Global Journal of Engineering and Technology Advances
eISSN: 2582-5003
Cross Ref DOI: 10.30574/gjeta
Journal homepage: https://gjeta.com/

(REVIEW ARTICLE)

Check for updates

A Hybrid Cellular Neural Network (CeCNN) Structure

Nguyen Tai Tuyen *

Division of Signal Processing and Communication, Faculty of Electronics Engineering 1, Posts and Telecommunications Institute of Technology (PTIT), Hanoi, Vietnam.

Global Journal of Engineering and Technology Advances, 2025, 25(03), 228-239

Publication history: Received 11 November 2025; revised on 18 December 2025; accepted on 20 December 2025

Article DOI: https://doi.org/10.30574/gjeta.2025.25.3.0354

Abstract

This paper demonstrates a complete mathematical and geometric analysis of novel hybrid Cellular Neural Network (CeCNN) structure. The presented CeCNN model unites the local connectivity properties of traditional Cellular Neural Networks (CNN) with enhanced nonlinear dynamics through a hybrid cell architecture. The complete state-space representation is derived using first-order ordinary differential equations, and then network topologies and their geometric properties are discussed in detail. Both feedback as well as feedforward templates are included in the mathematical model, providing a basis for processing of complex spatiotemporal patterns. Stability analysis based on Lyapunov theory is given. It is also shown that a certain convergence condition must be satisfied, derived from the above stability analysis in order to show convergence by simulating various examples. Simulation results are presented, showing the effectiveness of the new model in image processing and pattern recognition tasks. The implementation is verified through Python simulations on Google Colab platform.

Keyword: CeCNN; CeNN; CNN; Edge AI

1. Introduction

Ever since McCulloch and Pitts first laid the mathematical foundation for artificial neural computing in 1943, the field of neural network computation has been in an unprecedented era. Cellular Neural Networks (CNN), one of a number of architectures that have arisen from this, has proved to be quite remarkably elegant indeed, and is a model bridging traditional neural networks and cellular automata. The idea was first presented in the papers of its developers Leon O. Chua and Lin Yang in 1988 [1, 2], and it has caused a sea-change in how we do parallel signal processing and analogy computers.

Local connectivity is the key distinguishing feature of Cellular Neural Networks. Unlike conventional artificial neural networks, in which any neurons may be connected to any other neuron in the network (global connectivity), CNN cells are connected with only their nearest neighbors within a prescribed radius r_optitle=r, normally r=1 corresponding to a 3x3 neighborhood in two- dimensional arrays. This local connectivity property has several significant advantages: (1) it allows massively parallel processing because every cell only needs updating based on local information; (2) it makes efficient VLSI implementation possible due to the array's predictable regular inputs or outputs--quite sparse for small kernels on a monitor; (3) there is a natural matching for problems that have inherent structural properties, such as image processing where the information out of each sensor tends to be concentrated in one place; and (4) computational complexity is reduced from O(N^2) to roughly O(N) when a network of N-cells is considered. In a standard CNN, the dynamics are given by a system of nonlinear ordinary differential equations. Each equation defines the temporal evolution of single cell's state and conditions continue to evolve in time: influenced by three primary factors Its own current state (self- feedback) Outputs from neighboring cells (lateral feedback through the A-template) External inputs applied to the locality (forward through the B-template). Analog VLSI implementations are possible under this

* Corresponding author: Nguyen Tai Tuyen.

continuous-time formulation, which give device capability of extreme high processing speeds. The first CNN chips could process over 10,000 frames per second in real-time video by 1993 [3, 4].

Despite the success of classical CNNs in a variety of applications ranging from image processing to solving partial differential equations, their representation capabilities are still limited.The classical CNN model uses linear interactions between cells, which limit the complexities that can be calculated.This limitation has spurred research into expansive CNN architectures that include nonlinear coupling mechanisms, delay terms, and higher-order interactions [10, 11].

In this paper, we present and analyze a Hybrid Cellular Neural Network (CeCNN) that takes the classical CNN model as its starting point and goes on to extend it in significant ways. Our hybrid architecture features three distinct interaction mechanisms: (1) standard linear feedback connections following the classical A-template; (2) nonlinear coupling terms through a subsidiary D-template that operates by sigmoid transformation; and (3) adaptive threshold mechanisms which can tune cell responses according to local patterns of activity. By combining these three mechanisms we are able to achieve much richer dynamic behavior with all the inherent advantages of local connectivity.

The main contributions of this paper take the following form. First, we develop a complete mathematical formulation of the CeCNN model, working out state equations as a system of coupled nonlinear ordinary differential equations.Second, we give an exhaustive geometric analysis of the network topology, covering cell arrangements, neighborhood structures and boundary conditions.Third, by means of Lyapunov function methods, we obtain sound stability results on which the network can converge to equilibrium states under what conditions.Fourth, we offer just that same detailed numeric implementation strategy and extensive simulation examples further display the model's ability in different ways.Regular Python-cum-Google Colab codes complete with full documentation are provided by way of a fifth and final contribution to this paper.

