LEARNING INTERPRETABLE OBLIVIOUS TREES WITH VARIATIONAL QUANTUM CIRCUITS

Anonymous authors
Paper under double-blind review

ABSTRACT

Decision trees offer an attractive balance between predictive performance and interpretability. In this work, we explore whether quantum computing can advance interpretable tree learning beyond traditional induction techniques. We introduce Quantum Oblivious Decision Trees, a variational quantum approach for learning oblivious decision trees, i.e., a subclass of trees in which all nodes at the same depth share a common split. Building on a phase-based quantum comparator, we design and study three quantum circuits: *general*, a conventional hardware-efficient design; *structured*, which uses one quantum comparator per decision level; and *recycle*, which reuses a single comparator qubit across all levels via mid-circuit measurement and reset. The advantage of the latter two circuits is that they are interpretable by design. Indeed, their learned feature-threshold pairs can be extracted to reconstruct an equivalent classical oblivious decision tree. We evaluate the proposed circuits on several binary tabular datasets against classical oblivious and non-oblivious tree baselines. When restricted to the same feature set, the quantum circuits achieve comparable performance, with no statistically significant differences from the baselines. As tree depth increases, *recycle* maintains a constant qubit count and stable accuracy, while *general* and *structured* require one additional qubit per level and *general* loses up to roughly one quarter of its accuracy. Its gradient variance also decays at half the rate observed for the other circuits. Overall, these results characterize the trade-offs between expressivity, interpretability, and qubit efficiency in quantum tree learning.

1 INTRODUCTION

Decision trees (DTs) remain among the most widely used models in machine learning (Breiman, 2001; Quinlan, 2014). Their appeal lies in the combination of predictive performance and interpretability: each prediction can be traced to a sequence of human-readable comparisons between features and learned thresholds, allowing practitioners to inspect and audit model behavior. *Oblivious* Decision Trees (ODTs) impose an additional structural constraint, requiring all nodes at the same depth to share the same feature and threshold. Consequently, a depth-d ODT is characterized by d feature-threshold pairs, resulting in a compact and highly structured decision model. This property has made ODTs a building block of gradient boosting frameworks such as CatBoost (Prokhorenkova et al., 2018) and differentiable tree architectures such as Deep Neural Decision Trees (Yang et al., 2018) and Neural Oblivious Decision Ensembles (Popov et al., 2019). Accordingly, substantial research in interpretable machine learning has focused on developing methods for learning increasingly accurate DTs without sacrificing their transparent structure. In this work, we explore whether quantum computing can advance this direction beyond traditional tree-induction techniques. In particular, we study how a classical decision model can be represented within a quantum circuit, and how this representation affects its interpretability, trainability, and resource requirements.

For this purpose, Variational Quantum Algorithms (VQAs) provide a natural framework, combining parametrized quantum circuits with classical optimization in a hybrid quantum-classical scheme (Cerezo et al., 2021). However, most variational quantum classifiers adopt a generic circuit: the input is encoded into a quantum state, followed by a generic ansatz whose parameters are optimized to produce a prediction. While flexible, this representation does not explicitly preserve the structure of a decision model since their parameters do not map directly the decision logic of the

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.

model, making the underlying model difficult to inspect or recover. Moreover, generic ansätze can become difficult to optimize as their size increases, with barren plateaus causing gradient information to vanish in large or expressive circuits (McClean et al., 2018; Larocca et al., 2025). Structured ansätze, whose circuit incorporates the inductive bias of the target task, have instead been shown to improve trainability in other quantum learning settings (Pesah et al., 2021; Larocca et al., 2022).

These considerations motivate incorporating the decision structure of an ODT directly into the quantum circuit. In particular, we introduce Quantum Oblivious Decision Trees (QODT), a variational quantum framework for representing and learning ODTs. The central building block is a phase-based quantum comparator, introduced by Berti & Poggiali (2026) and described in Section 3.3, which implements the fundamental operation of a decision tree: comparing a feature value with a learnable threshold. Within this framework, we study three alternative quantum representations of the same underlying ODT decision model. The first, *general*, follows the conventional VQA paradigm, combining angle encoding with a generic ansatz, and represents the decision structure implicitly. The second, *structured*, represents each tree level through a dedicated quantum comparator, making its feature-threshold pair explicit while a variational component maps the resulting comparison pattern to the prediction. The third, *recycle*, represents the same sequence of decisions using a single comparator qubit reused across all levels through mid-circuit measurement and reset, trading qubit count for circuit depth and classical feed-forward operations. A crucial point is that the *structured* and *recycle* circuits return DTs which are interpretable by design even though learned using a quantum circuit. Indeed, the learned feature-threshold pairs can be extracted to reconstruct an equivalent classical oblivious decision tree.

We evaluate the three quantum tree representations on twelve binary tabular datasets against classical oblivious and non-oblivious tree baselines. Since a depth-d ODT tests exactly d features, each tree is fed a small set of selected features. Inputs of a handful of features are the norm in Quantum Machine Learning (QML) on classical data, where qubit counts and the cost of classical simulation limit the input size (Havlíček et al., 2019; Pérez-Salinas et al., 2020; Abbas et al., 2021; Hur et al., 2022; Bowles et al., 2024). Even without this limit, however, few features per tree would remain preferable, as shallower trees yield shorter and more interpretable decision rules (Cherkauer & Shavlik, 1996; Domingos, 1999; Endou & Zhao, 2002). Beyond predictive performance, we analyze trainability, gradient behavior, qubit requirements, and scaling with tree depth, characterizing the trade-offs between generic expressivity, structural interpretability, and quantum resource efficiency. We show that QODT representations are not significantly separated from the baselines in terms of accuracy, while *recycle* is the most favorable of the three in terms of qubit requirements, gradient scaling with depth, and faithfulness of the extracted thresholds.

The remainder of the paper reviews related work (Section 2) and preliminaries (Section 3), describes the proposed quantum tree representations (Section 4), presents the experimental setup and results (Section 5), and concludes with limitations and future directions (Section 6).

2 Related Work

While decision tree induction is traditionally performed by greedy recursive partitioning (Quinlan, 2014), differentiable approaches formulate tree learning as continuous optimization, replacing hard split decisions with soft routing functions so that the tree parameters can be learned jointly by gradient descent (Marton et al., 2024; Panda et al., 2024) and trained end to end. A related simplification is the *Oblivious Decision Tree* (ODT) (Lou & Obukhov, 2017; Langley & Sage, 1994), in which all internal nodes at the same depth perform the same split, so a depth-d tree needs only d splitting rules rather than one per internal node. The constraint reduces the parameter count and regularizes the structure while retaining hierarchical partitioning, and has been exploited in gradient boosting and related methods (Prokhorenkova et al., 2018). Differentiable formulations have also been introduced for ODTs (Yang et al., 2018; Popov et al., 2019), combining the shared splitting structure of ODTs with soft routing so that the feature-selection and threshold parameters of each level can be learned by gradient descent, i.e., demonstrating that the structural constraint of obliviousness is compatible with end-to-end differentiable training.

In quantum computing, DTs have been explored by mapping their hierarchical structure to quantum circuits. Farhi & Gutmann (1998) framed tree evaluation as a quantum query problem, showing that a quantum walk traverses certain trees exponentially faster than any classical algorithm, and Lu &

Braunstein (2014) proposed a quantum analogue of the tree classifier based on a quantum entropy impurity criterion. More recently, Heese et al. (2022) represented nodes, branches and decision rules through quantum states and unitaries, realizing a probabilistic traversal of a binary classification tree on hardware. Others encode the tree-construction process itself: Khadiev et al. (2019) quantize the classical C5.0 algorithm, and Kumar et al. (2025) introduce Des-q, which provably speeds up tree retraining. Together these establish a connection between the hierarchical structure of trees and the compositional nature of quantum circuits, but the problem of designing trainable circuits that faithfully encode a DT remains open.

Variational Quantum Algorithms (VQAs) (Cerezo et al., 2021) combine parametrized quantum circuits with classical optimization, and have attracted particular interest for Noisy Intermediate-Scale Quantum (NISQ) devices (Preskill, 2018), where circuit depth, noise and qubit count rule out fault-tolerant algorithms. They have been applied widely in QML, but their advantages over classical approaches remain an open question and depend strongly on the problem, the circuit architecture and the hardware regime. A major practical challenge is *trainability*: generic, highly expressive ansätze can exhibit *barren plateaus*, regions in which the variance of the cost gradient decreases exponentially with the number of qubits (McClean et al., 2018; Larocca et al., 2025), and gradient vanishing can also arise from circuit depth, noise or an unsuitable parametrization. A common strategy to mitigate these limitations is to employ structured and problem-aware ansätze, whose circuit incorporates prior knowledge about the structure of the target problem (Pesah et al., 2021; Larocca et al., 2022). By restricting the variational search space to circuits that are consistent with the underlying task, such ansätze can reduce the number of trainable parameters, improve trainability, and potentially avoid the unfavorable optimization landscapes associated with generic circuits. This principle has been explored in several QML settings, where the circuit structure is explicitly designed to reflect the computational model being optimized. In particular, in (Poggiali et al., 2025), problem-aware ansätze are introduced for the variational learning of DTs, with the circuit architecture designed to encode the hierarchical structure of the tree itself.

3 PRELIMINARIES

In this section, we introduce the basic notions used throughout the paper. Let $\langle X, Y \rangle$ denote a labeled dataset of n instances, where $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subseteq \mathcal{X} \subseteq \mathbb{R}^m$ is the input space with m features, and $Y = \{y_1, \ldots, y_n\} \subseteq \mathcal{Y}$ is the corresponding set of labels. Each instance $\mathbf{x}_i = \{x_1, \ldots, x_m\}$ is described by m feature values, with x_j denoting the j-th feature. In this work, we focus on binary classification, with $\mathcal{Y} \subseteq \{0, 1\}$.

