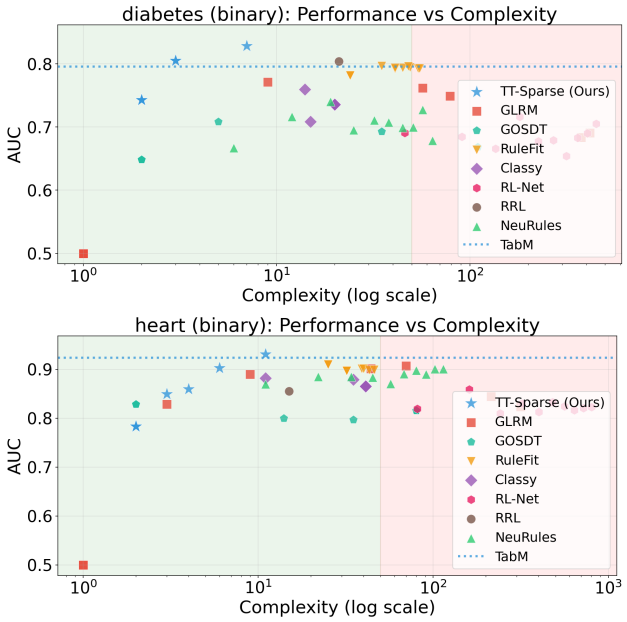


Figure 1. AUC score against rule log-complexity scatter plot of 8 interpretable models and TabM as SOTA black-box baseline across different hyperparameters, on diabetes and heart tabular datasets.

Rudin (2019) argues that, when possible, one should prefer *inherently interpretable* models over explanations of opaque ones, where transparency is a property of the model itself. Rule-based predictors are a natural target for such fundamental interpretability. However, interpretability is multifaceted. First, *global* interpretability means that a single set of rules governs the model's behavior for all inputs; this contrasts with *local* explanations that provide input-dependent rationales. Second, *exact* interpretability means that the extracted rules reproduce the model's inference exactly, rather than approximately (e.g., via feature-attribution scores). In this work, we focus on the demanding but practically valuable regime of **global and exact** interpretability, where the entire model's inference can be reduced to symbolic logic without approximation.

Crucially, interpretability does not guarantee human understandability. Human-subject evaluations demonstrate that cognitive load scales non-linearly; specifically, studies suggest that rule sets exceeding a complexity of 50 become functionally unintelligible (Lage et al., 2019). Hence, our work focuses on producing rule sets that minimize *rule*

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

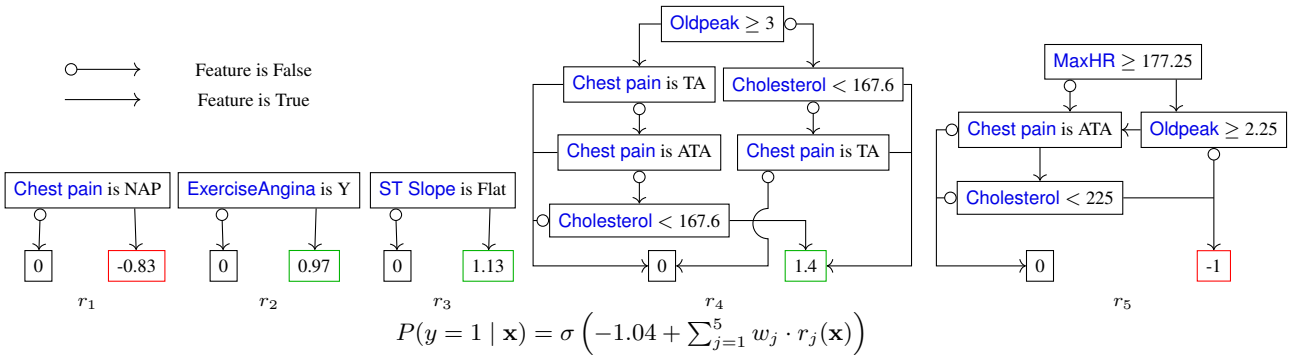


Figure 2. A TT-SPARSE model trained on Heart dataset converted to Boolean decision trees, achieving 91% test ROC-AUC score and complexity of 15. The sigmoid $\sigma(\cdot)$ function is applied to the final activated weights + intercept term to obtain the probability of heart disease existence between $[0, 1]$.

complexity while maintaining competitive predictive performance. Figure 1 showcases this: our model TT-SPARSE achieves superior predictive performance while maintaining low rule complexity (well within the complexity threshold for human intelligibility of 50 indicated by the green region), outperforming other rule models and remaining competitive with the state-of-the-art deep tabular model, TabM (Gorishniy et al., 2025).

Despite their appeal, learning high-performing rule sets while minimizing complexity remains challenging. Classical approaches often rely on discrete, heuristic search (e.g., greedy rule induction or tree growth/pruning), which can become trapped in suboptimal solutions and may struggle to exploit modern hardware efficiently (Cohen, 1995; Breiman et al., 1984). More recent lines of work pursue global optimization objectives or neural-symbolic relaxations that enable gradient-based training, but they impose restrictive logical forms that limit expressivity (Petersen et al., 2022a; 2024).

Truth table-based nodes offer a direct route to exact symbolic conversion via standard minimization procedures such as Quine-McCluskey (Benamira et al., 2023; 2024). Crucially, truth table-based nodes provide high expressivity because they are capable of representing any Boolean function over their inputs, effectively acting as the discrete counterpart to standard neural network (NN) nodes. Unlike neural-symbolic relaxations that approximate discrete logic through continuous functions, truth table nodes function as fundamental building blocks that directly map input patterns to outputs. However, making such nodes trainable at scale requires solving a key bottleneck: selecting a sparse, meaningful subset of inputs for each node in a way that remains compatible with backpropagation.

We introduce **TT-SPARSE**, a differentiable rule-learning architecture built around *Learnable Truth Table (LTT)* nodes that can be transformed exactly into Boolean formulas (DNF/CNF) after training. Each LTT node learns a Boolean function over a small set of selected input features; the entire model is then a composition of these rule activations with a lightweight prediction head. The central technical challenge is that choosing which k inputs feed each node is a discrete TOPK operation, which is non-differentiable. TT-SPARSE resolves this with a novel **soft TOPK** operator that provides a continuous relaxation while enforcing an exact cardinality constraint in expectation, enabling stable gradient flow through the connection-selection mechanism. Concretely, we use a straight-through strategy where the forward pass uses hard TOPK selections to preserve sparsity, while the backward pass uses the differentiable relaxation to update connection scores. After training, we enumerate each node's induced truth table and apply Quine-McCluskey (Quine, 1952; McCluskey, 1956) minimization to obtain compact symbolic rules, yielding global and exact interpretability.

Empirically, we evaluate TT-SPARSE along the Pareto frontier of performance versus complexity. Our experiments show that TT-SPARSE consistently achieves favorable trade-offs, matching or approaching the predictive performance of strong baselines while producing substantially more compact rule sets. Across a broad suite of tabular benchmarks spanning binary and multiclass classification as well as regression, TT-SPARSE consistently produces competitive accuracy with markedly reduced rule complexity compared to a range of interpretable baselines, while remaining competitive to the SOTA tabular architecture.

Contributions. Hence, the main contributions of this work include: (1) We propose a novel, efficient, and fully **differentiable relaxation of the discrete TOPK operator**, enabling end-to-end gradient-based optimization of discrete feature routing via backpropagation. (2) We introduce the **TT-SPARSE layer**, a versatile neural building block that leverages our Soft TOPK operator to dynamically learn sparse connections and extract higher-order Boolean interactions. This block is inherently flexible, allowing for seamless adaptation across diverse tasks. (3) We provide a **rule**

extraction pipeline that converts trained LTT nodes to compact DNF/CNF rules via truth table enumeration and Quine-McCluskey minimization, leveraging data-driven don't-care terms to reduce redundancy. We illustrate such exact rule extraction with no approximation in Figure 3. (4) Through extensive empirical validation on 28 datasets, we demonstrate that TT-SPARSE establishes a new **Pareto frontier in the performance-complexity landscape** while achieving predictive accuracy competitive with SOTA non-pretrained tabular model, TabM.

2. Related Work

In this section, we provide an overview of the literature on methods that achieve global and exact interpretability, as well as relaxations of the discrete TOPK function. We evaluate these approaches with TT-SPARSE in Section 4.

Tree-Based Approaches. While classical decision trees (Breiman et al., 1984) offer an intuitive approach, their greedy induction strategy often leads to suboptimal rule sets. To address this, GOSDT (Lin et al., 2020) employs dynamic programming to find provably optimal sparse trees, though at a high computational cost for large datasets. Conversely, ensemble methods like LightGBM (Ke et al., 2017) achieve state-of-the-art predictive performance but sacrifice interpretability by distributing the decision logic across hundreds of weaker trees. However, the tree topology inherently restricts each root-to-leaf path to a logical conjunction (AND) of splitting predicates.

Heuristic-Guided Bottom-Up Rule Induction. RIPPER (Cohen, 1995) is a classic technique that employs a greedy strategy to grow and prune rule sets, incrementally refining them to optimize predictive accuracy. Classy (Proença & van Leeuwen, 2020) advances this approach by utilizing the Minimum Description Length (MDL) principle to search for compact probabilistic rule lists without the need for hyperparameter tuning. Other approaches, such as GLRM (Wei et al., 2019) and RuleFit (Friedman & Popescu, 2008) improve scalability by leveraging column generation or tree ensembles to identify candidate rules. However, similar to the limitation with trees, these methods remain constrained by discrete heuristic generation: they build rules by iteratively adding conjunctions (ANDs), lacking the native flexibility to learn complex, nested Boolean structures (e.g., ORs) directly within a single rule unit.

Neuro-Symbolic Approaches. Neuro-symbolic models aim to combine the generalization power of neural networks with symbolic interpretability. RRL (Wang et al., 2021) learns a set of discrete rules by projecting logical structures into a differentiable space and applying gradient-based optimization. RL-Net (Dierckx et al., 2023) uses a neural architecture that mimics rule lists by enforcing a hierarchical activation structure. Most recently, NeuRules (Xu et al., 2025) integrate feature discretization, rule construction, and rule ordering into a single end-to-end differentiable pipeline.

Differentiable TOPK. Prior differentiable TopK approaches rely on iterative Optimal Transport solvers with the Sinkhorn algorithm (Xie et al., 2020; Cuturi, 2013) or stochastic noise injection (Cordonnier et al., 2021; Berthet et al., 2020), which introduces significant computational overhead or gradient variance. Most recently, Petersen et al. (2022b) used these novel differentiable TOPK operators to define a top-k cross-entropy loss, enabling the direct end-to-end optimization of top-k classification accuracy, achieving state-of-the-art fine-tuning results on ImageNet.

3. TT-SPARSE

In this section, we introduce TT-SPARSE, a **T**ruth **T**able-based **Sparse** neural block. Each node in the TT-SPARSE layer can be understood as a learnable truth table logic over its inputs, we call them Learnable Truth Table (LTT) nodes interchangeably. This block is the core interpretable unit that transforms input features into Boolean rules. Let $\vec{x}$ be the n-dimensional input vector to the layer, and there are M LTT nodes in the layer.

There are three aspects of a TT-SPARSE layer: the connection selection, the linear operation of the selected inputs, and the binarization. The connection selection mechanism leverages the soft TOPK operator we introduce to enable learning by backpropagation of the loss by relaxing the discrete, non-differentiable selection of the traditional TOPK.

3.1. Components

Soft TOPK. First, we define the space of operators

$$S_k : \mathbb{R}^n \to \mathcal{P}_k \subset [0,1]^n$$

where $\vec{x} \in \mathbb{R}^n$ denotes the input vector. This space is constrained by $\sum_{i=1}^n S_k(\vec{x})_i = k$.

The hard TOPK operator can be viewed as finding the vector y within the feasible set that maximizes $x^\top y$, achieved by assigning the components of y corresponding to the k largest values of x to 1, while setting the rest to 0. Since this discrete selection is not differentiable, we introduce an entropic regularizer to find a probability-like vector $\vec{y} \in [0,1]^n$ that is closest to the logits $\vec{x}$ (maximizing the dot product) while maintaining smoothness and summing exactly to k. The entropic regularizer is weighted by a predetermined parameter temperature $\tau > 0$. This leads us to the optimization objective

$$\vec{y}^* = \underset{\vec{y}\in[0,1]^n}{\text{argmax}} \left[\sum_{i=1}^n y_i x_i + \tau \sum_{i=1}^n H(y_i)\right]$$

subject to $\sum_{i=1}^n y_i = k$, where $H(y_i)$ is the binary entropy

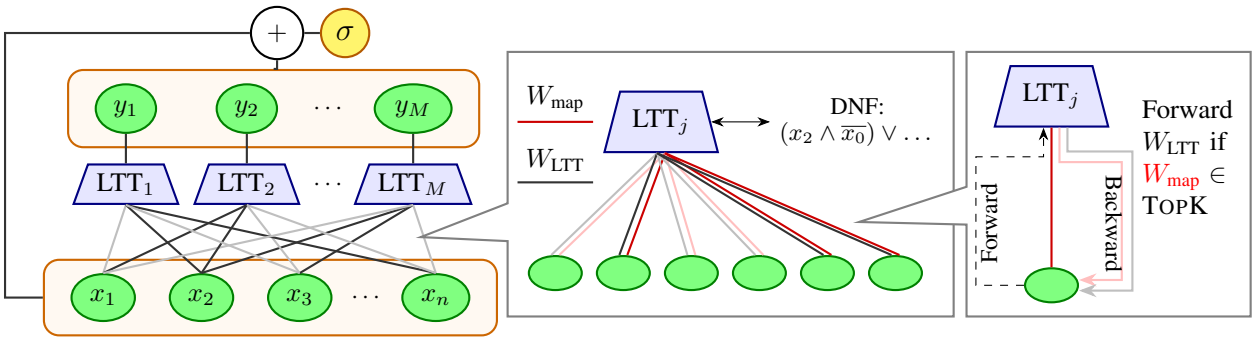


Figure 3. **Overview of the TT-SPARSE Architecture. (Left)** The hybrid model structure. The input vector is processed by a layer of Learnable Truth Table (LTT) nodes (blue trapezoids), which extract higher-order Boolean rules. These outputs are concatenated with the raw input features to form the final prediction. **(Middle)** Each potential connection is parameterized by the logic weight W_{LTT} and the mapping weight W_{map}. Darker lines highlight the "active" connections, those where W_{map} belongs to the subset of the k-highest mapping weights for that node. Each LTT node is convertible to an equivalent CNF/DNF equation (See Figure 4). **(Right)** In the forward pass, the input is fed forward and multiplied by W_{LTT} if the corresponding W_{map} is part of TOPK. In the backward pass, gradients flow with the Soft TOPK relaxation, updating both W_{map} and W_{LTT}. This design enables exact rule extraction while preserving gradient-based training through discrete connection selection.