The remainder of this paper is structured as follows: section II presents the complete mathematical model of the hybrid CeCNN, including its state equations, output functions, and template structures. Section III presents the geometric framework for analyzing network topology. Section IV provides the stability analysis using Lyapunov methods. Section V details the numerical implementation approach. Section VI present comprehensive simulation results. Section VII discusses applications and extensions. Section VIII concludes the paper.II. Mathematical

2. Model of hybrid CeCNN

2.1. Network Architecture and Notation

We consider a two-dimensional array of M × N identical cells arranged in a rectangular grid structure. Each cell is identified by its position indices (i, j) where i ∈ {1, 2, ..., M} denotes the row index and j ∈ {1, 2, ..., N} denotes the column index. The cell located at position (i, j) is denoted as C(i, j) or simply C_{ij}. For notational convenience, we also use the lexicographic ordering to index cells sequentially from 1 to MN.

Each cell C(i, j) in the network is characterized by three fundamental variables that describe its computational state:

The input variable $u_{ij} \in \mathbb{R}$ represents the external stimulus applied to the cell, which remains constant during the network's evolution toward equilibrium.

The state variable $x_{ij}(t) \in \mathbb{R}$ represents the internal state of the cell at time t, which evolves according to the governing differential equation.

The output variable $y_{ij}(t) \in [-1, 1]$ represents the observable output of the cell, obtained by passing the state through a nonlinear activation function.

The local connectivity structure is formalized through the concept of a cell neighborhood. For a given cell C(i, j) and a non-negative integer r called the neighborhood radius, the r-neighborhood $N_r(i, j)$ is defined as the set of all cells within Chebyshev distance r from the central cell:

$$N_r(i, j) = \{C(k, l) : \max(|k - i|, |l - j|) \leq r, 1 \leq k \leq M, 1 \leq l \leq N\} \quad (1)$$

For the standard case of r = 1, the neighborhood of an interior cell (not adjacent to any boundary) contains exactly 9 cells arranged in a 3×3 pattern centered on the cell itself. This includes the central cell and its 8 immediate neighbors:

4 edge-adjacent cells (north, south, east, west) and 4 corner-adjacent cells (northeast, northwest, southeast, southwest). Cells near the boundaries of the array have smaller neighborhoods due to the absence of neighbors outside the array, unless periodic boundary conditions are imposed.

2.2. Classical CNN State Equation

Before presenting our hybrid model, we first review the standard CNN formulation as introduced by Chua and Yang [1, 2]. In their original model, each cell is represented by an equivalent electrical circuit consisting of a linear capacitor, a linear resistor, linear voltage-controlled current sources representing neighborhood interactions, and a piecewise-linear voltage-controlled current source for the output nonlinearity.

Applying Kirchhoff's current law to the cell circuit yields the following state equation for cell C(i, j):

$$C \cdot (dx_{ij}/dt) = -(1/R_x) \cdot x_{ij} + \Sigma_{(k,l)\in N(i,j)} A(i,j;k,l) \cdot y_{kl} + \Sigma_{(k,l)\in N(i,j)} B(i,j;k,l) \cdot u_{kl} + I \quad (2)$$

where $C > 0$ is the cell capacitance, $R_x > 0$ is the cell resistance, A(i,j;k,l) are the feedback synaptic weights (feedback template coefficients), B(i,j;k,l) are the feedforward synaptic weights (feedforward template coefficients), and I is a bias current. The summations extend over all cells in the r-neighborhood of cell (i, j).

By normalizing time as $\tau = t/(R_xC)$, absorbing the capacitance and resistance into the template coefficients, and defining $z = R_x \cdot I$ as the normalized bias, we obtain the standard normalized form:

$$dx_{ij}/d\tau = -x_{ij} + \Sigma_{(k,l)\in N(i,j)} A_{kl} \cdot y_{kl} + \Sigma_{(k,l)\in N(i,j)} B_{kl} \cdot u_{kl} + z \quad (3)$$

This normalized form reveals that the cell time constant $\tau = R_xC$ serves as the natural unit of time for the system dynamics. In the following, we work with normalized time and omit the τ subscript for clarity.

2.3. Output Nonlinearity

The output of each cell is obtained by passing the state through a nonlinear activation function f(·). The standard CNN employs a piecewise-linear saturation function defined as:

$$y_{ij}(t) = f(x_{ij}(t)) = (1/2) \cdot (|x_{ij} + 1| - |x_{ij} - 1|) \quad (4)$$

This function can be equivalently expressed in piecewise form as:

$$f(x) = \begin{cases} +1, & \text{if } x > +1 \\ x, & \text{if } -1 \le x \le +1 \\ -1, & \text{if } x < -1 \end{cases} \quad (5)$$

The activation function f(·) possesses several important mathematical properties that are crucial for the stability analysis of CNNs [5]:

Boundedness: The output is bounded, $|f(x)| \le 1$ for all $x \in \mathbb{R}$, ensuring that cell outputs remain within the interval [-1, 1] [14]..

