Partition of Unity Neural Networks for Interpretable Classification with Explicit Class Regions

Anonymous authors
Paper under double-blind review

Abstract

We introduce Partition of Unity Neural Networks (PUNNs), a neural architecture that constructs class probabilities directly as a partition of unity, eliminating the need for a softmax layer. PUNNs produce nonnegative functions that sum to one via a recursive product of gate functions, guaranteeing valid probability distributions by design.

Our contributions are threefold. First, we prove that PUNNs are dense in the space of continuous probability maps on compact domains, establishing a universal approximation guarantee for probabilistic classification. Second, the recursive gate construction induces a hierarchical rejection chain that explicitly reveals how predictions are formed: each gate performs a sequential accept/reject decision, passing remaining probability mass onward. We demonstrate this on MNIST, where the resulting gate trace localizes model uncertainty and pinpoints specific gating failures in misclassified examples. Third, we evaluate PUNNs against multilayer perceptrons and Explainable Boosting Machines across MNIST, UCI benchmarks, and synthetic datasets. Under matched parameter budgets on MNIST, PUNNs achieve accuracy within 0.4–1.1 percentage points of MLPs, with performance stable across random class orderings; on UCI tabular benchmarks, the gap is at most one percentage point. When geometric priors align with the data structure, shape-informed gate parameterizations can achieve comparable accuracy with up to 300× fewer parameters.

We relate PUNNs to stick-breaking constructions from Bayesian nonparametrics, clarifying connections to probabilistic modeling while emphasizing the deterministic, input-dependent nature of the architecture. Overall, PUNNs provide a principled alternative to softmax-based classifiers, offering transparent class probability assignments through explicit gate decompositions, with controlled accuracy trade-offs.

1 Introduction

Modern neural classifiers typically produce class probabilities by applying a softmax transformation to unconstrained logits (Bishop, 2006; Goodfellow et al., 2016; Hastie et al., 2009; LeCun et al., 2015). While effective, softmax defines class probabilities through a global competition: all logits interact through the normalization denominator, and the resulting predictions do not, by construction, decompose into interpretable components that reflect a sequential or localized decision process (Lipton, 2018; Doshi-Velez & Kim, 2017). This global coupling obscures the contribution of individual classes and complicates post-hoc analysis of confidence and calibration (Guo et al., 2017).

In this work, we introduce *Partition of Unity Neural Networks (PUNNs)*, an architecture that constructs class probabilities directly as a partition of unity, eliminating the need for a softmax layer. The model produces nonnegative functions that sum to one via a recursive product of input-dependent gate functions, guaranteeing valid probability distributions by design. This structure yields a representation in which predictions are explicitly decomposed into interpretable accept/reject decisions, in contrast to post-hoc explanation methods that seek to rationalize predictions after training (Ribeiro et al., 2016; Lundberg & Lee, 2017).

The recursive gate construction induces a hierarchical decision process in which probability mass is sequentially allocated across classes. Each gate performs an accept/reject decision for its associated class and passes

remaining probability mass forward, producing a *gate rejection trace* that reveals how predictions are formed. This trace provides a concrete diagnostic tool: it identifies where probability mass is rejected or retained and allows misclassifications to be traced to specific gating failures. We illustrate this on MNIST, where the gate trace localizes model uncertainty and explains errors at the level of individual gating decisions.

We make the following contributions:

- **Architecture.** We introduce Partition of Unity Neural Networks, a class of models that construct class probabilities directly through a partition of unity, yielding valid probabilistic outputs without softmax.
- **Theory.** We prove that PUNNs are dense in the space of continuous probability maps on compact domains, establishing a universal approximation guarantee for probabilistic classification.
- **Interpretability.** We show that the recursive gate structure induces a hierarchical rejection chain that provides an explicit and interpretable explanation of model predictions and failures.
- **Empirical evaluation.** We compare PUNNs with multilayer perceptrons and Explainable Boosting Machines on MNIST, UCI benchmarks (Asuncion & Newman, 2007), and synthetic datasets. Under matched parameter budgets on MNIST, PUNNs achieve accuracy within 0.4–1.1 percentage points of MLPs, with performance stable across random class orderings; on UCI tabular benchmarks, the gap is at most one percentage point. When geometric priors align with the data structure, shape-informed gate parameterizations achieve comparable accuracy with up to 300× fewer parameters.

Finally, we relate PUNNs to stick-breaking constructions from Bayesian nonparametrics, clarifying their connection to probabilistic modeling while emphasizing their deterministic, input-dependent nature. Together, these results position PUNNs as a principled alternative to softmax-based classifiers, offering transparent class probability assignments through explicit gate decompositions, with controlled accuracy trade-offs.

2 Related Work

2.1 Partition of Unity Methods

Partition of unity methods have a long history in numerical analysis and approximation theory. The partition of unity finite element method (Babuška & Melenk, 1997) uses overlapping patches with partition of unity functions to construct flexible approximation spaces. Meshfree methods such as the element-free Galerkin method (Belytschko et al., 1994) rely on partition of unity constructions for function approximation without mesh constraints. Radial basis function (RBF) networks (Powell, 1987) use localized basis functions that can form partitions of unity, providing both interpolation guarantees and intuitive geometric interpretation.

PUNN draws directly from this tradition: our partition functions $h_i(x)$ satisfy the classical partition of unity property $\sum_i h_i(x) = 1$ by construction, enabling the same theoretical guarantees while operating within a neural network framework.

Related ideas have recently appeared in neural architectures that incorporate partition of unity constructions for function approximation and operator learning. For example, POUnets (Lee et al., 2021) use partition of unity principles to localize approximation and improve convergence properties in regression settings. These approaches focus on approximating real-valued functions or operators and do not address probabilistic classification or the construction of class probabilities.

In contrast, PUNN uses a partition of unity to define class probabilities directly, replacing softmax normalization with an architectural guarantee of probabilistic validity. This distinction is central: in PUNN, each partition function represents a probability by construction rather than acting as a localization weight in a function-approximation sum.

2.2 Connection to Stick-Breaking Processes

The recursive construction underlying PUNN is structurally related to the stick-breaking representation used in Bayesian nonparametrics, particularly in the construction of the Dirichlet process (Sethuraman, 1994). In stick-breaking, a sequence of Beta-distributed random variables v_i defines mixture weights via $w_i = v_i \prod_{j<i}(1 - v_j)$, yielding a random probability measure. PUNN adopts the same algebraic recursion but differs in purpose and mechanism: the "break points" $g_i(x)$ are deterministic, input-dependent functions learned by neural networks, and the resulting partition functions serve as class probabilities rather than mixture weights. This deterministic, input-conditioned use of the stick-breaking identity is what enables PUNN to define explicit, interpretable class regions—a property absent from the Bayesian setting where the weights are global and data-independent. We also note that the stick-breaking construction has been used in neural network contexts, for example in variational autoencoders with nonparametric priors (Nalisnick & Smyth, 2017), though not in the partition of unity framework with geometric gate parameterizations that we develop here.

2.3 Interpretable Machine Learning

Beyond softmax-based classifiers, related ideas have appeared in mixture-of-experts models, gating networks, and hierarchical classifiers, where multiple components are combined through learned selection mechanisms (Jacobs et al., 1991; Jordan & Jacobs, 1994; Shazeer et al., 2017; Kontschieder et al., 2015). While PUNN is superficially related to these approaches, the underlying computational structure is fundamentally different. In MoE, a gating network produces normalized weights that combine the outputs of multiple expert networks, yielding a weighted sum in which all experts contribute simultaneously. In contrast, PUNN does not combine expert outputs: the partition functions themselves are the class probabilities. These are constructed via a recursive product that sequentially allocates probability mass across classes. Each gate performs a local accept/reject decision that removes mass from subsequent classes, yielding a hierarchical decomposition of the prediction. Thus, in PUNN the partition of unity is the output of the model, not a routing mechanism over experts, and the resulting predictions admit an explicit sequential interpretation that is not present in standard MoE formulations.

Recent work on interpretable machine learning includes Neural Additive Models (NAM) (Agarwal et al., 2021), which restrict neural networks to additive structures for interpretability. Explainable Boosting Machines (Lou et al., 2012) combine boosting with generalized additive models. Attention mechanisms (Vaswani et al., 2017) provide some interpretability through attention weights, though their faithfulness has been questioned (Jain & Wallace, 2019). Post-hoc explanation methods such as feature attribution and surrogate models (Molnar, 2022) seek to rationalize predictions after training rather than enforcing interpretability at the architectural level.

PUNN takes a different approach to interpretability. Rather than restricting model capacity or relying on post-hoc explanation methods, PUNN constructs class probabilities through a hierarchical rejection structure that is interpretable by design. Each gate function g_i determines whether an input is assigned to class i or passed to subsequent classes, yielding an explicit accept/reject decision process that requires no additional interpretation machinery.

2.4 Approximation of Probability Maps

Universal approximation theorems establish that standard neural networks can approximate any continuous real-valued function on a compact domain to arbitrary precision. Classical results include Cybenko (1989) for sigmoidal networks and Hornik et al. (1989) for general activation functions.

Our work builds on this foundation by studying approximation within a *structured function class* relevant to classification. Rather than approximating arbitrary real-valued functions, we consider continuous *probability maps*, i.e., continuous functions taking values in the probability simplex. We show that the PUNN architecture is dense in this space, demonstrating that the partition of unity construction does not sacrifice expressive power for probabilistic classification despite imposing explicit structure and interpretability.

3 Partition of Unity Neural Networks

3.1 Architecture Definition

Let $x \in \mathbb{R}^d$ be an input. A Partition of Unity Neural Network with k partitions consists of:

1. **Gate functions:** $g_1, \ldots, g_{k-1} : \mathbb{R}^d \to [0,1]$, each parameterized as

$$g_i(x) = g(\theta_i(x)), \tag{1}$$

where $g : \mathbb{R} \to [0,1]$ is a smooth activation function and $\theta_i : \mathbb{R}^d \to \mathbb{R}$ is a parameterized function (e.g., a neural network or a geometric primitive).

Natural choices for g include:

- **Sigmoid:** $g(t) = \sigma(t) = 1/(1+e^{-t})$
- **Gaussian:** $g(t) = \exp(-t^2)$
- **Bump:** $g(t) = \begin{cases} \exp\left(-\frac{1}{1-t^2}\right) & |t| < 1 \\ 0 & |t| \geq 1 \end{cases}$ — C^∞ with compact support

The parameterization θ_i can range from simple geometric forms to flexible neural networks:

- **MLP-based:** $\theta_i(x) = \mathrm{MLP}_i(x)$ — maximum flexibility
- **Radial:** $\theta_i(x) = s_i(r_i - \|x - c_i\|)$ — spherical regions centered at c_i with radius r_i and scaling s_i
- **Ellipsoidal:** $\theta_i(x) = s_i(1 - (x-c_i)^\top A_i (x - c_i))$ — axis-aligned or rotated ellipsoids

2. **Partition functions:** $h_1, \ldots, h_k : \mathbb{R}^d \to [0,1]$ defined recursively as

$$h_1(x) = g_1(x), \tag{2}$$

$$h_i(x) = \left(\prod_{j=1}^{i-1} (1 - g_j(x)) \right) g_i(x), \quad i = 2, \ldots, k-1, \tag{3}$$

$$h_k(x) = \prod_{j=1}^{k-1} (1 - g_j(x)). \tag{4}$$

Proposition 1 (Partition of Unity Property)**.** *Let $k \geq 2$. For every $x \in \mathbb{R}^d$, the partition functions satisfy*

$$\sum_{i=1}^{k} h_i(x) = 1 \quad \textit{and} \quad h_i(x) \geq 0 \ \textit{for all } i.$$

Proof. Define, for $m \geq 1$,

$$S_m(x) := \sum_{j=1}^{m} \left(\prod_{i=1}^{j-1} (1 - g_i(x)) \right) g_j(x),$$

with the standard convention that an empty product equals 1.