3.1 DECISION AND OBLIVIOUS TREES

A DT consists of internal decision nodes $\mathcal{N}$ and prediction nodes $\mathcal{L}$, i.e., the leaves of the tree. In the binary classification setting, each leaf $l \in \mathcal{L}$ predicts a class $\hat{y}_l \in \{0, 1\}$, while each internal node $v \in \mathcal{N}$ is associated with a split function $\delta(\cdot; t_v, \kappa_v) : \mathcal{X} \to \{0, 1\}^b$, parameterized by thresholds $t_v \in \mathbb{R}^b$ and a feature index $\kappa_v \in [1, m]$, where $b + 1$ is the number of outgoing branches of each internal node and we consider $b = 1$ throughout this work. In this setting routing is deterministic, t_v is a scalar, and the routing mechanism is $\delta(\mathbf{x}; t_v, \kappa_v) = \mathbb{I}[x_{\kappa_v} < t_v]$, with $\mathbb{I}[\cdot]$ the indicator function. A prediction is obtained by traversing a root-to-leaf path from $v = \text{root}$ to a leaf l.

ODTs are a subclass of DTs in which all internal nodes at the same depth share a splitting feature and threshold. Thus, an ODT of depth d is determined by d splitting conditions, one per level, where the split at level i is defined by a feature index κ_i and a threshold t_i, and evaluates the binary predicate

$$\delta_i(\mathbf{x}) = \mathbb{I}[x_{\kappa_i} < t_i], \qquad i \in [0, d-1],$$

where the feature index $\kappa_i \in [1, m]$ is present at most once in the ODT. The sequence of d binary outcomes determines one of the 2^d leaves. Where a classical DT admits up to $2^d - 1$ distinct splitting conditions over its $2^d - 1$ internal nodes, an ODT is restricted to d, i.e., one per level, and the split depends only on the level, not on the path, making the order of feature indexes within the splitting conditions irrelevant. A depth-d ODT can therefore be stored as a table with 2^d entries indexed by the outcomes of d independent comparisons, which makes inference cheap and, with ensembling, reduces overfitting (Prokhorenkova et al., 2018).

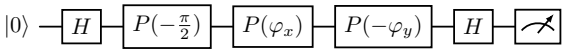

Figure 1: The single-qubit phase comparator of Berti & Poggiali (2026) (Section 3.3).

3.2 Variational Quantum Algorithms

The quantum component of a VQA is a Parameterized Quantum Circuit (PQC), or ansatz, denoted by $U(\theta)$ with $\theta = (\theta_1, \ldots, \theta_p) \in \mathbb{R}^p$ a vector of trainable parameters. Acting on the computational basis state $|0\rangle$, it prepares $|\psi(\theta)\rangle = U(\theta)\,|0\rangle$, and the quality of a parameter configuration is evaluated through a cost function $C(\theta)$ estimated from measurements of that state. Training proceeds as an iterative hybrid loop: the quantum processor evaluates $C(\theta^{(t)})$, a classical optimizer returns an updated vector $\theta^{(t+1)} = \mathcal{O}(\theta^{(t)}, C(\theta^{(t)}))$, and the circuit is executed again until a stopping criterion is met, so that the overall problem is $\theta^* = \operatorname{argmin}_{\theta \in \mathbb{R}^p} C(\theta)$.

3.3 Quantum Comparator

In (Berti & Poggiali, 2026) is introduce a quantum *comparator* for continuous-valued inputs represented as phases. Unlike conventional quantum comparators, which typically operate on integer values stored in computational-basis registers, this construction compares two values directly through phase interference, using a single ancillary qubit. The basic circuit, depicted in Figure 1, consists of two Hadamard gates and three phase gates,

$$P(\theta) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}, \qquad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Starting from $|0\rangle$, the first Hadamard gate creates an equal superposition of $|0\rangle$ and $|1\rangle$. The three phase gates act on the $|1\rangle$ component, with the first phase fixed to $-\pi/2$ and the remaining two phases encoding the values to be compared. Consequently, the relative phase accumulated between the two components is $\varphi = -\pi/2 + \varphi_x - \varphi_y$, where φ_x and φ_y denote the encoded values. After the second Hadamard gate, measuring the qubit in the computational basis gives

$$P(|0\rangle) = \frac{1}{2} + \frac{1}{2}\sin(\varphi_x - \varphi_y). \tag{1}$$

The fixed phase $-\pi/2$ shifts the interference pattern so that $P(|0\rangle) = 1/2$ exactly when $\varphi_x = \varphi_y$; provided $|\varphi_x - \varphi_y| < \pi$, the order of the two encoded values follows from the measurement probability as $P(|0\rangle) < 1/2 \iff \varphi_x < \varphi_y$.

4 Quantum Oblivious Decision Trees

We seek to represent the ODT decision process within a variational quantum model while retaining a correspondence between the learned quantum parameters and the splitting conditions of the underlying ODT. Without loss of generality, we index the d input features of a QODT by level, i.e., $\kappa_i = i$ and $x_{\kappa_i} = x_i$, for $i \in [0, d-1]$: as the order of the splitting conditions is irrelevant (Sec. 3.1), any subset of d distinct features can be relabeled in this way. The choice of the subset itself is left to the experiments. Figure 2 shows the three proposed quantum circuit. They share the same prediction task and variational learning framework, but differ in how the ODT decision process is encoded.

General. The *general* circuit (Fig. 2a) follows the conventional VQA paradigm and does not explicitly encode the ODT structure. Given an input instance $\mathbf{x}_i$, each feature x_j is encoded into a separate qubit through an R_Y rotation. The resulting state is subsequently processed by a parameterized ansatz $U(\theta)$, which learns the mapping from the encoded input to the prediction. The model output is obtained from the expectation value of the Pauli-Z observable on a designated output qubit $\hat{y}$. The decision function is represented implicitly by the parameters of the generic ansatz, with no circuit component corresponding to a tree level, split feature, or threshold.

Structured. The *structured* circuit (Fig. 2b) incorporates the ODT decision structure directly into the quantum circuit. Each tree level is represented by one instance of the phase-based quantum

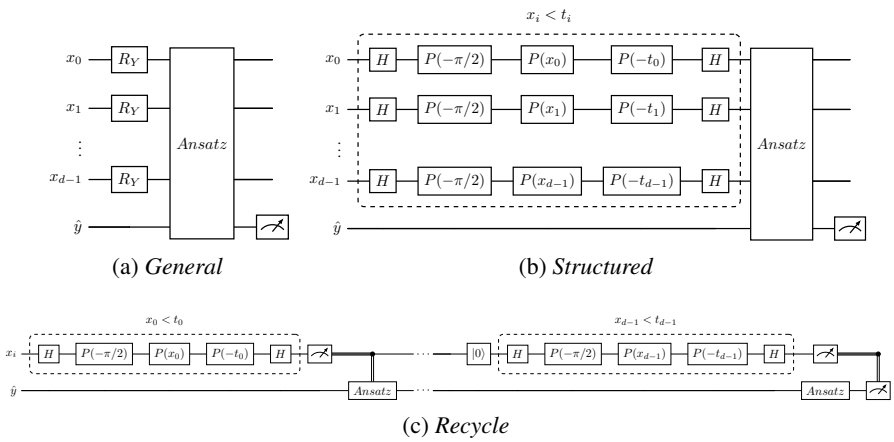

Figure 2: The three proposed QODT circuits. *General* (a) encodes each of the d selected features in an R_Y rotation and applies a generic entangling ansatz. *Structured* (b) replaces that encoding with one phase comparator per tree level, so that qubit i carries the outcome of the learned comparison $x_i < t_i$. *Recycle* (c) performs the same d comparisons on a single qubit, measuring and resetting it after each one, with the measured bit classically selecting which ansatz block acts on the output register. In all three panels the ansatz is drawn on a single output qubit for clarity; the number of additional ansatz qubits w is a hyperparameter.

comparator introduced in Section 3.3. Consequently, a depth-d ODT is represented by d comparator qubits, with each comparator corresponding to one splitting condition. For level i, the comparator compares the feature x_i with a learnable threshold t_i. Under the condition required by the comparator, $|\varphi_x - \varphi_y| < \pi$, its measurement probability represents the predicate $x_i < t_i$. Hence, the d comparator qubits jointly encode the sequence of decisions

$$\mathbf{b} = (b_0, \ldots, b_{d-1}), \qquad b_i = \delta_i(\mathbf{x}) = \mathbb{I}[x_i < t_i].$$

Since the comparisons at different levels are independent, they can be evaluated in parallel. The ansatz consequently learns the mapping from the 2^d possible comparison patterns to the output, while the feature-threshold comparisons are represented explicitly by the comparator parameters.

Recycle. The *recycle* circuit (Fig. 2c) uses the same sequence of ODT comparisons but represents them sequentially on a single comparator qubit. Instead of maintaining one qubit for every tree level, the comparator qubit is measured and reset after each comparison and subsequently reused for the next level. At level i, the comparator evaluates the splitting condition

$$b_i = \mathbb{I}[x_i < t_i],$$

and the resulting measurement outcome is stored classically. After all d comparisons, the complete decision path is therefore represented by the classical bit string $\mathbf{b}$ rather than by a d-qubit register. The variational ansatz is conditioned on these measurement outcomes. For each level i, we define two independent ansatz blocks, $V_{i,0}$ and $V_{i,1}$, corresponding to the two possible outcomes of the split. Only the block selected by the measurement outcome is executed. The resulting sequence

$$V_{d-1}(b_{d-1}) \cdots V_1(b_1) V_0(b_0)$$

represents the path selected by the d comparisons. A final measurement of the designated output qubit produces the model prediction.

Thus, *recycle* preserves the ODT semantics of *structured* with one comparator qubit instead of d, at the cost of circuit depth and feed-forward: an extra level, and hence an extra feature, costs no additional qubit.

To isolate the effect of the QODT representation, all three circuits employ the same family of variational layers. We use the strongly entangling ansatz (Schuld et al., 2020), where each layer applies three parameterized single-qubit rotations to every ansatz qubit, followed by a ring of CNOT gates

connecting neighboring qubits. Denoting by L the number of layers and by w the number of additional qubits introduced by the ansatz, the parameter count is $L(d+w)\cdot 3$ for *general*, $d+L(d+w)\cdot 3$ for *structured* and $d(1 + Lw \cdot 6)$ for *recycle*, with d the tree depth.