$H(y_i) = -y_i \log y_i - (1-y_i)\log(1-y_i)$. Solving this optimization via Lagrangian multipliers (see Appendix C), we obtain the solution $S_k(\vec{x}) := \vec{y}$ where $y_i := \sigma(\frac{x_i}{\tau} + c)$ for $i \in \{1, \dots, n\}$ and $c \in \mathbb{R}$ is the unique root of $f(c) = \sum_{j=1}^{n} \sigma(\frac{x_i}{\tau} + c) - k = 0$. Since $f(c)$ is strictly increasing, c is uniquely determined and solved by the bisection method.

This formulation allows for the derivation of exact gradients with implicit differentiation. Let $s'_i = \sigma'(\frac{x_i}{\tau} + c) = y_i(1 - y_i)$ and $s' = [s_1, \dots, s_n]^\top$.

$$\frac{\partial y_i}{\partial x_j} = s'_i \cdot \frac{\partial}{\partial x_j}\left(\frac{x_i}{\tau} + c\right) = \frac{1}{\tau} s'_i \left(\delta_{ij} - \frac{s'_j}{\|s'\|_1}\right) \quad (1)$$

$$\nabla_{\vec{x}} S_k(\vec{x}) = \frac{1}{\tau}\left[\text{diag}(s') - \frac{s'(s'^\top)}{\|s'\|_1}\right] \quad (2)$$

$$\frac{\partial y_i}{\partial k} = s'_i \cdot \frac{\partial}{\partial k}\left(\frac{x_i}{\tau} + c\right) = \frac{s'_i}{\|s'\|_1} \quad (3)$$

$$\nabla_k S_k(\vec{x}) = \frac{s'}{\|s'\|_1} \quad (4)$$

The complete derivation of the partials can be found in Section C.2. Notice that the gradient is scaled by the inverse temperature $\frac{1}{\tau}$, similar to the effect of temperature in the softmax function when approximating the Heaviside step function. As $\tau \to 0$, the relaxation gets closer to the hard, discrete counterpart. The effect of temperature τ on the distribution of probabilities is illustrated in Figure 6.

In the implementation of the bisection search of c, it takes around 25 halving iterations before reaching machine precision limits of `float32` ($\epsilon \approx 10^{-6}$), taking around 1 ms to complete for a batch of 16 input vectors. See Section C.4 for visualization of the convergence of the bisection method.

LTT Nodes. Let $\vec{x} \in \mathbb{R}^n$ denote the input to the TT-SPARSE layer with M LTT nodes. The block has 2 learnable modules: the connection mapping represented by a matrix $W_{\text{map}} \in \mathbb{R}^{n \times M}$ and the truth table logic represented by matrix $W_{\text{LTT}} \in \mathbb{R}^{n \times M}$ (augmented by a bias vector $b \in \mathbb{R}^M$).

For each LTT node $j \in \{1, \dots, M\}$, the objective is to select a sparse subset of k input features to form a truth table of size 2^k and learn effective truth table logic over the features. To determine the active inputs for node j, we define a selection mask vector $m^{(j)} \in \{0, 1\}^n$.

In the forward pass, this mask is discrete. It selects the indices corresponding to the top-k values in the j-th column of the mapping matrix:

$$m^{(j)}_{\text{forward},i} = \begin{cases} 1 & \text{if } W_{\text{map}}[i, j] \in \text{TopK}(W_{\text{map}}[:, j]) \\ 0 & \text{otherwise} \end{cases}$$

for $i \in [1, \dots, n]$.

The output of each LTT node is the linear combination of the active features

$$z_j(\vec{x}) = \sum_{i=1}^{n} m^{(j)}_i \cdot W_{\text{LTT}}[i, j] \cdot x_i + b_j \quad (5)$$

However, the traditional TOPK operation is discrete and not differentiable, so without relaxation, the connection matrix will not update to find better connections for the LTT logic. Thus, we leverage the soft TOPK operator S_k we propose in 3.1 to replace $m^{(j)}$ in the backward pass to:

$$m^{(j)}_{\text{backward}} = S_k(W_{\text{map}}[\cdot, j])$$

Consequently, the partial derivatives w.r.t. the connection weights are computed via the chain rule. Let $\delta_j = \frac{\partial \mathcal{L}}{\partial z_j}$ be

the gradient of the loss function $\mathcal{L}$ w.r.t. the node output z_j. The gradient flow to the connection matrix is given by

$$\frac{\partial \mathcal{L}}{\partial W_{\text{map}}[\cdot, j]} = \frac{\partial \mathcal{L}}{\partial z_j} \cdot \frac{\partial z_j}{\partial m^{(j)}_{\text{backward}}} \cdot \frac{\partial S_k(W_{\text{map}}[\cdot, j])}{\partial W_{\text{map}}[\cdot, j]} \quad (6)$$

The term $\frac{\partial z_j}{\partial m^{(j)}_{\text{backward}}}$ is an n-dimensional vector representing the potential contribution of each feature if selected, given by

$$\frac{\partial z_j}{\partial m^{(j)}_{\text{backward}}} = \vec{x} \odot W_{\text{LTT}}[\cdot, j]$$

The term $\frac{\partial S_k(W_{\text{map}}[\cdot,j])}{\partial W_{\text{map}}[\cdot,j]}$ is the Jacobian of the Soft TOPK operator, $J_{S_k} \in \mathbb{R}^{n \times n}$ given by equation (2). The full derivation of the gradient is provided in Appendix D. This mechanism allows the TT-SPARSE layer to dynamically re-route connections into the LTT node while learning the LTT logic during training.

Finally, to function as a Boolean logic gate, the continuous output z_j is binarized. Differentiability is maintained with a straight-through estimator (STE): $y_j = \mathbf{1}_{(z_j > 0)} + (z_j - \text{stop\_gradient}(z_j))$.

3.2. TT-SPARSE Design

The architecture of the overall model is designed as a hybrid neural block that integrates high-order feature interaction logic through the TT-SPARSE layer with linear single-order features.

The layer processes an n-dimensional input vector $\vec{x}$ to produce M binary rule activations $\vec{z} \in \{0,1\}^M$ via the LTT nodes. Then, we implement a skip-concatenation, so the final representation $\vec{h}$ is formed by concatenating the raw input features with the LTT rule outputs: $\vec{h} = [\vec{x} \parallel \vec{z}] \in \mathbb{R}^{n+M}$. This combined vector is then fed into a final classifier or regressor layer $f_{\text{cls}}(\vec{h}) = \sigma(W_{\text{cls}}\vec{h} + b_{\text{cls}})$, depending on the task (binary/multiclass/regression). The final linear classifier layer assigns weights to both the higher-order and single-order rules, before the task-specific activation function.

While the TT-SPARSE layer selects several inputs per node, not all connections or all nodes may be essential for a given task. To identify the most compact and effective rule set, we implement a post-hoc iterative magnitude pruning after the initial training phase. Connections in the TT-SPARSE layer are removed based on the weights' L_1 norm, followed by an iterative fine-tuning of the non-zeroed weights. Throughout this refinement process, the selected connections of the LTT nodes remain fixed to the selected indices by W_{map} during training. This refinement phase focuses on optimizing the weights of the established logical structure, finding the lightest and most effective rule logic for the given task.

3.3. Rule Extraction

Each node j in the TT-SPARSE layer represents a learnable Boolean logic of its selected inputs. Let $\mathcal{I}_j \subset \{1, \ldots, n\}$ be the selected indices of the input features by the TOPK operator on W_{map}, where $|\mathcal{I}_j| = k$. Recall that the output of the node $z_j(\vec{x})$ is determined by the thresholded linear combination of the inputs as in Equation (5).

To extract the explicit Boolean formula of the input features for node j, we perform a simple truth table enumeration. All 2^k possible binary inputs are efficiently iterated and for each input, the node's output activation is computed, obtaining the truth table $\mathcal{T}_j$ for node j

$$\mathcal{T}_j = \{(v_i, z_j(v_i)) | v_i \in \{0,1\}^k\}_{i=1}^{2^k}$$

where v_i denotes the i-th binary vector in the enumeration of the input space. The subset of the truth table where the output is 1, i.e. $\mathcal{M}_j = \{v_i \in \{0,1\}^k \mid z_j(v_i) = 1\}$, are the *minterms* of the Boolean function.

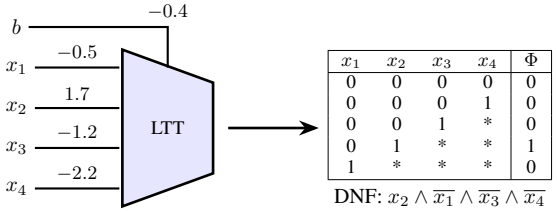


Figure 4. Conversion of an LTT node to DNF by truth table enumeration of 2^n input combinations and obtaining the binary outputs with the LTT weight and bias parameters.

To minimize rule complexity, we identify *Don't-Care Terms* (DCTs)—input combinations (v_i) that do not appear in the training data. Following the identification of minterms and DCTs, the truth table is synthesized into a CNF or DNF expression via the Quine-McCluskey (QMC) algorithm (Quine, 1952; McCluskey, 1956) (Algorithm 1). By optimally assigning binary values to these DCTs, the algorithm minimizes the resulting rule complexity while maintaining exact fidelity on the observed training distribution.

By modeling the node truth table logic as a thresholded linear combination of the selected inputs, TT-SPARSE requires only $n + 1$ learnable parameters (one weight per selected input plus a bias term) to represent the logic, a linear complexity compared to DiffLogicNet (Petersen et al., 2022a) double-exponential and DWN (Bacellar et al., 2024) exponential complexities.

DiffLogicNet introduces a differentiable relaxation of the Boolean logic to implement binary gates as nodes, by learning a probability distribution over the set of all possible binary operators for a given number of inputs. While effective for 2-input gates, this approach scales poorly as the number of possible Boolean functions grows double exponentially

Table 1. Binary Classification Results: AUC ↑ (top) and Complexity ↓ (bottom) for each dataset. Standard error in subscript. Best values in **bold**. The last row reports the avg. rank ↓ of each model's AUC across all datasets, as calculated for the Friedman statistical test. *OOM* indicates out-of-memory during training.