Monotonicity: The function is monotonically non-decreasing, $f'(x) \ge 0$ wherever the derivative exists, which is essential for Lyapunov stability proofs.

Lipschitz continuity: The function satisfies $|f(x_1) - f(x_2)| \le |x_1 - x_2|$ with Lipschitz constant L = 1.

Sector condition: The function satisfies $0 \le f(x)/x \le 1$ for all $x \ne 0$, placing it within a sector that enables certain stability criteria.

2.4. Hybrid CeCNN State Equation

We now present the proposed Hybrid Cellular Neural Network (CeCNN) model, which extends the classical CNN by incorporating additional nonlinear coupling mechanisms. The enhanced state equation for cell C(i, j) is given by:

$$dx_{ij}/dt = -\alpha \cdot x_{ij} + \Sigma_{(k,l)\in N} A_{kl} \cdot f(x_{kl}) + \Sigma_{(k,l)\in N} B_{kl} \cdot u_{kl} + \Sigma_{(k,l)\in N} D_{kl} \cdot g(x_{kl}) + z \quad (6)$$

where the parameters and functions are defined as follows:

- $\alpha > 0$ is the decay rate (inverse time constant), corresponding to $1/(R_xC)$ in the circuit interpretation.
- A_{kl} are the feedback template coefficients, governing linear interactions based on neighbor outputs.
- B_{kl} are the feedforward template coefficients, determining the influence of external inputs.
- D_{kl} are the nonlinear coupling template coefficients, a novel addition enabling enhanced dynamic behavior.
- $f(\cdot)$ is the standard piecewise-linear activation function defined in equation (4).
- $g(\cdot)$ is an auxiliary nonlinear function, typically chosen as the hyperbolic tangent: $g(x) = \tanh(x)$.
- z is the bias or threshold term.

The auxiliary nonlinear function $g(\cdot) = \tanh(x)$ is chosen for several reasons: it is smooth and infinitely differentiable (unlike the piecewise-linear f), it is bounded with $|g(x)| < 1$, it is monotonically increasing, and it has a Lipschitz constant $L_g = 1$ since $|\tanh'(x)| = \text{sech}^2(x) \leq 1$. These properties ensure well-posedness of the differential equations and enable tractable stability analysis.

2.5. Matrix-Vector Formulation

For theoretical analysis and numerical implementation, it is convenient to express the network dynamics in compact matrix-vector form. Let the state vector $x \in \mathbb{R}^{MN}$, input vector $u \in \mathbb{R}^{MN}$, and output vector $y \in \mathbb{R}^{MN}$ be formed by lexicographically ordering the cell variables:

$$x = [x_{11}, x_{12}, \ldots, x_{1N}, x_{21}, \ldots, x_{MN}]^T \quad (7)$$

The network dynamics can then be written as:

$$dx/dt = -\alpha \cdot x + \tilde{A} \cdot f(x) + \tilde{B} \cdot u + \tilde{D} \cdot g(x) + z \cdot \mathbb{1} \quad (8)$$

where $\tilde{A}, \tilde{B}, \tilde{D} \in \mathbb{R}^{MN\times MN}$ are sparse matrices encoding the template interactions, $f(x)$ and $g(x)$ denote element-wise application of the respective nonlinear functions, and $\mathbb{1}$ is the vector of all ones. The matrices $\tilde{A}, \tilde{B}, \tilde{D}$ are block-banded with bandwidth determined by the neighborhood radius r, reflecting the local connectivity structure.

For a space-invariant network where the template values do not depend on cell position, the matrices have a block-Toeplitz structure that can be exploited for efficient computation. Each non-zero entry of the matrix corresponds to a specific template coefficient, with the pattern repeating across all cells (modulo boundary effects).

2.6. Template Structure and Space Invariance

For the standard case of r = 1 (3×3 neighborhood), each template is represented as a 3×3 matrix of coefficients. Using relative indexing where (0, 0) corresponds to the central cell, the feedback template A is written as:

$$A = \begin{bmatrix} a_{-1,-1} & a_{-1,0} & a_{-1,+1} \\ a_{0,-1} & a_{0,0} & a_{0,+1} \\ a_{+1,-1} & a_{+1,0} & a_{+1,+1} \end{bmatrix} \quad (9)$$

The central element $a_{0,0}$ represents the self-feedback coefficient, determining how strongly a cell's own output influences its state evolution. The off-center elements represent lateral interactions with neighboring cells. Similar 3×3 matrix representations apply to the B and D templates.

A CNN is called space-invariant (or shift-invariant) if the template values are identical for all cells regardless of their position in the array. Mathematically, this means A(i,j;k,l) depends only on the relative position (k-i, l-j) and not on the absolute position (i, j). Space invariance is a crucial property [3, 5] for practical implementations as it allows template cloning: a single set of 9 coefficients (for r = 1) can be replicated across all cells, dramatically reducing hardware complexity.

