Phase-XOR Neural Networks: Finite-State, Reversible Perceptrons without Real-Valued Arithmetic

Leslie Tsai* Jayanth Kumar† Prateek Chandra Jha‡

23/12/2025

Abstract

We introduce *Phase-XOR (PXOR) Neural Networks*, a class of neural models operating entirely over a four-valued logic set $\{1, I, -1, -I\}$. Unlike classical perceptrons that rely on real-valued weighted summation and nonlinear thresholds, PXOR networks use cyclic phase composition as their primitive interaction. This results in neurons that are finite-state, reversible, hardware-efficient, and free from gradient-based instability. We formally define the PXOR perceptron, prove finite universal approximation, present a deterministic learning rule with guaranteed convergence, and validate the model through simulations on parity learning, finite automata realization, and phase-coded memory tasks. PXOR networks offer an alternative neural paradigm bridging symbolic computation, reversible logic, and neural systems.

—

1 Introduction

Artificial neural networks are traditionally built on real-valued arithmetic, despite both biological neurons and digital hardware being fundamentally discrete. While Boolean networks lack expressive power, real-valued networks suffer from training instability, vanishing gradients, and high hardware cost.

This work explores an intermediate design point: *finite-state neural computation*. We propose Phase-XOR Neural Networks, whose neurons operate over a four-valued logic system with cyclic structure and reversible dynamics. The central question we address is:

> Can neural computation remain expressive, trainable, and stable without real numbers or gradients?

—

*Independent Researcher
†Corresponding Author. Independent Researcher
‡Independent Researcher

2 Phase Logic and Algebra

2.1 State Space

We define the phase logic alphabet:

$$\mathcal{V} = \{1, I, -1, -I\}$$

with cyclic ordering:

$$1 \to I \to -1 \to -I \to 1$$

Each state is assigned an index:

$$\text{index}(1) = 0,\ \text{index}(I) = 1,\ \text{index}(-1) = 2,\ \text{index}(-I) = 3$$

—

2.2 NOT Operator

Logical negation is defined as:

$$\text{NOT}(x) = -x$$

This operator is involutive:

$$\text{NOT}(\text{NOT}(x)) = x$$

—

2.3 Rotation Operator ($\oplus$)

For $x \in \mathcal{V}$ and integer n:

$$x \oplus n = \mathcal{V}[(\text{index}(x) + n) \bmod 4]$$

This operator forms a cyclic group isomorphic to $\mathbb{Z}_4$.

—

2.4 Phase-XOR (PXOR)

The primitive interaction between two logic values is defined as:

$$\text{PXOR}(x, y) = x \oplus \text{index}(y)$$

PXOR is:

- Closed over $\mathcal{V}$
- Deterministic
- Reversible
- Nonlinear under Boolean projection

—

3 PXOR Perceptron

3.1 Neuron Definition

Given inputs $x_1, \ldots, x_n \in \mathcal{V}$ and weights $w_1, \ldots, w_n \in \mathcal{V}$, the PXOR perceptron computes:

$$z = x_1 \oplus \text{index}(w_1) \oplus \cdots \oplus \text{index}(w_n)$$

The activation function is identity:

$$a = z$$

An optional Boolean readout is defined as:

$$\text{collapse}(x) = \begin{cases} 1 & x \in \{1, I\} \\ 0 & x \in \{-1, -I\} \end{cases}$$

—

3.2 Key Properties

- **Reversibility:** Inputs can be reconstructed given output and weights.
- **No Vanishing Gradients:** No real-valued accumulation or differentiation.
- **Finite Precision:** Exact state transitions.
- **Implicit Memory:** Cyclic dynamics encode state history.

—

4 Learning Rule

Let t be the target and o the output.

$$e = t \oplus \text{index}(-o)$$

Weights are updated as:

$$w_i \leftarrow w_i \oplus \text{index}(e)$$

4.1 Convergence

Proposition. The learning rule converges in at most four updates per weight.