Note that, for $2 \leq l \leq k$

$$\sum_{i=1}^{l} h_i(x) = S_{l-1}(x) + h_l(x). \tag{5}$$

We claim that for all $m \geq 1$,

$$S_m(x) = 1 - \prod_{j=1}^{m} (1 - g_j(x)).$$

For $m = 1$, this is immediate since

$$S_1(x) = g_1(x) = 1 - (1 - g_1(x)).$$

Assume the identity holds for some $m \geq 1$. Then

$$\begin{aligned} S_{m+1}(x) &= S_m(x) + \prod_{j=1}^{m}(1 - g_j(x))\, g_{m+1}(x) \\ &= 1 - \prod_{j=1}^{m}(1 - g_j(x)) + \prod_{j=1}^{m}(1 - g_j(x))\, g_{m+1}(x) \\ &= 1 - \prod_{j=1}^{m}(1 - g_j(x))\big(1 - g_{m+1}(x)\big) \\ &= 1 - \prod_{j=1}^{m+1}(1 - g_j(x)). \end{aligned}$$

This completes the induction.

Taking $m = k - 1$ and $l = k$ in (5), and using the definition of h_k, we obtain

$$\sum_{i=1}^{k} h_i(x) = 1.$$

Finally, since each $g_i(x) \in [0, 1]$, all factors defining $h_i(x)$ are nonnegative, and hence $h_i(x) \geq 0$ for all i. □

3.2 PUNN for Classification

For a k-class classification problem, we set the number of partitions equal to the number of classes. The class probability is given directly by the partition function:

$$P(\text{class } i \mid x) = h_i(x). \tag{6}$$

No softmax or other normalization is needed because the partition of unity property guarantees $\sum_i h_i(x) = 1$.

Training. We minimize the negative log-likelihood:

$$\mathcal{L} = -\sum_{n=1}^{N} \log h_{y_n}(x_n), \tag{7}$$

where y_n is the true class label for sample x_n.

Remark 2 (Extension to Multi-Modal Classes)**.** *When classes contain multiple modes, one can use $k > C$ partitions and aggregate them by class: $P(\text{class } c \mid x) = \sum_{i:\pi(i)=c} h_i(x)$, where π maps partitions to classes. This maintains valid probabilities and allows the network to discover substructure within classes.*

3.3 Hierarchical Decision Structure

PUNN provides a natural hierarchical interpretation:

- $h_1 = g_1$: probability that the input is assigned to class 1;
- $h_2 = (1 - g_1) \cdot g_2$: probability of class 2, given rejection by gate 1;
- $h_i = (1 - g_1) \cdots (1 - g_{i-1}) \cdot g_i$: probability of class i, given rejection by gates $1, \ldots, i - 1$.

This structure therefore enables direct interpretation: the gate value $g_j(x)$ explicitly quantifies the degree to which class j accepts the input. Values of $g_j(x)$ close to zero indicate strong rejection by class j, while values close to one indicate strong acceptance. Unlike softmax-based classifiers, where all logits interact through normalization, each gate in PUNN produces an explicit accept/reject score without requiring competition with other classes.

4 Theoretical Analysis

4.1 Density of PUNN in the space of continuous probability maps

Definition 3 ($(k-1)$-Simplex)**.** *The $(k-1)$-dimensional probability simplex is defined by*

$$\Delta^{k-1} = \left\{ (p_1, \ldots, p_k) \in \mathbb{R}^k \ : \ p_i \geq 0 \text{ for all } i, \ \sum_{i=1}^{k} p_i = 1 \right\}.$$

Its relative interior is

$$\mathrm{ri}(\Delta^{k-1}) = \left\{ (p_1, \ldots, p_k) \in \mathbb{R}^k \ : \ p_i > 0 \text{ for all } i, \ \sum_{i=1}^{k} p_i = 1 \right\}.$$

Definition 4 (Probability Map)**.** *Let $K \subset \mathbb{R}^d$ be compact. A function $p : K \to \Delta^{k-1}$ is called a* probability map*. If each coordinate function p_i is continuous, we say that p is a* continuous probability map*.*

Remark 5. *Standard universal approximation extends immediately to simplex-valued targets: any continuous $f : K \to \mathrm{ri}(\Delta^{k-1})$ on a compact $K \subset \mathbb{R}^d$ can be uniformly approximated by an MLP composed with a softmax output layer. The substantive question for the present paper is therefore not whether Δ^{k-1}-valued MLPs are dense, but whether the PUNN cascade parametrization, in which probabilities are built sequentially through input-dependent stick-breaking rather than via a single softmax head, retains this density. The novelty lies not in density itself, but in showing that this highly structured sequential parametrization preserves universal approximation. Theorem 6 establishes that it does.*

Theorem 6. *Let $K \subset \mathbb{R}^d$ be compact and let $p : K \to \mathrm{ri}(\Delta^{k-1})$ be a continuous probability map.*

Let $g : \mathbb{R} \to (0,1)$ be a strictly monotone continuous function (e.g. sigmoid). Consider PUNN partition functions defined by

$$\begin{aligned}
h_1(x) &= g(\theta_1(x)), \\
h_i(x) &= \Big(\prod_{j=1}^{i-1} \big(1 - g(\theta_j(x))\big)\Big) g(\theta_i(x)), \quad i = 2, \ldots, k-1, \\
h_k(x) &= \prod_{j=1}^{k-1} \big(1 - g(\theta_j(x))\big),
\end{aligned}$$

where each $\theta_i : K \to \mathbb{R}$ is a feedforward neural network.

Then for every $\varepsilon > 0$, there exist neural networks $\theta_1, \ldots, \theta_{k-1}$ such that

$$\max_{1 \leq i \leq k} \|h_i - p_i\|_{C(K)} < \varepsilon.$$

Proof. Let $p = (p_1, \ldots, p_k) : K \to \Delta^{k-1}$ be a continuous probability map. In particular,

$$p_i(x) \geq 0 \quad \text{and} \quad \sum_{i=1}^{k} p_i(x) = 1 \quad \text{for all } x \in K.$$

Define continuous functions $\gamma_1, \dots, \gamma_{k-1} : K \to (0,1)$ recursively by

$$\begin{aligned}\gamma_1(x) &= p_1(x),\\ \gamma_i(x) &= \frac{p_i(x)}{\sum_{j=i}^{k} p_j(x)}, \quad i = 2, \dots, k-1.\end{aligned}$$

Since p is continuous and $\sum_{j=i}^{k} p_j(x) > 0$ for all x, each γ_i is continuous and takes values in $(0,1)$.

Let $g : \mathbb{R} \to (0,1)$ be a continuous strictly monotone function, and define

$$g_i(x) := g(\phi_i(x)), \quad \text{where } \phi_i(x) := g^{-1}(\gamma_i(x)).$$

Since g is strictly monotone and continuous, it follows that g^{-1} is also monotone and continuous and that $\phi_i := g^{-1} \circ \gamma_i$ is continuous on K.

Define the PUNN partition functions $h_1, \dots, h_k$ by

$$\begin{aligned}h_1(x) &= g_1(x),\\ h_i(x) &= \left(\prod_{j=1}^{i-1}(1 - g_j(x))\right) g_i(x), \quad i = 2, \dots, k-1,\\ h_k(x) &= \prod_{j=1}^{k-1}(1 - g_j(x)).\end{aligned}$$

A direct computation then shows that

$$h_i(x) = p_i(x), \qquad i = 1, \dots, k.$$

By the classical universal approximation theorem for feedforward neural networks (e.g. Cybenko, 1989; Hornik et al., 1989), for every $\delta > 0$ there exist neural networks $\theta_i : K \to \mathbb{R}$ such that

$$\|\theta_i - \phi_i\|_{C(K)} < \delta, \quad i = 1, \dots, k-1. \tag{8}$$

Since ϕ_i is continuous on the compact set K, its range is contained in a compact interval $[-M, M] \subset \mathbb{R}$. Choosing δ sufficiently small in (8), we may assume that $\theta_i(K) \subset [-M-1, M+1]$ for all i. Since g is continuous, it is uniformly continuous on the compact interval $[-M-1, M+1]$. Therefore, for every $\varepsilon > 0$ there exists $\delta > 0$ such that

$$\|g(\theta_i) - g(\phi_i)\|_{C(K)} < \varepsilon \quad \text{for all } i.$$

Let $\tilde{h}_1, \dots, \tilde{h}_k$ denote the partition functions obtained by replacing ϕ_i with θ_i in the above construction. Since the range of $\tilde{h}_i, h_i$ is included in the interval $[0,1]$ and since each h_i and $\tilde{h}_i$ is a finite product of the functions g_j and $1 - g_j$, all of which take values in $[0,1]$, possibly reducing δ in (8), we obtain

$$\max_{1 \le i \le k} \|\tilde{h}_i - h_i\|_{C(K)} < \varepsilon.$$

To make this explicit, consider two typical factors appearing in the products, for instance

$$\tilde{h}_2 = g(\theta_1)(1 - g(\theta_2)) \quad \text{and} \quad h_2 = g(\phi_1)(1 - g(\phi_2)).$$

We write

$$\begin{aligned}\tilde{h}_2 - h_2 &= g(\theta_1)(1 - g(\theta_2)) - g(\phi_1)(1 - g(\phi_2)) \\ &= \big(g(\theta_1) - g(\phi_1)\big)(1 - g(\theta_2)) + g(\phi_1)\big(g(\phi_2) - g(\theta_2)\big).\end{aligned}$$

Taking supremum norms and using that all terms take values in $[0, 1]$, we obtain

$$\begin{aligned}\|\tilde{h}_2 - h_2\|_{C(K)} &= \|g(\theta_1)(1 - g(\theta_2)) - g(\phi_1)(1 - g(\phi_2))\|_{C(K)} \\ &\leq \|g(\theta_1) - g(\phi_1)\|_{C(K)} + \|g(\theta_2) - g(\phi_2)\|_{C(K)}.\end{aligned}$$

The same argument applies to products involving more factors and follows by induction.

Since $h_i = p_i$, this yields

$$\max_{1 \leq i \leq k} \|\tilde{h}_i - p_i\|_{C(K)} < \varepsilon,$$

which completes the proof. □

Theorem 6 applies to strictly monotone activations such as the sigmoid.

4.2 Flexibility in Gate Function Parameterization

The proof of Theorem 6 requires only that the gate arguments θ_i can approximate continuous real-valued functions on compact sets.

Importantly, different gates within the same PUNN may use different activation functions and parameterizations, allowing heterogeneous inductive biases across classes. For example, shape-informed gates can be used for geometrically simple classes, while MLP-based gates can be employed for more complex or irregular class regions.

Each gate $g_i : \mathbb{R}^d \to [0, 1]$ can be constructed as a composition $g_i(x) = g(\theta_i(x))$, where g is an activation function (e.g., sigmoid, Gaussian, bump) and θ_i is the argument parameterization. These two choices are independent: any activation can be combined with any parameterization.

MLP-based gates use $\theta_i(x) = \mathrm{MLP}_i(x)$, providing maximum flexibility as universal approximators. In our experiments, we use the naming convention **PUNN-[Activation]** for this case (e.g., PUNN-Sigma uses sigmoid activation with MLP arguments).

Shape-informed gates encode geometric priors directly into θ_i, dramatically reducing parameter count when class regions have known structure. Examples include spherical shells, ellipsoids, and direction-dependent radii via Fourier series or spherical harmonics. Section 6 develops these parameterizations in detail and demonstrates their effectiveness.

Remark 7 (Partition of Unity Preservation)**.** *The partition of unity property* $\sum_{i=1}^{k} h_i(x) = 1$ *is preserved regardless of the choice of* g *or* θ_i*, as it depends only on the recursive product construction.*

5 Experiments

We evaluate PUNN on synthetic, UCI benchmark, and image classification datasets, comparing against standard MLPs.

