

Article

Design of Oscillatory Neural Networks Using Machine-Learned Templates

Mitra Moayed and Gyorgy Csaba * 

Faculty of Information Technology and Bionics, Pázmány Péter Catholic University, 1083 Budapest, Hungary; mitra.moayed@itk.ppke.hu

* Correspondence: csaba.gyorgy@itk.ppke.hu

Abstract

Oscillatory neural networks (ONNs) provide a neuromorphic computing framework that exploits the phase dynamics of coupled oscillators for parallel and energy-efficient pattern recognition. In this study, we design a single-layer, fully connected ONN to classify handwritten digits from the MNIST dataset. Input images were downsampled to 6×6 binary patterns, which were optimized using a genetic algorithm to evolve effective templates, as experiments with higher-resolution inputs showed only marginal accuracy improvements at significantly increased computational and energy costs. Coupling weights were determined using Hebbian learning, and the network dynamics were simulated using the Kuramoto model to encode information via phase relationships. To the best of our knowledge, this is the first work to apply genetic algorithm optimization to design the templates used by an ONN and to combine evolutionary template generation with Hebbian-based ONN training for image classification. The results show that the ONN achieves 75–76% accuracy in the full 10-class MNIST task, with outputs exhibiting stable sinusoidal behavior and resilience to moderate noise. These findings highlight the potential of ONNs as a practical, low-power alternative to conventional deep learning models, particularly for real-time edge-level applications where energy efficiency and robustness are critical.

Keywords: oscillatory neural networks; neuromorphic computing; VO₂ oscillators; kuramoto dynamics; hebbian learning; GA-based optimization; low-power image recognition

1. Introduction

In this work, we design an oscillatory neural network (ONN) for classifying handwritten digits from the MNIST dataset. The system uses a single-layer network of 36 coupled oscillators governed by the Kuramoto model. Input images are downsampled to 6×6 binary patterns and set as initial phases. Reference patterns for each class are optimized with a genetic algorithm, and coupling weights are formed using the Hebbian learning rule. Classification relies on the final phase relationships among the oscillators. Simulations show 75–76% in the full 10-class problem, with stable sinusoidal outputs even under moderate noise. This design addresses the need for energy-efficient alternatives to neural networks, especially for edge devices with limited resources.

The concept of synchronization-based computation traces back to Winfree's early work on coupled oscillators [1], later formalized by Kuramoto's phase-interaction model [2], which forms the basis of modern ONN architectures. ONNs offer several advantages over conventional deep learning models such as CNNs: they rely on phase alignment rather than gradient-based updates, enabling inherently parallel and low-power computation.



Academic Editor: Fernando De la Prieta Pintado

Received: 15 May 2026

Revised: 25 June 2026

Accepted: 26 June 2026

Published: 2 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

CMOS and VO₂ oscillators provide scalable, room-temperature implementations with energy consumption in the femtojoule range [3–5].

Moreover, recent demonstrations of large-scale neuromorphic arrays, including wafer-scale optoelectronic computing devices [6] and monolithic oscillatory chemoreceptive chips [7], further validate the practical scalability of oscillatory computing architectures.

These properties make ONNs well suited for pattern recognition at the edge. Recent experimental studies on VO₂ and CMOS oscillators have shown that their phase dynamics can be accurately modeled using Kuramoto-type abstractions, suggesting that ONN architectures are compatible with practical hardware implementations [8].

For an ONN to operate as an associative circuit, each class requires a template that the network can match against its input. To optimize ONN templates without extensive manual trial and error, genetic algorithms (GAs) provide a powerful tool. Inspired by natural evolution, GAs evolve populations of candidate solutions through selection, crossover, and mutation to identify effective parameters such as coupling strengths or training templates. The concept was first introduced by Holland in his seminal work on adaptation in natural and artificial systems [9], and subsequent studies have demonstrated their versatility across engineering and machine learning applications [10]. In neuromorphic contexts, GAs have been shown to complement Hebbian learning by improving robustness under noisy conditions [11]. Building on this, our work employs GA-based optimization to generate effective training patterns for Hebbian-trained ONNs [12,13].

The Kuramoto-based ONN architecture we introduce in this work is in fact a model of a real analog circuit. Large ONNs were built for different purposes, such as the Ising-machine architectures [14]; our design is a relatively simple one for classification tasks.

1.1. Hopfield Neural Networks (HNNs) and Auto-Associative Memory (AAM)

Hopfield Neural Networks (HNNs) are classical energy-based models capable of auto-associative memory, where stored patterns are retrieved by converging to stable attractor states [12,13]. This property enables robust pattern recall from noisy or incomplete inputs and has inspired a wide range of neuromorphic architectures, including phase-based and oscillator-driven models such as phase-locked loop networks [15] and more recent deep oscillatory frameworks [16]. While HNNs illustrate how collective dynamics can support memory and computation, their scalability and classification performance remain limited. Building on this associative principle, the proposed ONN replaces binary neuron states with continuous phase dynamics, enabling more robust and scalable synchronization-based computation. Here, we employ an ONN layer with Hebbian-inspired coupling to encode 6×6 binary patterns through coupled phase dynamics.

1.2. Preliminary Work and Basics

In ONN architectures, information is typically encoded in the relative phase difference between oscillators. For classification tasks, two stable synchronization modes are commonly employed: in-phase locking ($\Delta\theta \approx 0^\circ$) representing one class and anti-phase locking ($\Delta\theta \approx 180^\circ$) representing the other. Coupling between oscillators is generally achieved through resistive or capacitive elements. For instance, resistive coupling enhances the interaction between oscillators whose phases align during training and allows for weight adjustment via continuous analog modulation rather than discrete digital updates. Operating fully in the analog domain enables ONNs to perform continuous, parallel computation without clock cycles, which is advantageous for real-time edge applications. Previous studies have demonstrated that ONNs employ nonlinear devices to achieve phase synchronization under specific operating conditions. Sharma et al. demonstrated how phase coupling in oxide-based oscillators can be controlled to support neuromorphic comput-

ing [17]. Similarly, Wang et al. modeled memristor-based oscillators for ONN pattern recognition, highlighting their potential for robust synchronization [18].