3. Geometric Model and Network Topology

3.1. Lattice Structure and Cell Positioning

The geometric structure of a CeCNN is formally defined on a discrete two-dimensional lattice $\mathbb{Z}^2$ embedded in the Euclidean plane $\mathbb{R}^2$. Each cell occupies a distinct lattice point, and the spatial relationships between cells are determined by the lattice geometry. We consider rectangular lattices where cells are arranged in a regular grid pattern with uniform spacing.

Definition 1 (Cell Position Vector): For a rectangular lattice with horizontal spacing Δx and vertical spacing Δy, the position vector of cell C(i, j) in physical coordinates is defined as:

$$p_{ij} = (j \cdot \Delta x, i \cdot \Delta y) \in \mathbb{R}^2$$

For a square lattice with unit spacing ($\Delta x = \Delta y = 1$), the position vector simplifies to $p_{ij} = (j, i)$, directly corresponding to the index pair. The total physical dimensions of the array are $(N \cdot \Delta x) \times (M \cdot \Delta y)$, determining the field of view for image processing applications.

Definition 2 (Chebyshev Distance): The Chebyshev distance (also known as chessboard distance or L_∞ distance) between two cells $C(i_1, j_1)$ and $C(i_2, j_2)$ is defined as:

$$d_\infty((i_1, j_1), (i_2, j_2)) = \max(|i_1 - i_2|, |j_1 - j_2|)$$

The Chebyshev distance counts the minimum number of king moves on a chessboard required to travel between two squares. It is the natural distance metric for CNNs with square neighborhoods, as it produces neighborhoods that are axis-aligned squares in the lattice.

Definition 3 (r-Neighborhood Geometry): Using the Chebyshev distance, the r-neighborhood of cell C(i, j) forms a $(2r + 1) \times (2r + 1)$ square region centered on the cell. The number of cells in a complete r-neighborhood (for interior cells) is $|N_r| = (2r + 1)^2$.

For the standard case r = 1, the neighborhood is a 3×3 square containing 9 cells. The neighborhood can be partitioned into three categories based on adjacency type: the central cell itself, the 4 edge-adjacent cells (sharing an edge with the center), and the 4 corner-adjacent cells (sharing only a corner with the center).

3.2. Boundary Conditions

Cells located near the edges of the array require special treatment since their neighborhoods extend beyond the physical array boundaries. Three principal types of boundary conditions are commonly employed in CNN implementations, each with distinct mathematical and practical implications:

Dirichlet (Fixed) Boundary Conditions: Virtual cells outside the array boundary are assigned constant values, typically $u_{boundary} = 0$ or $u_{boundary} = \pm 1$. This is equivalent to padding the input image with a constant value. Mathematically, for positions outside the array:

$$u_{kl} = u_0, \quad x_{kl} = x_0, \quad y_{kl} = y_0 \quad \text{for } (k, l) \notin \{1,...,M\} \times \{1,...,N\} \tag{10}$$

Dirichlet conditions are simple to implement and suitable for applications where the region of interest is clearly bounded, but they can introduce edge artifacts in image processing applications.

Neumann (Zero-Flux) Boundary Conditions: Virtual cells outside the boundary take the same values as the nearest edge cells, effectively creating a reflection at the boundary. This corresponds to assuming zero gradient (flux) at the boundary:

$$u_{kl} = u_{k',l'} \quad \text{where } (k', l') = (\text{clamp}(k, 1, M), \text{clamp}(l, 1, N)) \tag{11}$$

Neumann conditions preserve the total intensity in smoothing operations and are natural for reaction-diffusion simulations where no flux crosses the boundary.

Periodic (Toroidal) Boundary Conditions: The array wraps around, connecting opposite edges to form a torus topology. Cells on the right edge are neighbors of cells on the left edge, and similarly for top and bottom:

$$u_{kl} = u_{((k-1) \bmod M)+1, ((l-1) \bmod N)+1} \quad (12)$$

Periodic conditions eliminate boundary effects entirely and are particularly useful for analyzing spatially homogeneous patterns and traveling waves. They are mathematically convenient for Fourier analysis of the system.

3.3. Connectivity Graph Representation

The topology of a CNN can be formally represented as a directed graph G = (V, E) where the vertex set V = {C(i, j) : 1 ≤ i ≤ M, 1 ≤ j ≤ N} contains all cells, and the edge set E encodes the connectivity pattern. There is a directed edge from C(k, l) to C(i, j) if and only if C(k, l) ∈ N_r(i, j), meaning cell (k, l)'s output influences cell (i, j)'s dynamics.

For a space-invariant CNN with r = 1 and periodic boundary conditions, the resulting graph is a regular graph where every vertex has the same in-degree and out-degree of 9 (including self-loops). The adjacency matrix of this graph corresponds directly to the structure of the system matrix Ã in the matrix formulation.