5.1 Experimental Setup

PUNN: Each gate function g_i is parameterized as a two-hidden-layer network with ReLU activations and sigmoid output. Hidden dimension is 64 unless otherwise specified.

MLP Baseline: Two hidden layers with dimensions [128, 64] and ReLU activations, followed by softmax.

Training: Adam optimizer, learning rate 0.001, 300 epochs. Results averaged over 3–5 random seeds.

5.2 Synthetic Datasets

We begin our empirical evaluation with synthetic two-dimensional classification problems. These experiments serve two purposes: (1) to verify that the partition of unity constraint does not limit the expressiveness of PUNN, and (2) to visualize the learned partition functions, which is only feasible in low dimensions.

5.2.1 Datasets

We consider four binary classification datasets with increasing complexity:

Moons Two interleaving half-circles generated using `sklearn.datasets.make_moons`.

Circles Two concentric circles (inner vs. outer class) generated using `sklearn.datasets.make_circles` with factor 0.5.

XOR Four Gaussian clusters at $(\pm 1, \pm 1)$ with XOR labeling, i.e., clusters at $(-1, -1)$ and $(1, 1)$ belong to class 0.

Helix Two interleaved spirals with two full rotations, parameterized as $(t\cos(t), t\sin(t))$ and $(t\cos(t + \pi), t\sin(t + \pi))$ for $t \in [0, 4\pi]$.

Each dataset contains 1,000 samples with additive Gaussian noise ($\sigma = 0.1$). We use an 80/20 train/test split and standardize features to zero mean and unit variance.

5.2.2 Models

We evaluate three instantiations of PUNN, differing only in the gate function architecture:

PUNN-Sigma. Each gate g_i is a two-hidden-layer MLP:

$$g_i(x) = \sigma(W_3 \cdot \mathrm{ReLU}(W_2 \cdot \mathrm{ReLU}(W_1 x + b_1) + b_2) + b_3)$$

with 32 hidden units per layer, yielding 1,185 parameters per gate.

PUNN-Bump. Each gate uses the C^∞ bump function with compact support, composed with an NN:

$$g_i(x) = \phi(\tanh(\theta(x))), \quad \text{where } \phi(t) = \begin{cases} \exp\left(-\frac{1}{1-t^2}\right) & \text{if } |t| < 1 \\ 0 & \text{if } |t| \geq 1 \end{cases}$$

The tanh ensures the NN output stays within the bump's support $(-1, 1)$. For complex shapes like Helix, a larger NN (128 hidden units) enables the domain warping to follow the spiral.

PUNN-Gaussian. Each gate uses a Gaussian function composed with an NN:

$$g_i(x) = \exp(-\theta(x)^2)$$

where θ has the same architecture as PUNN-Sigma, yielding 1,185 parameters per gate.

MLP Baseline. A standard two-hidden-layer MLP (32 units each) with softmax output, having 1,218 parameters.

For binary classification, PUNN uses a single gate ($K - 1 = 1$ for $K = 2$ classes), with partition functions $h_0(x) = g_0(x)$ and $h_1(x) = 1 - g_0(x)$. Note that PUNN-Sigma in this binary setting reduces to a standard neural network with sigmoid output. However, for $K > 2$ classes, PUNN-Sigma differs from standard softmax classifiers: the $K - 1$ gate product formula produces partition functions with different properties than softmax normalization. The PUNN framework also becomes distinct when using alternative gate functions (bump, Gaussian) or multiple partitions per class.

Table 1: Test accuracy (%) on synthetic datasets. All PUNN variants achieve comparable accuracy to the MLP baseline, demonstrating that the partition of unity constraint does not limit expressiveness.

Dataset	PUNN-Sigma	PUNN-Bump	PUNN-Gaussian	MLP
Moons	100.0	100.0	100.0	100.0
Circles	99.0	99.0	98.5	98.5
XOR	100.0	100.0	100.0	100.0
Helix	100.0	99.5	98.5	99.5
Parameters	1,185	1,186	1,185	1,218

5.2.3 Training Protocol

All models are trained for 200 epochs using Adam (Kingma & Ba, 2015) with learning rate 0.01 and batch size 64. PUNN minimizes the cross-entropy loss

$$\mathcal{L} = -\sum_{n=1}^{N} \sum_{k=0}^{K-1} y_n^{(k)} \log\big(h_k(x_n) + \epsilon\big)$$

where $\epsilon = 10^{-10}$ ensures numerical stability. The MLP baseline uses the standard cross-entropy loss with softmax. All experiments use random seed 42.

5.2.4 Results

Table 1 reports test accuracy for all methods. All PUNN variants achieve near-perfect accuracy, matching or exceeding the MLP baseline across all datasets.

Figure 1 visualizes the learned decision boundaries. The background color encodes $h_1(x)$, the predicted probability of class 1, with the decision boundary (black contour) at $h_1(x) = 0.5$. All gate functions learn appropriate nonlinear boundaries, including the challenging spiral structure of the Helix dataset.

5.2.5 Visualizing the Partition of Unity

A distinguishing feature of PUNN is that the outputs $h_i(x)$ form a partition of unity by construction: $\sum_{i=0}^{K-1} h_i(x) = 1$ for all $x \in \mathbb{R}^d$. This property enables direct interpretation of $h_i(x)$ as $P(\text{class } i \mid x)$ without requiring softmax normalization.

Figure 2 visualizes the partition functions $h_0(x)$ and $h_1(x)$ learned by PUNN-Sigma on each dataset. The complementary structure is evident: $h_0(x) + h_1(x) = 1$ everywhere, with high values of h_0 corresponding exactly to low values of h_1. This visualization provides direct insight into how PUNN partitions the input space into class regions.

5.2.6 Discussion

These experiments yield three main observations:

Unified framework. All three gate functions follow the same design principle: a base gating function ψ composed with a neural network θ that warps the domain, i.e., $g(x) = \psi(\theta(x))$. The sigmoid ($\psi(t) = \sigma(t)$), bump ($\psi(t) = \phi(t)$), and Gaussian ($\psi(t) = \exp(-t^2)$) functions each provide different inductive biases while the NN adapts the gate's support to the data.

Expressiveness. The partition of unity constraint does not limit PUNN's ability to learn complex decision boundaries. All configurations achieve near-perfect accuracy on problems requiring highly nonlinear separators, including the two-spiral Helix dataset. PUNN-Sigma achieves 100% on Helix with only 1,185 parameters, demonstrating that the sigmoid gate with NN-parametrized domain is highly efficient.

Table 2: Parameter efficiency via multiple partitions on the Helix dataset using the bump gate. Increasing the number of partitions while shrinking each gate's hidden dimension preserves accuracy at a fraction of the parameter count, up to a point: very small per-gate capacity becomes unstable across seeds. Mean ± std over 5 seeds; best configuration in bold.

# Partitions	Hidden dim	Parameters	Test acc. (%)
16	4	570	92.20 ± 5.31
8	8	742	78.30 ± 11.18
2	32	1,186	99.10 ± 0.80
16	8	1,590	94.10 ± 5.05
8	**16**	**2,366**	$\mathbf{99.60 \pm 0.20}$
4	32	3,558	97.40 ± 3.76
2	64	4,418	99.30 ± 0.75
2	128	17,026	99.50 ± 0.32

Parameter efficiency via multiple partitions. For gates with compact support like the bump function, the partition-of-unity construction allows a trade-off between the capacity of each individual gate and the number of partitions. Table 2 explores this trade-off on the Helix dataset. A single high-capacity bump gate (2 partitions, 128 hidden units, 17,026 parameters) achieves $99.50 \pm 0.32\%$ accuracy. Replacing it with 8 partitions of 16 hidden units each preserves accuracy ($99.60 \pm 0.20\%$) using 2,366 parameters—a $7\times$ parameter reduction with comparable or slightly improved performance. This supports the intuition that, for complex or disconnected decision regions, multiple smaller gates can collectively cover the boundary more efficiently than a single large gate.

Pushing this further is not free. Configurations with very small per-gate capacity (e.g., 8 partitions of 8 hidden units, or 16 partitions of 4) become unstable across random seeds, with standard deviations exceeding 5 percentage points. The sweet spot is intermediate: enough partitions to localize the decision boundary, but enough capacity per gate that each individual decision is reliably learned.

Interpretability. Unlike standard neural networks, PUNN provides built-in probabilistic outputs through its partition of unity structure. The visualizations in Figure 2 demonstrate how the input space is explicitly partitioned, offering insight into the model's decision-making that is not readily available from softmax-based classifiers.

5.3 Real-World Datasets

Having validated PUNN on synthetic problems, we now evaluate its performance on real-world classification tasks. These experiments assess whether PUNN's partition of unity structure scales to higher-dimensional data while remaining competitive with standard neural network classifiers.

5.3.1 MNIST Handwritten Digits

Dataset. MNIST (LeCun et al., 1998) consists of 70,000 grayscale images of handwritten digits (0–9), each of size 28×28 pixels. We use the standard split of 60,000 training and 10,000 test images. Each image is flattened to a 784-dimensional vector and normalized to zero mean and unit variance.

Models. We sweep over a wide range of total parameter counts for both PUNN-Sigma and an MLP baseline, so that any accuracy gap we observe is attributable to the partition-of-unity architecture itself rather than to the size of the model.

PUNN-Sigma. Uses $K - 1 = 9$ sigmoid gates, $g_i(x) = \sigma(\theta_i(x))$, where each θ_i is a 2-hidden-layer MLP with H_g units per layer and ReLU activations. We sweep $H_g \in \{16, 32, 64, 128, 256\}$, covering 115K to 2.4M total parameters.

Table 3: MNIST test accuracy as a function of total parameter count for PUNN-Sigma (gate hidden dim H_g) and the MLP baseline (hidden dim H_m). At every parameter budget, the MLP outperforms PUNN-Sigma by 0.4–1.1 percentage points.

Model (H)	Parameters	Test Acc. (%)
PUNN-Sigma ($H_g = 16$)	115,641	96.78
PUNN-Sigma ($H_g = 32$)	235,881	97.11
PUNN-Sigma ($H_g = 64$)	490,185	97.38
PUNN-Sigma ($H_g = 128$)	1,054,089	97.43
PUNN-Sigma ($H_g = 256$)	2,403,081	97.85
MLP ($H_m = 32$)	26,506	97.00
MLP ($H_m = 64$)	55,050	97.19
MLP ($H_m = 128$)	118,282	97.92
MLP ($H_m = 256$)	269,322	98.24
MLP ($H_m = 512$)	669,706	98.09
MLP ($H_m = 1024$)	1,863,690	98.26

Table 4: MNIST gate-function comparison at matched width $H_g = 256$. The bump gate underperforms sigmoid by approximately one percentage point.

Model	Parameters	Test Acc. (%)
PUNN-Sigma	2,403,081	97.85
PUNN-Bump	2,403,090	97.00

PUNN-Bump. Uses $K - 1 = 9$ bump gates with tanh domain compression,

$$g_i(x) = \phi(\tanh(\theta_i(x))), \quad \phi(t) = \exp\Big(-\tfrac{1}{1-t^2}\Big) \text{ for } |t| < 1,$$

with the same θ_i architecture as PUNN-Sigma. The tanh confines the network output to the bump's support $(-1, 1)$. We report PUNN-Bump only at $H_g = 256$ for direct comparison with PUNN-Sigma at matched width (Table 4).

MLP baseline. A 2-hidden-layer MLP with H_m units per layer and ReLU activations, followed by softmax. We sweep $H_m \in \{32, 64, 128, 256, 512, 1024\}$, covering 27K to 1.86M parameters.