1.3. State of the Art in HNNs and ONNs for Image Classification

Hopfield Neural Networks (HNNs) and Oscillatory Neural Networks (ONNs) have recently been investigated as energy-based alternatives for image classification. Early work by Belyaev and Velichko [19] applied the Storkey learning rule to Hopfield networks and achieved approximately 61% accuracy on a deskewed 14 × 14 MNIST dataset, highlighting the limitations of associative memory models when used for classification. Building on this, Abernot and Todri-Sanial [20] extended the approach to ONNs, reporting 65.2% accuracy for HNNs and 59.1% for ONNs on a simplified 10 × 10 MNIST set. They also introduced the AAM-EP algorithm, which improved performance to 67% and 62.6% respectively. More recently, Sabo and Todri-Sanial [21] proposed ClassONN, a Kuramoto-based ONN model, and demonstrated classification of the full 28 × 28 MNIST digits with approximately 70% accuracy on the training set and 72% on the test set. Recent reviews such as Todri-Sanial et al. [22] emphasize that ONNs are energy-based computing models whose dynamics can be described by synchronization phenomena and formalized through the Kuramoto model. Synchronization not only enables information encoding in phase relationships but also provides intrinsic memory to the network. Other works explore advanced training strategies, including machine-learning-based ONN design [23]. In parallel, Cai et al. (2025) introduced OscNet v1.5, a Hopfield-inspired ONN implemented on CMOS oscillator fabrics for energy-efficient image classification [24]. While these works highlight the potential of more complex and hardware-oriented ONN architectures, our approach focuses on a simple and lightweight design, which makes implementation easier while still achieving competitive performance.

To the best of our knowledge, this is the first work to apply genetic algorithm optimization directly to the design of oscillatory neural networks, combining evolutionary template generation with Hebbian-based ONN training for image classification. Table 1 summarizes these frameworks and positions our work among them.

Table 1. Summary of progress in neuromorphic computing frameworks.

Framework	MNIST Acc.	Key Idea
HNN (Storkey rule)	~61%	Energy-based associative memory
Kuramoto ONN (10 × 10)	59–65%	Phase synchronization
ClassONN	70–72%	Full-size Kuramoto ONN
This Work	75–76%	GA-optimized Kuramoto ONN

The closest prior approach is ClassONN [21], which uses the same Kuramoto ONN dynamics and Hebbian-trained templates on the same task; the key distinction is our GA-based template optimization, which improves accuracy beyond manually designed or Hebbian-only templates without adding network complexity.

2. Materials and Methods

We developed an oscillator-based neural network (ONN) to perform image classification. The model leverages synchronization dynamics of coupled oscillators, with parameters optimized through a genetic algorithm. The following subsections describe the system architecture, dynamical framework, and evaluation methods in detail.

2.1. Kuramoto Dynamics and Parallel Classification Framework

The classification mechanism is governed by the Kuramoto model dynamics:

$$\frac{d\theta_i}{dt} = \omega + \sum_{j=1}^N W_{ij} \cdot \sin(\theta_j - \theta_i),$$

where θ_i denotes the phase of the i -th oscillator and ω is the uniform natural frequency across all oscillators. The initial phases $\theta(0)$ are provided by the preprocessed MNIST images, mapped to $\{-\pi/2, +\pi/2\}$, drawn from the training split during GA optimization and from the test split during final evaluation, while the coupling weights W , derived from prototype patterns via the Hebbian rule, remain fixed during classification. The system dynamics are numerically integrated using the Runge–Kutta (RK45) method to ensure accuracy and stability. To reduce computational cost, simulations for different test images are distributed across multiple CPU cores. The Kuramoto equation yields temporal phase trajectories for each test image, from which binary outputs are extracted. Oscillator phases are mapped into binary states through a normalization and thresholding process, similar to approaches in oscillatory neural networks where binary states are represented by synchronized versus desynchronized phase relations [15]. The ONN architecture consists of a network of phase-coupled oscillators governed by the Kuramoto model, with class-dependent coupling structures generated through a Hebbian learning rule and refined via a genetic algorithm.

2.2. System Architecture and Evolutionary Training Process

Our oscillatory neural network (ONN) consists of 36 coupled oscillators. Data preprocessing employed the MNIST handwritten digit dataset for binary classification. Each 28×28 pixel image was cropped to 24×24 , downsampled to 6×6 resolution (yielding a 36-dimensional vector), binarized, and then mapped to binary values $\{-1, +1\}$. As part of preprocessing, these values were multiplied by $\pi/2$, resulting in initial oscillator phases of $\{-\pi/2, +\pi/2\}$ to align with the system dynamics. In this configuration, each pixel of the 6×6 image corresponds to one oscillator, resulting in a network of 36 oscillators that collectively represent the input pattern. The test images serve as the initial phases in the system. The preprocessing steps described are illustrated in Figure 1A. Prototype patterns for each class were generated using a genetic algorithm. These patterns then determine the coupling weights according to the Hebbian learning rule, so that the weight matrix is not directly evolved but instead derived from the optimized class prototypes. The resulting weights constitute the trained network parameters that process the inputs through the Kuramoto dynamics. The GA-optimized templates enhance the ONN by producing class-specific patterns that maximize phase separability and improve the quality of the Hebbian weight matrix.