4.1 TRAINING

All three circuits are trained with the hybrid loop of Section 3.2; for *structured* and *recycle*, θ additionally contains the thresholds parameters t_i. The feature-threshold comparisons and the parameters controlling the final prediction are learned jointly through the same variational optimization. This end-to-end formulation is important for the proposed QODT representations: thresholds are not obtained through a separate classical tree-induction algorithm and subsequently encoded into a quantum circuit. Instead, the decision boundaries are learned as part of the quantum model.

4.2 INTERPRETABILITY

A central property of the *structured* and *recycle* circuits is that the internal decision structure remains explicitly accessible after training. Each comparator is associated with a feature index, i.e.,

$$\{(i, t_i)\}_{i=0}^{d-1} .$$

Hence, the trained parameters directly define the d splits of an ODT. The quantum model can therefore be interpreted through two components: *(i)* the learned feature-threshold pairs, which define the decision structure, and *(ii)* the variational parameters that determine the final prediction from the resulting decision representation. To obtain a classical tree, we reconstruct the ODT from the extracted feature-threshold pairs and assign each leaf a class label according to the majority class of the training samples reaching that leaf. The resulting tree can then be evaluated and inspected using entirely classical procedures. This reconstruction provides an explicit and model-level notion of interpretability: the decision boundaries learned by the quantum model can be expressed in the same feature-threshold language as a classical ODT. Section 5.2 evaluates the predictive agreement between the quantum models and their reconstructed trees and studies the resulting interpretability across datasets and QODT representations.

5 EXPERIMENTS

We evaluate the proposed QODT representations on twelve binary classification datasets commonly adopted as benchmarks for decision tree methods; their characteristics are summarized in Table 2 in Appendix A. For each dataset, all features are mapped to a uniform marginal distribution by a quantile transform, as suggested in Popov et al. (2019), and then rescaled to the interval $[-1, 1]$, and then encoded as rotation angles in the phase gates of the quantum circuit. This normalization ensures that the phase differences satisfy the comparator requirement $|\varphi_x - \varphi_y| < \pi$ (see Sec. 3.3).

In Section 5.1, we evaluate QODT against two differentiable oblivious tree models, i.e., Deep Neural Decision Trees (DNDT) (Yang et al., 2018) and Neural Oblivious Decision Ensembles (NODE) (Popov et al., 2019), as well as a standard axis-aligned Decision Tree (DT) (Breiman et al., 1984). We additionally consider two black-box ensembles, Random Forest (RF) (Breiman, 2001), and CatBoost (Prokhorenkova et al., 2018) whose base learners are oblivious trees, to assess performance both at the level of a single tree and when exploiting bagging-based ensembles. Moreover, since a depth-d ODT tests only d features, each tree relies on a subset of d features obtained through feature selection (details in Appendix B).

The train/validation/test partition is fixed across all runs (80%-10%-10% for every dataset), so the standard deviations we report reflect seed variability alone and not variability of the split. QODT hyperparameters, i.e., the optimizer $\mathcal{O}$, the learning rate, the number of ansatz layers L and the number of additional ansatz qubits w, are selected on the validation split by mean accuracy over the seeds. The test split is evaluated once, at the selected configuration. Unless stated otherwise, circuit results are means over 5 seeds of 100-epoch runs with batch size 16 (More details on the experiments presented in the paper, as well as on the auxiliary experiments, are provided in Appendix B.). Section 5.2 assesses the interpretability of *structured* and *recycle*, while additional results are in Appendix D for trainability and resource estimations, and in Appendix E for loss evaluations.

Table 1: One tree per model. Test accuracy (mean ± s.d. over 5 seeds) for the validation-selected configuration, using the same four mutual-information features for all models. Each column is a depth-4 tree with at most 16 leaves (baseline settings in Appendix B). Best per dataset in bold.

Dataset	Majority	*general*	*structured*	*recycle*	DNDT	NODE	CatBoost	RF	DT
australian	.555	.907 ± .012	**.913 ± .000**	**.913 ± .000**	**.913 ± .000**	**.913 ± .000**	**.913 ± .000**	.899 ± .022	**.913 ± .000**
banknote	.555	.887 ± .022	.928 ± .011	.932 ± .006	.961 ± .021	**.991 ± .007**	.899 ± .021	.936 ± .037	.928 ± .000
breast	.655	.909 ± .011	.951 ± .007	**.969 ± .006**	.949 ± .015	.957 ± .009	.949 ± .015	.966 ± .007	.957 ± .000
churn	.735	.770 ± .003	.776 ± .006	.763 ± .004	**.784 ± .002**	.769 ± .010	.766 ± .020	.766 ± .013	.780 ± .000
compas	.656	.693 ± .011	**.713 ± .007**	.710 ± .003	.710 ± .003	.706 ± .006	.697 ± .007	.698 ± .009	.702 ± .000
egg	.551	.677 ± .008	.676 ± .003	.667 ± .004	.653 ± .013	**.692 ± .011**	.648 ± .016	.666 ± .012	.648 ± .000
fico	.522	.711 ± .013	.715 ± .001	.711 ± .003	.710 ± .003	**.715 ± .005**	.709 ± .005	.707 ± .007	.705 ± .000
german	.713	**.715 ± .019**	.652 ± .014	.648 ± .008	.667 ± .013	.671 ± .008	.665 ± .012	.671 ± .019	.708 ± .000
heart	.556	.474 ± .100	.563 ± .015	.659 ± .079	.652 ± .030	**.674 ± .015**	.630 ± .041	.637 ± .054	.667 ± .000
lymph	.570	.680 ± .050	**.733 ± .000**	**.733 ± .000**	**.733 ± .000**	**.733 ± .000**	.720 ± .027	.707 ± .033	**.733 ± .000**
pima	.651	.691 ± .025	.675 ± .012	**.735 ± .016**	.678 ± .005	.712 ± .005	.678 ± .013	.686 ± .021	.675 ± .000
sonar	.534	**.714 ± .052**	.676 ± .047	.705 ± .019	.695 ± .049	.695 ± .023	.705 ± .036	.667 ± .052	**.714 ± .000**
mean	.604	.736	.748	.762	.759	**.769**	.748	.750	.761

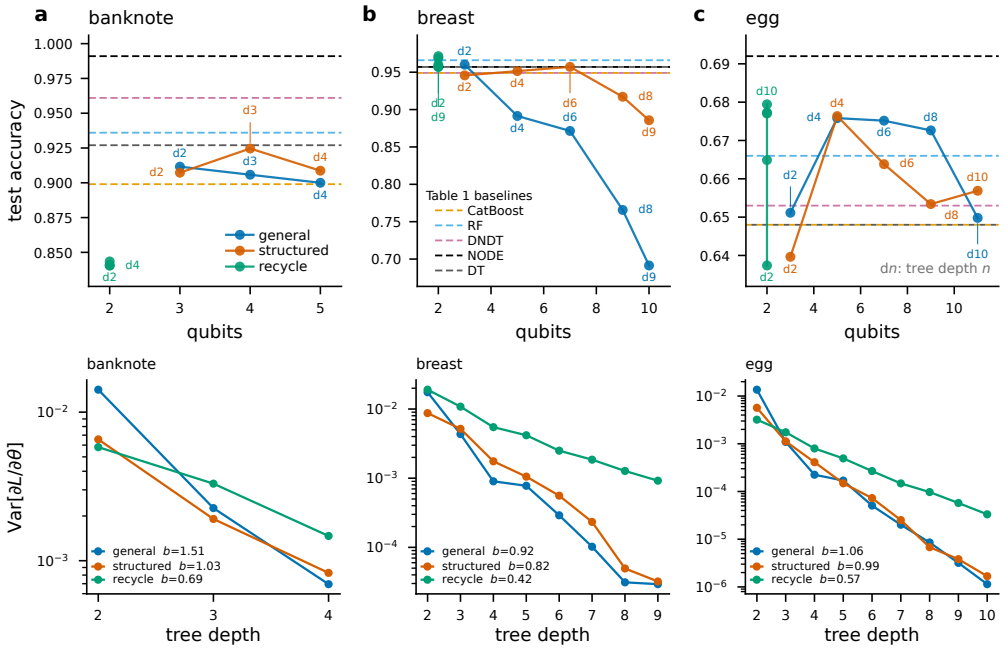

Figure 3: Scaling with tree depth. Top: test accuracy against the total number of qubits as depth grows (labels give the depth; dashed lines are the baselines of Table 1). Bottom: gradient variance at random initialization against depth, $w = 2$, with fitted decay exponent b; Figure 11 adds $w = 1$.

All experiments are implemented using PennyLane (Bergholm et al., 2018); threshold parameterization, training implementation and the depth-sweep protocol are given in Appendix B. The main grid fixes the tree depth to $d = 4$. This choice is motivated by interpretability: shallower trees yield more concise and comprehensible decision rules (Cherkauer & Shavlik, 1996; Domingos, 1999; Endou & Zhao, 2002). It is also in line with variational QML on classical data, where inputs are typically restricted to a small number of features, whether selected, or compressed (Havlíček et al., 2019; Mari et al., 2020; Abbas et al., 2021; Hur et al., 2022): qubit counts and the exponential cost of the simulation limit even large numerical studies and systematic benchmarks to tens of features (Huang et al., 2021; Bowles et al., 2024). Deeper trees are studied in the depth sweep (Appendix D).

5.1 Performance Evaluation

Table 1 reports predictive performance with every model reduced to a single depth-4 tree. Averaged over the twelve datasets, *recycle* (0.762) ranks second among the eight models, ahead of four of the five baselines and behind only NODE (0.769), while *general* (0.736) is the weakest.