Dataset	TT-SPARSE	GOSDT	GLRM	RuleFit	Classy	NeuRules	RL-Net	RRL	DWN	TabM
bank	**.9096**$_{.00}$ 112_{37}	.6662$_{.01}$ $\mathbf{5}_{0}$	.8928$_{.00}$ 38_{2}	.8963$_{.01}$ 58_{20}	.9042$_{.00}$ 370_{13}	.8648$_{.01}$ 902_{45}	.6917$_{.01}$ 1318_{6}	.8913$_{.01}$ 73_{7}	.8480$_{.01}$ 533_{31}	.9351$_{.00}$ -
blood	**.7554**$_{.02}$ 15_{8}	.6155$_{.04}$ 15_{5}	.7093$_{.05}$ 30_{5}	.7059$_{.05}$ 43_{7}	.6628$_{.03}$ 9_{2}	.7116$_{.02}$ 165_{3}	.5618$_{.05}$ 201_{0}	.6712$_{.04}$ $\mathbf{5}_{2}$	.6545$_{.05}$ 465_{29}	.7193$_{.04}$ -
calhousing	**.9307**$_{.00}$ 62_{11}	.8111$_{.01}$ $\mathbf{25}_{0}$	.9205$_{.01}$ 108_{7}	.9259$_{.00}$ 131_{46}	.9072$_{.01}$ 457_{17}	.8804$_{.01}$ 229_{8}	.8004$_{.03}$ 88_{1}	.8684$_{.01}$ 45_{3}	.7961$_{.01}$ 572_{21}	.9667$_{.00}$ -
compas	**.7301**$_{.01}$ $\mathbf{9}_{2}$	.6765$_{.01}$ 13_{0}	.7236$_{.01}$ 38_{4}	.7270$_{.01}$ 31_{16}	.7080$_{.01}$ 39_{2}	.7129$_{.01}$ 280_{6}	.6851$_{.02}$ 961_{1}	.6825$_{.01}$ 17_{5}	.6629$_{.01}$ 1358_{28}	.7311$_{.01}$ -
covertype	.8554$_{.00}$ 194_{31}	*OOM*	.8437$_{.00}$ 49_{1}	.8465$_{.00}$ $\mathbf{39}_{6}$	**.8913**$_{.00}$ 2183_{102}	.8320$_{.00}$ 284_{28}	.7672$_{.01}$ 194_{16}	.8522$_{.00}$ 325_{8}	.8002$_{.00}$ 464_{16}	.9801$_{.00}$ -
cc_default	**.7915**$_{.01}$ 128_{19}	.7011$_{.01}$ $\mathbf{6}_{2}$	.7705$_{.01}$ 56_{24}	.7802$_{.00}$ 110_{41}	.7631$_{.01}$ 116_{3}	.7596$_{.01}$ 281_{20}	.7233$_{.00}$ 440_{2}	.7454$_{.02}$ 100_{20}	.6601$_{.02}$ 445_{18}	.7883$_{.01}$ -
creditg	.7919$_{.04}$ 54_{36}	.5788$_{.02}$ 27_{6}	.6851$_{.04}$ 56_{5}	**.7948**$_{.04}$ 97_{24}	.6768$_{.02}$ $\mathbf{13}_{2}$	.7313$_{.05}$ 781_{148}	.6625$_{.04}$ 246_{0}	.6745$_{.05}$ 35_{8}	.7234$_{.07}$ 2444_{36}	.7329$_{.03}$ -
diabetes	**.8208**$_{.02}$ 14_{9}	.6443$_{.03}$ $\mathbf{5}_{2}$	.7796$_{.02}$ 87_{10}	.8024$_{.01}$ 19_{7}	.7260$_{.02}$ 14_{3}	.6854$_{.05}$ 18_{1}	.6442$_{.02}$ 91_{0}	.7609$_{.03}$ 24_{7}	.7186$_{.04}$ 588_{26}	.7957$_{.01}$ -
electricity	.8823$_{.00}$ 99_{29}	.7734$_{.00}$ $\mathbf{20}_{0}$	.8705$_{.00}$ 61_{3}	.8745$_{.00}$ 186_{32}	**.8895**$_{.00}$ 731_{12}	.8397$_{.01}$ 240_{6}	.7459$_{.01}$ 1121_{0}	.8387$_{.01}$ 56_{6}	.8160$_{.00}$ 2913_{96}	.9666$_{.00}$ -
eye	**.6312**$_{.02}$ 76_{43}	.5836$_{.00}$ $\mathbf{8}_{2}$	.6278$_{.01}$ 16_{2}	.6216$_{.01}$ 46_{18}	.5913$_{.02}$ 38_{6}	.5981$_{.01}$ 715_{24}	.5815$_{.01}$ 1921_{0}	.5930$_{.01}$ 64_{11}	.5711$_{.02}$ 340_{19}	.6799$_{.02}$ -
heart	**.9308**$_{.01}$ 11_{15}	.8140$_{.01}$ $\mathbf{2}_{0}$	.9149$_{.02}$ 39_{6}	.9151$_{.01}$ 34_{16}	.8749$_{.01}$ 31_{2}	.8663$_{.01}$ 100_{9}	.8301$_{.01}$ 641_{0}	.8661$_{.03}$ 29_{10}	.8719$_{.02}$ 474_{20}	.9236$_{.02}$ -
income	**.9147**$_{.00}$ 152_{40}	.7341$_{.01}$ $\mathbf{11}_{2}$	.8958$_{.00}$ 28_{2}	.9121$_{.00}$ 155_{47}	.8799$_{.00}$ 684_{22}	.8800$_{.01}$ 413_{58}	.7707$_{.02}$ 4345_{1351}	.8962$_{.01}$ 307_{18}	.8642$_{.01}$ 1045_{23}	.9230$_{.00}$ -
jungle	.8917$_{.00}$ 262_{75}	.7804$_{.01}$ $\mathbf{27}_{4}$	.8904$_{.00}$ 51_{4}	.8630$_{.01}$ 103_{48}	**.9476**$_{.00}$ 891_{25}	.8702$_{.01}$ 163_{1}	.7685$_{.03}$ 421_{0}	.8781$_{.01}$ 28_{4}	.7900$_{.01}$ 440_{25}	.9970$_{.00}$ -
road_safety	.7764$_{.00}$ 236_{54}	.7197$_{.01}$ 15_{8}	.7603$_{.00}$ 24_{1}	OOM	**.8435**$_{.00}$ 1348_{33}	.7842$_{.01}$ 186_{9}	.7465$_{.01}$ 1948_{38}	.8293$_{.01}$ 119_{10}	.7295$_{.02}$ 425_{13}	.8852$_{.01}$ -
Avg. rank ↓	**1.53**	8.27	3.6	3.07	3.8	4.8	7.93	4.87	7.13	

with the number of inputs, (2^{2^n}). The DWN model addresses this scalability bottleneck by parameterizing nodes as Lookup Tables (LUTs) and introducing an Extended Finite Difference method to estimate gradients. This reduces the parameter complexity from 2^{2^n} to 2^n, the size of the lookup table itself. However, by parameterizing a regular lookup table, a standard LUT is strictly position-dependent; the meaning of its weights is tied to the specific ordering of its address. If the input selection mechanism permutes the selected features (e.g. swapping index i with index j), the LUT weights do not accommodate this shift, causing performance inefficiency. In contrast, the TT-SPARSE node uses a linear combination of the inputs, which is commutative, making the node robust to input reordering during the connection search phase.

Pruning a single input connection from a dense LUT is also non-trivial. To make a LUT invariant to a specific input bit, one must constrain half of the table's parameters to be identical to the other half (based on Shannon expansion). In the TT-SPARSE formulation, removal of a redundant input connection can simply be done by pushing the corresponding linear weight to zero via L_1 norm.

We validate these theoretical advantages by replacing the TT-SPARSE block in our model with the DWN block in our experimental evaluation (Tables 1-3).

4. Experiments

In this section, we validate the efficacy of TT-SPARSE across the 3 main tabular tasks: binary and multiclass classification, and regression. Our evaluation aims to answer 2 fundamental questions: (1) Can TT-SPARSE generate rule sets that are Pareto-competitive, offering a better trade-off between predictive performance and complexity compared to existing interpretable models? (2) How does the performance of TT-SPARSE compare to TABM (Gorishniy et al., 2025), the current state-of-the-art non-pretrained deep learning architecture for tabular data? The code for this project is publicly available at [1].

Datasets and Tasks. We conduct experiments on 28 datasets (14 binary, 7 multiclass, 7 regression). We evaluate the Area Under the Receiver Operating Characteristic Curve

[1] https://anonymous.4open.science/r/sparse-CF97

Table 2. Multiclass Classification Results: AUC ↑ (top) and Complexity ↓ (bottom) for each dataset. Standard error in subscript. Best values in **bold**. The last row reports the avg. rank ↓ of each model's AUC across all datasets, as calculated for the Friedman statistical test.

Dataset	TT-SPARSE	GOSDT	Classy	NeuRules	RL-Net	DWN	TabM
car	**.9901**$_{.00}$ 252_{110}	.6636$_{.03}$ **41**$_{12}$	.9727$_{.01}$ 136_{6}	.8930$_{.02}$ 227_{11}	.8631$_{.03}$ 73_{6}	.6967$_{.06}$ 2445_{63}	1.0000$_{.00}$ -
ecoli	**.9663**$_{.01}$ 148_{77}	.8315$_{.05}$ **16**$_{4}$	.8968$_{.02}$ 34_{4}	.9340$_{.01}$ 201_{3}	.7551$_{.05}$ 36_{0}	.8841$_{.02}$ 1358_{3}	.9629$_{.01}$ -
iris	**.9993**$_{.00}$ 26_{17}	.9500$_{.02}$ **7**$_{2}$	.9633$_{.02}$ 12_{0}	.9883$_{.01}$ 25_{1}	.9040$_{.05}$ 301_{0}	.8090$_{.04}$ 2406_{68}	.9963$_{.01}$ -
penguins	**1.0000**$_{.00}$ **6**$_{1}$	.9519$_{.03}$ 15_{2}	.9795$_{.02}$ 19_{3}	.9893$_{.01}$ 212_{8}	.9551$_{.02}$ 50_{2}	.9503$_{.02}$ 1065_{55}	.9998$_{.00}$ -
satimage	**.9802**$_{.00}$ 157_{18}	.8408$_{.01}$ **17**$_{3}$	.9587$_{.00}$ 408_{12}	.9488$_{.01}$ 378_{9}	.8094$_{.03}$ 740_{2}	.8652$_{.02}$ 554_{26}	.9925$_{.00}$ -
wine	**1.0000**$_{.00}$ **7**$_{2}$	.9592$_{.01}$ 16_{3}	.9430$_{.03}$ 13_{0}	.9807$_{.02}$ 95_{6}	.9543$_{.06}$ 1121_{0}	.9528$_{.01}$ 795_{21}	1.0000$_{.00}$ -
yeast	**.8457**$_{.02}$ 77_{14}	.6767$_{.02}$ **37**$_{6}$	.7809$_{.02}$ 135_{11}	.7787$_{.01}$ 307_{8}	.7082$_{.03}$ 42_{5}	.6721$_{.04}$ 3050_{41}	.8564$_{.02}$ -
Avg. rank ↓	**1**	4.71	3	2.43	4.71	5.14	

(**ROC-AUC**) score for the binary and multiclass (One-vs-Rest) classification to judge predictive performance while handling class imbalance. For regression tasks, we evaluate the $\mathbf{R^2}$ score.

Baselines. We compare TT-SPARSE against the interpretable baselines we discussed on the Related Work (section 2): **GOSDT** (Lin et al., 2020), **GLRM** (Wei et al., 2019), **RuleFit** (Friedman & Popescu, 2008), **Classy** (Proença & van Leeuwen, 2020), **NeuRules** (Xu et al., 2025), **RL-Net** (Dierckx et al., 2023), **RRL** (Wang et al., 2021), and **DWN** (Bacellar et al., 2024).

Our objective is to maximize predictive performance (AUC/R^2) while minimizing rule complexity. The **complexity** of a rule set is simply defined as the sum of the number of rules and the number of Boolean operators in all the rules. Crucially, for multiclass tasks, we calculate a single global complexity score: a rule or feature contributes to the complexity count exactly once if it holds a non-zero weight for any target class, ensuring that logic shared across classes is not penalized multiple times. Refer to Appendix F for examples of rules generated by each interpretable model and its complexity. Because the output of a rule-based model depends on its hyperparameters, a single performance number does not provide a complete picture. For comprehensive evaluation, we visualize the performance complexity Pareto frontier for every model on every dataset (see Appendix E). The results presented in Tables 1, 2, and 3 report the performance of the model configuration that achieves the best performance-complexity of this Pareto grid. Due to architectural limitations, GLRM and RuleFit cannot handle multiclass classification, and the remaining models are limited to classification.

For all binary classification datasets except covertype, jungle, and road_safety (Table 1), TT-SPARSE consistently pushes the Pareto limit, achieving comparable or better performance with lower complexity. While GOSDT often yields the lowest complexity rules with sparse decision trees, it frequently suffers from performance issues. For some datasets (blood, compas, cc_default, creditg, diabetes, heart, income), TT-SPARSE remains competitive with the SOTA DL baseline TabM, notably surpassing it on the Heart dataset with low complexity.

Multiclass classification (Table 2) highlights the strongest advantage of our approach. With GLRM and RuleFit unable to operate in this setting, TT-SPARSE comfortably outperforms the other models in all datasets as the Pareto frontier, and is very close to achieving TabM performance while maintaining full transparency. TT-SPARSE also shows

Table 3. Regression Results: R^2 ↑ (top) and Complexity ↓ (bottom) for each dataset. Standard error in subscript. Best values in **bold**. The last row reports the avg. rank ↓ of each model's R^2 across all datasets, as calculated for the Friedman statistical test.

Dataset	TT-SPARSE	GLRM	RuleFit	TabM
abalone	**.5328**$_{.01}$ 43_{16}	.4103$_{.02}$ 53_{6}	.5324$_{.01}$ **23**$_{7}$	.5494$_{.02}$ -
bike	.6956$_{.01}$ 299_{21}	.1879$_{.05}$ **50**$_{5}$	**.7380**$_{.01}$ 127_{27}	.9547$_{.00}$ -
boston	.8566$_{.01}$ 77_{41}	.6541$_{.10}$ **74**$_{14}$	**.8576**$_{.05}$ 89_{17}	.9042$_{.01}$ -
california	**.7102**$_{.01}$ 135_{30}	.5473$_{.04}$ 197_{20}	.7006$_{.01}$ **101**$_{34}$	.8432$_{.01}$ -
crime	.3647$_{.08}$ 3003_{22}	.4893$_{.04}$ **68**$_{5}$	**.6540**$_{.01}$ 79_{33}	.6513$_{.02}$ -
parkinsons	.9232$_{.01}$ 103_{90}	.8598$_{.02}$ 96_{6}	**.9270**$_{.01}$ **82**$_{15}$	.9909$_{.00}$ -
wine_reg	**.3121**$_{.02}$ 111_{46}	.2151$_{.04}$ **59**$_{8}$	.3090$_{.03}$ 85_{21}	.3629$_{.10}$ -
Avg. rank ↓	1.71	2.86	**1.43**	

strong competitiveness for the regression tasks (Table 3), effectively outperforming GLRM and underperforming RuleFit in rank by a thin margin.