The Kuramoto model governs their dynamics, with the phase of each oscillator evolving according to the coupling interactions. Figure 1B illustrates the ensemble of binary classifiers used in this work. Specifically, we employ 45 two-class classifiers, which together cover all pairwise combinations of the ten digit classes (0–9). As an example, one such binary classifier distinguishing between class 0 and class 1 is shown in this part of the figure. Given the input pattern and the synaptic weight matrix constructed by the ONN, the network produces a binary output representation. The decision of each binary classifier is then obtained by processing the ONN output using three different decision methods, which are described in detail later in this paper. Figure 1C presents the decision aggregation process and the final classification stage. In this step, the outputs of all binary classifiers are collected and accumulated into a vote count matrix, where each classifier contributes

one vote to one of the digit classes. Among the proposed decision strategies, the minimum energy criterion is visualized in this figure. Based on the computed energy values for each class, the corresponding votes are assigned and the vote matrix is populated. Finally, the predicted label is determined using majority voting, where the class receiving the highest number of votes is selected as the final output. This ensemble-based voting mechanism improves robustness and ensures a reliable final classification result.

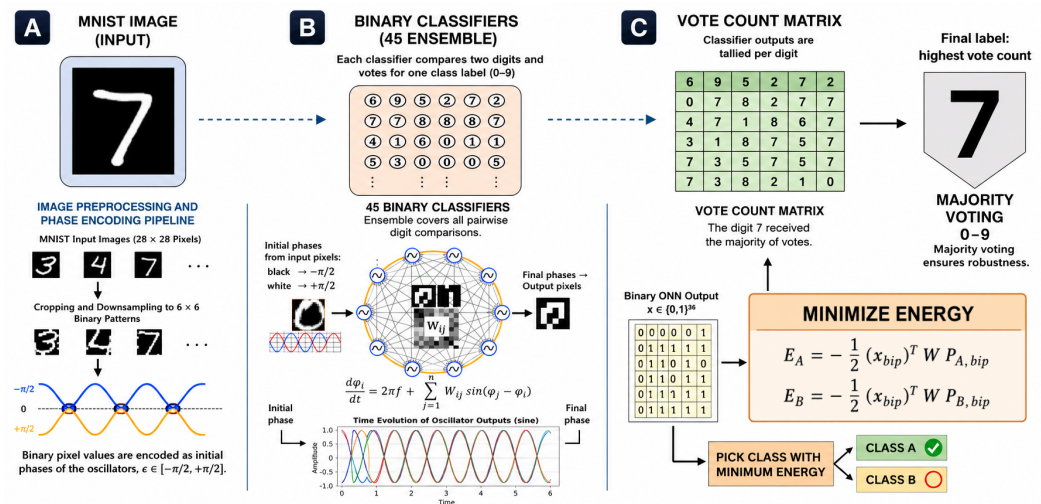


Figure 1. Overall pipeline of the proposed ONN for MNIST digit classification. (A) Input preprocessing and phase encoding of the 6×6 binary pattern. (B) Ensemble of 45 binary ONN classifiers covering all digit pairs. (C) Decision aggregation via a vote-count matrix and final majority voting.

In the following, we describe in detail the process of constructing the weight matrices using the Hebbian learning rule and the genetic algorithm. The overall procedure is illustrated in Figure 2. The coupling weights w_{ij} adapt via the Hebbian learning rule, defined by the weight matrix:

$$W_{ij} = \begin{cases} \frac{1}{N} x_i x_j & \text{if } i \neq j, \\ 0 & \text{if } i = j, \end{cases} \quad (1)$$

This formula computes the weight matrix from the outer product of the input pattern with its transpose, normalized by the pattern dimension N . The resulting matrix W captures the correlations between the input patterns, which is fundamental to the Hebbian learning rule. Here, $x_i \in \{-1, +1\}$ are binary vectors, $N = 36$ is the pattern dimension, and $W_{ii} = 0$ prevents self-coupling. This supports stable synchronization, local learning, and sparse connectivity. This setup lets us examine how the system preserves synchronization under different initial conditions, coupling matrices, and input patterns.

To optimize the binary prototypes for each digit class, we implement a genetic algorithm (GA) that iteratively evolves a population of candidate templates. The GA begins with an initial population of 30 individuals, where each chromosome represents a 6×6 binary pattern (36 bits, encoded as 0/1 for phase mapping). The search space is bounded to integer values $[0, 1]$ for each bit, ensuring feasible binary representations. The algorithm runs for a maximum of 100 generations or terminates early if no improvement is observed for 100 consecutive iterations, balancing exploration and convergence.

Selection employs tournament sampling with a tournament size of 4, where the fittest parent from each group advances, promoting diversity while favoring high-performing candidates. Crossover uses a two-point method with a probability of 0.5, recombining segments from two parents to generate offspring. Mutation applies bit-flip operations at a

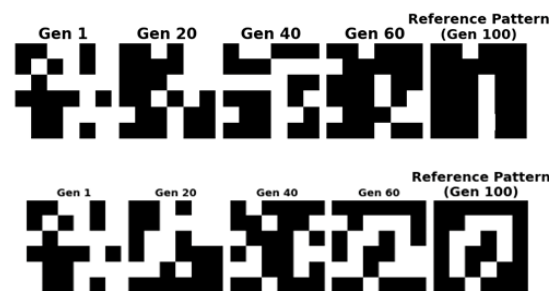
probability of 0.06, introducing small variations to escape local optima. An elite ratio of 0.5 preserves the top 50% of the population unchanged, and parents constitute 50% of the next generation, ensuring a balanced mix of inheritance and innovation.

These parameter values were chosen to balance exploration and exploitation in a 36-bit binary search space. A tournament size of 4 provides sufficient selective pressure without premature convergence, while crossover at 0.5 and mutation at 0.06 are moderate rates effective for binary optimization problems of this scale. The elite ratio of 0.5 and 50% parent contribution preserve high-quality solutions while allowing adequate variation. These settings follow standard recommendations and common practices for genetic algorithms in template optimization tasks [25], supporting stable convergence toward digit-specific features that enable robust phase synchronization in the ONN.