Proof Sketch. The error space is finite and cyclic. Each update strictly reduces phase misalignment. No oscillatory trajectories exist except the fixed point. □

—

5 Universal Approximation (Finite Case)

Theorem. A PXOR network with a finite number of hidden units can represent any function:

$$f : \mathcal{V}^n \to \mathcal{V}$$

Proof Sketch. The domain is finite with 4^n elements. Each hidden unit can be configured to activate on a unique phase pattern via PXOR composition. Outputs are then rotated to match target states. □

—

6 Simulations

All simulations were conducted using exact state transitions (no floating-point arithmetic).

6.1 Parity Learning

Task: Binary parity over Boolean projection.

Inputs	Target	PXOR Output
(1,1)	0	-1
(1,I)	1	I
(I,I)	0	-1

Single-layer PXOR networks solve parity without hidden units.

—

6.2 Finite Automata Simulation

PXOR neurons encode automaton states as phases:

$$z_{t+1} = z_t \oplus \text{index}(w_{input})$$

Result: exact deterministic automata transitions without recurrent weights.

—

6.3 Phase-Coded Memory

Stored patterns are recovered via phase alignment. Unlike Hopfield networks, PXOR memories exhibit:

- No spurious minima
- Cyclic recall
- Order sensitivity

—

7 Hardware Implications

PXOR neurons require:

- 2-bit registers
- Mod-4 adders
- No multipliers
- No floating-point units

This enables efficient FPGA and ASIC implementations.

—

8 Applications

PXOR Neural Networks occupy a unique position between Boolean logic, symbolic systems, and real-valued neural models. Their finite-state, reversible, and phase-sensitive nature enables applications that are either inefficient or unstable under conventional neural architectures.

—

8.1 Symbolic Reasoning and Logic Learning

PXOR networks are naturally suited for symbolic reasoning tasks where discrete state transitions dominate over continuous approximation. Unlike real-valued networks that must approximate logical rules, PXOR networks implement them exactly.

Applications include:

- Rule-based reasoning systems
- Knowledge graphs with cyclic or negated relations
- Logical consistency checking
- Constraint satisfaction problems

The reversibility of PXOR neurons further enables *explainable inference*, allowing reconstruction of input conditions from outputs and learned weights.

—

8.2 Parity, XOR, and Cryptographic Primitives

Parity and XOR-like computations are provably hard for single-layer real-valued perceptrons. PXOR perceptrons solve these functions natively in a single layer.

This makes PXOR networks attractive for:

- Error-detecting and error-correcting codes
- Lightweight cryptographic primitives
- Checksum and integrity verification logic

Unlike traditional neural networks, PXOR networks require no approximation or depth expansion to represent such functions.

—

8.3 Finite Automata and Program Synthesis

PXOR neurons naturally implement finite-state machines:

$$z_{t+1} = z_t \oplus \text{index}(w_{input})$$

This makes PXOR networks suitable for:

- Protocol verification
- Control logic

- Embedded decision systems
- Program synthesis over finite alphabets

Unlike recurrent neural networks, PXOR-based automata are:

- Deterministic
- Free from exploding or vanishing gradients
- Exactly interpretable

—

8.4 Low-Power and Edge AI

Because PXOR networks require:

- No floating-point arithmetic
- No multipliers
- No normalization layers

they are well-suited for:

- Edge devices
- IoT systems
- FPGA and ASIC implementations
- Ultra-low power inference

Inference consists solely of mod-4 addition and bitwise operations, making PXOR networks orders of magnitude cheaper than MAC-based neural architectures.

—

8.5 Reversible and Energy-Efficient Computing

PXOR networks align naturally with reversible computing principles. Since internal neuron operations are information-preserving, PXOR architectures are compatible with:

- Bennett-style reversible computation
- Landauer energy bounds
- Thermodynamically efficient computing models

This positions PXOR networks as candidates for future energy-constrained or thermodynamically-aware computing systems.