Training. All models are trained for 20 epochs using Adam (Kingma & Ba, 2015) with learning rate 0.001 and batch size 128. PUNN minimizes the cross-entropy loss over the partition functions:

$$\mathcal{L} = -\sum_{n=1}^{N} \sum_{k=0}^{K-1} y_n^{(k)} \log\big(h_k(x_n) + \epsilon\big)$$

where $\epsilon = 10^{-10}$ for numerical stability. All experiments use random seed 42.

Results. Table 3 and Figure 3 show that the MLP baseline consistently outperforms PUNN-Sigma by 0.4–1.1 percentage points at every parameter budget. This gap represents the cost of the partition of unity construction: because PUNN distributes parameters across $K - 1 = 9$ independent gate networks, each gate has access to fewer parameters than a single unified MLP of the same total size.

Discussion. *Parameter efficiency gap.* The parameter sweep reveals that PUNN's accuracy disadvantage is not simply an artifact of mismatched model sizes—it persists at every budget. This is an inherent cost of the modular gate architecture, where each gate independently processes the 784-dimensional input. However,

Table 5: Failure analysis of PUNN-Sigma on the full MNIST test set (10,000 examples; 257 misclassified, 97.43% accuracy). For each error we record the true-class gate value $g_y(x)$ (or $h_9(x)$ for the leftover class). Two distinct failure modes emerge depending on gate position.

True class y	#errors	mean g_y on errors	frac. $g_y < 0.1$	dominant failure mode
0	11	0.07	72.7%	gate did not fire
1	9	0.10	55.6%	gate did not fire
2	33	0.17	60.6%	gate did not fire
3	34	0.18	67.6%	gate did not fire
4	33	0.25	54.5%	gate did not fire
5	29	0.24	62.1%	gate did not fire
6	21	**0.64**	28.6%	mass already absorbed
7	16	**0.79**	12.5%	mass already absorbed
8	22	**0.84**	9.1%	mass already absorbed
9 (leftover)	49	$h_9 = 0.06$	—	leftover squeezed out

the gap is modest (under 1.1 percentage points even at the smallest budget) and may be acceptable in applications where the interpretability of the gate rejection structure is valued.

Diminishing returns. Both architectures exhibit diminishing returns with increased parameters. PUNN-Sigma plateaus near 97.4–97.9%, while the MLP plateaus near 98.1–98.3%. This suggests that neither architecture benefits substantially from additional capacity beyond a moderate size on MNIST.

Gate function comparison. Table 4 compares the two gate parameterizations at matched width $H_g = 256$. PUNN-Bump achieves 97.00% compared to PUNN-Sigma's 97.85%. The bump function's compact support likely limits gradient flow compared to sigmoid's smooth transition over the entire real line.

Interpretability: a concrete trace. Consider an uncertain test image of the digit "7" (Figure 4a). The PUNN gate chain produces, in order,

$$g_0 = 0.00,\ g_1 = 0.00,\ g_2 = \mathbf{0.31},\ g_3 = \mathbf{0.37},\ g_4{=}g_5{=}g_6 = 0.00,\ g_7 = \mathbf{1.00},$$

yielding partition probabilities $h_2 = 0.31$, $h_3 = 0.26$, $h_7 = 0.43$ (others ≈ 0). The model's prediction is class 7, but the trace makes the residual ambiguity explicit: gates 2 and 3 partially fire, indicating that the image plausibly resembles those classes, and gate 7 fully accepts what mass remains. A misclassified example (Figure 4b: true class 5, predicted 6) localizes the failure to a single gate: $g_5 = 0.023$ when it should have been near 1, after which gate 6 absorbs nearly all probability. This kind of stepwise diagnosis is unavailable for softmax classifiers, where the predicted distribution is the joint output of all logits and cannot be decomposed into per-class accept/reject decisions.

Systematic failure analysis. Beyond individual examples, the gate trace enables systematic diagnosis of all misclassifications. Table 5 analyzes the 257 errors made by PUNN-Sigma on the full MNIST test set, revealing two distinct failure modes that depend on gate position.

For early-position classes ($y \leq 5$), misclassifications occur predominantly because the true-class gate did not fire: the mean g_y on errors ranges from 0.07 to 0.25, and 55–73% of errors have $g_y < 0.1$. For late-position classes ($y \geq 6$), a different failure mode dominates. The true-class gate *did* fire strongly on errors (mean g_y between 0.64 and 0.84), ruling out a gate-did-not-fire explanation. Given the sigma construction, in which $p_y \propto g_y \prod_{k<y}(1-g_k)$, the natural reading is that earlier gates had already absorbed most of the mass, leaving insufficient residual for the true class. Class 9, which receives only leftover probability mass by construction, accounts for 49 of the 257 errors—the largest share—with a mean $h_9 = 0.06$. Figure 5 (left) confirms this bimodal structure: the distribution of g_y on errors splits into a peak near zero (gate failure) and a peak near one (mass absorption). Importantly, the error pattern does not cluster at $\hat{y} - y = \pm 1$ (Figure 5, middle), confirming that the sequential gate ordering does not systematically bias errors toward adjacent classes.

Table 6: Shared-backbone PUNN-Sigma matches the accuracy of the largest independent-gate variant ($H_g = 256$) with ∼10× fewer parameters. Backbone: $784 \to 256 \to 128$ with ReLU; per-class gate heads: $128 \to 1$ with sigmoid.

Model	Parameters	Test Acc. (%)
PUNN-Sigma ($H_g = 128$, independent)	1,054,089	97.43
PUNN-Sigma ($H_g = 256$, independent)	2,403,081	97.85
PUNN-Sigma (shared backbone)	**235,017**	**97.85**
MLP ($H_m = 256$)	269,322	98.24

This analysis demonstrates a key advantage of PUNN's interpretability: not only can individual misclassifications be diagnosed through the gate trace, but the architecture exposes systematic failure patterns—such as position-dependent error modes—that are invisible in softmax classifiers.

Shared backbone variant. The parameter overhead of independent gates can be mitigated by sharing a feature extractor across all gates. We replace the $K - 1$ independent gate networks with a shared backbone ($784 \to 256 \to 128$, ReLU) followed by per-class linear gate heads ($128 \to 1$, sigmoid). The partition of unity structure is unchanged: the recursive product of gate outputs still guarantees $\sum_i h_i(x) = 1$.

Table 6 shows that the shared-backbone variant achieves 97.85% accuracy—matching the best independent-gate configuration—with only 235,017 parameters, comparable to the MLP baseline (269,322). Figure 6 shows that the trace remains diagnostic on the shared-backbone variant. The remaining accuracy gap of 0.39 percentage points represents the intrinsic cost of the partition of unity constraint rather than a parameter inefficiency.

The shared backbone preserves PUNN's primary interpretability mechanism: the gate rejection trace operates identically, and the hierarchical accept/reject structure remains fully intact. The individual gate decisions now operate on learned representations rather than raw inputs, which represents a natural point on the interpretability-efficiency spectrum that PUNN offers. In settings where per-gate feature interpretability is essential, independent gates can be used; when parameter efficiency is prioritized, the shared backbone provides it without sacrificing the diagnostic gate trace.

5.3.2 UCI Classification Benchmarks

To evaluate PUNN across diverse problem settings, we test on seven UCI classification benchmarks with varying numbers of classes, features, and samples.

Datasets.

- **Iris**: Fisher's classic iris flower classification (4 features, 3 classes, 150 samples)
- **Wine**: Wine cultivar recognition (13 features, 3 classes, 178 samples)
- **Breast Cancer**: Wisconsin breast cancer diagnosis (30 features, 2 classes, 569 samples)
- **Digits**: Optical recognition of 8×8 handwritten digits (64 features, 10 classes, 1,797 samples)
- **Pendigits**: Pen-based digit recognition (16 features, 10 classes, 10,992 samples)
- **Satimage**: Satellite image classification (36 features, 6 classes, 6,430 samples)
- **Optdigits**: Optical digit recognition (64 features, 10 classes, 5,620 samples)

All datasets are standardized to zero mean and unit variance. We use 80/20 train/test splits with stratified sampling, averaging results over 5 random seeds.

Table 7: Test accuracy (%) on the larger UCI classification benchmarks (single seed, seed 42). PUNN-Sigma is within 0.3 percentage points of the MLP baseline on three of four datasets and exceeds it on Optdigits. Iris, Wine, and Breast Cancer are reported separately with error bars in Table 8.

Dataset	Classes	Samples	PUNN-Sigma	PUNN-Bump	MLP
Digits	10	1,797	97.8	96.4	**98.0**
Pendigits	10	10,992	99.3	99.2	**99.5**
Satimage	6	6,430	91.5	90.4	**91.8**
Optdigits	10	5,620	**98.7**	97.9	98.4

Table 8: Test accuracy (mean ± std, %, 5 seeds) on UCI benchmarks comparing PUNN-Sigma against a softmax MLP and the Explainable Boosting Machine (EBM). EBM is competitive with or exceeds both neural methods, particularly on Breast Cancer; PUNN-Sigma matches the MLP and lies within one standard deviation of EBM on Iris and Wine.

Dataset	PUNN-Sigma	MLP	EBM
Iris	94.67 ± 2.67	95.33 ± 2.67	94.67 ± 2.67
Wine	97.22 ± 1.76	97.22 ± 2.48	$\mathbf{97.78 \pm 2.08}$
Breast Cancer	95.26 ± 0.89	95.44 ± 1.02	$\mathbf{97.02 \pm 0.89}$

Models. We compare three architectures:

PUNN-Sigma. Uses $K-1$ sigmoid gates, each with a 2-hidden-layer MLP (64 units per layer).

PUNN-Bump. Uses $K-1$ bump gates with tanh domain compression:

$$g_i(x) = \phi(\tanh(\theta_i(x))), \quad \phi(t) = \exp\left(-\frac{1}{1-t^2}\right) \text{ for } |t| < 1$$

MLP Baseline. A 2-hidden-layer MLP (128, 64 units) with softmax output.

Training. All models trained for 300 epochs using Adam with learning rate 0.001 and batch size 64.

Results. Table 7 shows that PUNN-Sigma is within 0.3 percentage points of the MLP baseline on Digits, Pendigits, and Satimage, and exceeds it on Optdigits (98.7% vs 98.4%). PUNN-Bump is consistently about one percentage point below PUNN-Sigma. Results on the smaller UCI datasets (Iris, Wine, Breast Cancer), reported separately with error bars over 5 seeds in Table 8, show all three methods within one standard deviation of each other on Iris and Wine. This confirms that the partition of unity structure does not compromise classification accuracy on real-world tabular data.

Gate Function Comparison. The bump gate consistently underperforms the sigmoid gate by approximately 1 percentage point. This gap likely stems from the bump function's compact support: unlike the sigmoid which transitions smoothly over the entire real line, the bump function is zero outside $|t| < 1$, potentially limiting gradient flow during training. However, the bump gate's compact support may offer advantages for interpretability, as it creates sharply defined regions of activation.

Comparison with Interpretable Baselines. Since PUNN is motivated by interpretability, we also compare against Explainable Boosting Machines (EBM) (Lou et al., 2013; Nori et al., 2019), a leading glass-box additive model, on three UCI datasets.

Table 8 shows that all three methods are within one standard deviation of each other on Iris and Wine, while EBM's advantage on Breast Cancer (97.0% vs 95.3%) is more substantial. These results place PUNN among established interpretable methods in terms of accuracy, while offering a qualitatively different form

of interpretability: EBM provides feature-level additive explanations, whereas PUNN provides an explicit hierarchical decomposition of class probabilities into per-class accept/reject decisions.

6 Shape-Informed Gating

A key advantage of the PUNN framework is that the gate functions $g_i(x)$ can encode domain knowledge directly. While the previous section used neural networks to parametrize gates, here we show that *geometric priors* can be encoded directly into the gate structure, leading to dramatic reductions in parameter count when the prior matches the data geometry.