4.1. Ablation Study

A core contribution of TT-SPARSE is the use of the Soft TOPK operator to select the input connections for each LTT node. To validate the necessity and efficiency of this design, we perform an ablation study comparing it against a baseline connection selection mechanism leveraging the softmax operator as a relaxation of the argmax operator, used in DWN (Bacellar et al., 2024).

Baseline. In the baseline configuration we ablate against, instead of selecting k features from a single importance vector, the k inputs to a truth table are treated as distinct, ordered "slots". Mathematically, the connection matrix for the layer is $W_{\text{map}} \in \mathbb{R}^{n \times kM}$ instead of $W_{\text{map}} \in \mathbb{R}^{n \times M}$. For each node j in the layer, the k inputs to the node are determined by taking an argmax over k columns in the matrix, and, in the backpropagation, the weights are updated by taking the relaxation of argmax with softmax.

This baseline exhibits two disadvantages relative to our proposed soft TOPK approach: (1) Parameter inefficiency, where mapping complexity scales as $\mathcal{O}(M \cdot n \cdot k)$ rather than $\mathcal{O}(M \cdot n)$; and (2) Input redundancy, where independent slots may converge on the same feature index, effectively collapsing the dimension of the truth table. Our soft TOPK operator inherently prevents this by selecting the k most relevant distinct indices from the columns of the mapping matrix.

Experimental results (Figure 5) demonstrate that TT-SPARSE (x-axis) consistently yields superior predictive performance compared to the Softmax baseline (y-axis).

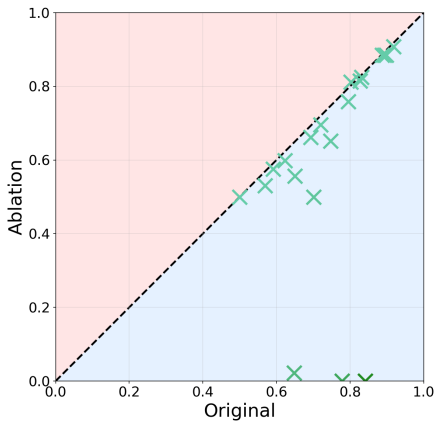


Figure 5. Predictive performance comparison (AUC/R^2, higher is better) between TT-SPARSE (Soft TOPK) and the slot-based Softmax baseline. Points below the diagonal indicate superior performance by TT-SPARSE.

5. Conclusion

In this work, we introduced TT-SPARSE, a new neural building block designed to achieve highly performant rule sets while minimizing complexity. By reformulating truth tables as differentiable modules with sparse connectivity learned through the new soft TOPK operator, we enable the end-to-end learning of Boolean logic via standard gradient descent. Furthermore, as a fully differentiable layer, TT-SPARSE offers significant architectural flexibility; it can be seamlessly integrated into standard neural pipelines to learn interpretable rules across diverse problem settings, including binary, multiclass, and regression tasks.

5.1. Limitations

Continuous features must be encoded before training, fixing the representation of the literals. Second, the utilization of the soft TOPK operator makes training reliant on the initialization of the temperature parameter τ. While these parameters must be well-initialized, our ablations in Appendix H show that TT-SPARSE is robust to this constraint, maintaining high performance after as few as 5 encoding bits.

5.2. Future Work

Future research will focus on enhancing the architectural expressivity of TT-SPARSE for broader data domains. We aim to adapt the LTT nodes for time-series analysis by introducing recurrent latent dimensions, enabling the Soft TOPK operator to sparsely select historical states and extract interpretable temporal logic. Additionally, we plan to extend the feature selection mechanism to support literals formed by learnable linear combinations of input variables (e.g. $w_1 x_1 + w_2 x_2 \leq z$) to capture complex decision boundaries, thereby reducing the Boolean complexity required to approximate non-linear functions.

6. Impact Statement

This work advances the field of interpretable machine learning by introducing a scalable framework for achieving global and exact interpretability without sacrificing predictive performance. This directly addresses the transparency requirements of high-stakes sectors such as healthcare, finance, and criminal justice, where auditability and accountability are essential. By enabling domain experts to inspect, verify, and reason about model behavior, TT-SPARSE facilitates algorithmic oversight, strengthens institutional trust, and supports informed, responsible use of machine-learning systems under emerging regulatory frameworks.

References

Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., and Kim, B. Sanity checks for saliency maps. *Advances in neural information processing systems*, 31, 2018.

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

Benamira, A., Guérand, T., Peyrin, T., and Soegeng, H. Neural network-based rule models with truth tables. In Gal, K. et al. (eds.), *ECAI 2023: 26th European Conference on Artificial Intelligence*, volume 372 of *Frontiers in Artificial Intelligence and Applications*, pp. 223–230. IOS Press, 2023. doi: 10.3233/FAIA230274. URL https://ebooks.iospress.nl/doi/10.3233/FAIA230274.

Benamira, A., Peyrin, T., Yap, T., Guérand, T., and Hooi, B. Truth table net: Scalable, compact & verifiable neural networks with a dual convolutional small boolean circuit networks form. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2024.

Berthet, Q., Blondel, M., Teboul, O., Cuturi, M., Vert, J.-P., and Bach, F. Learning with differentiable perturbed optimizers. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9508–9519. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6bb56208f672af0dd65451f869fedfd9-Paper.pdf.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees*. Wadsworth, 1984. ISBN 0-534-98053-8.

Cohen, W. W. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on International Conference on Machine Learning*, ICML'95, pp. 115–123, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1558603778.

Cordonnier, J.-B., Mahendran, A., Dosovitskiy, A., Weissenborn, D., Uszkoreit, J., and Unterthiner, T. Differentiable patch selection for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2351–2360, June 2021.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. (eds.), *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL https://proceedings.neurips.cc/paper_files/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf.

Dierckx, L., Veroneze, R., and Nijssen, S. Rl-net: Interpretable rule learning with neural networks. In *Advances in Knowledge Discovery and Data Mining: 27th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD 2023, Osaka, Japan, May 25–28, 2023, Proceedings, Part I*, pp. 95–107, Berlin, Heidelberg, 2023. Springer-Verlag. ISBN 978-3-031-33373-6. doi: 10.1007/978-3-031-33374-3_8. URL https://doi.org/10.1007/978-3-031-33374-3_8.

Doshi-Velez, F. and Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

Erickson, N., Purucker, L., Tschalzev, A., Holzmüller, D., Desai, P. M., Salinas, D., and Hutter, F. Tabarena: A living benchmark for machine learning on tabular data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=jZqCqpCLdU.

Friedman, J. H. and Popescu, B. E. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3): 916–954, 2008. ISSN 19326157. URL http://www.jstor.org/stable/30245114.

Gorishniy, Y., Kotelnikov, A., and Babenko, A. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=Sd4wYYOhmY.

Grinsztajn, L., Oyallon, E., and Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data? In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 507–520. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/0378c7692da36807bdec87ab043cdadc-Paper-Datasets_and_Benchmarks.pdf.

Hegselmann, S., Buendia, A., Lang, H., Agrawal, M., Jiang, X., and Sontag, D. Tabllm: Few-shot classification of tabular data with large language models. In *International Conference on Artificial Intelligence and Statistics*, volume 206, pp. 5549–5581. PMLR, 2023.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. Lightgbm: A highly efficient gradient boosting decision tree. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

Lage, I., Chen, E., He, J., Narayanan, M., Kim, B., Gershman, S. J., and Doshi-Velez, F. Human evaluation of models built for interpretability. *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, 7(1):59–67, Oct. 2019. doi: 10.1609/hcomp.v7i1.5280. URL https://ojs.aaai.org/index.php/HCOMP/article/view/5280.

Lin, J., Zhong, C., Hu, D., Rudin, C., and Seltzer, M. Generalized and scalable optimal sparse decision trees. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

McCluskey, E. J. Minimization of boolean functions. *The Bell System Technical Journal*, 35(6):1417–1444, 1956. doi: 10.1002/j.1538-7305.1956.tb03835.x.

Petersen, F., Borgelt, C., Kuehne, H., and Deussen, O. Deep differentiable logic gate networks. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 2006–2018. Curran Associates, Inc., 2022a. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/0d3496dd0cec77a999c98d35003203ca-Paper-Conference.pdf.

Petersen, F., Kuehne, H., Borgelt, C., and Deussen, O. Differentiable top-k classification learning. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 17656–17668. PMLR, 17–23 Jul 2022b. URL https://proceedings.mlr.press/v162/petersen22a.html.

Petersen, F., Kuehne, H., Borgelt, C., Welzel, J., and Ermon, S. Convolutional differentiable logic gate networks. *Advances in Neural Information Processing Systems*, 37: 121185–121203, 2024.

Proença, H. M. and van Leeuwen, M. Interpretable multiclass classification by mdl-based rule lists. *Information Sciences*, 512:1372–1393, 2020. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2019.10.050. URL https://www.sciencedirect.com/science/article/pii/S0020025519310138.

Quine, W. V. The problem of simplifying truth functions. *The American Mathematical Monthly*, 59(8):521–531, 1952. ISSN 00029890, 19300972. URL http://www.jstor.org/stable/2308219.

Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019. doi: 10.1038/s42256-019-0048-x.

Seshia, S. A., Sadigh, D., and Sastry, S. S. Toward verified artificial intelligence. *Communications of the ACM*, 65 (7):46–55, 2022.

Slack, D., Hilgard, S., Jia, E., Singh, S., and Lakkaraju, H. Fooling lime and shap: Adversarial attacks on post hoc explanation methods. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 180–186, 2020.

Wang, Z., Zhang, W., Liu, N., and Wang, J. Scalable rule-based representation learning for interpretable classification. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 30479–30491. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/ffbd6cbb019a1413183c8d08f2929307-Paper.pdf.

Wei, D., Dash, S., Gao, T., and Gunluk, O. Generalized linear rule models. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6687–6696. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/wei19a.html.

Xie, Y., Dai, H., Chen, M., Dai, B., Zhao, T., Zha, H., Wei, W., and Pfister, T. Differentiable top-k with optimal transport. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 20520–20531. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/ec24a54d62ce57ba93a531b460fa8d18-Paper.pdf.

Xu, S., Walter, N. P., and Vreeken, J. Neural rule lists: Learning discretizations, rules, and order in one go. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=oBf5eZSeBT.

A. Datasets

We benchmark with 28 publicly available datasets (14 binary, 7 multiclass, 7 regression) used in prior works (Xu et al., 2025; Grinsztajn et al., 2022; Erickson et al., 2025; Wei et al., 2019; Hegselmann et al., 2023).

Table 4. Dataset information: the tabular task, the dataset name, the number of rows, the number of continuous columns, categorical columns, the number of instances per class for binary and multiclass datasets.

Task	Dataset	# Rows	# Cont.	# Cat.	Classes
Binary	bank	45,210	8	8	39921/5289
	blood	748	4	0	570/178
	calhousing	20,640	8	0	10323/10317
	compas	4,966	3	8	2483/2483
	covertype	423,680	10	44	211840/211840
	credit_card_default	13,272	20	1	6636/6636
	creditg	1,000	7	13	300/700
	diabetes	768	8	0	500/268
	electricity	38,474	7	1	19237/19237
	eye_movements	7,608	20	3	3804/3804
	heart	918	5	6	410/508
	income	48,842	6	8	37155/11687
	jungle	44,819	6	0	21757/23062
	road_safety	111,762	29	3	55881/55881
Multiclass	car	1,728	0	6	1210/384/69/65
	ecoli	327	5	2	143/77/35/20/52
	iris	150	4	0	50/50/50
	penguins	333	4	3	146/68/119
	satimage	6,435	36	0	1533/703/1358/626/707/1508
	wine	178	13	0	59/71/48
	yeast	1,479	6	2	244/429/463/44/35/51/163/30/20
Regression	abalone	4,177	7	1	-
	bike	17,379	4	8	-
	boston	506	11	2	-
	california	20,640	8	0	-
	crime	1,994	122	5	-
	parkinsons	5,875	19	1	-
	wine_reg	6,497	11	0	-

B. Implementation Details

B.1. Hardware

For all experiments, we use 4 Nvidia GeForce RTX 3090 GPUs and 48 cores Intel(R) Core(TM) i7-8650U CPU clocked at 1.90 GHz, 16 GB RAM. The hardware chosen was based on availability; neural networks under PyTorch (TT-SPARSE, NeuRules, RL-Net, RRL, DWN, TabM) were run on GPU, while tree-based model GOSDT and rule induction based models (Classy, GLRM, RuleFit) were run on CPU.

B.2. Hyperparameters and Training Protocol

For each dataset, we reserve 20% of the data as a hold-out test set using a fixed seed. The remaining 80% is used for hyperparameter tuning. We perform a grid search by further splitting this development set into 80% training and 20% validation. Once the optimal hyperparameters are identified based on the validation metric (AUC for classification, R^2 for regression), we retrain the model on the full development set. This process is repeated across 5 different random seeds and we report the model's performance on the hold-out test set.