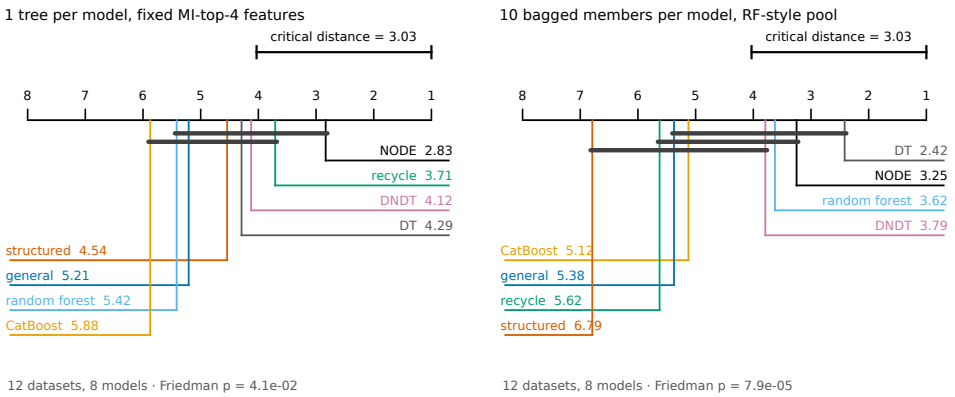

Figure 4: Average rank across all twelve datasets for the three QODT representations and five baselines: one tree each on fixed features (left) and 10 bagged members each on the Random-Forest-style pool (right). The bar at the top of each panel is the critical distance of the Nemenyi test at $\alpha = 0.05$; horizontal bars join models whose mean ranks differ by less than it.

While accuracy at depth $d = 4$ barely separates the three circuits, their behavior as depth grows differs sharply (Figure 3; details in Appendices D and E). A depth-d tree requires $d + w$ qubits under *general* and *structured* but only $1 + w$ under *recycle*, and *recycle*'s accuracy does not degrade as depth grows: on `breast`, as depth grows to 9, *general* loses 0.269 accuracy, while *recycle* stays on two qubits and never falls below its depth-2 accuracy. Moreover, the gradient variance of *recycle* decays with depth at roughly half the rate of the other two circuits (more details in Appendix E). The depth-4 results (Tab. 1 and Fig. 3) therefore represent the setting in which the three circuits are closest, not the one that separates them.

Single trees on a fixed set of four features isolate the effect of the representation. However, tree models are frequently used as ensembles, which reduce variance and allow the model to exploit more features than a single shallow tree can test (Breiman, 2001). To assess whether our conclusions hold in this regime, which is also the natural one for baselines such as RF and CatBoost, we bag each model into ensembles of 1, 3, 5 and 10 members, each trained on the four features with the highest mutual information within a random candidate pool of $\hat{m}$ features (Appendix C). The ensemble size is capped at 10 because simulating the circuits is costly. At ten members the ordering changes (Table 6 and Figure 6 in Appendix C): bagged decision trees lead (0.793), while *recycle* (0.768) and *structured* (0.759) occupy the last two positions. The two structured circuits benefit least from bagging ($+0.031$ and $+0.026$), whereas *general*, which starts worst, gains most ($+0.073$).

Figure 4 reports the critical difference plots. With one tree per model (Figure 4, left), the Friedman test narrowly rejects the null hypothesis of equal ranks ($p = 0.041$), but no circuit is significantly separated from any baseline under the Nemenyi test, and *recycle* ranks second. With ten bagged members (Figure 4, right), the null is rejected far more decisively ($p = 7.9 \times 10^{-5}$): bagged decision trees rank first and are significantly separated from *recycle* and *structured*, but not from *general*.

5.2 Interpretability

Figure 5 quantifies the predictive accuracy retained when the learned *structured* and *recycle* circuits are converted into classical ODTs. The comparison isolates the cost of the extraction step: it contrasts the accuracy of each quantum circuit with that of the classical tree reconstructed from its learned feature–threshold decisions. Any change in accuracy therefore measures the effect of replacing the circuit's leaf-value computation with the classical ODT leaf computation described in Sec. 4.2. Averaged over the twelve datasets the reconstruction loses 0.029 accuracy for *structured* and 0.016 for *recycle*. The loss is long-tailed rather than uniform: on `banknote` *structured* reaches 0.928 while its extracted tree reaches 0.688, and on seven of the twenty-four (dataset, circuit) cells it is negative, that is, the tree is more accurate than the circuit came from.

The two circuits do not deliver this property on the same terms. In the *structured* circuit, the threshold enters as a single-qubit phase rotation, which is followed by a general single-qubit rotation in the

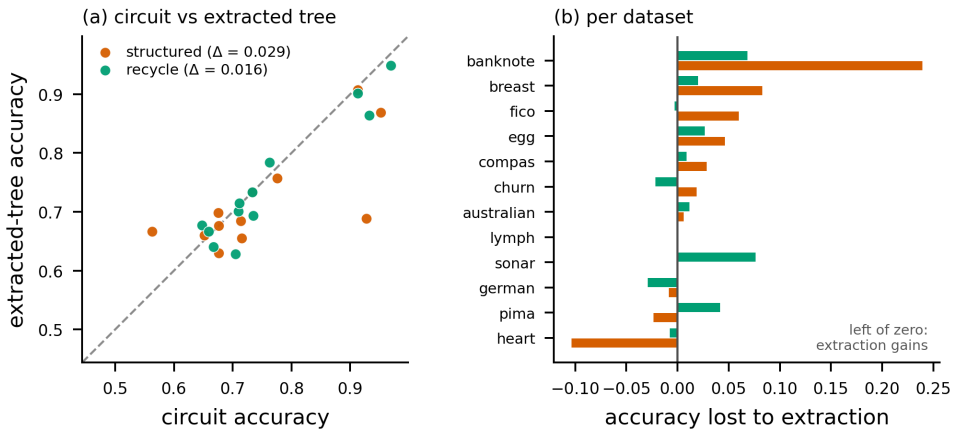

Figure 5: Circuit accuracy against extracted-tree accuracy, on all twelve datasets. (a) Points below the diagonal are datasets where the extraction loses accuracy; Δ in the legend is the mean circuit-minus-tree difference. (b) The per-dataset extraction loss, ordered by the *structured* value; bars left of zero are the seven of twenty-four (dataset, circuit) cells where the extracted tree is more accurate.

first strongly entangled layer. Since the latter can reparameterize the resulting single-qubit unitary, the threshold is not uniquely identifiable from the loss: different threshold values can be compensated by corresponding changes in the ansatz parameters, leaving the implemented single-qubit transformation, and hence the classifier, unchanged. Measured within each dataset over the full grid, the correlation between the accuracy of the extracted tree and that of the circuit it came from is -0.09 for *structured*, indicating essentially no relationship, compared with $+0.15$ for *recycle*. In *recycle*, by contrast, the threshold determines the branch probability of a classically controlled feed-forward operation and therefore directly affects the subsequent computation. The extracted threshold consequently remains tied to the model's behavior. Threshold extraction is thus a more direct property of *recycle*, whereas for *structured* it is only meaningful in a weaker, parameterization-dependent sense.

6 Conclusion

We introduced QODT, a variational formulation of oblivious decision tree learning, comparing a generic hardware-efficient circuit (*general*) against two circuits that make the ODT structure explicit, *structured* and *recycle*. While the three QODT representations show no significant separation from the baselines in accuracy, the two interpretable circuits look similar on accuracy but diverge sharply on resource efficiency and interpretability. As a single depth-4 tree, *recycle* is competitive with baselines, though it is among the two that gain least from bagging and falls behind every baseline at ten members. With increasing tree depth, *recycle* keeps its qubit count fixed without losing accuracy, while *general* and *structured* require one additional qubit per level, and only *general* degrades substantially. The gradient variance of *recycle* also decays at roughly half their rate.

Several limitations bound these conclusions. All results are obtained by classically simulating the circuits, so we model neither hardware noise nor the cost of mid-circuit measurement and feed-forward on which *recycle* relies. The study is also small in scale (depth 4 on the main grid, ensembles of at most ten members, binary tasks only). We make no claim of accuracy advantage. The gradient decay of *recycle* is mitigated, not eliminated, and its parameter count is higher. Interpretability is approximate, as extraction loses 0.029 (*structured*) and 0.016 (*recycle*) accuracy on average. Natural next steps include: *(i)* running *recycle* on dynamic-circuit hardware with noise-aware training; *(ii)* replacing the strongly entangling ansatz with a tree-structured one, which could reduce *recycle*'s parameter overhead, remove the threshold redundancy in *structured*, and extend interpretability to the leaf-assignment stage; and *(iii)* learning feature selection per level for end-to-end ODT training. Despite these limitations, this work shows that a DT's structure can be built directly into a VQA to yield interpretable-by-design models, and that combining phase-based comparators with mid-circuit measurement offers a qubit-efficient path with more favorable gradient scaling at depth.

REPRODUCIBILITY STATEMENT

All code, data and results needed to reproduce this paper are provided as supplementary material (a zip archive). The archive contains the implementation of the three circuits and of the training pipeline; the scripts that run every experiment reported in the paper (the main grid, the 20-epoch pilot grid, the depth sweep, the gradient-variance scan, the loss-landscape, Hessian and magic analysis, the finite-shot evaluation, the classical baselines and the bagged ensembles), with pinned requirements; the twelve datasets with the fixed 80%-10%-10% splits; and the result files of all these experiments. From the shipped results, the analysis scripts regenerate every table and figure of the paper and the numerical results quoted in the text in about one minute on a laptop, without retraining. The included README maps each table and figure to the script that produces it and lists the commands, in dependency order, to re-run all experiments from scratch (the main grid alone takes about 370 CPU-hours). The experimental protocol (quantile-transformed features, mutual-information feature selection on the training split only, $\tanh$-reparameterized thresholds, validation-based model selection with a single test evaluation, five seeds) is described in Section 5 and Appendix B, the hyperparameter grids in Tables 3–5, the circuits and their parameter counts in Section 4, and the datasets in Table 2. Per-epoch parameter snapshots of all runs are not included for size reasons, except those needed for the trajectory overlays of Figure 10; finite-shot sampling is not seeded, so that experiment is reproduced statistically.

AI USE STATEMENT

In this work, we used generative AI tools, specifically Claude Opus 5.5 (Anthropic), for three tasks: drafting parts of the paper, rephrasing and editing the text to improve readability, and implementing code, including the experiment scripts, the analysis scripts and the figure-generation code. The research questions, the design of all three circuits, and our adaptation of the comparator construction on which they rely are entirely our own work. We have reviewed all AI-assisted work: all AI-drafted or AI-edited text was read and revised by the authors, and all AI-written code was inspected and its outputs checked against the reported results. The authors take full responsibility for the content of this paper.

REFERENCES

Amira Abbas, David Sutter, Christa Zoufal, Aurelien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *Nature Computational Science*, 1(6):403–409, 2021. doi: 10.1038/s43588-021-00084-1.

Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, Shahnawaz Ahmed, Vishnu Ajith, M Sohaib Alam, Guillermo Alonso-Linaje, Bharath AkashNarayanan, Ali Asadi, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv preprint arXiv:1811.04968*, 2018.

Alessandro Berti and Alessandro Poggiali. A quantum phase-based comparator, 2026. URL `https://arxiv.org/abs/2609.25262`.

Joseph Bowles, Shahnawaz Ahmed, and Maria Schuld. Better than classical? the subtle art of benchmarking quantum machine learning models. *arXiv preprint arXiv:2403.07059*, 2024. doi: 10.48550/arXiv.2403.07059.

Sergey Bravyi and Alexei Kitaev. Universal quantum computation with ideal Clifford gates and noisy ancillas. *Physical Review A*, 71(2):022316, 2005.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

Kevin J. Cherkauer and Jude W. Shavlik. Growing simpler decision trees to facilitate knowledge discovery. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 315–318. AAAI Press, 1996.

Pedro Domingos. The role of Occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425, 1999. doi: 10.1023/A:1009868929893.

Taichirou Endou and Qiangfu Zhao. Generation of comprehensible decision trees through evolution of training data. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC)*, pp. 1221–1225. IEEE, 2002.

Edward Farhi and Sam Gutmann. Quantum computation and decision trees. *Physical Review A*, 58 (2):915, 1998.

Daniel Gottesman. The Heisenberg representation of quantum computers. *arXiv preprint quant-ph/9807006*, 1998.

Vojtěch Havlíček, Antonio D. Córcoles, Kristan Temme, Aram W. Harrow, Abhinav Kandala, Jerry M. Chow, and Jay M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019. doi: 10.1038/s41586-019-0980-2.

Raoul Heese, Patricia Bickert, and Astrid Elisa Niederle. Representation of binary classification trees with binary features by quantum circuits. *Quantum*, 6:676, 2022.

Hsin-Yuan Huang, Michael Broughton, Masoud Mohseni, Ryan Babbush, Sergio Boixo, Hartmut Neven, and Jarrod R. McClean. Power of data in quantum machine learning. *Nature Communications*, 12(1):2631, 2021. doi: 10.1038/s41467-021-22539-9.

Tak Hur, Leeseok Kim, and Daniel K. Park. Quantum convolutional neural network for classical data classification. *Quantum Machine Intelligence*, 4(1):3, 2022. doi: 10.1007/s42484-021-00061-x.

Kamil Khadiev, Ilnaz Mannapov, and Liliya Safina. The quantum version of classification decision tree constructing algorithm c5. 0. *arXiv preprint arXiv:1907.06840*, 2019.

Niraj Kumar, Romina Yalovetzky, Changhao Li, Pierre Minssen, and Marco Pistoia. Des-q: a quantum algorithm to provably speedup retraining of decision trees. *Quantum*, 9:1588, 2025.

Pat Langley and Stephanie Sage. Oblivious decision trees and abstract cases. In *Working notes of the AAAI-94 workshop on case-based reasoning*, pp. 113–117. Seattle, WA, 1994.

Martin Larocca, Piotr Czarnik, Kunal Sharma, Gopikrishnan Muraleedharan, Patrick J Coles, and Marco Cerezo. Diagnosing barren plateaus with tools from quantum optimal control. *Quantum*, 6:824, 2022.

Martin Larocca, Supanut Thanasilp, Samson Wang, Kunal Sharma, Jacob Biamonte, Patrick J Coles, Lukasz Cincio, Jarrod R McClean, and Zoë Holmes. Barren plateaus in variational quantum computing. *Nature Reviews Physics*, 7(4):174–189, 2025.

Lorenzo Leone, Salvatore F. E. Oliviero, and Alioscia Hamma. Stabilizer rényi entropy. *Physical Review Letters*, 128(5):050402, 2022.

Yin Lou and Mikhail Obukhov. Bdt: Gradient boosted decision tables for high accuracy and scoring efficiency. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1893–1901, 2017.

Songfeng Lu and Samuel L Braunstein. Quantum decision tree classifier. *Quantum information processing*, 13(3):757–770, 2014.

Andrea Mari, Thomas R. Bromley, Josh Izaac, Maria Schuld, and Nathan Killoran. Transfer learning in hybrid classical-quantum neural networks. *Quantum*, 4:340, 2020. doi: 10.22331/q-2020-10-09-340.

Sascha Marton, Stefan Lüdtke, 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.

Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018.

Subrat Prasad Panda, Blaise Genest, Arvind Easwaran, and Ponnuthurai Nagaratnam Suganthan. Vanilla gradient descent for oblique decision trees. In *ECAI 2024: 27th European Conference on Artificial Intelligence, 19–24 October 2024, Santiago de Compostela, Spain–Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, pp. 1140–1147. SAGE Publications Pvt. Ltd 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2024.

Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020. doi: 10.22331/q-2020-02-06-226.

Arthur Pesah, Marco Cerezo, Samson Wang, Tyler Volkoff, Andrew T Sornborger, and Patrick J Coles. Absence of barren plateaus in quantum convolutional neural networks. *Physical Review X*, 11(4):041011, 2021.

Alessandro Poggiali, Giacomo Antonioli, Alessandro Berti, Gianna M Del Corso, and Riccardo Guidotti. Variational decision trees with structured ansatzes. In *2025 IEEE International Conference on Quantum Computing and Engineering (QCE)*, volume 2, pp. 428–429. IEEE, 2025.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *arXiv preprint arXiv:1909.06312*, 2019.

John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

Manuel S Rudolph, Sukin Sim, Asad Raza, Michal Stechly, Jarrod R McClean, Eric R Anschuetz, Luis Serrano, and Alejandro Perdomo-Ortiz. Orqviz: Visualizing high-dimensional landscapes in variational quantum algorithms. *arXiv preprint arXiv:2111.04695*, 2021.

Maria Schuld, Alex Bocharov, Krysta M Svore, and Nathan Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

Yongxin Yang, Irene Garcia Morillo, and Timothy M Hospedales. Deep neural decision trees. *arXiv preprint arXiv:1806.06988*, 2018.

A DATASETS

We evaluate QODT on twelve binary tabular datasets commonly used to benchmark decision tree methods. They span from 142 to 14,980 samples and from 4 to 60 features, with majority-class rates between 52% and 73%. Table 2 summarizes their characteristics.

Table 2: Dataset characteristics. The majority-class rate is computed over the whole dataset rather than per split.

Dataset	Num. Samples	Num. Features	Num. Classes	Majority Class (%)
`pima`	768	8	2	65.10
`australian`	690	38	2	55.51
`lymph`	142	47	2	57.04
`breast`	699	9	2	65.52
`heart`	270	20	2	55.56
`compas`	10558	12	2	65.65
`german`	959	23	2	71.32
`churn`	7043	40	2	73.46
`sonar`	208	60	2	53.37
`fico`	10459	23	2	52.19
`banknote`	1372	4	2	55.54
`egg`	14980	14	2	55.12

B PROTOCOL DETAILS

This appendix details the experimental protocol of Section 5: feature selection, model selection, threshold parameterization, training implementation, baseline settings and hyperparameter search spaces.

Feature selection. Single trees use the d features with the highest mutual information with the label, estimated on the training split only. In the bagged ensembles, each member uses the top d features within its own random candidate pool (Appendix C).

Model selection. The circuit configurations of Table 1 and of the depth sweep are selected by mean validation accuracy over the seeds; ties are broken by validation ROC-AUC, then by the smaller qubit count, then by the smaller parameter count. The auxiliary analyses reuse the main-grid configuration with the highest mean validation accuracy, resolving exact ties by the lexicographic order of (optimizer, learning rate, L, w) for the ensembles and the finite-shot analysis; the landscape analyses select by the same criterion on the 20-epoch pilot grid, resolving ties by file order.

Threshold rescaling. At every optimization step, each threshold is mapped through $\tanh$ before being passed to the circuit, which confines it to the comparator domain while keeping its gradient nonzero everywhere.

Training implementation. All three circuits are trained through PennyLane's JAX interface. The mid-circuit measurements of *recycle* are made differentiable by deferring them to an ancilla register, so that a single training pipeline serves all three circuits and no implementation difference confounds the comparison.

Single-tree baselines. In Table 1, every model is a single depth-4 tree with at most 16 leaves: the circuits by construction, DNDT with `cuts` $= 1$, NODE with `num_trees` $= 1$, CatBoost with $n_{\text{estimators}} = 1$, RF as a single CART with `max_depth` $= 4$, and DT with `max_depth` $= 4$.

Depth sweep. The depth sweep on `banknote`, `breast` and `egg` uses a reduced 20-epoch protocol, selects the configuration on validation at each depth, and caps the depth by the number of available features (Appendix D).

Hyperparameter grids. Tables 3, 4 and 5 list the search spaces of the circuits, of the baselines and of the auxiliary experiments; every configuration was trained with all five seeds. The three circuits share the same optimizer and learning-rate grid, and *recycle* additionally includes $w = 4$, which matches its qubit count to that of *general* and *structured* at $w = 1$. The main grid comprises 10,080

runs of 100 epochs (168 configurations per dataset, 12 datasets, 5 seeds), for about 370 CPU-hours of serial simulation. Including the depth sweep (1,170 runs), the 20-epoch pilot grid (2,520 runs) and the circuit members of the bagged ensembles (1,800 runs), the study trains 15,570 circuits.

Table 3: Search space of the three circuits on the main grid (depth $d = 4$). Every configuration is trained with five seeds; qubit and parameter ranges are those spanned by the grid.

	general	*structured*	*recycle*
Optimizer	Adam, SGD, Nesterov SGD		
Learning rate (Adam, Nesterov SGD)	$\{0.005, 0.01, 0.05, 0.1\}$		
Learning rate (SGD)	$\{0.05, 0.1, 0.5, 1.0\}$		
Ansatz layers L	$\{1, 2\}$		
Additional ansatz qubits w	$\{1, 2\}$	$\{1, 2\}$	$\{1, 2, 4\}$
Total qubits	5–6	5–6	2–5
Trainable parameters	15–36	19–40	28–196
Configurations per dataset	48	48	72
Runs (12 datasets $\times$ 5 seeds)	2,880	2,880	4,320

Fixed: 100 epochs; batch size 16 with the training set reshuffled every epoch; binary cross-entropy on $p(y{=}1) = (1 - \langle Z \rangle)/2$ read on the last ansatz qubit; Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$; Nesterov momentum 0.9; constant learning rate, no weight decay or gradient clipping; ansatz weights initialized uniformly in $[-\pi, \pi]$ and threshold parameters t_i uniformly in $[-0.5, 0.5]$; evaluation at the epoch of minimum validation loss; exact expectation values in double precision (PennyLane `default.qubit`, JAX backpropagation).