6.1 Motivation

Standard neural network classifiers treat all classification problems uniformly: the same architecture is applied regardless of whether class regions are spherical, elongated, or irregularly shaped. This flexibility comes at a cost—the network must learn the geometry from scratch using many parameters.

In contrast, PUNN allows us to design gates that directly encode geometric assumptions. When the assumption holds, this reduces parameters by orders of magnitude.

6.2 Direction-Dependent Shell Parametrization

We introduce a unified framework for shape-informed gates using direction-dependent radii in spherical coordinates. For $x \in \mathbb{R}^d$, let $\hat{n} = (x - c)/\|x - c\| \in S^{d-1}$ denote the unit direction from a center c to x.

Definition 8 (Direction-Dependent Shell Gate)**.** *Given a center $c \in \mathbb{R}^d$ and direction-dependent radius functions $r_1, r_2 : S^{d-1} \to \mathbb{R}_{\geq 0}$, define the normalized radial coordinate*

$$t(x) = \frac{\|x - c\| - r_1(\hat{n})}{r_2(\hat{n}) - r_1(\hat{n})}, \qquad \hat{n} = \frac{x - c}{\|x - c\|},$$

and the shell gate

$$g(x; c, r_1, r_2) = \phi\big(t(x)\big),$$

where $\phi : \mathbb{R} \to [0, 1]$ is a smooth transition function.

The geometry of the shell is encoded by the level sets of the normalized radius $t(x)$. The inner and outer shell boundaries are given by the surfaces $t(x) = 0$ and $t(x) = 1$, respectively. Different choices of the transition function ϕ yield different smooth parameterizations of the same underlying geometry.

For example, if ϕ is a bump function supported on $[0, 1]$, then the gate acts as a smooth indicator of the direction-dependent shell region

$$\Omega = \{\, c + r\hat{n} : \hat{n} \in S^{d-1},\ r_1(\hat{n}) \leq r \leq r_2(\hat{n}) \,\}.$$

6.3 Special Cases

The direction-dependent shell subsumes many useful geometric shapes:

Spherical Shell. Constant radii $r_1(\hat{n}) = r_1$, $r_2(\hat{n}) = r_2$ yield a spherical shell with $d + 2$ parameters (center plus two radii). Setting $r_1 = 0$ gives a solid ball.

Ellipsoid. Setting $r_1 = 0$ and $r_2(\hat{n}) = (\hat{n}^\top A \hat{n})^{-1/2}$ for a positive definite matrix A yields an ellipsoid centered at c. This requires $d + d(d + 1)/2$ parameters.

Star-Shaped Regions (2D). In $\mathbb{R}^2$, parametrize the outer radius using Fourier series:

$$r_2(\theta) = a_0 + \sum_{k=1}^{K} \big(a_k \cos(k\theta) + b_k \sin(k\theta)\big)$$

This captures non-convex star-shaped regions with $d + 2K + 1$ parameters.

Table 9: Shape-informed gates vs. a small parameter-matched MLP baseline ([32, 32] hidden layers, ∼1.2K–1.3K parameters depending on feature and class count). Direction-dependent parametrizations achieve comparable or superior accuracy with 66–304× fewer parameters. For reference, the full MLP baseline used in the main experiments ([128, 64], Section 5.3) achieves $95.33 \pm 2.67\%$ on Iris over 5 seeds (Table 8).

Dataset	Gate Type	Accuracy (%)	Parameters	Reduction
Circles	Small MLP ([32, 32])	98.6	1,218	—
Circles	Spherical Shell	**98.9**	**4**	**304**×
Moons	Small MLP ([32, 32])	99.9	1,218	—
Moons	Fourier Shell	99.3	**18**	**68**×
Conc. Rings (3-class)	Small MLP ([32, 32])	100.0	1,251	—
Conc. Rings (3-class)	Spherical Shell	**100.0**	**10**	**125**×
Iris	Small MLP ([32, 32])	93.3	1,315	—
Iris	Harmonics (shape-agnostic)	**95.3**	40	33×
Iris	Ellipsoid (shape-specific)	**95.3**	**20**	**66**×

Spherical Harmonics (Higher Dimensions). In $\mathbb{R}^d$, use spherical harmonics $Y_\ell^m(\hat{n})$ to parametrize direction-dependent radii, enabling complex shapes with controlled parameter count.

6.4 Experiments

We compare shape-informed PUNN against standard NN-parametrized PUNN on datasets where geometric priors are appropriate.

Table 9 demonstrates dramatic parameter reductions when shape priors match data geometry. On the Circles dataset, a spherical shell gate with only **4 parameters** achieves 98.9% accuracy—*exceeding* the 1,218-parameter MLP (98.6%) while using **304× fewer parameters**. The Moons dataset, with crescent-shaped classes, benefits from the Fourier shell parametrization achieving 99.3% with only 18 parameters (68× reduction). On the 3-class Concentric Rings dataset, spherical shell gates achieve **perfect 100% accuracy** matching the MLP while using **125× fewer parameters**. Figures 7 and 8 visualize the Circles result.

On Iris, we compare two shape-informed parametrizations: (1) a general spherical-harmonic parametrization that makes no assumptions about the shape of class regions, and (2) an ellipsoid parametrization that assumes axis-aligned elongation. Both reach 95.3% accuracy, exceeding the small parameter-matched MLP (93.3% with 1,315 parameters), but at very different costs: the ellipsoid uses only 20 parameters (66× reduction relative to this MLP), compared with 40 for the harmonic parametrization (33× reduction). This illustrates a key trade-off: *when the shape prior is known, more restrictive parametrizations are more efficient; when it is unknown, general direction-dependent parametrizations still yield substantial savings.*

6.5 Discussion

The direction-dependent shell provides a unified framework for encoding geometric priors in PUNN. By choosing appropriate parametrizations for $r_1(\hat{n})$ and $r_2(\hat{n})$, practitioners can trade off between model complexity and shape flexibility. Shape-informed gates are most effective when:

1. Class regions have known or suspected geometric structure.

2. The feature space has low-to-moderate dimensionality, so that radii on S^{d-1} can be parametrized cheaply and concentration of measure has not yet washed out the radial signal.

3. Parameter efficiency is critical (embedded systems, interpretability).

Table 10: Effect of harmonics degree on Iris accuracy. Higher degrees do not improve performance, suggesting spherical regions suffice for this dataset.

Degree	Accuracy (%)	Parameters
0 (spherical)	95.3 ± 3.4	12
1 (linear)	94.7 ± 2.7	20
2 (quadratic)	95.3 ± 2.7	40

Table 11: Effect of number of partitions on Circles (2-class). Using more partitions than classes improves accuracy by allowing finer coverage of class regions.

Partitions	Accuracy (%)	Parameters
2	88.6 ± 6.2	4
4	99.1 ± 0.6	12
6	99.0 ± 0.7	20
8	99.2 ± 0.5	28

PUNN naturally supports *hybrid* configurations: some gates can use geometric parametrizations while others use neural networks, allowing partial domain knowledge to be encoded while retaining flexibility for complex regions.

6.6 Ablation Studies

We conduct ablation studies to understand the effect of key hyperparameters on shape-informed PUNN performance.

Effect of Harmonics Degree. Table 10 shows the effect of varying the polynomial degree in the spherical harmonics parametrization on Iris. Surprisingly, even degree 0 (spherical) achieves 95.3% accuracy, matching the more flexible degree 2 parametrization. This suggests that for Iris, the class structure is well-captured by simple spherical regions, and higher-degree terms provide no additional benefit.

Effect of Number of Partitions. Table 11 shows the effect of using more partitions than classes on the Circles dataset (2 classes). With only 2 partitions (minimum), accuracy is 88.6%. Increasing to 4 partitions improves accuracy to 99.1%, as multiple partitions can better cover the circular class regions. Beyond 4 partitions, returns diminish.

6.7 Computational Efficiency

Shape-informed gates are computationally efficient due to their low parameter count. On Iris, the shape-informed model (40 parameters) achieves the same accuracy as an MLP (1,315 parameters) while requiring 33× fewer parameters. Training is also faster: 0.13s for shape-informed vs 0.37s for MLP (500 epochs), a 3× speedup. Both models have sub-millisecond inference times on CPU.

7 Discussion

7.1 When to Use PUNN

PUNN is particularly well suited for applications where interpretability is a primary concern. Its partition-of-unity structure induces an explicit geometric decomposition of the input space into local regions, enabling transparent analysis of the decision mechanism and the role of each local model. This makes PUNN appropriate for:

Table 12: Class-ordering sensitivity of PUNN-Sigma. We sample 5 permutations of the class-to-gate assignment (including the identity) and retrain. On both datasets the spread across orderings is comparable to the spread across random seeds.

Dataset	Mean acc. (%)	Std across orderings	Range	Min–Max
UCI Digits	97.39	0.39	1.11	96.67–97.78
MNIST	97.59	0.16	0.42	97.41–97.83

UCI Digits: 5 orderings × 3 seeds, $H_g = 64$, 100 epochs, Adam lr 10^{-3}.
MNIST: 5 orderings × 1 seed, $H_g = 128$ (1.05M params), 20 epochs, Adam lr 10^{-3}.

- **High-stakes domains:** Medical diagnosis, legal decisions, and financial applications where understanding the decision process is as important as the prediction itself.

- **Structured class geometry:** Problems where classes occupy geometrically coherent regions (spherical, ellipsoidal, or other shape-informed structures), enabling dramatic parameter reductions through shape-informed gates.

- **Rejection reasoning:** Settings where understanding "why not class X?" is important. The hierarchical gate structure explicitly shows which classes were rejected and with what confidence.

To illustrate concretely what PUNN's interpretability would provide in a high-stakes setting, consider a hypothetical medical-diagnosis task with classes *healthy*, *diabetes*, *hypertension*, and *cardiac risk*, in that order. When PUNN classifies a patient as cardiac risk, the gate chain might produce a rejection trace such as $g_1 = 0.03$ (healthy gate rejects), $g_2 = 0.05$ (diabetes gate rejects), $g_3 = 0.12$ (hypertension gate partially activates), with the remaining probability mass $h_4 = (1 - g_1)(1 - g_2)(1 - g_3) \approx 0.81$ assigned to cardiac risk by construction. A clinician reading this trace sees not only the final prediction but also which conditions were ruled out and how confidently: hypertension was considered as a secondary possibility ($h_3 \approx 0.11$ is nontrivial), while diabetes was confidently dismissed. A softmax classifier producing the same top-1 prediction provides no such decomposition: the predicted distribution is the joint output of all logits through a global normalization, and there is no way to extract which conditions were rejected or how confidently. Figure 4 shows the same kind of trace on real MNIST predictions, exposing residual ambiguity when the model is correct and isolating the responsible gate when it is wrong.

7.2 Limitations

Parameter efficiency gap. PUNN requires $k - 1$ gate networks for k classes, and our parameter sweep on MNIST (Table 3, Figure 3) shows that independent-gate PUNN underperforms the MLP baseline by 0.4–1.1 percentage points at matched parameter budgets. A shared-backbone variant (Table 6) closes most of this gap, achieving 97.85% with 235K parameters compared to 98.24% for the MLP with 269K—a residual gap of only 0.39 percentage points. This remaining gap represents the intrinsic cost of the partition of unity constraint rather than parameter inefficiency.

Class ordering sensitivity. The hierarchical construction imposes an ordering on classes: class 1 is evaluated first, and the final class receives probability mass only after all previous gates reject. While this does not affect the model's expressive power (as shown by our density result), it could in principle influence optimization dynamics. To assess this empirically, we retrained PUNN-Sigma under 5 different class orderings on both UCI Digits and MNIST (Table 12). On MNIST the standard deviation across orderings is only 0.16%, with a total range of 0.42 percentage points; on UCI Digits the spread is 0.39%. In both cases, the variation across orderings is comparable to the variation across random seeds, indicating that the class ordering is not a meaningful hyperparameter in practice.