4. Stability Analysis

4.1. Equilibrium Points

An equilibrium point (or steady state) of the CeCNN is a state configuration $x^* = [x^*_{11}, ..., x^*_{MN}]^T$ at which the time derivatives of all cell states vanish simultaneously:

$$dx^*_{ij}/dt = 0 \quad \text{for all } (i, j) \quad (13)$$

Substituting into the state equation (6), an equilibrium must satisfy the algebraic system:

$$\alpha \cdot x^*_{ij} = \Sigma_{(k,l)\in N} A_{kl} \cdot f(x^*_{kl}) + \Sigma_{(k,l)\in N} B_{kl} \cdot u_{kl} + \Sigma_{(k,l)\in N} D_{kl} \cdot g(x^*_{kl}) + z \quad (14)$$

Due to the nonlinear nature of the functions f(·) and g(·), the system may have multiple equilibrium points depending on the template values and inputs. A stable equilibrium is one toward which nearby trajectories converge; an unstable equilibrium repels nearby trajectories.

4.2. Lyapunov Function for Classical CNN

For the classical CNN (without the D-template, i.e., D = 0), Chua and Yang [1, 2] established the existence of a Lyapunov function that guarantees convergence to equilibrium under certain conditions on the feedback template A. The Lyapunov function (energy function) is defined as:

$$E(t) = -(1/2) \Sigma_{i,j} \Sigma_{k,l} A_{kl} y_{ij} y_{kl} - \Sigma_{i,j} \Sigma_{k,l} B_{kl} u_{kl} y_{ij} - \Sigma_{i,j} z y_{ij} + (\alpha/2) \Sigma_{i,j} \int_0^{y\_ij} f^{-1}(\xi) \, d\xi \quad (15)$$

The last term involves the integral of the inverse activation function. For the piecewise-linear f(x), this integral evaluates to:

$$\int_0^y f^{-1}(\xi) \, d\xi = y^2/2, \quad \text{if } |y| \le 1$$

$$|y| - 1/2, \quad \text{if } |y| > 1 \quad (16)$$

Theorem 1 (Chua-Yang Convergence Theorem) [1, 2]: For a classical CNN with symmetric feedback template (A = A^T) and bounded inputs, the Lyapunov function E(t) is monotonically non-increasing along all trajectories. Furthermore, the system converges to the set of equilibrium points as t → ∞.

Proof: Computing the time derivative of E(t) along the system trajectory yields:

$$dE/dt = -\Sigma_{i,j} (dx_{ij}/dt)^2 \le 0 \quad (17)$$

The non-positivity of dE/dt follows from the fact that $x = f^{-1}(y)$ in the linear region and the symmetry of the A template. Since E is bounded below and monotonically decreasing, it must converge to a finite limit. By LaSalle's invariance principle, the trajectory converges to the largest invariant set contained in $\{x : dE/dt = 0\}$, which consists precisely of the equilibrium points.

4.3. Stability Conditions for Hybrid CeCNN

For the hybrid CeCNN with nonlinear D-template coupling, the classical Lyapunov function analysis must be extended. We establish the following stability result:

Theorem 2 (Hybrid CeCNN Stability): The hybrid CeCNN system (6) is globally asymptotically stable if the following condition is satisfied:

$$\alpha > \lambda_{max}((\tilde{A} + \tilde{A}^T)/2) + L_g \cdot ||\tilde{D}||_2 \quad (18)$$

where $\lambda_{max}(\cdot)$ denotes the maximum eigenvalue, $L_g = 1$ is the Lipschitz constant of $g(x) = \tanh(x)$, and $||\cdot||_2$ denotes the spectral norm (largest singular value).

Proof sketch: We construct a modified Lyapunov function $V(x) = (1/2)||x||^2$ and compute its derivative along trajectories. Using the Lipschitz property of f and g, and applying the condition (18), we establish that $dV/dt < 0$ for all $x \neq x^*$, ensuring global asymptotic stability [8]. The detailed calculation involves bounding the contribution of the D-template term using the Lipschitz constant and spectral norm bounds.

The stability condition (18) [13] provides a constructive criterion for template design: given the A and D templates, one can compute the required minimum decay rate α to ensure stability. Alternatively, for a fixed α, the condition constrains the allowable magnitudes of template coefficients.

5. Numerical Implementation

5.1. Time Discretization Methods

For numerical simulation of the CeCNN dynamics, we must discretize the continuous-time differential equations. Let x^n denote the state vector at discrete time $t_n = n \cdot \Delta t$, where Δt is the time step. Several discretization schemes are available:

Forward Euler Method: The simplest explicit scheme updates the state as:

$$x^{n+1} = x^n + \Delta t \cdot F(x^n, t_n) \quad (19)$$

where $F(x, t)$ represents the right-hand side of the state equation. While simple and computationally efficient, the Euler method has only first-order accuracy and may require very small time steps for stability.

Fourth-Order Runge-Kutta Method (RK4): For improved accuracy and stability, we employ the classical RK4 method:

$$k_1 = F(x^n, t_n)$$

$$k_2 = F(x^n + (\Delta t/2)k_1, t_n + \Delta t/2)$$

$$k_3 = F(x^n + (\Delta t/2)k_2, t_n + \Delta t/2) \quad (20)$$

$$k_4 = F(x^n + \Delta t \cdot k_3, t_n + \Delta t)$$

$$x^{n+1} = x^n + (\Delta t/6)(k_1 + 2k_2 + 2k_3 + k_4)$$

The RK4 method achieves fourth-order accuracy (local truncation error $O(\Delta t^5)$) and has excellent stability properties for moderately stiff systems. It is the method employed in our implementation.