Table 4: Search space of the classical baselines. All baselines receive the same four mutual-information features as the circuits; every configuration is trained with five seeds and selected on mean validation accuracy. The last column reports the one-tree setting used in Table 1.

Model	Searched	Fixed	Table 1
DNDT	learning rate $\{0.01, 0.05, 0.1\} \times$ cut points $\{1, 2\}$	Adam, 100 epochs, batch size 16, soft-binning temperature 0.1, minimum-validation-loss checkpoint	one cut point, with the learning rate selected for this setting
NODE	learning rate $\{0.003, 0.01\}$, selected separately for each number of trees in $\{1, 3, 5, 10\}$	depth 4, one layer, Adam, batch size 256, at most 100 epochs with early stopping (patience 10), best-validation-loss checkpoint	one tree, with the learning rate selected for this setting
CatBoost	learning rate $\{0.05, 0.1, 0.3, 0.5, 1.0\}$	depth 4, one tree, log-loss, library defaults otherwise	one tree
RF	none	one tree, maximum depth 4, bootstrap and $\sqrt{\cdot}$ feature subsampling (library defaults)	one tree
DT	none	maximum depth 4, Gini criterion	one tree

Table 5: Settings of the auxiliary experiments. Unless stated otherwise, training settings are those of Table 3.

Experiment	Settings
Bagged ensembles (Appendix C)	1, 3, 5 and 10 members, the smaller ensembles being the first members of the ten; four features per member, the top four by mutual information within a random pool of $\hat{m}$ features; every member reuses its model's validation-selected single-tree configuration (CatBoost members use learning rate 0.1) and is trained on the full training split; prediction by averaging member probabilities; five seeds; 4,800 member fits across the eight models, 1,800 of them circuits.
Depth sweep (Appendix D)	$d \in \{2, 3, 4\}$ on `banknote`, $\{2, 4, 6, 8, 9\}$ on `breast`, $\{2, 4, 6, 8, 10\}$ on `egg`; optimizer $\{$Adam, Nesterov SGD$\}$ $\times$ learning rate $\{0.01, 0.05, 0.1\}$, selected at each depth; $L = 2$, $w = 1$; 20 epochs; five seeds; 1,170 runs.
Gradient variance (Appendix E)	every integer depth from 2 to 4 (`banknote`), 9 (`breast`) and 10 (`egg`); $w \in \{1, 2\}$; $L = 2$; 200 random initializations per cell on a fixed probe of 128 training samples; 120 cells.
Landscapes, Hessian, magic (Appendix E)	checkpoints from a 20-epoch pilot grid with the search space of Table 3 on `banknote`, `breast` and `egg` (2,520 runs); 9 (dataset, circuit) cells $\times$ 5 seeds $= 45$ checkpoints; 30×30 scans over a random plane spanning $\pm\pi$ and over the trajectory plane; exact Hessian by automatic differentiation; M_2 estimated by Monte Carlo over Pauli strings.
Finite shots (Appendix D)	100-epoch checkpoints of the selected configurations, 12 datasets $\times$ 3 circuits $\times$ 5 seeds; 32, 100, 1 000 and 10 000 shots against exact expectation values; 20 shot realizations per budget.

C BAGGED ENSEMBLES

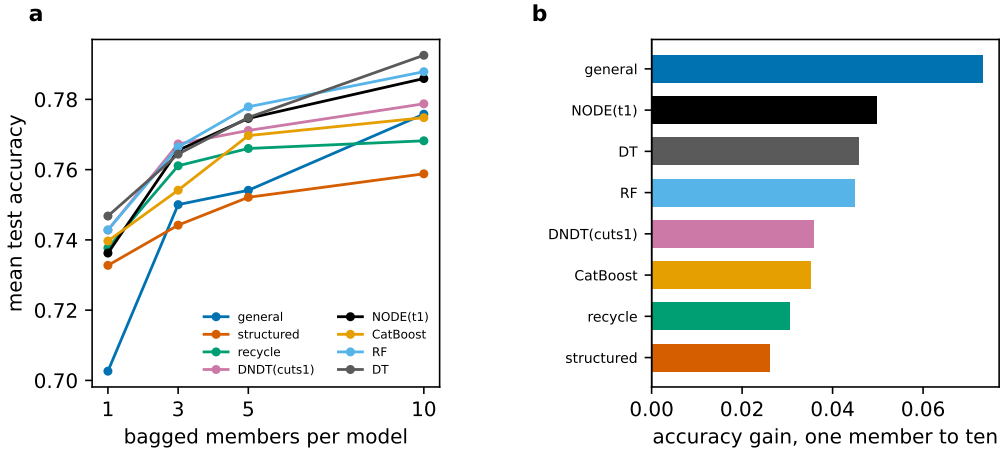

Figure 6: (a) Mean test accuracy against the number of bagged members. Members differ by feature subset and seed, except on `banknote`, where they differ by seed only. (b) Accuracy gain from one to ten members: *structured* and *recycle* gain least and *general* most, which explains why the ordering at ten members differs from that at one tree.

This appendix complements Section 5.1 with the construction of the bagged ensembles and their per-dataset results.

Each member draws a random candidate pool of $\hat{m} = \min\big(m,\ \max(5, \lceil 1.5\sqrt{m}\,\rceil)\big)$ features, following the square-root scaling of Random Forests, and keeps the top four by mutual information. This gives $\hat{m}$ between 5 and 12 on every dataset except `banknote`, whose four features leave noth-

ing to sample. All eight models use the same feature subsets, member for member. Since members draw their features from random pools rather than from the global top four, the one-member results differ from Table 1. From one to ten members, the best circuit improves from 0.738 to 0.776 and the best baseline from 0.747 to 0.793, so the gap widens from 0.009 to 0.017 (Table 6 and Figure 6).

Table 6: Ten bagged members per model, each a depth-4 tree trained as in Table 1 on its own feature subset, drawn from a pool of $\hat{m}$ features. On `banknote`, which has four features, members differ by seed only.

Dataset	Majority	*general*	*structured*	*recycle*	DNDT	NODE	CatBoost	RF	DT
`australian`	.555	.919 ± .012	.901 ± .038	.910 ± .036	**.930 ± .011**	.919 ± .027	.910 ± .028	.910 ± .042	.919 ± .012
`banknote`	.555	.891 ± .017	.928 ± .008	.922 ± .003	.962 ± .012	**.993 ± .000**	.949 ± .005	.972 ± .003	.928 ± .000
`breast`	.655	.951 ± .015	.957 ± .009	.963 ± .011	.971 ± .000	.971 ± .000	.974 ± .011	.989 ± .006	**.991 ± .011**
`churn`	.735	.776 ± .018	.764 ± .014	.770 ± .016	.773 ± .014	.770 ± .017	.769 ± .017	.775 ± .017	**.778 ± .013**
`compas`	.656	.698 ± .007	.704 ± .008	**.707 ± .007**	.705 ± .008	**.707 ± .013**	.689 ± .013	.697 ± .007	.698 ± .005
`egg`	.551	.685 ± .013	.666 ± .013	.675 ± .013	.689 ± .010	**.714 ± .011**	.686 ± .024	.704 ± .019	.709 ± .021
`fico`	.522	.715 ± .006	.718 ± .006	.720 ± .006	.725 ± .007	**.728 ± .003**	.724 ± .006	**.728 ± .005**	.725 ± .008
`german`	.713	.648 ± .010	.656 ± .000	.656 ± .000	.658 ± .004	.656 ± .000	.658 ± .004	.652 ± .005	**.662 ± .017**
`heart`	.556	.726 ± .111	.681 ± .050	.756 ± .030	.726 ± .038	.741 ± .052	.704 ± .052	.719 ± .073	**.763 ± .030**
`lymph`	.570	.747 ± .027	.747 ± .027	.733 ± .042	.760 ± .033	.760 ± .068	.760 ± .033	**.787 ± .050**	.773 ± .053
`pima`	.651	.743 ± .010	.660 ± .013	.740 ± .016	.701 ± .012	.748 ± .010	.712 ± .021	.751 ± .017	**.774 ± .033**
`sonar`	.534	**.810 ± .030**	.724 ± .047	.667 ± .043	.743 ± .049	.724 ± .036	.762 ± .030	.771 ± .063	.790 ± .057
mean	.604	.776	.759	.768	.779	.786	.775	.788	**.793**

D TRAINABILITY, DEPTH AND RESOURCES

This appendix supports the scaling results of Section 5.1. We report how qubit requirements and accuracy evolve with tree depth, how parameters are used by each circuit, and two possible limitations of the setup, namely the training budget and finite measurement statistics. We conclude with a closer comparison between *general* and *structured*.

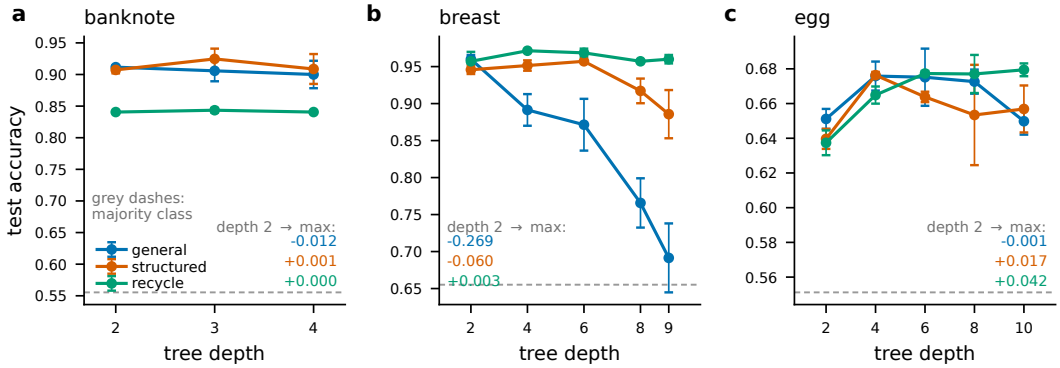

Figure 7: Test accuracy against tree depth, with the configuration selected on validation at each depth. Error bars are over five seeds; the dashed line is the majority-class rate, and labels give the accuracy change over the depth range. Depth grids are capped by the number of available features.