Table 13: Calibration comparison. Top-1 accuracy, expected calibration error (ECE), Brier score, and test negative log-likelihood (NLL). Lower is better for ECE, Brier, and NLL. Despite producing probabilities by construction, PUNN-Sigma is consistently slightly less well calibrated than the softmax MLP and EBM.

Dataset	Model	Acc. (%)	ECE	Brier	NLL
MNIST	PUNN-Sigma	97.43	0.0181	0.0436	0.1503
	MLP	97.90	**0.0151**	**0.0353**	**0.1089**
UCI Digits	PUNN-Sigma	97.69	0.0213	0.0413	0.1445
	MLP	98.15	0.0170	0.0323	0.1030
	EBM	98.06	**0.0137**	**0.0300**	**0.0618**
UCI Breast Cancer	PUNN-Sigma	95.03	0.0426	0.0791	0.2511
	MLP	96.20	0.0371	0.0673	0.2568
	EBM	96.49	**0.0349**	**0.0598**	**0.1984**

7.3 Calibration Analysis

Since PUNN produces probabilities by construction rather than through softmax normalization, one might expect improved calibration. To test this, we measure expected calibration error (ECE, 15-bin), Brier score, and test negative log-likelihood (NLL) on MNIST and two UCI datasets, comparing PUNN-Sigma against the softmax MLP and EBM.

Table 13 shows that PUNN-Sigma is slightly less well calibrated than both the softmax MLP and EBM across all three datasets. This may seem surprising given that PUNN produces valid probabilities by construction. However, the partition of unity property guarantees only that outputs sum to one and are nonnegative—it does not guarantee that the learned probabilities match the true conditional class frequencies. Calibration depends on the training dynamics and the capacity of the gate networks, not solely on the architectural constraint. Improving PUNN's calibration through techniques such as temperature scaling or calibration-aware training objectives is a promising direction for future work.

7.4 Future Directions

Several extensions of PUNN merit further investigation:

- **Learned class ordering:** Although our experiments show that random orderings have minimal impact on accuracy (Table 12), developing methods to automatically learn an optimal ordering during training could further improve convergence speed and reduce variance.
- **Hierarchical PUNN:** Extend the architecture to tree-structured partitions, where each node recursively partitions its region. This would enable coarse-to-fine classification and natural handling of class taxonomies.
- **Approximation rates:** Derive theoretical bounds on how approximation error decreases as a function of the number of partitions and gate complexity, analogous to classical approximation theory results for neural networks.
- **Convolutional PUNN:** Extend gate functions to operate on spatial feature maps for image classification, potentially combining the interpretability benefits of PUNN with the representation learning capabilities of convolutional architectures.

8 Conclusion

We introduced Partition of Unity Neural Networks (PUNN), an architecture that produces class probabilities directly through a partition of unity construction. By designing gate functions $g_i(x)$ that recursively partition

the input space, PUNN guarantees valid probability distributions by construction, eliminating the need for softmax normalization.

Our theoretical analysis established that PUNN is dense in the space of continuous probability maps on compact domains, showing that the partition of unity construction preserves expressive power for probabilistic classification. This result ensures that replacing softmax with a partition of unity does not restrict the class of probability distributions that can be represented.

Empirically, we demonstrated that PUNN-Sigma achieves accuracy within roughly one percentage point of standard MLPs across UCI benchmarks and across two orders of magnitude of MNIST parameter budgets. A shared-backbone variant on MNIST closes the gap to 0.39 percentage points at comparable parameter count, demonstrating that the partition of unity constraint imposes only a modest accuracy cost. When geometric priors are available, shape-informed gate parameterizations achieve comparable accuracy with up to 300$\times$ fewer parameters, demonstrating that domain knowledge can further offset the structural overhead.

The key advantage of PUNN lies in its interpretability: each partition function $h_i(x)$ explicitly defines the region associated with class i, and the hierarchical gate structure reveals which classes were rejected and why. This transparency is achieved by design, not through post-hoc explanation methods.

We believe PUNN represents a promising direction for building neural network classifiers that are both accurate and interpretable, addressing a critical need in high-stakes applications where understanding model decisions is essential.

Reproducibility Statement

All experiments are fully reproducible. Source code for all PUNN variants (Sigma, Bump, Gaussian) and shape-informed gate implementations (spherical shell, ellipsoid, Fourier shell, spherical harmonics) will be made publicly available upon acceptance. All hyperparameters are documented in Appendix A. Synthetic datasets are generated with fixed random seeds and standard scikit-learn functions. UCI datasets are publicly available. Results on UCI benchmarks are averaged over 5 random seeds; synthetic and MNIST experiments use seed 42. The theoretical results require no computational verification beyond the proofs provided.

Broader Impact Statement

PUNN is designed to improve the interpretability of neural network classifiers, which we believe is beneficial for high-stakes applications where understanding model decisions is important. We do not foresee specific negative societal impacts from this work. As with any classification method, practitioners should be mindful of biases in training data and the limitations of the model when deploying it in sensitive domains.

References

Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E. Hinton. Neural additive models: Interpretable machine learning with neural nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4699–4711, 2021.

Arthur Asuncion and David Newman. Uci machine learning repository. `https://archive.ics.uci.edu/ml`, 2007.

Ivo Babuška and Jens M. Melenk. The partition of unity method. *International Journal for Numerical Methods in Engineering*, 40(4):727–758, 1997.

Ted Belytschko, Yun Yun Lu, and Lei Gu. Element-free Galerkin methods. *International Journal for Numerical Methods in Engineering*, 37(2):229–256, 1994.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

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

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1321–1330, 2017.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, New York, 2 edition, 2009.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

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

Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019.

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

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

Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulò. Deep neural decision forests. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1467–1475, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Kookjin Lee, Nathaniel A. Trask, Ravi G. Patel, Mamikon A. Gulian, and Eric C. Cyr. Partition of unity networks: Deep hp-approximation. In *Proceedings of the AAAI Spring Symposium on Combining Artificial Intelligence and Machine Learning with Physical Sciences (AAAI-MLPS)*, volume 2964 of *CEUR Workshop Proceedings*, 2021. URL `https://ceur-ws.org/Vol-2964/article_180.html`.

Zachary C. Lipton. The mythos of model interpretability. *Queue*, 16(3):30–57, 2018.

Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2012.

Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 623–631, 2013.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Christoph Molnar. *Interpretable Machine Learning*. Independently published, 2 edition, 2022. URL `https://christophm.github.io/interpretable-ml-book/`.

Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. In *International Conference on Learning Representations (ICLR)*, 2017.

Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

Michael J. D. Powell. Radial basis functions for multivariable interpolation: A review. In J. C. Mason and M. G. Cox (eds.), *Algorithms for Approximation*, pp. 143–167. Clarendon Press, Oxford, 1987.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1135–1144, 2016.

Jayaram Sethuraman. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations (ICLR)*, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

A Hyperparameter Details

This section documents the hyperparameters used in all experiments. All models were implemented in PyTorch and trained using the Adam optimizer.

A.1 Synthetic Experiments

Table 14: Hyperparameters for synthetic dataset experiments (Moons, Circles, XOR, Helix).

Parameter	**Value**
Training samples	800 (80% of 1,000)
Test samples	200 (20% of 1,000)
Noise level	$\sigma = 0.1$
Gate hidden dimension	32
Gate architecture	2-layer MLP with ReLU
Number of gates	$k - 1 = 1$ (binary classification)
Optimizer	Adam
Learning rate	0.01
Batch size	64
Epochs	200
Random seed	42

A.2 UCI Experiments

A.3 MNIST Experiments

A.4 Shape-Informed Experiments

Table 15: Hyperparameters for UCI benchmark experiments.

Parameter	**Value**
Train/test split	80% / 20% (stratified)
Preprocessing	Standardization (zero mean, unit variance)
PUNN gate hidden dimension	64
PUNN gate architecture	2-layer MLP with ReLU
MLP hidden dimensions	[128, 64]
MLP architecture	2-layer MLP with ReLU + softmax
Optimizer	Adam
Learning rate (PUNN-Sigma)	0.001
Learning rate (PUNN-Gaussian)	0.0001
Learning rate (MLP)	0.001
Batch size	64
Epochs	300
Number of runs	5 (different random seeds)
Base random seed	42

Table 16: Hyperparameters for MNIST experiments, including the parameter sweep.

Parameter	**Value**
Training samples	60,000
Test samples	10,000
Input dimension	784 (flattened 28×28)
Preprocessing	Standardization (zero mean, unit variance)
PUNN gate hidden dimensions	{16, 32, 64, 128, 256} (sweep)
PUNN gate architecture	2-layer MLP with ReLU
Number of gates	9 (for 10 classes)
MLP hidden dimensions	{32, 64, 128, 256, 512, 1024} (sweep)
MLP architecture	2-layer MLP with ReLU + softmax
Optimizer	Adam
Learning rate	0.001
Batch size	128
Epochs	20
Random seed	42

Table 17: Hyperparameters for shape-informed gate experiments.

Parameter	**Value**
Spherical shell parameters	Center $c \in \mathbb{R}^d$, radius $r > 0$, sharpness $s > 0$
Ellipsoid parameters	Center $c \in \mathbb{R}^d$, axis radii $\{r_j\}_{j=1}^d$, sharpness $s > 0$
Fourier shell harmonics	$K = 5$ (Moons dataset)
Spherical harmonics degree	$L = 2$ (Iris dataset)
Optimizer	Adam
Learning rate	0.01
Epochs	500
Number of runs	5 (different random seeds)
Base random seed	42