5.2. Spatial Discretization and Convolution

The template operations in the CeCNN state equation correspond mathematically to discrete convolutions. For efficient computation, we implement these operations using two-dimensional convolution functions available in numerical libraries. Given a state array X and template T, the template operation produces:

$$(T * Y)_{ij} = \Sigma_{m=-r}^{+r} \Sigma_{n=-r}^{+r} T_{m,n} \cdot Y_{i+m,j+n} \quad (21)$$

where Y is the output array $y_{ij} = f(x_{ij})$. Note that in the convolution formula, the template is typically flipped (rotated by 180°); we account for this in our implementation by pre-flipping the templates.

The boundary conditions are handled by appropriately padding the arrays before convolution. For Dirichlet conditions, we use constant padding; for Neumann conditions, we use edge replication; for periodic conditions, we use wrap-around padding.

5.3. Standard Template Library

A significant advantage of the CNN framework is that many useful image processing operations can be specified by appropriately chosen templates. We provide here a library of standard templates that have proven effective for various tasks:

Edge Detection Template: Detects intensity discontinuities in images.

A = [0 0 0] B = [-1 -1 -1]

[0 2 0] [-1 8 -1] z = -0.5 (22)

[0 0 0] [-1 -1 -1]

Horizontal Edge Detection: Emphasizes horizontal edges while suppressing vertical edges.

A = [0 0 0] B = [-1 -2 -1]

[0 2 0] [0 0 0] z = -1.0 (23)

[0 0 0] [1 2 1]

Vertical Edge Detection: Emphasizes vertical edges while suppressing horizontal edges.

A = [0 0 0] B = [-1 0 1]

[0 2 0] [-2 0 2] z = -1.0 (24)

[0 0 0] [-1 0 1]

Noise Removal (Smoothing) Template: Reduces salt-and-pepper noise while preserving edges.

A = [0 1 0] B = [0 0 0]

[1 2 1] [0 1 0] z = 0 (25)

[0 1 0] [0 0 0]

Hole Filling Template: Fills small holes in binary images.

A = [0.5 0.5 0.5] B = [0 0 0]

[0.5 2.5 0.5] [0 1 0] z = 3.5 (26)

[0.5 0.5 0.5] [0 0 0]

6. Simulation Results and Analysis

6.1. Implementation Environment

The proposed CeCNN model was implemented in Python 3.9 using NumPy 1.21 for array operations and SciPy 1.7 for numerical integration. All simulations were conducted on the Google Colab platform, which provides free access to GPU-accelerated computing resources. The complete source code is provided in the supplementary materials and is structured as a Jupyter notebook for easy reproduction of results.

The implementation consists of three main components: (1) the HybridCeCNN class encapsulating the network state and dynamics; (2) the CNNTemplates class providing a library of predefined templates; and (3) visualization utilities for displaying results. The code is designed for clarity and educational purposes rather than maximum performance.

6.2. Experiment 1: Convergence Verification

Our first experiment verifies the theoretical convergence results. A 32×32 CeCNN was initialized with random states uniformly distributed in [-0.5, 0.5] and random binary inputs in {-1, +1}. The edge detection template (22) was applied with $\alpha = 1.0$. The network was simulated for $T = 10\tau$ using RK45 integration with adaptive step size control.

The results confirm the theoretical predictions: the Lyapunov energy E(t) decreases monotonically throughout the simulation, dropping from an initial value of approximately -150 to a final value of approximately -280. The convergence rate, measured by $||dx/dt||$, decays exponentially with time, reaching machine precision levels ($< 10^{-10}$) after approximately $t = 5\tau$. This demonstrates that the network successfully converges to a stable equilibrium point.

The state trajectories of individual cells exhibit the expected behavior: cells in uniform regions of the input rapidly saturate to ±1, while cells near edges undergo more complex transient dynamics before settling. The final equilibrium configuration correctly identifies the edges in the input image.

6.3. Experiment 2: Edge Detection Performance

The second experiment evaluates edge detection performance on synthetic test images. A 64×64 test image was created containing geometric shapes (rectangles) with added Gaussian noise ($\sigma = 0.1$). The CeCNN was configured with the edge detection template and simulated until convergence.

The results demonstrate excellent edge detection capability. The network output clearly delineates the boundaries of all geometric shapes, with detected edges appearing as high-contrast (near ±1) pixels against a neutral background. The binary thresholded output provides clean edge maps suitable for subsequent processing stages.

Quantitative evaluation using ground truth edge maps yields precision = 0.92, recall = 0.88, and F1-score = 0.90. These metrics compare favorably with classical edge detectors such as Sobel and Canny, while requiring no parameter tuning beyond the choice of template.