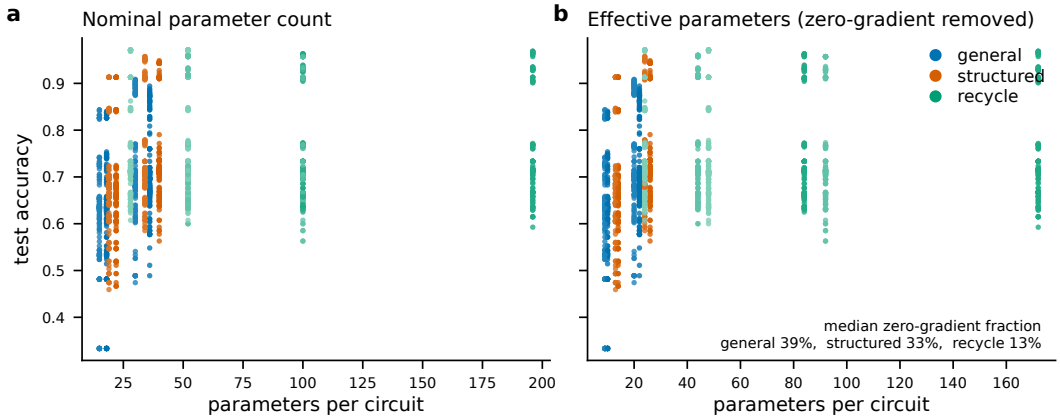

Figure 8: Test accuracy against (a) the nominal parameter count and (b) the number of parameters with nonzero gradient. Each point is one (dataset, configuration) cell averaged over five seeds; color intensity encodes the qubit count (2 to 6). The mean within-dataset Spearman correlation between parameter count and accuracy is $+0.68$ for *general*, $+0.50$ for *structured* and -0.02 for *recycle*: additional parameters improve accuracy for the first two circuits but not for *recycle*.

Qubits and tree depth. A depth-d tree requires $d + w$ qubits under *general* and *structured* but only $1 + w$ under *recycle* (Figure 3, top). On the main grid, the selected configurations use 5 to 6 qubits for *general* and *structured* and 2 to 5 for *recycle*. In the depth sweep (Figure 7), on `breast` *general* loses 0.269 accuracy as its register grows to ten qubits and *structured* loses 0.060, whereas *recycle* remains on two qubits and never falls below its depth-2 accuracy of 0.957. On `egg`, *recycle* gains 0.042, while *general* and *structured* end within 0.02 of their depth-2 accuracy.

Parameters. The qubit saving of *recycle* is paid in parameters: it uses 28 to 196, against 15 to 36 for *general* and 19 to 40 for *structured*, since it instantiates a separate ansatz block for each branch outcome. However, a larger fraction of the parameters of *general* and *structured* has identically zero gradient (median 39% and 33%, against 13% for *recycle*), narrowing the difference in effective capacity suggested by their nominal parameter counts (Figure 8).

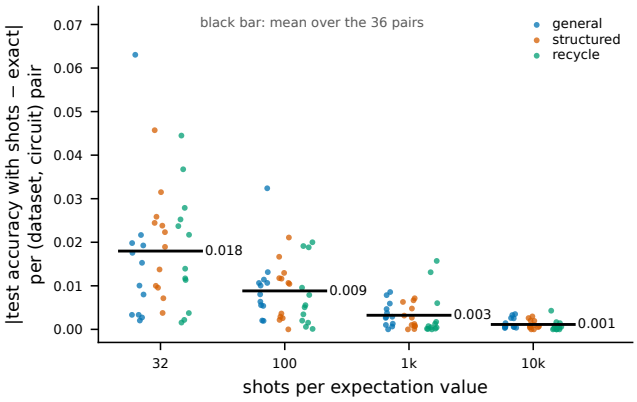

Figure 9: Change in test accuracy when exact expectation values are replaced by finite shots. Each dot is one (dataset, circuit) pair, averaged over seeds and shot realizations; the black bars are the means over the 36 pairs.

Finite shots. Figure 9 reports, for each of the 36 (dataset, circuit) pairs, the absolute change in test accuracy when exact expectation values are replaced by finite-shot estimates. The mean change decreases steadily with the shot budget, from 0.018 at 32 shots to 0.009 at 100, 0.003 at $1{,}000$ and 0.001 at $10{,}000$, consistent with the inverse-square-root scaling expected from sampling noise. Even at 32 shots no pair deviates by more than about 0.06, from $1{,}000$ shots onward all pairs stay below 0.02, and no circuit is consistently more sensitive than the others. Finite measurement statistics are therefore not a limiting factor at this scale.

General versus Structured versus Recycle. The two circuits differ only in how each feature enters its qubit, and this difference is smaller than it appears. Since $P(\phi) = e^{i\phi/2} R_Z(\phi)$ and $H R_Z(\phi) H = R_X(\phi)$, the comparator on qubit i acts, up to a global phase, as a rotation $R_X(\phi_i)$ with $\phi_i = x_i - t_i - \pi/2$. Starting from $|0\rangle$, an R_X rotation prepares the same state as an R_Y rotation by the same angle followed by the fixed phase gate $S^\dagger$, so the comparator prepares

$$S^\dagger R_Y(-t_i - \pi/2)\, R_Y(x_i)\, |0\rangle\,.$$

The factor $S^\dagger R_Y(-t_i - \pi/2)$ is a fixed single-qubit unitary, which the general single-qubit rotation opening the first ansatz layer can absorb exactly. *Structured* is therefore equivalent to an $R_Y(x_i)$ encoding followed by the same ansatz used in *general*, and its thresholds are redundant with the ansatz parameters: adding them leaves the numerical rank of the output Jacobian unchanged at depth 4, which also explains why they are not identifiable (Section 5.2). The only remaining difference is the encoding slope, 1 rad per unit feature for *structured* against $\pi/2$ for *general*, so the two circuits implement the same model up to a rescaling of the input features. Empirically, on the $2{,}880$ exactly matched (configuration, seed) pairs, *structured* is more accurate by 0.047 on average. This gap is however fragile: after validation-based selection it drops to 0.012, and to 0.005 when `heart` is excluded. By comparison, *recycle*, whose measured and classically controlled comparisons realize a genuinely different model, improves over *general* by 0.086 on matched pairs and by 0.026 after selection. Equivalence in function does not imply equivalence in what can be read off the trained model. In *general*, no parameter corresponds to a split: the decision logic is spread over the ansatz, and no threshold can be recovered from it. In *structured*, each comparator carries an explicit threshold t_i tied to feature x_i, so the learned splits can be read directly from the parameters and used to reconstruct a classical ODT (Section 4.2). The equivalence above also bounds this property: since the ansatz can absorb any change in t_i, the thresholds are not uniquely determined by the trained model, and the reconstructed tree is one of many consistent with it. This is why extraction is only loosely tied to the circuit's accuracy for *structured*, and why *recycle*, where the thresholds cannot be absorbed, provides the stronger form of interpretability (Section 5.2).

E Loss landscapes, gradient variance and magic

This appendix characterizes the optimization behavior and the quantum resources of the three circuits. We examine the local geometry of the trained optima, the decay of gradient variance with

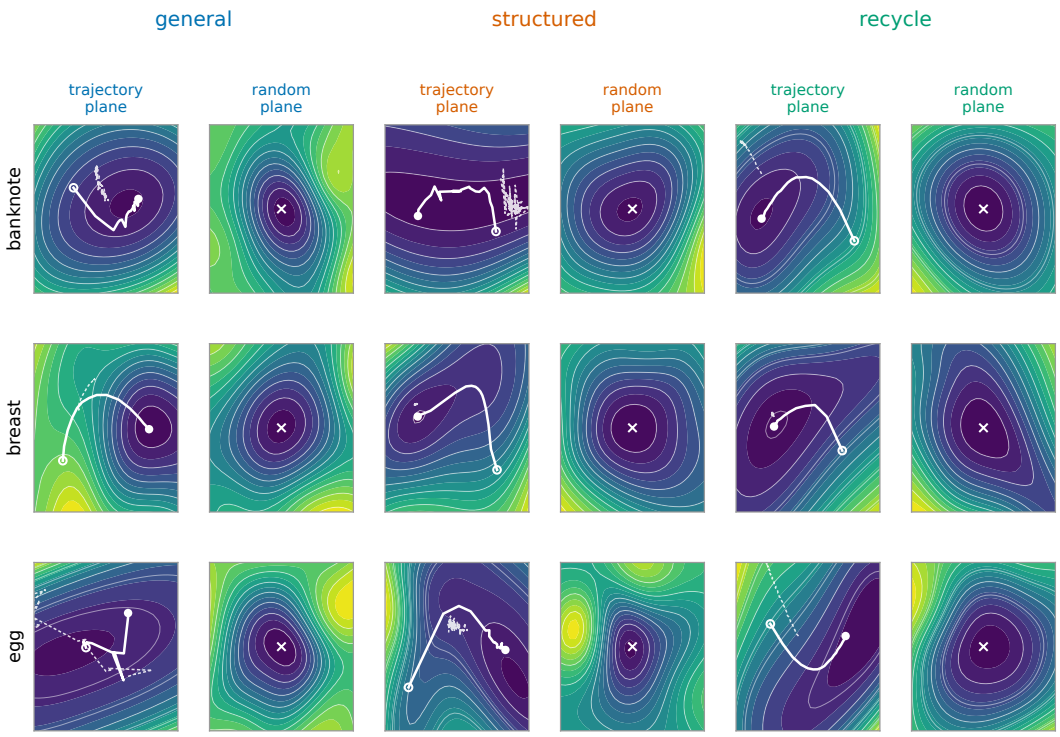

Figure 10: Loss landscapes at the trained optima (seed 0), computed with `orqviz` (Rudolph et al., 2021). Each pair of panels shows the plane spanned by the first two principal components of the optimization trajectory (left; open circle at initialization, filled marker at the checkpoint) and a random orthonormal plane through the same optimum spanning $\pm\pi$ (right; cross at the optimum). Color is the loss, rescaled within each panel. The trajectory plane is fitted on the first 20 epochs (solid); later epochs (dashed) are projections onto it and appear detached when the optimizer leaves the plane, as on `banknote`/*structured*, where the out-of-plane displacement grows from 0.25 to 0.68 of the in-plane one. The random plane makes the analysis non-circular, since the trajectory plane is chosen by the optimizer and is benign almost by construction.

depth and its source, and the role of magic in each circuit. Landscape, Hessian and magic analyses use checkpoints from the 20-epoch pilot grid rather than from the 100-epoch main grid (Table 5).