We tune the key hyperparameters for all models by performing GridSearch:

- **TT-SPARSE:** Number of bits $\in \{4, 5, 6\}$, Number of LTT nodes $\in \{1, 20, 30, 40, 50\}$, Soft TOPK temperature $\tau \in \{0.001, 0.01, 0.05\}$.
- **RuleFit:** max_rules $\in \{5, 20, 40, 60, 80\}$ and regularization strength $Cs \in \{1, 5, 10, 50\}$.
- **GLRM:** Regularization penalty $\lambda_0 \in \{0.05, 0.01, 0.005\}$. We fix $\lambda_1 = 0.2 \cdot \lambda_0$ as per (Wei et al., 2019).
- **Classy (MDL-Rule-List):** min_support $\in \{5, 10, 15, 20, 30\}$, maximum depth $\in \{3, 5, 7\}$, and number of cutpoints $\in \{5, 7\}$.
- **GOSDT:** Regularization parameter $\in \{0.005, 0.01, 0.05\}$ and depth budget $\in \{3, 5, 7\}$.
- **NeuRules:** n_rules $\in \{5, 10, 20, 40, 60, 80\}$, predicate temperature $\in \{0.1, 0.2\}$, and selector temperature $\in \{0.5, 1.0\}$.
- **RL-Net:** n_rules $\in \{5, 10, 20, 40, 60, 80\}$, conjunction penalty $\lambda_{and} \in \{0.01, 0.005, 0.001\}$, and L_2 regularization $\in \{0, 0.001, 0.01\}$.
- **DWN (Linear):** Hidden layer sizes $\in \{20, 30, 40, 50\}$ and Look-Up Table (LUT) size $\in \{4, 5, 6\}$. To ensure a fair comparison with TT-SPARSE, we implemented the classifier layer immediately following the LUT layer. This architecture introduces a linear operation by assigning a weight to each LUT and adding a bias term before the activation function. The logic rules are derived similarly as TT-SPARSE, utilizing Quine-McCluskey conversion to transform LUT minterms into DNF equations.
- **TabM**: We use the AdamW optimizer and a batch size of 512. Number of layers $\in \{1, 2, 3\}$, hidden size $\in \{64, 128, 256, 512\}$, dropout rate $\in \{0, 0.1, 0.2\}$, learning rate $\in \{0.01, 0.005, 0.001\}$, bin count for embeddings $\in \{8, 16, 32, 64\}$, and embedding dimension $\in \{8, 16, 32, 64\}$.

C. Soft TOPK

$S_k(\vec{x}) = [y_1, \ldots, y_n]$ where $y_i = \sigma(\frac{x_i}{\tau} + c)$, constrained by $\sum_{j=1}^{n} \sigma(\frac{x_i}{\tau} + c) = k$. $\sigma(x) = \frac{1}{1+e^{-x}}$ is the sigmoid function and $\sigma'(x) = \sigma(x)(1 - \sigma(x))$. Let $s'_i = \sigma'(\frac{x_i}{\tau} + c) = \frac{d}{d(\frac{x_i}{\tau}+c)}\sigma(\frac{x_i}{\tau} + c) = y_i(1 - y_i)$ and $s' = [s_1, \ldots, s_n]^\top$.

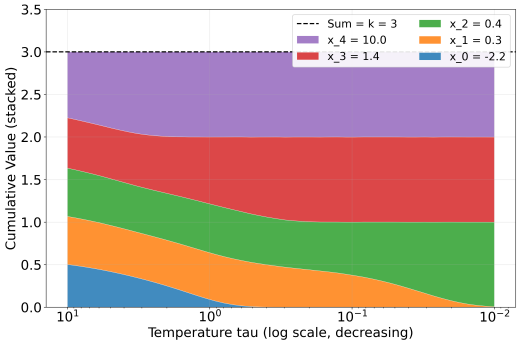


Figure 6. The output distribution of the soft TOPK operator with $k = 3$ applied on input vector $[-2.2, 0.3, 0.4, 1.4, 10]$ across different temperatures τ. y-axis represents the output mass for each index of the input vector.

C.1. Solving the optimization objective via Lagrangian multipliers

Recall the optimization objective

$$\vec{y}^* = \underset{\vec{y} \in [0,1]^n}{\operatorname{argmax}} \left[\sum_{i=1}^{n} y_i x_i + \tau \sum_{i=1}^{n} H(y_i) \right] \text{ subject to } \sum_{i=1}^{n} y_i = k \tag{7}$$

where $H(y_i)$ is the binary entropy $H(y_i) = -y_i \log y_i - (1 - y_i) \log(1 - y_i)$. We then derive the optimum by introducing a Lagrange multiplier λ for the $\sum_{i=1}^{n} y_i = k$ constraint such that the problem becomes

$$\mathcal{L}(\vec{y}, \lambda) = \sum_{i=1}^{n} y_i x_i + \tau \sum_{i=1}^{n} \Big(- y_i \log y_i - (1 - y_i) \log(1 - y_i) \Big) - \lambda \left(\sum_{i=1}^{n} y_i - k \right) \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial y_i} = x_i + \tau \left(\log \frac{1 - y_i}{y_i} \right) - \lambda = 0 \tag{9}$$

Setting the derivative 0, we obtain the solution $y_i = \frac{1}{1+e^{-\frac{(x_i-\lambda)}{\tau}}} = \sigma(\frac{x_i-\lambda}{\tau})$, where $\sigma(\cdot)$ is the sigmoid function. Thus, setting $c = -\frac{\lambda}{\tau}$, we define $S_k(\vec{x}) := \vec{y}$ where $y_i := \sigma(\frac{x_i}{\tau} + c)$ for $i \in \{1, \ldots, n\}$ and $c \in \mathbb{R}$ is the unique root of $f(c) = \sum_{j=1}^{n} \sigma(\frac{x_i}{\tau} + c) - k = 0$. Since $f(c)$ is strictly increasing, c is uniquely determined and solved by the bisection method.

C.2. Obtaining $\nabla_{\vec{x}} S_k(\vec{x})$

We want the Jacobian $J \in R^{n \times n}$ where $J_{ij} = \frac{\partial y_i}{\partial x_j}$.

First, we apply the chain rule to $y_i = \sigma(\frac{x_i}{\tau} + c)$:

$$\frac{\partial y_i}{\partial x_j} = \sigma'\left(\frac{x_i}{\tau} + c\right) \cdot \frac{\partial}{\partial x_j}\left(\frac{x_i}{\tau} + c\right)$$

Since $\frac{\partial x_i}{\partial x_j} = \delta_{ij}$:

$$\frac{\partial y_i}{\partial x_j} = s'_i \left(\frac{1}{\tau}\delta_{ij} + \frac{\partial c}{\partial x_j}\right) \tag{10}$$

where $\delta_{ij} = \mathbf{1}_{\{i=j\}}$ is the Kronecker delta. Next, we differentiate the constraint function w.r.t. x_j to obtain:

$$\frac{\partial}{\partial x_j} \sum_{m=1}^{n} \sigma\left(\frac{x_m}{\tau} + c\right) = 0. \tag{11}$$

$$\sum_{m=1}^{n} s'_m \left(\frac{1}{\tau}\delta_{mj} + \frac{\partial c}{\partial x_j}\right) = 0. \tag{12}$$

$$\frac{1}{\tau} s'_j + \left(\sum_{m=1}^{n} s'_m\right) \frac{\partial c}{\partial x_j} = 0. \tag{13}$$

$$\frac{\partial c}{\partial x_j} = -\frac{1}{\tau} \frac{s'_j}{\|s'\|_1} \tag{14}$$

Substituting this back to Equation (1):

$$\frac{\partial y_i}{\partial x_j} = \frac{1}{\tau} s'_i \left(\delta_{ij} - \frac{s'_j}{\|s'\|_1}\right) \tag{15}$$

$$\nabla_{\vec{x}} S_k(\vec{x}) = \frac{1}{\tau}\left[\text{diag}(s') - \frac{s'(s')^\top}{\|s'\|_1}\right] \tag{16}$$

C.3. Obtaining $\nabla_k S_k(\vec{x})$

$$\frac{\partial y_i}{\partial k} = s'_i \cdot \frac{\partial}{\partial k}\left(\frac{x_i}{\tau} + c\right) \tag{17}$$

$$\frac{\partial y_i}{\partial k} = s'_i \cdot \frac{\partial c}{\partial k} \tag{18}$$

Then, similarly, we differentiate both sides of the constraint function w.r.t. k to obtain:

$$\frac{\partial}{\partial k} \sum_{m=1}^{n} \sigma\left(\frac{x_m}{\tau} + c\right) = 1 \tag{19}$$

$$\sum_{m=1}^{n} s'_m \cdot \frac{\partial c}{\partial k} = 1 \tag{20}$$

$$\frac{\partial c}{\partial k} = \frac{1}{\|s'\|_1} \tag{21}$$

Substituting this back to Equation (9):

$$\frac{\partial y_i}{\partial k} = \frac{s'_i}{\|s'\|_1} \tag{22}$$

$$\nabla_k S_k(\vec{x}) = \frac{s'}{\|s'\|_1} \tag{23}$$

C.4. Convergence of bisection method

To compute the constant c, we implement vectorized bisection search that solves for c across an entire batch simultaneously on GPU. The bisection method halves the search space in every iteration, so the absolute error $|\sum S_k(\vec{x})_i - k|$ decays with $\epsilon_t \propto 2^{-t}$.

Figure 7 illustrates the convergence of this method. The soft TOPK is applied to a batch of 16 input vectors $\vec{x} \in \mathbb{R}^{100}$ with target $k = 10$. The theoretical linear convergence (linear slope on the semi-log plot) is observed empirically. The error magnitude drops below 10^{-3} within 15 iterations and reaches the limit of `Float32` machine precision in around 30 iterations. Our experiments also show negligible overhead with around 1 ms for a batch of size 16, giving a throughput of more than 10000 samples per second.

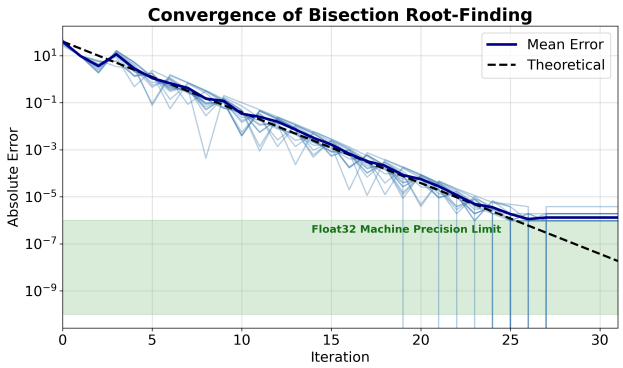


Figure 7. Absolute error $|\sum y_i - k|$ (log scale) vs. iterations for a batch of 16-input vectors $\vec{x} \in \mathbb{R}^{100}$ with target $k = 10$. Each light blue line represents the error plot of an input vector in the batch.

D. TT-SPARSE Parameter Gradients

Let $\vec{x} \in \mathbb{R}^n$ denote the input to the TT-SPARSE layer with M LTT nodes. The layer is parameterized by $W_{\text{map}} \in \mathbb{R}^{n \times M}$, $W_{\text{LTT}} \in \mathbb{R}^{n \times M}$, and $b \in \mathbb{R}^M$. For each node $j \in \{1, \ldots, M\}$, the output is

$$z_j = \sum_{i=1}^{n} m_i^{(j)} \cdot W_{\text{LTT}}[i, j] \cdot x_i + b_j.$$

Leveraging the soft TOPK operator S_k in the backpropagation, $m^{(j)} = S_k(W_{\text{map}}[\cdot, j])$ is the operator applied on the j-th column of W_{map} which corresponds to the j-th node. Let $\mathcal{L}$ be the upstream loss.

D.1. Gradient w.r.t. W_{map}

With the chain rule, we have

$$\frac{\partial \mathcal{L}}{\partial W_{\text{map}}[\cdot, j]} = \frac{\partial \mathcal{L}}{\partial z_j} \cdot \frac{\partial z_j}{\partial S_k(W_{\text{map}}[\cdot, j])} \cdot \frac{\partial S_k(W_{\text{map}}[\cdot, j])}{\partial W_{\text{map}}[\cdot, j]}$$

As $m^{(j)} = S_k(W_{\text{map}}[\cdot, j])$, we have

$$\frac{\partial z_j}{\partial S_k(W_{\text{map}}[\cdot, j])} = x_i \odot W_{\text{LTT}}[i, j]. \tag{24}$$

The third term is the Jacobian of the soft TOPK operator applied on the W_{map}. Using the Jacobian formula obtained in Equation (13), where $s' = m^{(j)} \odot (1 - m^{(j)})$ is the derivative of the sigmoid:

$$\frac{\partial S_k(W_{\text{map}}[\cdot, j])}{\partial W_{\text{map}}[\cdot, j]} = \frac{1}{\tau} \left[\text{diag}(s') - \frac{s'(s')^\top}{\|s'\|_1} \right] \tag{25}$$

Let $g_j = \frac{\partial \mathcal{L}}{\partial z_j} \cdot (x \odot W_{\text{LTT}}[\cdot, j])$ be the local gradient from the first and second terms of the derivative. Combining these, we obtain the vector-Jacobian product:

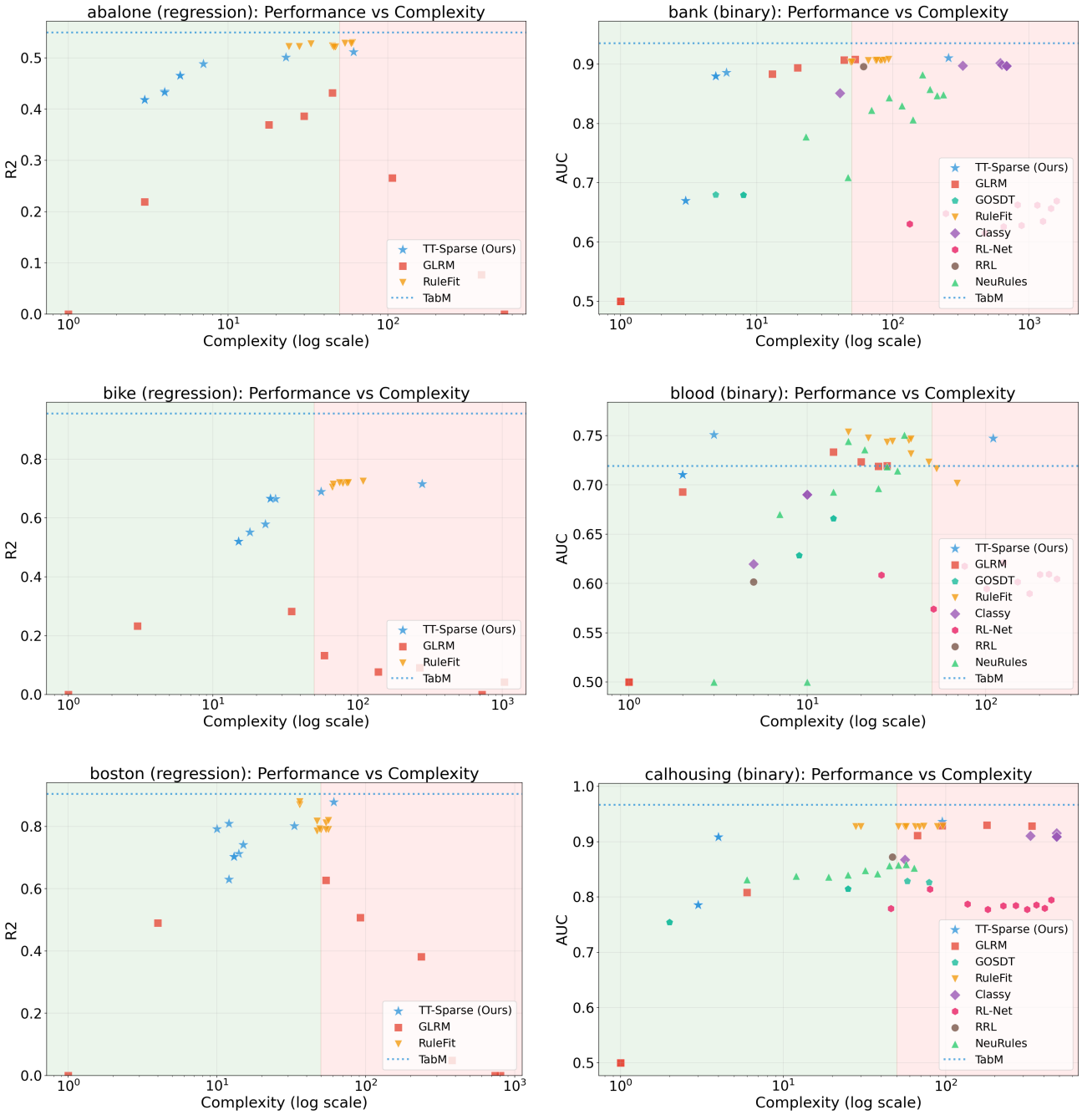
$$\frac{\partial \mathcal{L}}{\partial W_{\text{map}}[\cdot, j]} = \frac{1}{\tau}\left[\text{diag}(s')g_j - \frac{s'(s')^\top}{\|s'\|_1} g_j\right] = \frac{1}{\tau}\left[s' \odot g_j - \frac{s'(s' \cdot g_j)}{\|s'\|_1}\right] \tag{26}$$

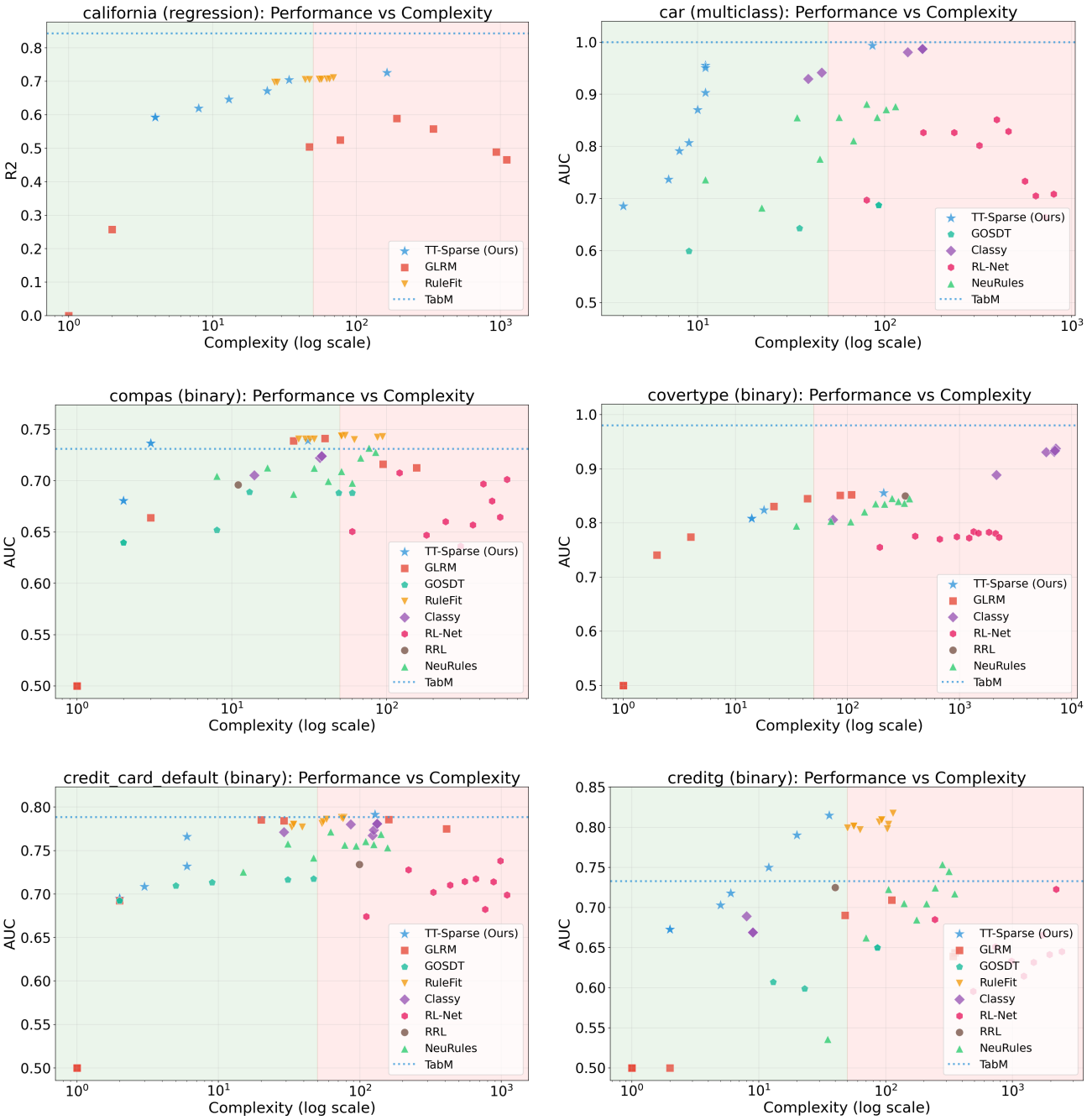
D.2. Gradient w.r.t. W_{LTT} and b

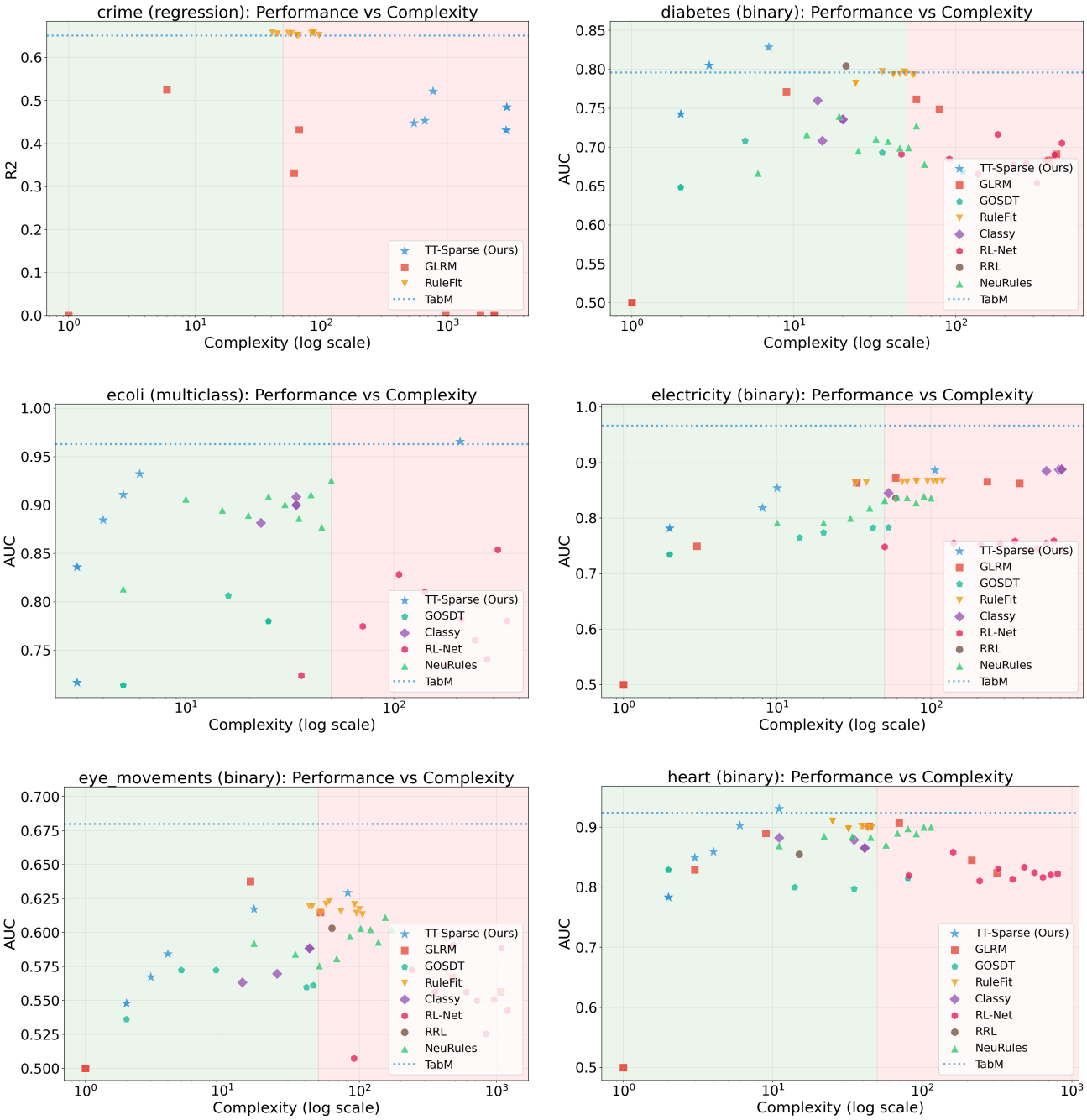
$$\frac{\partial \mathcal{L}}{\partial W_{\text{LTT}}[i,j]} = \frac{\partial \mathcal{L}}{\partial z_j} \cdot \frac{\partial z_j}{\partial W_{\text{LTT}}[i,j]} = \frac{\partial \mathcal{L}}{\partial z_j} \cdot m_i^{(j)} \cdot x_i \tag{27}$$

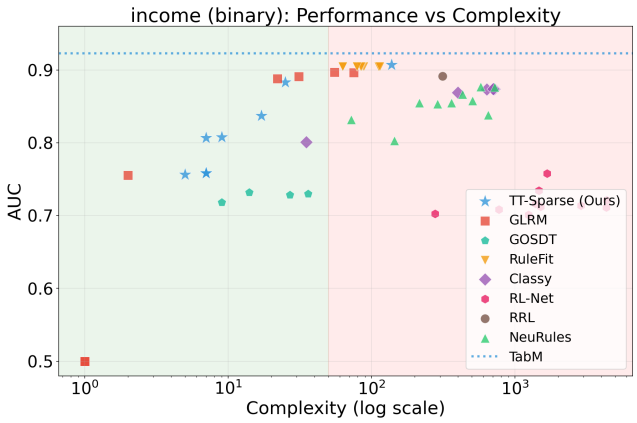
$$\frac{\partial \mathcal{L}}{\partial b} = \frac{\partial \mathcal{L}}{\partial z_j} \cdot \frac{\partial z_j}{\partial b} = \frac{\partial \mathcal{L}}{\partial z_j} \tag{28}$$