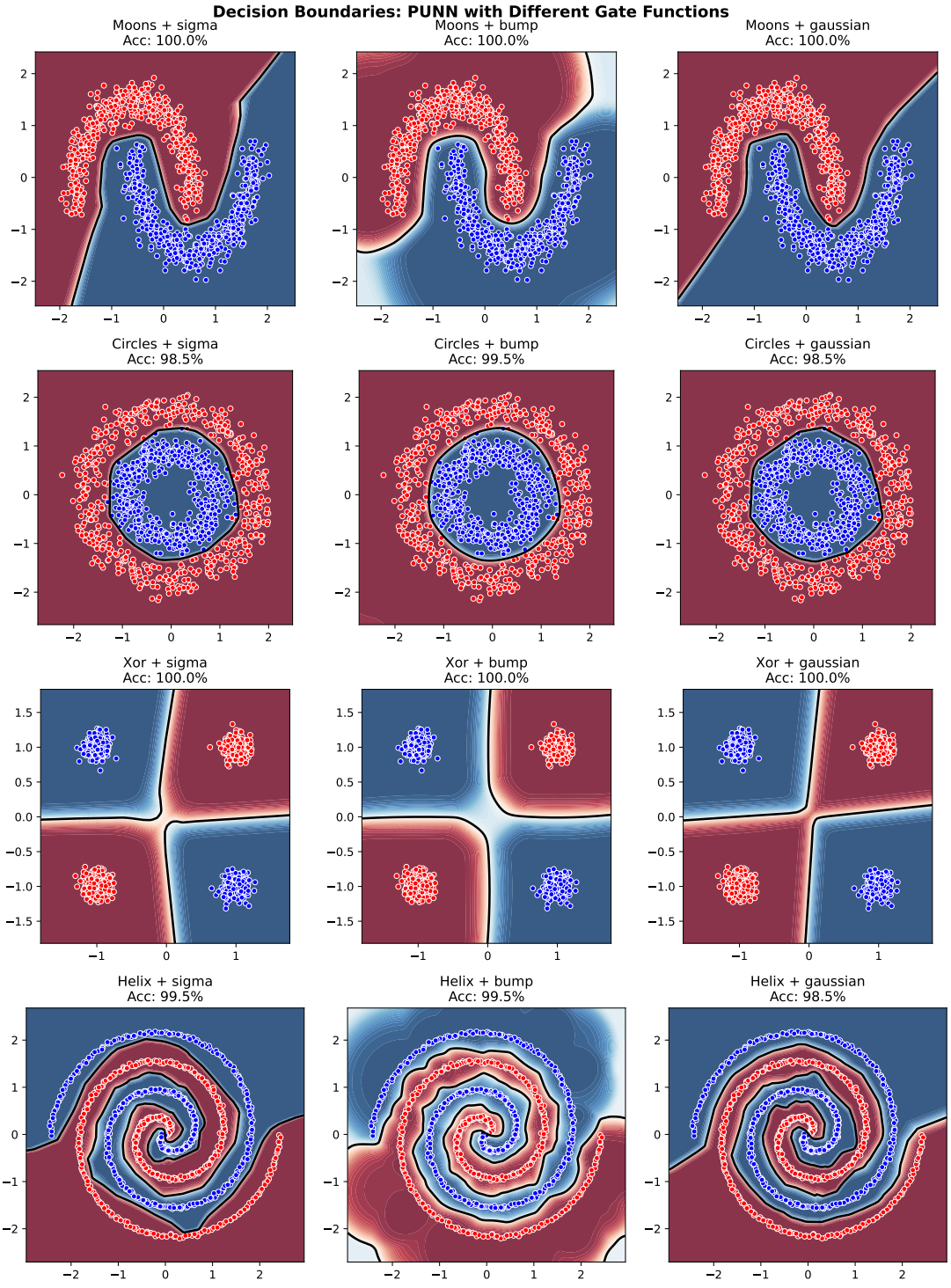

Figure 1: Decision boundaries learned by PUNN on synthetic datasets. Rows: Moons, Circles, XOR, Helix. Columns: Sigma, Bump, Gaussian gates. Background color indicates $h_1(x)$ (blue = 1, red = 0); black contour shows the decision boundary at $h_1(x) = 0.5$.

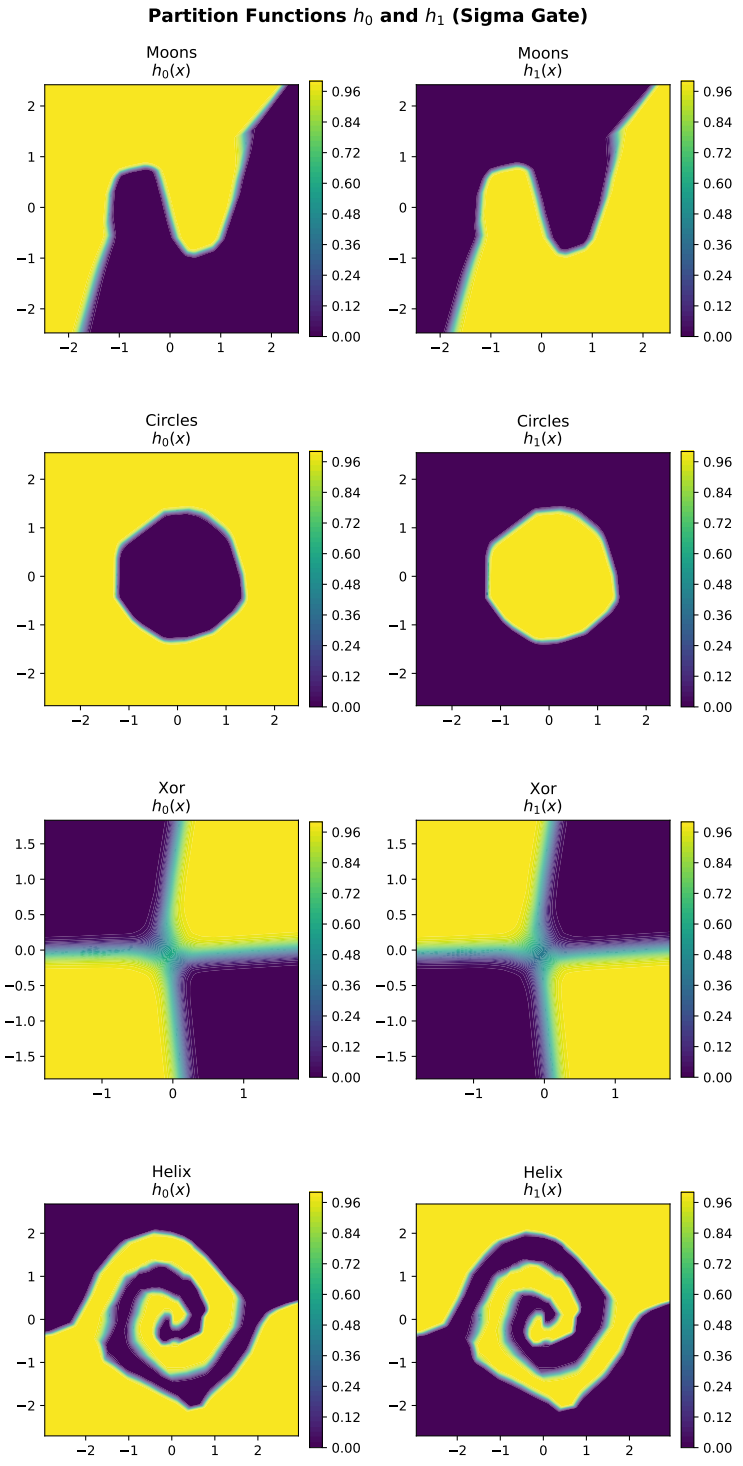

Figure 2: Partition functions $h_0(x)$ and $h_1(x)$ learned by PUNN-Sigma. Left column: $h_0(x)$; right column: $h_1(x)$. Rows correspond to Moons, Circles, XOR, and Helix datasets. The complementary structure ($h_0 + h_1 = 1$) is evident, enabling direct probabilistic interpretation.

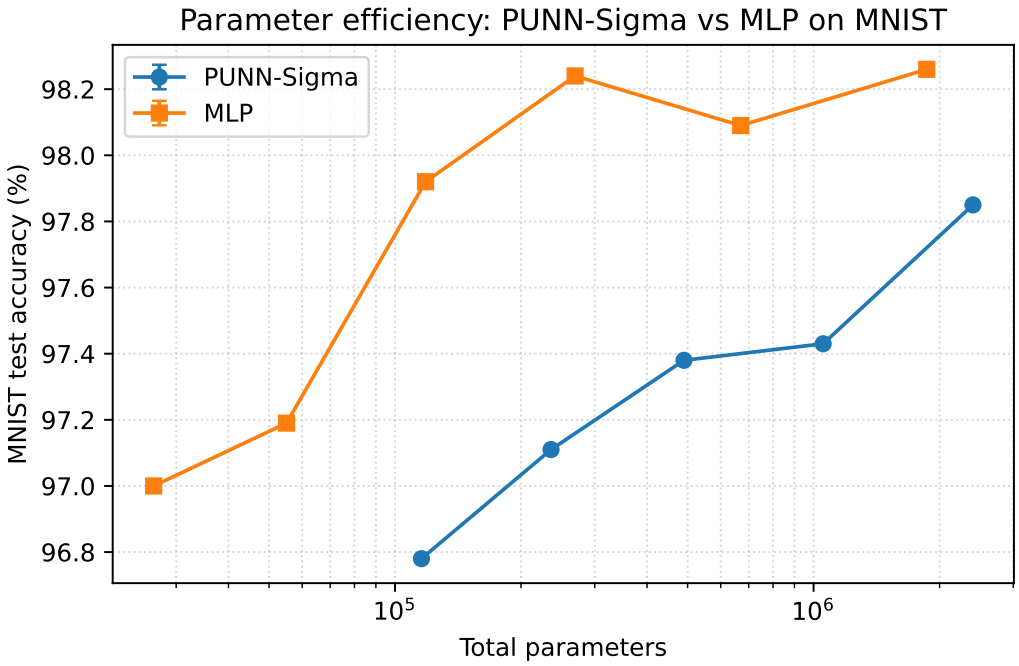

Figure 3: MNIST parameter-efficiency curve. The MLP baseline outperforms PUNN-Sigma by roughly 0.4–1.1 percentage points across two orders of magnitude of total parameters. PUNN's accuracy plateaus near 97.4–97.9%, suggesting that the partition of unity construction imposes a small but consistent capacity cost on this task.

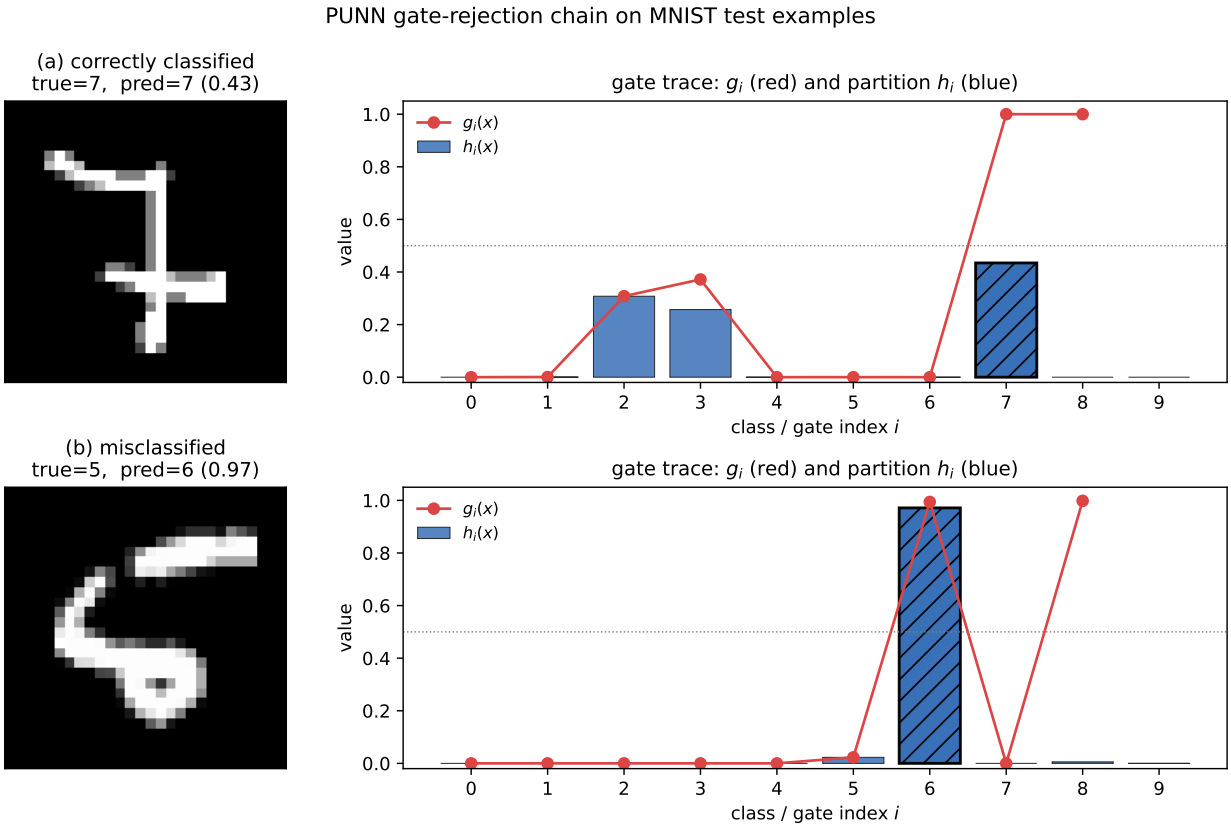

Figure 4: PUNN gate-rejection chain on two MNIST test examples. **(a)** A correctly classified but uncertain "7" (predicted with confidence 0.43). Gates 2 and 3 partially activate ($g_2 = 0.31$, $g_3 = 0.37$), assigning 30.8% and 25.7% probability to classes 2 and 3 respectively before gate 7 fully accepts the remaining mass. The chain exposes the model's ambiguity directly. **(b)** A misclassified example (true class 5, predicted 6 with confidence 0.97). The failure point is visible at gate 5, which should have accepted the true class but produced only $g_5 = 0.023$; gate 6 then captures nearly all remaining probability. Red curve: gate values $g_i(x)$. Blue bars: partition probabilities $h_i(x)$. Hatched bar: predicted class.