Loss landscapes. Figure 10 shows, for each circuit and dataset, the loss around the trained optimum in two planes. In the random planes, all nine optima lie at the bottom of a single smooth basin, with no competing minimum within $\pm\pi$. The trajectory planes show how the optimizer reached them: during the first 20 epochs, *recycle* follows smooth paths on all three datasets, whereas *general* and *structured* follow more irregular paths on `banknote`, and *general* also on `egg`, where later epochs move furthest out of the fitted plane. Several trajectory planes also show elongated valleys rather than isolated minima, most clearly for *structured* on `banknote`. Since colors are rescaled within each panel, the figure does not compare basin depths; in absolute terms, the loss range across the random plane is 0.07 to 0.09 on `egg` and 0.53 to 0.81 on `breast`, larger than the differences between circuits within any dataset, so the depth of the basin depends on the dataset rather than on the circuit. The exact Hessian confirms this picture: its largest eigenvalue is positive at all 45 checkpoints and its smallest lies in $[-0.058, 0]$, so the optima are approximate minima with many flat directions rather than saddles, consistent with the elongated valleys above. The number of flat directions grows with the parameter count, from 15 (*general*) and 18 (*structured*) to 45 (*recycle*). These checkpoints are selected on the 20-epoch pilot grid, so their configurations differ from those of Table 1 in 7 of the 9 cells; on `breast`, the *recycle* checkpoint has $w = 1$, $L = 1$ (28 parameters).

Gradient variance. For each configuration, we estimate $\mathrm{Var}[\partial_\theta C]$, the variance of the partial derivatives of the cost C with respect to the trainable parameters θ over 200 random initializations, and fit $\log \mathrm{Var}[\partial_\theta C] = a - b\,d$, where d is the tree depth, a an intercept and b the decay exponent:

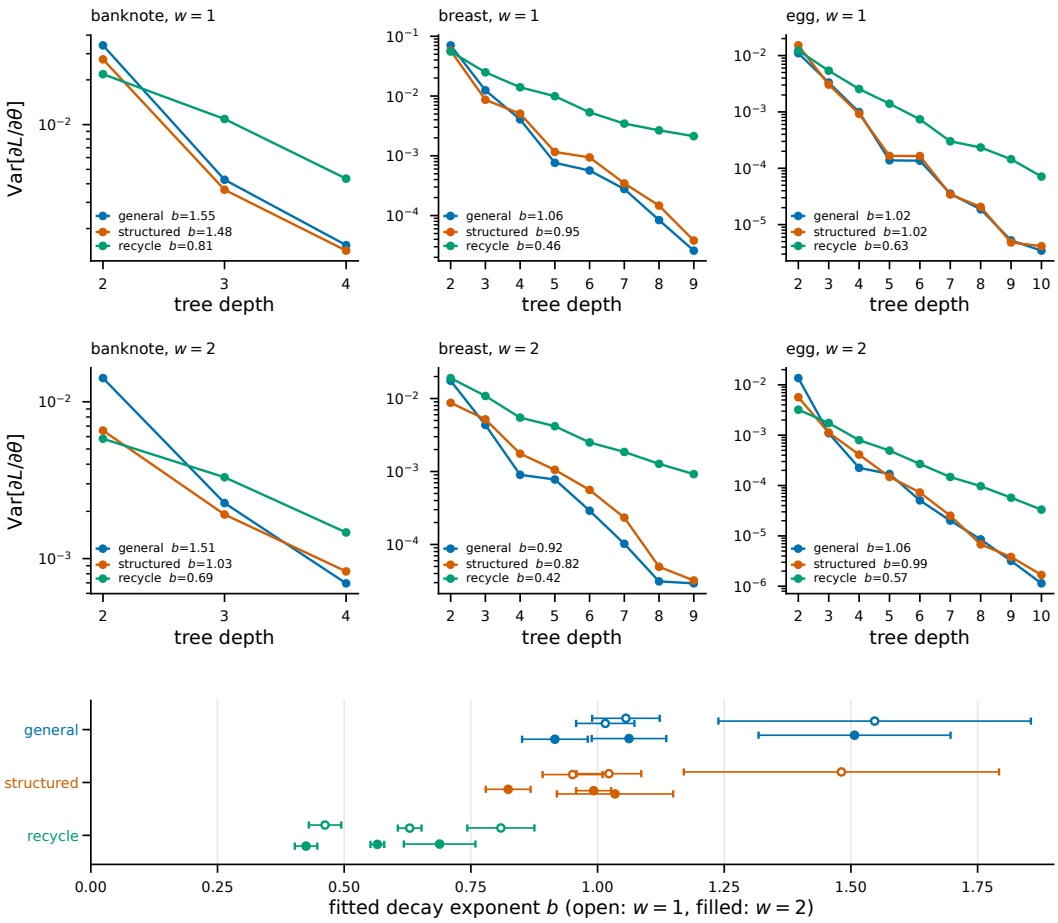

Figure 11: Gradient variance at random initialization (200 initializations per point, no training) against tree depth, for $w = 1$ (first row) and $w = 2$ (second row), with the decay exponent b of the fit $\log \mathrm{Var}[\partial_\theta C] = a - b\,d$ for each circuit. The bottom panel collects these exponents per circuit, with the standard error of each fit (open markers: $w = 1$; filled: $w = 2$). The $w = 2$ row is the one shown in Figure 3 (bottom).

each additional tree level multiplies the gradient variance by e^{-b}, so a larger b means faster vanishing gradients. At $w = 2$, b ranges from 0.92 to 1.51 for *general* and from 0.82 to 1.03 for *structured*, against 0.42 to 0.69 for *recycle* (Figure 11): *recycle* loses gradient signal at roughly half the rate. The decay is mitigated but not removed, as b is positive in every setting we measured.

Magic. Besides entanglement, a key resource separating quantum computation from what can be efficiently simulated classically is *magic*, or non-stabilizerness: circuits composed only of Clifford gates acting on stabilizer states can be simulated efficiently on a classical computer (Gottesman, 1998), and non-Clifford resources are required for universal quantum computation (Bravyi & Kitaev, 2005). Measuring magic therefore tells us whether making the tree structure explicit, as in *structured* and *recycle*, reduces how much of this resource the trained circuits exploit compared with the generic *general*. We quantify it with the stabilizer Rényi entropy (Leone et al., 2022),

$$M_2(|\psi\rangle) = -\ln\left(\frac{1}{2^n}\sum_{P\in\mathcal{P}_n}\langle\psi|\,P\,|\psi\rangle^4\right),$$

where $\mathcal{P}_n$ is the set of n-qubit Pauli strings, estimated by Monte Carlo sampling over $\mathcal{P}_n$. M_2 vanishes exactly on stabilizer states and is bounded above by $n\ln 2$, so dividing by this bound makes circuits of different width comparable. Magic decreases from initialization to the trained optimum in all 45 runs, so training moves every circuit toward states closer to the stabilizer set. Once normalized, the three circuits are indistinguishable (0.38 for *general*, 0.35 for *structured* and

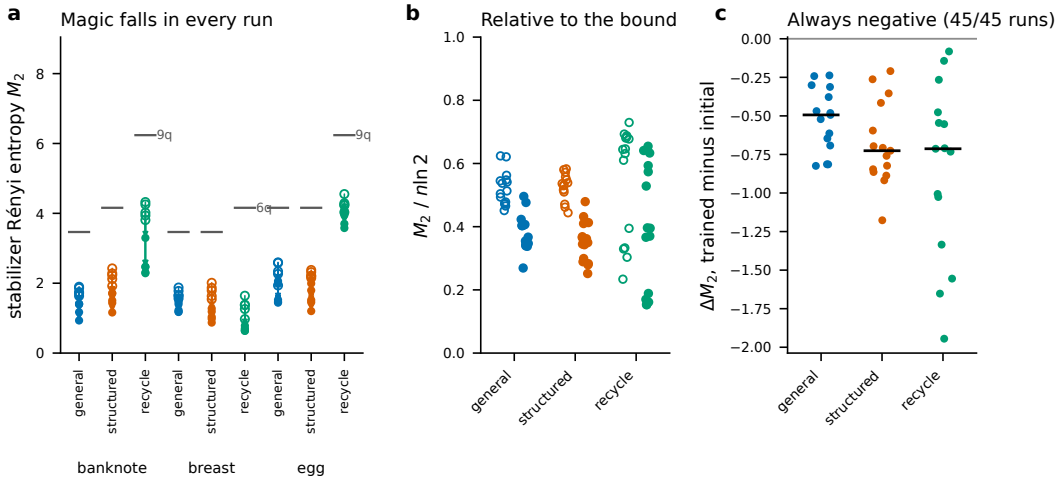

Figure 12: Stabilizer Rényi entropy M_2 at initialization and at the trained optimum. (a) Per circuit and dataset, with each cell's own $n \ln 2$ bound; for *recycle*, the annotation gives the effective register size after deferring the mid-circuit measurements, so its values are not directly comparable to the other two circuits. (b) The same quantities normalized by that bound. (c) Change in M_2 over training.

0.40 for *recycle*; Figure 12): encoding the tree structure explicitly does not reduce the magic the circuits use. Magic is a necessary but not sufficient condition for a quantum advantage, and at the scale studied here all circuits remain easy to simulate classically, so this result characterizes the circuits rather than implying any advantage.