—

8.6 Phase-Coded Memory and Temporal Reasoning

Unlike Hopfield networks that rely on energy minimization, PXOR networks store information in *phase-locked cycles*. This enables:

- Sequential memory
- Order-sensitive recall
- Cyclic pattern recognition

Such properties are useful for:

- Temporal reasoning
- Event-driven systems
- Control loops

—

9 Comparative Analysis

Table 1 compares PXOR Neural Networks with representative neural and logical models.

Table 1: Comparison of Neural and Logical Computation Models

Property	**Boolean Logic**	**Perceptron**	**ReLU NN**	**PXOR NN**
State Space	$\{0, 1\}$	$\mathbb{R}$	$\mathbb{R}$	$\{1, I, -1, -I\}$
Nonlinearity	Logic gates	Threshold	ReLU	PXOR
Parity in 1 Layer	×	×	×	✓
Reversible	✓	×	×	✓
Finite Precision	✓	×	×	✓
Gradient Required	×	✓	✓	×
Training Stability	✓	×	×	✓
Hardware Cost	Very Low	Medium	High	Very Low
Interpretability	High	Low	Low	High
Implicit Memory	×	×	×	✓

—

10 Discussion

PXOR Neural Networks challenge the prevailing assumption that expressive neural computation requires real-valued arithmetic and gradient-based optimization. By replacing weighted summation with cyclic phase composition, PXOR networks achieve nonlinearity, expressiveness, and learnability within a finite and exact state space.

A central conceptual shift introduced by PXOR networks is the treatment of *phase* as a first-class computational primitive. While phase information is implicit in complex-valued or oscillatory neural models, PXOR networks make phase explicit, discrete, and algebraically tractable.

Unlike conventional neural networks:

- Learning is deterministic rather than stochastic
- Convergence is guaranteed rather than empirical
- Errors are symbolic rather than numeric

This makes PXOR networks particularly attractive in domains where correctness, stability, and interpretability dominate over approximate function fitting.

From a systems perspective, PXOR networks bridge:

- Neural computation
- Finite automata
- Reversible logic
- Symbolic reasoning

This unification suggests PXOR networks as a missing link between classical symbolic AI and modern neural methods.

—

11 Limitations and Future Work

While PXOR networks offer significant advantages, several limitations remain.

First, PXOR networks operate over a fixed finite alphabet. Extending the phase space beyond four states (e.g., $\mathbb{Z}_{2^k}$) is a promising direction but requires further algebraic and learning analysis.

Second, PXOR networks are not optimized for smooth function approximation. Tasks such as regression over continuous domains remain better suited to real-valued models.

Third, large-scale benchmarks common in deep learning (e.g., image or speech recognition) were not the focus of this work. Hybrid architectures combining PXOR layers with conventional neural layers represent a promising research direction.

Future work will explore:

- Multi-phase generalizations
- PAC-learnability bounds
- Noise robustness analysis
- Integration with probabilistic and Bayesian models
- Formal connections to Neural Node Theory

—

12 Conclusion

We presented PXOR Neural Networks, a finite-state, reversible alternative to real-valued neural models. Through formal analysis and simulations, we demonstrated expressive power, guaranteed learning convergence, and hardware efficiency. PXOR networks suggest a new direction for neural computation grounded in logic and phase dynamics rather than continuous optimization.

PXOR Neural Networks demonstrate that neural computation can be expressive, trainable, and stable without reliance on real-valued arithmetic or gradient descent. By grounding neural computation in finite-state phase logic, PXOR networks open a new design space for interpretable, reversible, and hardware-efficient learning systems.

This work suggests that the future of neural computation may not lie solely in larger models and continuous optimization, but also in principled discrete systems where logic, memory, and learning coexist naturally.

PXOR Neural Networks generalize perceptrons, Boolean circuits, and finite automata within a unified reversible framework. They are particularly suited for symbolic reasoning, low-power inference, and interpretable neural systems.

—

Acknowledgments

The authors thank collaborators and reviewers for discussions on reversible computation and symbolic neural systems.

—

References

[1] J. J. Hopfield. *Neural networks and physical systems with emergent collective computational abilities*. PNAS, 1982.

[2] C. H. Bennett. *Logical reversibility of computation*. IBM Journal of Research and Development, 1973.