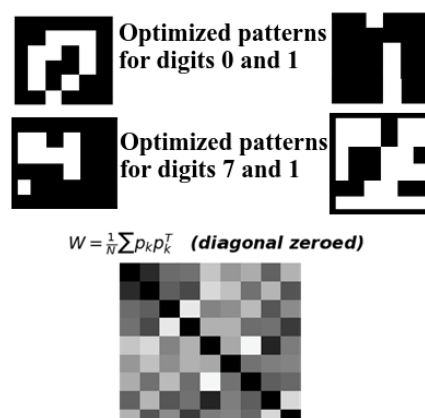
For each digit pair, fitness was evaluated on 2000 training samples; all final accuracy results are reported on the held-out MNIST test set of 10,000 images. For each prototype $\mathbf{p}$, Hebbian weights are computed as $\mathbf{W} = \frac{1}{N} \mathbf{p} \mathbf{p}^T$ (with diagonal zeroed). This metric directs the GA toward templates that capture key digit features, such as loops or lines.

The GA-evolved prototype patterns are optimized for phase separability within the Kuramoto dynamics and are not necessarily visually resemble the target digit classes. We observed that for certain digits, such as digit 0, the evolved prototypes retain some resemblance to the actual digit shape, whereas for others, such as digits 1 and 7, the prototypes differ substantially in visual appearance from the corresponding digit.

Genetic Algorithm Optimization of Reference Patterns



Optimized Reference Patterns → Hebbian Weight Matrix Construction



Weight matrix $\mathbf{W}$ (36×36) constructed via the Hebbian rule using GA-optimized reference patterns

Figure 2. Hebbian weight matrix (36×36) derived from GA-optimized reference patterns for the digit pairs (0–1 and 1–7). The 0–1 patterns look similar to their digits, whereas the 7–1 patterns do not visually match the digit shapes.

Figure 2 illustrates the methodology of the evolutionary algorithm applied to pattern generation. The top row visualizes the evolutionary process for Class 0, where each chromosome is represented as a 6×6 binary matrix. The process initializes with entirely random configurations (Generation 1) and undergoes iterative selection and variation, progressively converging toward a coherent structural pattern over successive generations.

The row below depicts the analogous evolutionary pathway for Class 1, with each generation's chromosome directly rendered as a 6×6 image. The algorithm initiates from random prototypes for both classes and refines them through a defined fitness function. The highest classification accuracy for the system was attained at Generation 100. The optimized reference patterns for both Class 0 and Class 1 were subsequently extracted from the population at this convergence point. Once the reference patterns have been obtained, the corresponding weight matrix is subsequently constructed from them and presented as a heatmap in Figure 2. The genetic algorithm improves each class-specific prototype by evolving a population of candidate patterns and selecting those that yield better class separation under the Hebbian-based coupling.

2.3. Performance Evaluation, Noise Robustness, and Output Analysis Methods

We assess robustness by introducing random phase deviations to the input phases in a simple binary (0 vs. 1) classification task. This setup reflects realistic imperfections that may occur in analog implementations, where the programmed initial phases can deviate from their intended values. The updated results, now presented in Figure 3, demonstrate that the GA-ONN maintains stability across a wide range of noise levels: the accuracy remains above 97% for moderate deviations and only decreases to 94.5% at the maximum tested perturbation ($\sigma = 0.8$ rad).

Resilience of GA-ONN to Increasing Phase Noise in Binary Classification

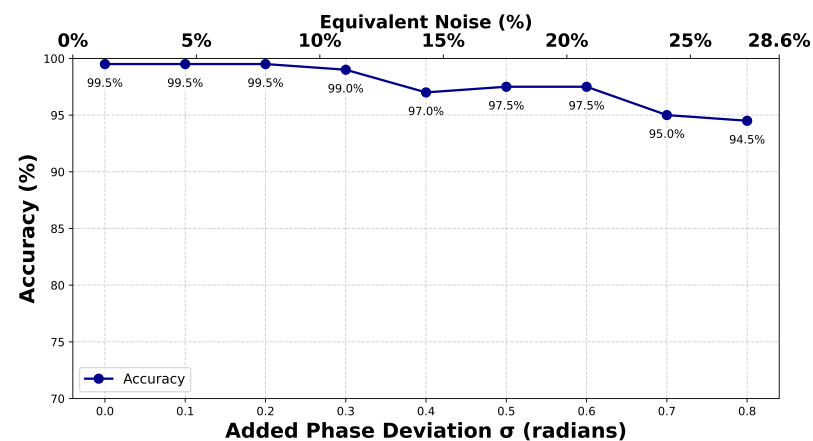


Figure 3. Resilience of the GA-ONN to increasing phase deviations in the binary (0 vs. 1) classification task. The plot shows how accuracy changes as added phase noise (σ in radians) increases, with the upper axis indicating the equivalent percentage of the total phase range.

To assess the effectiveness of the classification system, the output states from the Kuramoto dynamics were compared with the optimized prototype patterns using several complementary criteria, including distance-based similarity measures and energy-based analysis. Detailed comparisons of the individual metrics are presented in Section 3.

2.4. Implementation Details

All simulations were implemented in Python 3.12 using the NumPy and SciPy libraries. The most time-consuming step is the GA optimization to find the optimal binary prototype

patterns for each digit pair. However, they have to be done only ‘once’ and this pays off for inference-heavy tasks.

The Kuramoto dynamics were integrated with the Runge–Kutta (RK45) solver. To accelerate computation, parallel processing was employed across multiple CPU cores, allowing independent simulations for different test images to run concurrently. All reported multi-class accuracy results were evaluated on the full MNIST test set of 10,000 samples. The optimized prototype patterns for all digit pairs are publicly available in the GitHub repository. Using these pre-computed patterns on a 32-core CPU, the full inference on the MNIST test set requires approximately 3 to 4 h.

3. Results

3.1. Dynamics of Oscillator Synchronization

The temporal evolution of oscillator phases highlights how synchronization gradually develops across the network, which is essential for classification. Figure 4 shows how oscillator outputs converge toward stable states, while Figure 5 illustrates the stabilization of phase relationships over time.