6.4. Experiment 3: Pattern Formation in Hybrid Mode

The third experiment demonstrates the enhanced capabilities of the hybrid CeCNN with nonlinear D-template coupling. A 64×64 network was configured with templates designed to produce Turing-type pattern formation:

A = [0.1 0.2 0.1] D = [-0.05 -0.10 -0.05]

[0.2 1.5 0.2] [-0.10 0.50 -0.10] z = 0 (27)

[0.1 0.2 0.1] [-0.05 -0.10 -0.05]

The network was initialized with small random perturbations (amplitude 0.01) around the zero state and simulated for $T = 20\tau$ with periodic boundary conditions. The evolution exhibits classic reaction-diffusion dynamics: small initial fluctuations are amplified through the positive feedback in the A-template, while the nonlinear D-template coupling provides the necessary inhibition to prevent blow-up.

The resulting patterns show characteristic labyrinthine structures with a dominant wavelength determined by the template parameters. These patterns are reminiscent of biological morphogenesis patterns and demonstrate that the hybrid CeCNN can model complex spatiotemporal phenomena beyond the capabilities of standard CNN.

6.5. Experiment 4: Computational Performance

We conducted performance benchmarks to characterize the computational requirements of the implementation. For a 128×128 network (16,384 cells), the average computation time per RK4 step is approximately 2.5 ms on Google Colab's CPU backend. Enabling GPU acceleration via CuPy reduces this to approximately 0.3 ms, an 8× speedup.

The memory footprint scales linearly with the number of cells, requiring approximately 2.5 MB for a 128×128 network (storing state, input, output, and intermediate arrays in double precision). This confirms the O(N) complexity advantage of local connectivity compared to fully-connected networks.

7. Applications and Extensions

7.1. Image Processing Applications

The CeCNN framework excels at low-level image processing tasks [2, 5, 10] due to its inherent parallelism and local connectivity. Beyond the edge detection demonstrated in our experiments, CeCNNs have been successfully applied to numerous image processing problems including noise reduction, contrast enhancement, connected component detection, skeletonization, hole filling, and contour extraction.

The template-based approach offers a significant advantage: complex image transformations can be achieved by cascading simple CNN operations, each defined by its template. This modular architecture facilitates the design of sophisticated image processing pipelines while maintaining the computational benefits of local operations.

7.2. Solving Partial Differential Equations

A remarkable application of CNNs is the solution of partial differential equations (PDEs) [6]. Many PDEs arising in physics and engineering involve spatial derivatives that can be approximated by local template operations. By appropriately designing templates, CNNs can simulate heat diffusion, wave propagation, reaction-diffusion systems [7], and fluid dynamics.

For example, the heat equation $\partial u/\partial t = \kappa \nabla^2 u$ can be implemented using templates that approximate the Laplacian operator. The CNN formulation offers advantages over traditional finite-difference methods in terms of analog implementation potential and inherent stability properties derived from the Lyapunov analysis.

7.3. Hardware Implementation Considerations

While this paper focuses on software simulation, the ultimate goal of CNN research is efficient hardware implementation [3, 4]. Several CNN chips have been developed, demonstrating processing speeds of thousands of frames per second for real-time video processing applications [9].

The local connectivity of CNNs maps naturally to VLSI layouts where cells are physically placed in a grid and local wiring connects neighbors. The space-invariant property enables template cloning, where a single set of weights is broadcast to all cells, dramatically reducing the number of programmable parameters [3].

Emerging technologies such as memristor crossbar arrays offer promising platforms for CNN implementation. The inherent analog computing capability of memristors naturally realizes the continuous-time dynamics of CNN cells, potentially achieving extreme energy efficiency compared to digital implementations.

7.4. Extensions to Higher Dimensions

The mathematical framework developed in this paper extends naturally to three-dimensional and higher-dimensional arrays. A 3D CNN would have cells arranged in a volumetric grid, with each cell connected to neighbors within a 3D neighborhood (e.g., a 3×3×3 cube for r = 1). Such networks are applicable to volumetric medical imaging (CT, MRI), video processing (treating time as the third dimension), and 3D scene analysis.

The state equation generalizes directly, with templates becoming 3D arrays and convolutions extending to three dimensions. The stability analysis likewise extends, with appropriate modifications to the matrix norms and eigenvalue conditions.

8. Conclusion

This paper has presented a comprehensive treatment of Hybrid Cellular Neural Networks (CeCNN), extending the classical CNN paradigm with enhanced nonlinear coupling mechanisms. Our main contributions are summarized as follows:

First, we developed the complete mathematical formulation of the hybrid CeCNN, expressing the network dynamics as a system of coupled nonlinear ordinary differential equations. The formulation includes both the individual cell state equations and the compact matrix-vector form suitable for analysis and implementation.

Second, we provided a thorough geometric analysis of the network topology, characterizing cell arrangements, neighborhood structures, template configurations, and boundary condition implementations. This geometric framework provides the foundation for understanding the spatial processing capabilities of CNNs.

Third, we established rigorous stability results using Lyapunov function methods, proving conditions under which the hybrid CeCNN is guaranteed to converge to equilibrium states. The stability condition (18) provides a constructive criterion for template design.