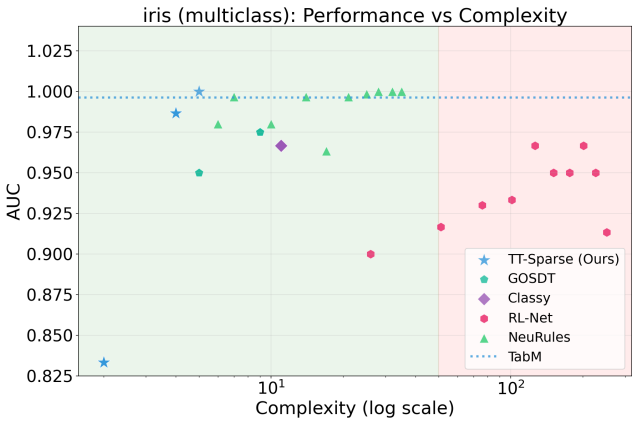
E. Performance-Complexity Pareto Grids

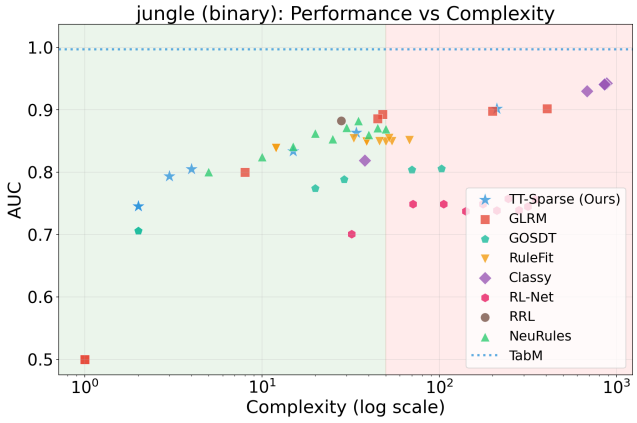

abalone (regression): Performance vs Complexity
R2
Complexity (log scale)
bank (binary): Performance vs Complexity
AUC
Complexity (log scale)
bike (regression): Performance vs Complexity
R2
Complexity (log scale)
blood (binary): Performance vs Complexity
AUC
Complexity (log scale)
boston (regression): Performance vs Complexity
R2
Complexity (log scale)
calhousing (binary): Performance vs Complexity
AUC
Complexity (log scale)
TT-Sparse (Ours)
GLRM
GOSDT
RuleFit
Classy
RL-Net
RRL
NeuRules
TabM

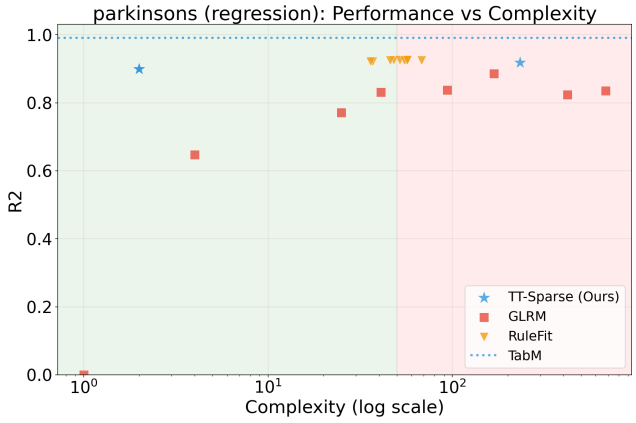

california (regression): Performance vs Complexity
R2
Complexity (log scale)
car (multiclass): Performance vs Complexity
AUC
Complexity (log scale)
compas (binary): Performance vs Complexity
AUC
Complexity (log scale)
covertype (binary): Performance vs Complexity
AUC
Complexity (log scale)
credit_card_default (binary): Performance vs Complexity
AUC
Complexity (log scale)
creditg (binary): Performance vs Complexity
AUC
Complexity (log scale)
TT-Sparse (Ours)
GLRM
GOSDT
RuleFit
Classy
RL-Net
RRL
NeuRules
TabM

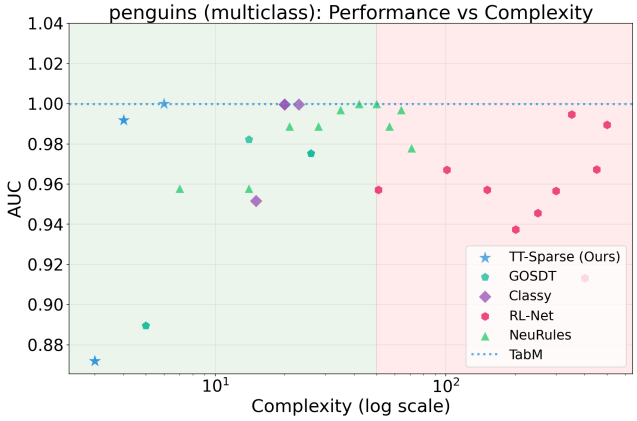

crime (regression): Performance vs Complexity
R2
Complexity (log scale)
diabetes (binary): Performance vs Complexity
AUC
Complexity (log scale)
ecoli (multiclass): Performance vs Complexity
AUC
Complexity (log scale)
electricity (binary): Performance vs Complexity
AUC
Complexity (log scale)
eye_movements (binary): Performance vs Complexity
AUC
Complexity (log scale)
heart (binary): Performance vs Complexity
AUC
Complexity (log scale)
TT-Sparse (Ours)
GLRM
GOSDT
RuleFit
Classy
RL-Net
RRL
NeuRules
TabM

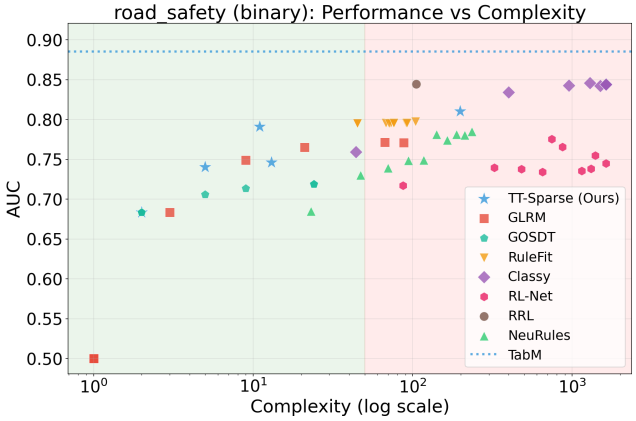

income (binary): Performance vs Complexity
AUC
Complexity (log scale)
TT-Sparse (Ours)
GLRM
GOSDT
RuleFit
Classy
RL-Net
RRL
NeuRules
TabM

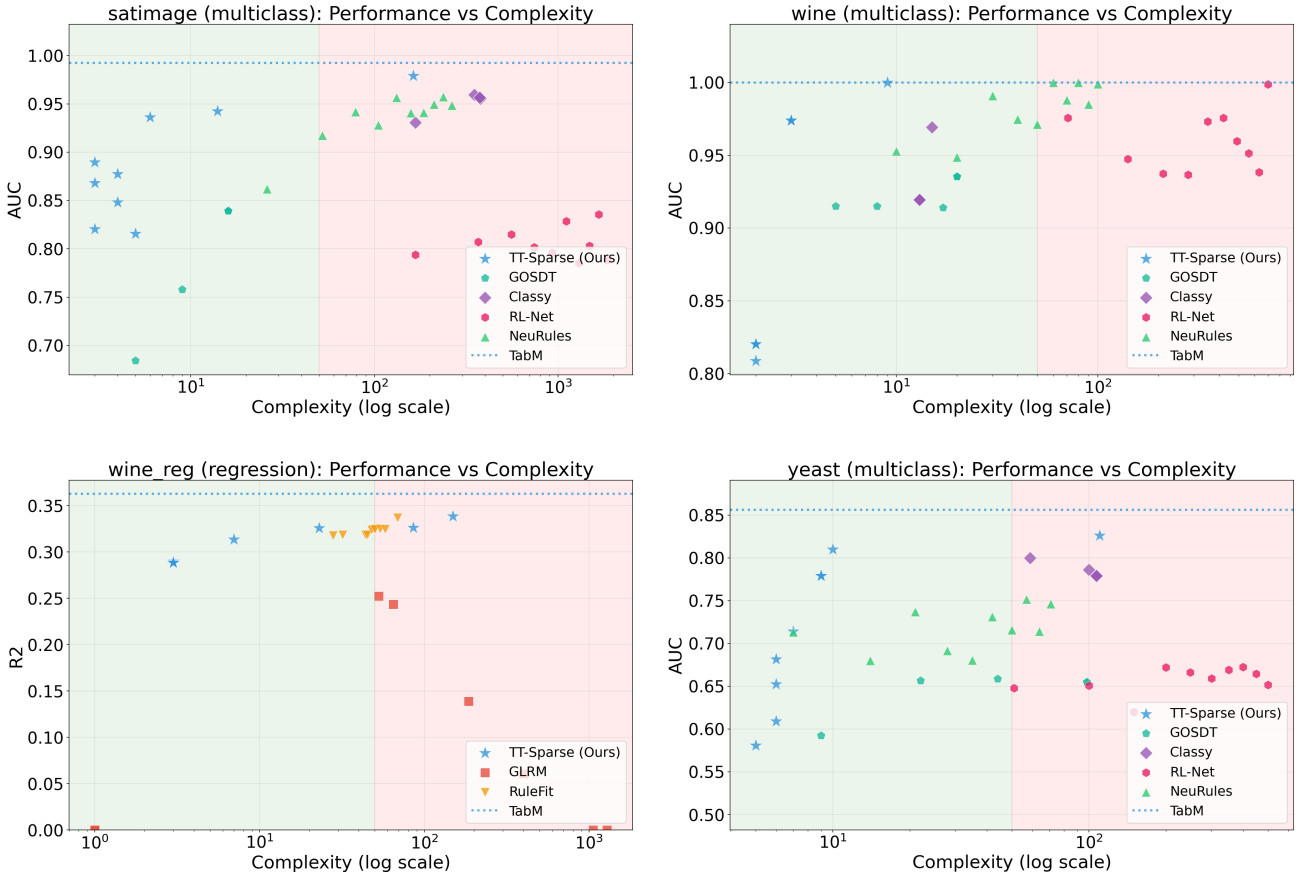

iris (multiclass): Performance vs Complexity
AUC
Complexity (log scale)
TT-Sparse (Ours)
GOSDT
Classy
RL-Net
NeuRules
TabM


jungle (binary): Performance vs Complexity
AUC
Complexity (log scale)
TT-Sparse (Ours)
GLRM
GOSDT
RuleFit
Classy
RL-Net
RRL
NeuRules
TabM


parkinsons (regression): Performance vs Complexity
R2
Complexity (log scale)
TT-Sparse (Ours)
GLRM
RuleFit
TabM


penguins (multiclass): Performance vs Complexity
AUC
Complexity (log scale)
TT-Sparse (Ours)
GOSDT
Classy
RL-Net
NeuRules
TabM


road_safety (binary): Performance vs Complexity
AUC
Complexity (log scale)
TT-Sparse (Ours)
GLRM
GOSDT
RuleFit
Classy
RL-Net
RRL
NeuRules
TabM


satimage (multiclass): Performance vs Complexity
AUC
Complexity (log scale)
wine (multiclass): Performance vs Complexity
AUC
Complexity (log scale)
wine_reg (regression): Performance vs Complexity
R2
Complexity (log scale)
yeast (multiclass): Performance vs Complexity
AUC
Complexity (log scale)
TT-Sparse (Ours)
GOSDT
Classy
RL-Net
NeuRules
GLRM
RuleFit
TabM

F. Example rules

GLRM example rule for eye_movements dataset with complexity 15:
0.84 – titleNo ≤ 2; **0.34** – wordNo ≤ 2; **-0.33** – nextWordRegress $= 1$; **0.31** – regressDur ≤ 139; **0.13** – nextWordRegress $= 0$; **-0.13** – wordNo ≤ 1; **0.15** – regressDur ≤ 0; **-0.18** – prevFixPos ≤ 149.5; **-0.16** – landingPos ≤ 71.6; **-0.10** – firstSaccLen ≤ 250.1; **-0.04** – firstSaccLen ≤ 204.0; **-0.53** – (lastSaccLen $\leq 503.2 \wedge$ landingPos $\leq 117.3 \wedge 1 <$ wordNo ≤ 8); (Bias: -0.23)

GOSDT example rule in the form of a tree for blood dataset with complexity 14:
Class 0 – Feature$_0 > -0.62$; **Class 0** – Feature$_0 \leq -0.62 \wedge$Feature$_3 > 0.34$; **Class 1** – Feature$_0 \leq -0.62 \wedge$Feature$_3 \leq 0.34 \wedge$ Feature$_6 > 2.14$; **Class 0** – Feature$_0 \leq -0.62 \wedge$ Feature$_3 \leq 0.34 \wedge$ Feature$_6 \leq 2.14 \wedge$ Feature$_8 > 0.34$; **Class 1** – Feature$_0 \leq -0.62 \wedge$ Feature$_3 \leq 0.34 \wedge$ Feature$_6 \leq 2.14 \wedge$ Feature$_8 \leq 0.34$;

Classy (MDL Rule List) example decision list for blood dataset with complexity 8:
Class 1 ($P = 0.57$) – (monetary $\geq 1500 \wedge$ time $< 59 \wedge$ recency < 7); **Class 0** ($P = 0.71$) – recency < 7; **Default Class 0** ($P = 0.89$) – else.