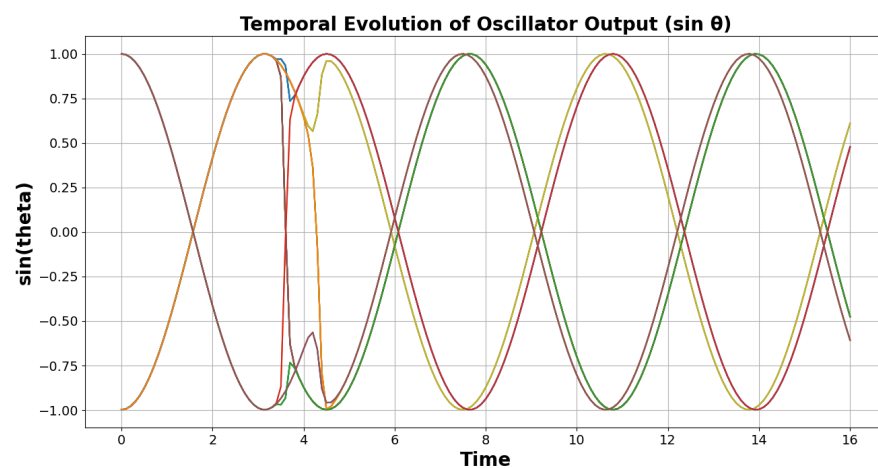


Figure 4. Temporal evolution of oscillator outputs ($\sin \theta$) showing convergence to stable synchronization states during classification. Each colored curve represents the sinusoidal output of an individual oscillator in the 36-oscillator network.

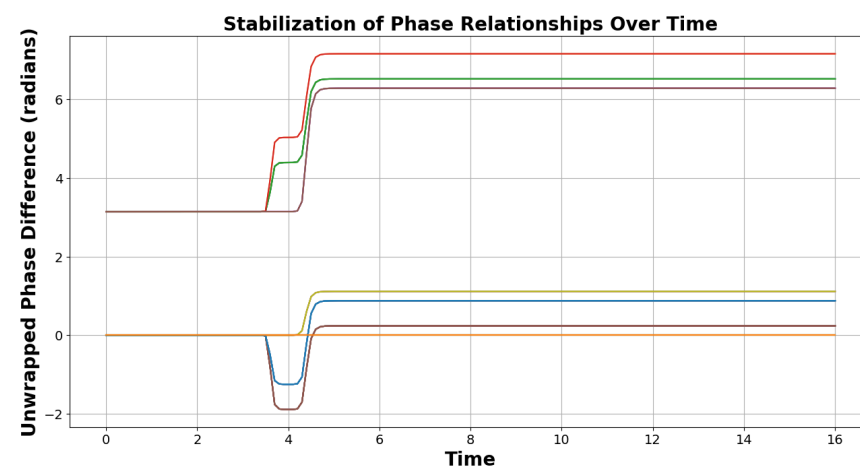


Figure 5. Stabilization of phase relationships over time. The unwrapped phase differences between the reference oscillator and all other oscillators converge to stable values, indicating successful pattern encoding through phase synchronization. Each colored curve corresponds to one oscillator’s phase difference relative to the reference.

As shown in Figure 4, the oscillator outputs align within roughly 5 s of simulation time, producing consistent sinusoidal patterns across the network. This convergence is crucial for reliable classification because it guarantees that the network produces stable and consistent outputs for every input. Figure 5 demonstrates how phase differences settle into steady values, reflecting coherent synchronization among oscillators. During the initial transient phase (0–5 s), the system adjusts to the input pattern. This is followed by stable phase locking, which encodes the pattern and enables accurate recognition.

3.2. Multi-Class Architecture and Recognition Performance

A single ONN layer that would alone classify all 10 MNIST classes is likely not feasible—even more involved training algorithms [23] did not yield reasonable results.

To extend digit recognition to all ten classes (0–9), we developed a framework based on multiple binary ONN classifiers in a one-vs-one configuration. As shown in Figure 1, the system processes MNIST digits through 45 specialized binary classifiers, with the final decision obtained through majority voting.

The classification pipeline consists of four main stages:

- **Input Preprocessing:** Each MNIST image is cropped and resized into a 6×6 binary grid, then mapped into the phase domain for oscillator compatibility.
- **Binary Classifier Array:** A set of 45 binary ONN classifiers, each optimized to distinguish between digit pairs (i, j) where $0 \leq i < j \leq 9$.
- **Output Matrix Construction:** Classifier decisions populate a symmetric output matrix, where each element M_{ij} indicates the preferred class for the corresponding digit pair.
- **Majority Voting:** The final digit label is selected as the class receiving the highest number of votes across all binary decisions.

Together, these stages extend binary ONN classification to the full 10-digit task, with majority voting over the 45 pairwise classifiers producing the final label.

3.3. Classification Strategies

Beyond the multi-class architecture, we evaluated three distinct strategies for interpreting the oscillator network outputs. Each method leverages the binary nature of the system in a different way, as described below.

3.3.1. Energy-Based Classification (Accuracy: 75.5%)

In binary classification, each ONN classifier is trained to distinguish between two digit classes, denoted as patterns $\mathbf{p}_A$ and $\mathbf{p}_B$. For a given test output vector $\mathbf{x} \in \{0, 1\}^n$,

$$\mathbf{x}^{\text{bip}} = 2\mathbf{x} - \mathbf{1} \in \{-1, +1\}^n$$

Each reference pattern is similarly converted:

$$\mathbf{p}_A^{\text{bip}} = 2\mathbf{p}_A - \mathbf{1}, \quad \mathbf{p}_B^{\text{bip}} = 2\mathbf{p}_B - \mathbf{1}$$

The Hebbian weight matrix is constructed as

$$\mathbf{W} = \mathbf{p}_A^{\text{bip}}(\mathbf{p}_A^{\text{bip}})^T + \mathbf{p}_B^{\text{bip}}(\mathbf{p}_B^{\text{bip}})^T$$