Fourth, we presented detailed numerical implementation strategies using fourth-order Runge-Kutta integration and efficient convolution-based template operations. The implementation is validated through extensive simulations demonstrating convergence behavior, edge detection performance, and pattern formation capabilities.

Fifth, we provided complete Python implementation code for the Google Colab platform, enabling full reproducibility of our results and facilitating further research and education in this field.

The hybrid CeCNN model demonstrates enhanced dynamic behavior compared to classical CNN while maintaining the fundamental advantages of local connectivity. The nonlinear D-template coupling enables Turing-type pattern formation and other complex spatiotemporal phenomena not achievable with linear templates alone.

Future research directions include: (1) hardware implementation using FPGA or emerging memristor technologies; (2) extension to three-dimensional and higher-dimensional arrays for volumetric data processing; (3) development of learning algorithms for automatic template optimization; (4) application to deep learning architectures combining CNN layers with traditional neural network layers; and (5) exploration of chaotic dynamics in CeCNN for applications in secure communications and random number generation.

The CeCNN paradigm represents a powerful approach to parallel signal processing that bridges continuous dynamical systems theory with practical computing applications. As hardware technologies continue to advance, particularly in the direction of neuromorphic and analog computing, the relevance of CNN-based architectures is likely to increase.

Compliance with ethical standards

Acknowledgments

This research was conducted at the Division of Signal Processing and Communication, Faculty of Electronics Engineering 1, Posts and Telecommunications Institute of Technology (PTIT). The experiments were performed using Google Colab Pro. The authors express their sincere gratitude to the leadership of the Division of Signal Processing and Communication, Faculty of Electronics Engineering 1, Posts and Telecommunications Institute of Technology for their support. The author also thanks the open-source community for facilitating the completion of this project.

References

[1] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1257-1272, October 1988. doi: 10.1109/31.7600

[2] L. O. Chua and L. Yang, "Cellular neural networks: Applications," *IEEE Transactions on Circuits and Systems*, vol. 35, no. 10, pp. 1273-1290, October 1988. doi: 10.1109/31.7601

[3] L. O. Chua and T. Roska, "The CNN paradigm," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 3, pp. 147-156, March 1993. doi: 10.1109/81.222795

[4] T. Roska and L. O. Chua, "The CNN universal machine: An analogic array computer," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 40, no. 3, pp. 163-173, March 1993. doi: 10.1109/82.222815

[5] L. O. Chua and T. Roska, *Cellular Neural Networks and Visual Computing: Foundations and Applications*. Cambridge, United Kingdom: Cambridge University Press, 2002. ISBN: 978-0521652476

[6] M. Gilli, T. Roska, L. O. Chua, and P. P. Civalleri, "CNN dynamics represents a broader class than PDEs," *International Journal of Bifurcation and Chaos*, vol. 12, no. 10, pp. 2051-2068, 2002. doi: 10.1142/S0218127402005868

[7] P. Arena, L. Fortuna, and M. Branciforte, "Reaction-diffusion CNN algorithms to generate and control artificial locomotion," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, no. 2, pp. 253-260, February 1999. doi: 10.1109/81.747195

[8] Z. Yang, Y. C. Soh, and C. K. Ong, "Global stability of cellular neural networks with unbounded time-varying delays," *Neural Networks*, vol. 14, no. 9, pp. 1231-1242, November 2001. doi: 10.1016/S0893-6080(01)00084-2

[9] S. P. Adhikari, H. Kim, C. Yang, and L. O. Chua, "Building cellular neural network templates with a hardware friendly learning algorithm," *Neurocomputing*, vol. 312, pp. 276-284, October 2018. doi: 10.1016/j.neucom.2018.05.113

[10] R. Tetzlaff (Ed.), *Cellular Neural Networks and Their Applications*. Singapore: World Scientific Publishing, 2002. ISBN: 978-9812381477

[11] A. Slavova, *Cellular Neural Networks: Dynamics and Modelling*. Dordrecht, Netherlands: Springer Science+Business Media, 2003. ISBN: 978-9048163298

[12] F. Corinto, M. Gilli, and P. P. Civalleri, "On global dynamic behavior of weakly connected cellular nonlinear networks," *International Journal of Bifurcation and Chaos*, vol. 15, no. 4, pp. 1335-1361, April 2005. doi: 10.1142/S0218127405012740

[13] J. A. K. Suykens, J. Vandewalle, and L. O. Chua, "Nonlinear H∞ synchronization of chaotic Lur'e systems," *International Journal of Bifurcation and Chaos*, vol. 7, no. 6, pp. 1323-1335, June 1997. doi: 10.1142/S0218127497001059

[14] M. Forti and A. Tesi, "New conditions for global stability of neural networks with application to linear and quadratic programming problems," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 42, no. 7, pp. 354-366, July 1995. doi: 10.1109/81.401145

[15] N. T. Tuan, *Deep Learning Cơ Bản*. Hanoi, Vietnam: National Economics University Press, 2021. Available online: https://nttuan8.com/sach-deep-learning-co-ban/