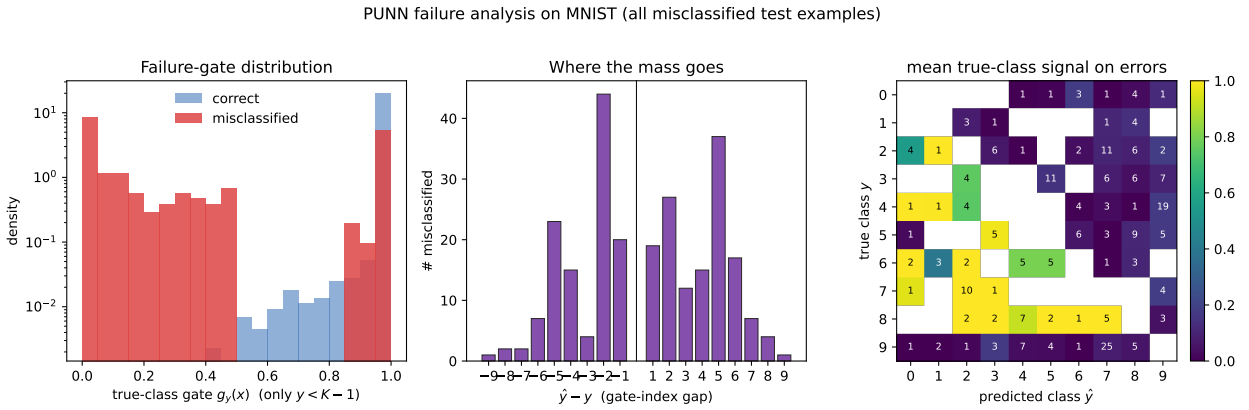

Figure 5: PUNN-Sigma failure analysis on the full MNIST test set. **Left:** density of the true-class gate value $g_y(x)$ on correct (blue) versus misclassified (red) examples. The distribution on errors is bimodal: about half have $g_y < 0.1$ (gate did not fire), while a substantial mass concentrates near $g_y = 1$ (gate fired but mass had already been claimed by an earlier gate). **Middle:** histogram of $\hat{y} - y$ on misclassified examples; errors do not cluster at ± 1, indicating the recursive construction does not bias mistakes toward neighboring gate positions. **Right:** confusion heatmap colored by mean g_y with raw counts overlaid; the $y \in \{6, 7, 8\}$ rows visibly trend toward $g_y \approx 1$ while $y \in \{0, \ldots, 5\}$ rows have $g_y \approx 0$, exposing the position-dependent failure modes.

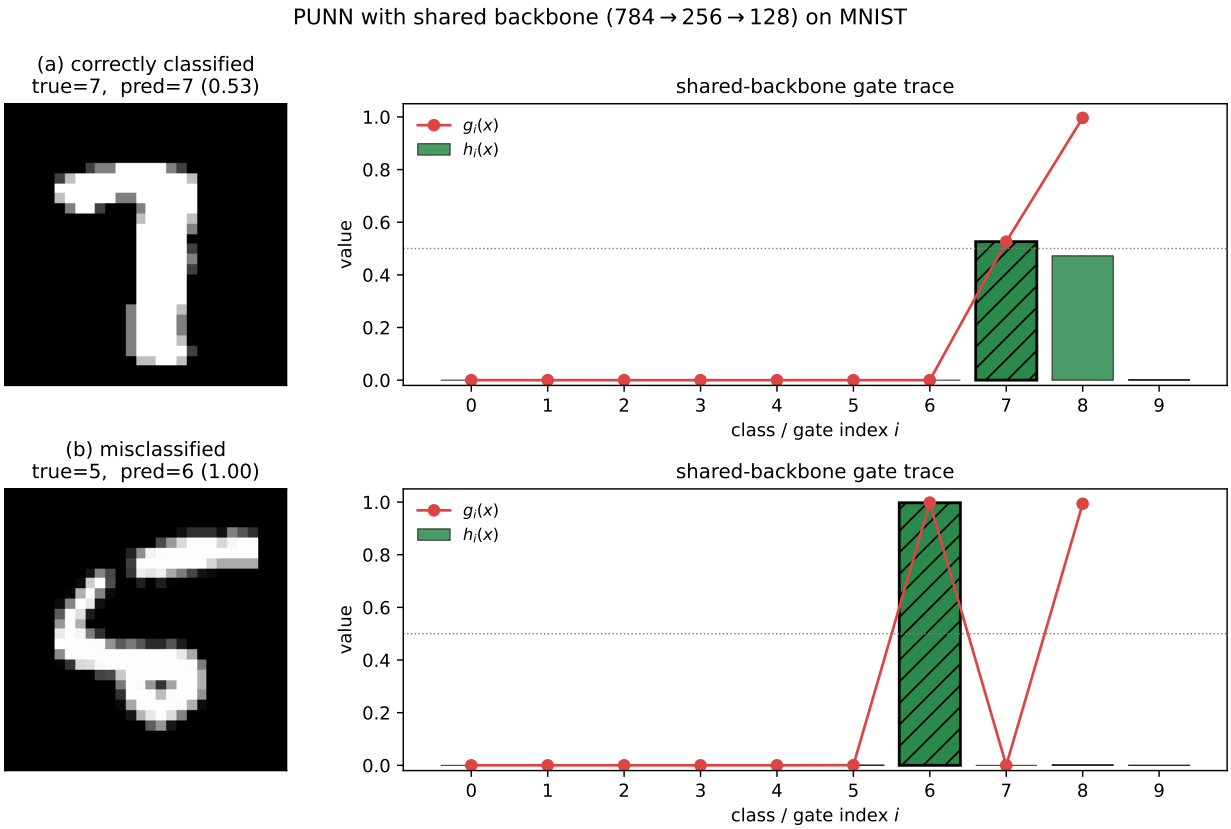

Figure 6: Gate-rejection chain for the shared-backbone PUNN-Sigma on MNIST (235K parameters, 97.85% test accuracy). **(a)** A correctly classified but ambiguous "7" (test index 8519): gate 7 fires at $g_7 = 0.53$, claiming 53% of the probability mass, while gate 8 fires at $g_8 = 0.996$, claiming 47% of the remaining mass, exposing a near-tie between classes 7 and 8. **(b)** A misclassified example (test index 8, true class 5, predicted 6): the failure localizes to gate 5, which produces $g_5 = 7 \times 10^{-4}$ when it should have fired near 1. The shared-backbone variant preserves the same per-gate accept/reject diagnosis as the independent-gate construction (Figure 4).

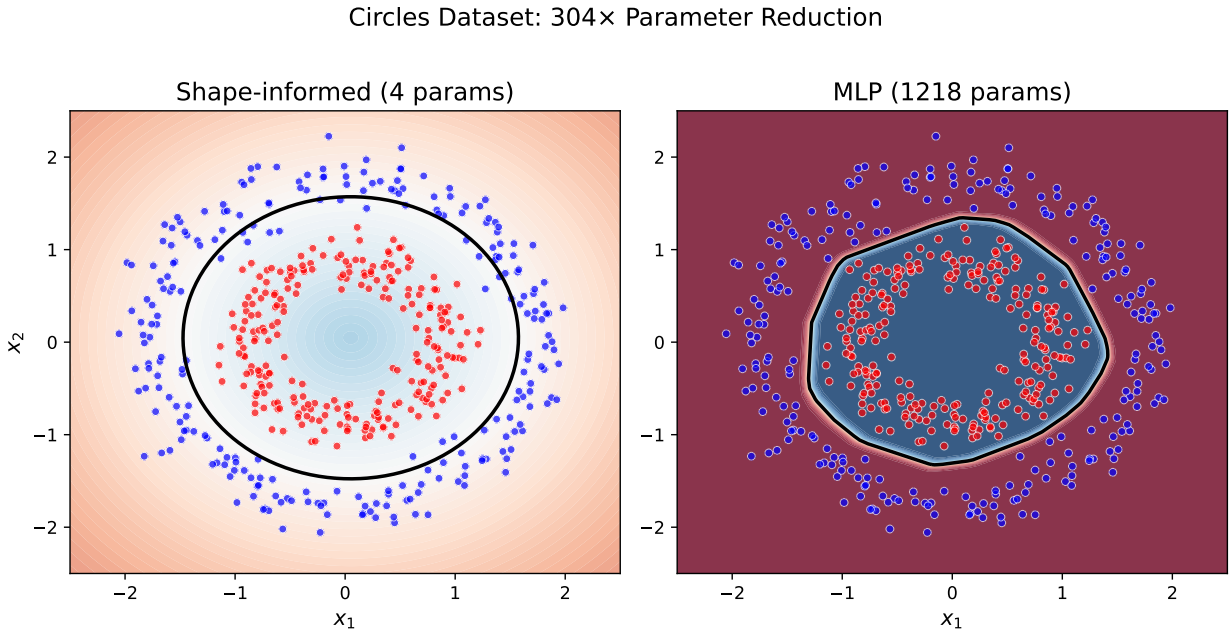

Figure 7: Decision boundaries on Circles dataset. Left: Shape-informed spherical shell (4 parameters) learns a smooth circular boundary. Right: MLP (1,218 parameters) learns an irregular boundary. The shape-informed gate achieves **304× parameter reduction** while producing a more interpretable decision region.

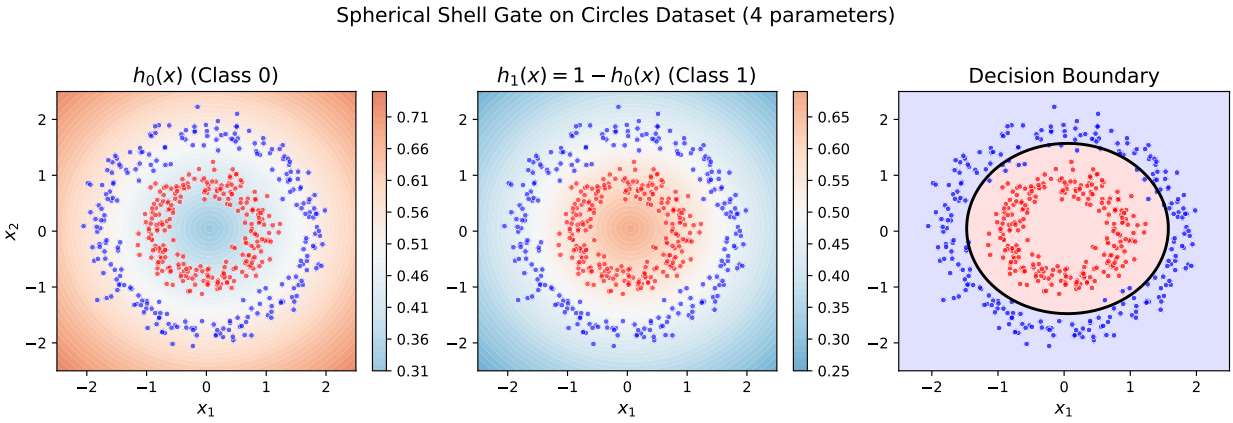

Figure 8: Partition functions learned by a spherical shell gate on Circles. Left: $h_0(x)$ assigns high probability to the inner circle. Middle: $h_1(x) = 1 - h_0(x)$ assigns high probability to the outer ring. Right: Decision boundary at $h_0(x) = 0.5$. The complementary structure ($h_0 + h_1 = 1$) enables direct probabilistic interpretation with only 4 parameters.