Then, the energy of the test output with respect to each pattern is computed as

$$E_A = -\frac{1}{2}(\mathbf{x}^{\text{bip}})^T \mathbf{W} \mathbf{p}_A^{\text{bip}}, \quad E_B = -\frac{1}{2}(\mathbf{x}^{\text{bip}})^T \mathbf{W} \mathbf{p}_B^{\text{bip}}$$

The predicted label is assigned to the pattern with lower energy:

$$\hat{y} = \begin{cases} A, & \text{if } E_A < E_B \\ B, & \text{otherwise} \end{cases}$$

To account for possible inversion in oscillator outputs, the same energy computation is repeated for the complement $1 - \mathbf{x}$, and the final decision is based on the minimum energy across both forms:

$$\hat{y} = \arg \min_{y \in \{A, B\}} \{E_y(\mathbf{x}), E_y(1 - \mathbf{x})\}$$

3.3.2. Reference Pattern Matching (Accuracy: 75%)

In this binary classification strategy, each ONN classifier compares the test output vector $\mathbf{x} \in \{0, 1\}^{36}$ directly against two reference patterns: $\mathbf{p}_A$ and $\mathbf{p}_B$. The classification decision is based on exact or approximate similarity.

A test output $\mathbf{x}$ is assigned label A if it satisfies any of the following conditions:

- **Exact match:** $\mathbf{x} = \mathbf{p}_A$.
- **Inverse match:** $\mathbf{x} = 1 - \mathbf{p}_A$.
- **Threshold similarity:** $\sum_{j=1}^{36} |x_j - p_{A,j}| \leq \theta$.
- **Inverse threshold similarity:** $\sum_{j=1}^{36} |(1 - x_j) - p_{A,j}| \leq \theta$.

Similarly, label B is assigned if any of the same conditions are satisfied with respect to $\mathbf{p}_B$. The threshold $\theta = 6$ allows for up to six pixel mismatches, enabling tolerance to minor distortions in the oscillator output. If the number of mismatches exceeds 6 pixels (i.e., $H(\mathbf{x}, \mathbf{p}_A) > 6$ and $H(\mathbf{x}, \mathbf{p}_B) > 6$), the input is considered an unrecognized/distorted image (or rejected as not belonging to either class A or B).

3.3.3. Hamming Distance Classification (Accuracy: 76%)

This strategy relies on minimizing the Hamming distance between the test output and two reference patterns. For each binary ONN classifier, the reference patterns $\mathbf{p}_A$ and $\mathbf{p}_B$ represent the two digit classes. Given a test output vector $\mathbf{x} \in \{0, 1\}^{36}$, we compute the Hamming distance to both patterns and their complements. The Hamming distance between $\mathbf{x}$ and a pattern $\mathbf{p}$ is defined as

$$d(\mathbf{x}, \mathbf{p}) = \sum_{j=1}^{36} (x_j \neq p_j)$$

For each pattern, we compute

$$d_A = \min(d(\mathbf{x}, \mathbf{p}_A), d(\mathbf{x}, 1 - \mathbf{p}_A))$$

$$d_B = \min(d(\mathbf{x}, \mathbf{p}_B), d(\mathbf{x}, 1 - \mathbf{p}_B))$$

The predicted label is assigned to the pattern with the smaller distance:

$$\hat{y} = \begin{cases} A, & \text{if } d_A < d_B \\ B, & \text{otherwise} \end{cases}$$

The consistent performance across all three methods (75–76% accuracy on MNIST) confirms the robustness of oscillatory neural networks for pattern recognition tasks. Although the three methods yield close accuracies (75%, 75.5%, and 76%), the Hamming distance approach is preferred due to its simplicity and computational efficiency. Unlike

the energy-based method, which requires matrix-vector multiplications, both the Hamming distance and reference pattern matching methods rely only on simple comparisons. Among these, the Hamming distance method offers a slight accuracy advantage while maintaining low computational cost. These results underscore the potential of phase-coupled oscillators as energy-efficient alternatives to conventional computing architectures for low-power intelligent systems, particularly in edge computing applications where power constraints are paramount.

As illustrated in Figure 6, the diagonal entries of the confusion matrix represent correctly classified samples, while the off-diagonal values indicate misclassifications between classes. The visualization highlights that the model achieves high accuracy for several digits, such as 0 and 1, and reasonably strong performance for digit 6, which shows relatively low misclassification rates. These errors are largely concentrated between visually similar digits, such as 4, 7, and 9, which are often confused in handwritten form due to overlapping structural features.

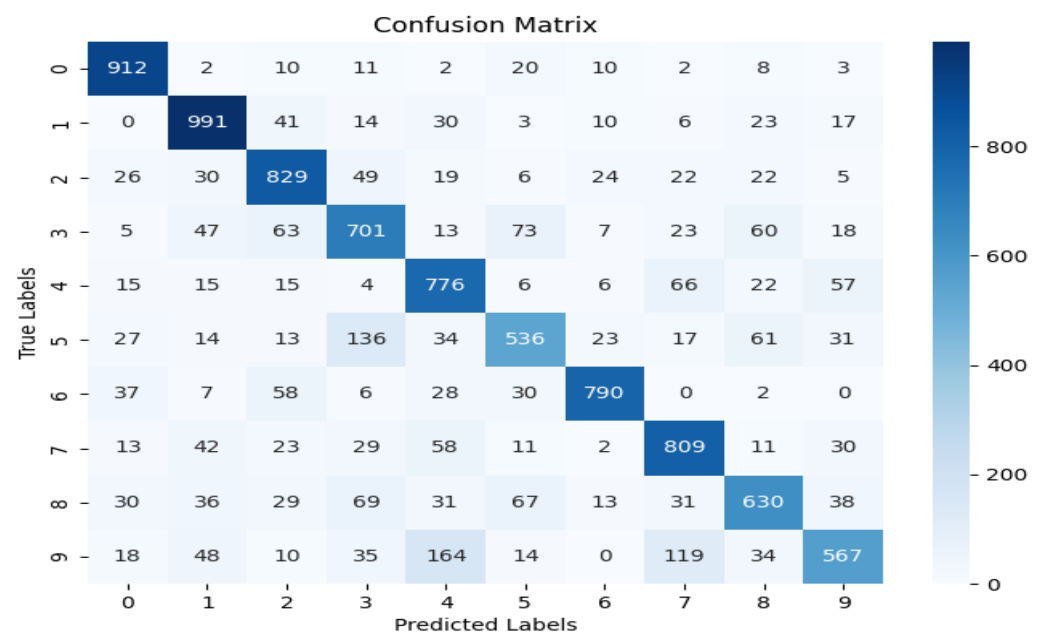


Figure 6. The strong diagonal dominance reflects a high agreement between predicted and true class labels, indicating reliable classification performance.