NeuRules example rule for diabetes dataset with complexity 17:
3.19 – $(0.82 <$ Feature_0 $< 1.69) \rightarrow$ Class $1(71.5\%)$; **8.41** – $(0.46 <$ Feature_4 $\wedge -0.40 <$ Feature_7 $< 3.50) \rightarrow$ Class $1(74.2\%)$; **4.56** – $(0.62 <$ Feature_4 $\wedge -0.15 <$ Feature_7 $< 3.31) \rightarrow$ Class $1(62.8\%)$; **5.97** – (Feature_0 $< 2.49 \wedge$ Feature_1 $< 0.58 \wedge$ Feature_6 $< 4.55) \rightarrow$ Class $0(80.9\%)$; **1.00** – $(-3.49 <$ Feature_1 $< 1.67 \wedge 0.18 <$ Feature_4 $< 4.98 \wedge -1.14 <$ Feature_6 $< 5.58 \wedge -0.35 <$ Feature_7 $< 3.50) \rightarrow$ Class $1(64.4)$;

RL-Net example rule for calhousing dataset with complexity 87:
Class 1 – (Feature$_0 \wedge \neg$Feature$_4 \wedge \neg$Feature$_6 \wedge \neg$Feature$_7$); **Class 1** – ($\neg$Feature$_0 \wedge$ Feature$_1 \wedge \neg$Feature$_2 \wedge$ Feature$_3 \wedge \neg$Feature$_4 \wedge$ Feature$_5 \wedge \neg$Feature$_6 \wedge \neg$Feature$_7$); **Class 1** – ($\neg$Feature$_0 \wedge$ Feature$_1 \wedge$ Feature$_2 \wedge$ Feature$_3 \wedge \neg$Feature$_4 \wedge$ Feature$_5 \wedge \neg$Feature$_6 \wedge \neg$Feature$_7$); **Class 1** – (Feature$_0 \wedge \neg$Feature$_1 \wedge$ Feature$_2 \wedge$ Feature$_3 \wedge$ Feature$_4 \wedge$ Feature$_5 \wedge \neg$Feature$_6 \wedge \neg$Feature$_7$); **Class 0** – ($\neg$Feature$_0 \wedge$ Feature$_1 \wedge \neg$Feature$_2 \wedge$ Feature$_3 \wedge$ Feature$_4 \wedge$ Feature$_5 \wedge$ Feature$_6 \wedge$ Feature$_7$); **Class 0** – ($\neg$Feature$_0 \wedge$ Feature$_1 \wedge \neg$Feature$_2 \wedge \neg$Feature$_3 \wedge$ Feature$_4 \wedge$ Feature$_5 \wedge \neg$Feature$_6 \wedge$ Feature$_7$); **Class 0** – ($\neg$Feature$_0 \wedge$ Feature$_1 \wedge \neg$Feature$_2 \wedge$ Feature$_3 \wedge \neg$Feature$_4 \wedge$ Feature$_5 \wedge \neg$Feature$_6 \wedge$ Feature$_7$); **Class 0** – ($\neg$Feature$_0 \wedge$ Feature$_1 \wedge \neg$Feature$_2 \wedge$ Feature$_3 \wedge$ Feature$_4 \wedge$ Feature$_5 \wedge \neg$Feature$_6 \wedge \neg$Feature$_7$); Default: **Class 0**

RRL example rule with complexity 12:
0.12 – frequency > 17.85; **-0.05** – frequency ≤ 3.14; **0.04** – recency > 19.73; **0.04** – (recency $> 32.27 \vee$ time ≤ 31.02); **0.19** – (recency $> 12.79 \wedge$ time $> 10.96 \wedge$ frequency ≤ 1.75); **0.09** – (time $> 16.72 \wedge$ frequency $\leq 2.10 \wedge$ monetary ≤ 1131.57); *(Bias: -0.02)*

RuleFit example rule on abalone dataset with complexity 15:
12.63 – Feature_5; **-11.86** – Feature_6; **-3.28** – Feature_7; **3.12** – Feature_8; **1.83** – Feature_4; **1.36** – Feature_3; **-0.74** – Feature_0; **-0.12** – Feature_2; **0.08** – Feature_1; **-0.30** – (Feature_8 ≤ -0.62); **0.09** – (Feature_8 $> -0.64 \wedge$ Feature_8 ≤ 0.83); **-0.43** – (Feature_8 $> -0.64 \wedge$ Feature_8 > 0.83); (Bias: 10.24)

TT-SPARSE example rule (same as Figure 2) with complexity 15:
-0.83 – Feature_ChestPainType = 'NAP'; **0.97** – Feature_ExerciseAngina = 'Y'; **1.13** – Feature_ST_Slope = 'Flat'; **1.40** – ((Feature_Cholesterol $< 167.63 \wedge$ Feature_Oldpeak $\geq 3.00) \vee$ (Feature_ChestPainType $\neq$ 'TA' $\wedge$ Feature_Oldpeak $\geq 3.00) \vee$ (Feature_ChestPainType $\neq$ 'TA' $\wedge$ Feature_ChestPainType $\neq$ 'ATA' $\wedge$ Feature_Cholesterol < 167.63)); **-1.03** – ((Feature_MaxHR $\geq 177.25 \wedge$ Feature_Oldpeak $< 2.25) \vee$ (Feature_ChestPainType = 'ATA' $\wedge$ Feature_Cholesterol < 224.88)); (Bias: -1.04)

TT-SPARSE example multiclass rule (Iris dataset) with complexity 7: **Class 0: -3.66** – sepal-length; **Class 0: 7.10, Class 2: -2.91** – sepal-width; **Class 0: -9.47, Class 2: 6.42** – petal-length; **Class 0: -10.65, Class 1: -4.34, Class 2: 5.77** – petal-width; **Class 1: -10.55, Class 2: 8.85** – (petal-length $\geq 5.76 \vee$ petal-width ≥ 1.87); (Bias: **Class 0: 2.70, Class 1: 8.71, Class 2: -2.82**)

G. Quine-McCluskey

Algorithm 1 Quine-McCluskey with XOR/XNOR

Require: Set of minterms M_1, set of don't-cares M_{dc}, flag use_xor
Ensure: Minimal set of implicants R covering M_1

procedure SIMPLIFY(M_1, M_{dc})
  $T \leftarrow$ BinaryStringRep($M_1 \cup M_{dc}$)
  $PI \leftarrow$ GETPRIMEIMPLICANTS(T, use_xor)
  $EI \leftarrow$ EXTRACTESSENTIAL(PI, M_{dc})
  $R \leftarrow$ REDUCEIMPLICANTS(EI, M_{dc})
  return R
end procedure

function GETPRIMEIMPLICANTS(T, use_xor)
  if use_xor **then**
    $T \leftarrow T \cup \{\text{MergeXOR}(t_i, t_j) \mid t_i, t_j \in T\}$ {Pre-pass for simple XOR/XNOR}
  end if
  $marked \leftarrow \emptyset$
  while changes occur in T **do**
    Group T by tuple $(n_{\text{ones}}, n_{\oplus}, n_{\odot})$
    $T_{new} \leftarrow \emptyset$
    for each group G_k and adjacent group G_{next} **do**
      for $t_1 \in G_k, t_2 \in G_{next}$ **do**
        if $d_H(t_1, t_2) = 1$ **then** {Standard QM Merge}
          $t_{new} \leftarrow \text{ReplaceDiff}(t_1, t_2, \text{'-'})$
          $T_{new} \leftarrow T_{new} \cup \{t_{new}\}$; $marked \leftarrow marked \cup \{t_1, t_2\}$
        else if $use\_xor \wedge \text{IsComplement}(t_1, t_2)$ **then** {Extended Merge}
          $t_{new} \leftarrow \text{ReplaceDiff}(t_1, t_2, \oplus \text{ or } \odot)$
          $T_{new} \leftarrow T_{new} \cup \{t_{new}\}$; $marked \leftarrow marked \cup \{t_1, t_2\}$
        end if
      end for
    end for
    $PI \leftarrow PI \cup (T \setminus marked)$
    $T \leftarrow T_{new}$
  end while
  return PI
end function

function REDUCEIMPLICANTS(I, M_{dc})
  Def $E(t)$: Set of minterms covered by term t excluding M_{dc}
  Def $C(t)$: Complexity cost of term t (weighted sum of operators)
  {Step 1: Orthogonal Combination}
  repeat
    Find pair $a, b \in I$ and merger m such that $E(m) = E(a) \cup E(b)$
    if exists m **then**
      $I \leftarrow (I \setminus \{a, b\}) \cup \{m\}$
    end if
  until no combinations found
  {Step 2: Redundancy Elimination}
  repeat
    Find $t \in I$ such that $E(t) \subseteq \bigcup_{k \in I \setminus \{t\}} E(k)$
    if exists t **then**
      Select $t_{worst} \in \{t \mid t \text{ is redundant}\}$ maximizing $C(t)$
      $I \leftarrow I \setminus \{t_{worst}\}$
    end if
  until no redundant terms exist
  return I
end function
=0

H. Ablation Studies

We conduct ablation studies to isolate the impact of specific components and hyperparameters within the TT-SPARSE framework. In Figure 13, we analyze the effect of the quantization bit-width for continuous features on both predictive performance and rule complexity. We observe a performance plateau beyond 7 bits, indicating that TT-SPARSE effectively captures decision boundaries without requiring tighter bins on the continuous feature encoding. Figures 14 and 15 examine the model's sensitivity to key structural parameters: the sparsity degree (number of input bits per LTT node), the model capacity (number of LTT nodes), and the temperature τ of the Soft TOPK operator. Finally, we validate the necessity of our hybrid design by ablating the global skip connections that link input features directly to the final classifier. We observe that removing this skip connection degrades performance.

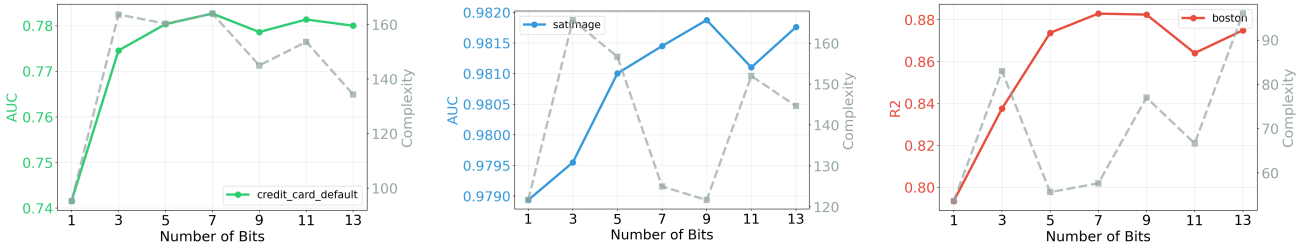


Figure 13. Ablation study on the number of bits used for continuous feature encoding, showing the performance metric (AUC or R^2) on the left side of the y-axis and rule complexity (in gray) on the right.

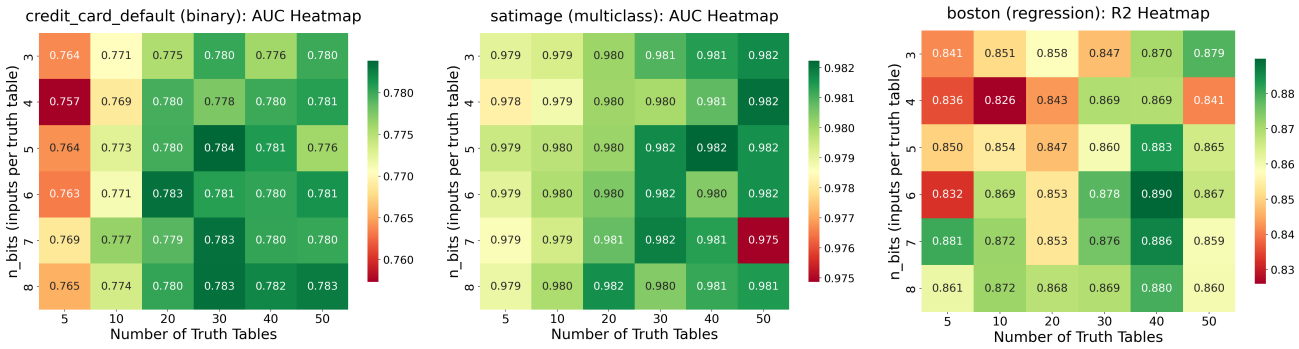


Figure 14. Ablation study on the number of input bits into each LTT node and the number of LTT nodes in the layer, visualized with a heatmap of the performance metric (AUC or R^2).

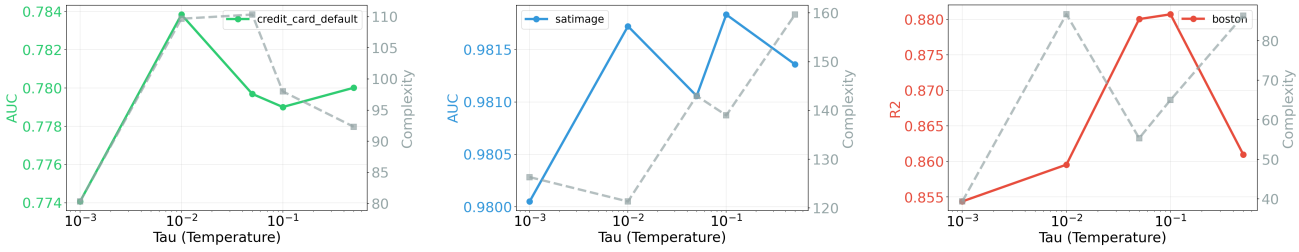


Figure 15. Ablation study on the temperature τ of the soft TOPK operator (lower τ gets it closer to discrete TOPK).

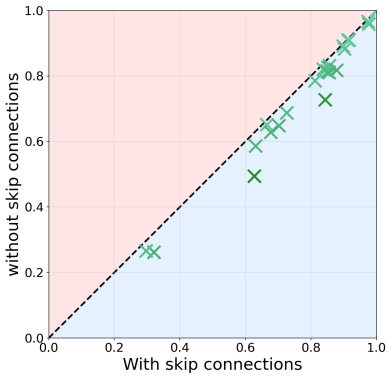


Figure 16. Ablation study on the integration of skip connection from the input features directly to the final classifier, concatenated with the outputs of the TT-SPARSE block. Each points show the evaluation metric (AUC/R2) achieved by the model with and without skip connections.