4. Discussion

This study shows that a simple oscillatory neural network (ONN) with 36 coupled oscillators, optimized by a genetic algorithm and trained with Hebbian learning, can achieve 75–76% accuracy in the full 10-class MNIST task. The outputs remained stable under moderate noise, which confirms that phase synchronization is a reliable approach for pattern recognition. We emphasize that both the training and the inference can eventually be done on real oscillators (i.e., actual analog circuits)—even if our present study is a computational study on a digital computer.

4.1. Scalability, Resolution Analysis, and Baseline Comparison

To evaluate the scalability of the proposed ONN architecture, we increased the input resolution from 6×6 to 14×14 . This change does not significantly affect the ONN's performance on the simple 0–1 binary task. Across all tested resolutions, the network maintains accuracy above 99%, indicating stable behavior under different oscillator counts.

To further assess the performance of the proposed ONN architecture, we compared the GA-ONN against a standard MLP under identical preprocessing conditions. The GA-ONN

achieves an accuracy of 99.28% with only 630 coupling weights, compared to 99.39% for the MLP with 1172 parameters. This marginal difference indicates that the GA-ONN remains competitive with gradient-trained baselines while relying solely on local Hebbian learning without any gradient computation.

4.2. Hardware Feasibility

The paper uses the Kuramoto model, which itself is not directly related to a physical oscillator model (such as to the circuit-level description of a ring oscillator or relaxation oscillator) but can be ‘translated’ to a real circuit as it was done in several cited papers. With the likely valid assumption that the Kuramoto oscillators can be implemented by ring oscillators, one can make a back-of-envelope calculation of the net power consumption and on the benefits of ONN hardware.

As a back-of-the-envelope estimate, following the arguments of [23] and references therein, a single ring oscillator consumes approximately 1 pJ per inference. For the oscillatory neural network considered in this work, consisting of 1620 oscillators, this corresponds to an estimated energy consumption of $E_{\text{inf}} \approx 1620 \times 10^{-12} \text{ J} = 1.62 \times 10^{-9} \text{ J}$, per inference, i.e., approximately 1.6 nJ per inference. It should be noted that this estimate is likely conservative. The assumption of a constant 1 pJ per oscillator is derived from a different, fully connected hardware architecture and does not account for the significantly simpler connectivity of the present network, where only small blocks of oscillators should be connected directly. Consequently, the actual energy consumption of a hardware implementation could be substantially lower than this simple scaling estimate.

For comparison, highly optimized lightweight conventional neural-network accelerators achieve energy efficiencies in the range of 1 μJ per inference for comparable tasks [23], while GPU-based implementations typically consume considerably more energy in exchange for higher model complexity and accuracy. Even under the pessimistic assumptions used here, the projected energy consumption of the oscillatory neural network is approximately three orders of magnitude lower than that of state-of-the-art lightweight digital neural-network solutions, suggesting a significant potential advantage in power efficiency.

4.3. Future Work

The main takeaway is that GA-optimized templates deliver strong classification accuracy while preserving the simple structure of an ONN classifier.

A natural next step is to replace the Kuramoto abstraction with a CMOS-level circuit model of the oscillators. While the Kuramoto framework provides a fast and analytically tractable approximation of phase dynamics, validating the GA-optimized templates on transistor-level or behavioral CMOS oscillator models would bridge the gap between algorithmic simulation and practical hardware. Such an investigation would clarify how inverter-level delay variations, supply fluctuations, and coupling nonidealities influence the stability and separability of the learned phase patterns. Another promising direction is to integrate the proposed ONN as a front-end processing layer within larger neuromorphic or hybrid pipelines. Because the ONN efficiently extracts low-dimensional phase features from binary or low-resolution inputs, it could serve as an energy-efficient preprocessing stage before a lightweight digital classifier or a secondary ONN module. Exploring multi-stage or hierarchical ONN architectures—where early layers perform coarse feature extraction and later layers refine the decision—may significantly improve multi-class performance while preserving low power consumption. Overall, extending the model toward CMOS-accurate oscillator simulations and embedding ONN modules within larger hybrid architectures represent two key steps toward practical, hardware-efficient oscillatory computing systems.

Author Contributions: Conceptualization, funding acquisition: G.C. Model and code development, data analysis: M.M. All authors contributed to the writing and discussed the results together. All authors have read and agreed to the published version of the manuscript.

Funding: The authors are grateful to the Stipendium Hungaricum program and the EU-funded PHASTRAC project (European Union’s Horizon Europe research and innovation programme under grant agreement No. 101092096) for financial support.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The code supporting the findings of this study is openly available at <https://github.com/Mitmoayed/ONN-Classifier-Code> (accessed on 25 June 2026).

Acknowledgments: The authors are grateful to the PHASTRAC team members, and especially to the project coordinator Aida Todri-Sanial, for motivating discussions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Winfree, A.T. Biological rhythms and the behavior of populations of coupled oscillators. *J. Theor. Biol.* **1967**, *16*, 15–42. [[CrossRef](#)] [[PubMed](#)]
2. Kuramoto, Y. Self-entrainment of a population of coupled non-linear oscillators. In *International Symposium on Mathematical Problems in Theoretical Physics*; Araki, H., Ed.; Lecture Notes in Physics; Springer: Berlin/Heidelberg, Germany, 1975; Volume 39. [[CrossRef](#)]
3. Csaba, G.; Porod, W. Coupled oscillators for computing: A review and perspective. *Appl. Phys. Rev.* **2020**, *7*, 011302. [[CrossRef](#)]
4. Moy, W.; Ahmed, I.; Chiu, P.-W.; Moy, J.; Sapatnekar, S.S.; Kim, C.H. A 1,968-node coupled ring oscillator circuit for combinatorial optimization problem solving. *Nat. Electron.* **2022**, *5*, 310–317. [[CrossRef](#)]
5. Maffezzoni, P.; Daniel, L.; Shukla, N.; Datta, S.; Raychowdhury, A. Modeling and simulation of vanadium dioxide relaxation oscillators. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2015**, *62*, 2207–2215. [[CrossRef](#)]
6. Zheng, Y.; Xu, H.; Xu, H.; Li, Z.; Li, L.; Zhong, B.; Liu, L.; Deng, F.; Zhu, K.; Xue, H.; et al. Twelve-inch electrically anisotropic boridene for optoelectronic computing. *Nat. Nanotechnol.* **2026**, *21*, 571–578. [[CrossRef](#)] [[PubMed](#)]
7. Zhong, B.; Qin, X.; Xu, H.; Deng, F.; Wang, H.; Li, L.; Li, Z.; Zhang, W.; Lou, Z.; Wang, L. Monolithic cell-on-memristor architecture enables wafer-scale integration of oscillatory chemoreceptors for bio-realistic gustatory chips. *Nat. Mater.* **2026**, *25*, 275–284. [[CrossRef](#)] [[PubMed](#)]
8. Maher, O.; Jiménez, M.; Delacour, C.; Harnack, N.; Núñez, J.; Avedillo, M.J.; Linares-Barranco, B.; Todri-Sanial, A.; Indiveri, G.; Karg, S. A CMOS-compatible oscillation-based VO₂ Ising machine solver. *Nat. Commun.* **2024**, *15*, 3334. [[CrossRef](#)] [[PubMed](#)]
9. Holland, J.H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, USA, 1975
10. Alam, T.; Qamar, S.; Dixit, A.; Benaïda, M. Genetic Algorithm: Reviews, Implementations, and Applications. *arXiv* **2020**, arXiv:2007.12673.
11. Jiménez, M.; Avedillo, M.J.; Linares-Barranco, B.; Núñez, J. Learning Algorithms for Oscillatory Neural Networks as Associative Memory for Pattern Recognition. *Front. Neurosci.* **2023**, *17*, 1257611. [[CrossRef](#)] [[PubMed](#)]
12. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [[CrossRef](#)] [[PubMed](#)]
13. Ramsauer, H.; Schäfl, B.; Lehner, J.; Seidl, P.; Widrich, M.; Adler, T.; Gruber, L.; Holzleitner, M.; Kreil, D.; Kopp, M.; et al. Hopfield networks is all you need. *arXiv* **2021**, arXiv:2008.02217. [[CrossRef](#)]
14. Kim, S.-J.; Shukla, N.; Raychowdhury, A. A scalable Ising machine based on coupled oscillator arrays. *Nat. Electron.* **2023**, *6*, 764–772. [[CrossRef](#)]
15. Hoppensteadt, F.C.; Izhikevich, E.M. Pattern recognition via synchronization in phase-locked loop neural networks. *IEEE Trans. Neural Netw.* **2000**, *11*, 734–738. [[CrossRef](#)] [[PubMed](#)]
16. Rajagopal Rohan, N.; Vigneswaran, C.; Ghosh, S.; Rajendran, K.; Gaurav, A.; Chakravarthy, V.S. Deep Oscillatory Neural Network. *Sci. Rep.* **2025**, *15*, 40968. [[CrossRef](#)] [[PubMed](#)]
17. Sharma, A.A.; Bain, J.A.; Weldon, J.A. Phase coupling and control of oxide-based oscillators for neuromorphic computing. *IEEE J. Explor.-Solid-State Comput. Devices Circuits* **2015**, *1*, 58–66. [[CrossRef](#)]
18. Wang, H.; Qi, M.; Wang, B. PPV modeling of memristor-based oscillators and application to ONN pattern recognition. *IEEE Trans. Circuits Syst. II Express Briefs* **2017**, *64*, 610–614. [[CrossRef](#)]

19. Belyaev, M.A.; Velichko, A.A. Classification of handwritten digits using the Hopfield network. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *862*, 052048. [[CrossRef](#)]
20. Abernot, M.; Todri-Sanial, A. Training energy-based single-layer Hopfield and oscillatory networks with unsupervised and supervised algorithms for image classification. *Neural Comput. Appl.* **2023**, *35*, 18505–18518.
21. Sabo, F.; Todri-Sanial, A. ClassONN: Classification with Oscillatory Neural Networks using the Kuramoto Model. In *Proceedings of the DATE 2024 Conference*; IEEE: New York, NY, USA, 2024; pp. 1–6.
22. Todri-Sanial, A.; Delacour, C.; Abernot, M.; Sabo, F. Computing with oscillators from theoretical underpinnings to applications and demonstrators. *npj Unconv. Comput.* **2024**, *1*, 14. [[CrossRef](#)] [[PubMed](#)]
23. Rudner, T.; Porod, W.; Csaba, G. Design of oscillatory neural networks by machine learning. *Front. Neurosci.* **2024**, *18*, 1307525. [[CrossRef](#)] [[PubMed](#)]
24. Cai, W.; Li, Z.; Wang, I.; Wang, Y.-N.; Lee, T.H. OscNet v1.5: Energy Efficient Hopfield Network on CMOS Oscillators for Image Classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, Honolulu, HI, USA, 19–23 October 2025; pp. 4733–4741.
25. Eiben, A.E.; Smith, J.E. *Introduction to Evolutionary Computing*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 2015.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.