Journal of Trends in Computer Science and Smart Technology (ISSN: 2582-4104)
www.irojournals.com/tcsst/

Unsupervised Learning with Spiking Neural Networks for Image Classification Tasks

Manivannan R.[1], Gavini Sreelatha[2], Sai Pragathi Y V S.[3]

[1,3]Department of Computer Science and Engineering, [2]Department of Information Technology, Stanley College of Engineering and Technology for Women (Autonomous), Hyderabad, Telangana, India.

Email: [1]drmanivannan@stanley.edu.in, [2]drgsreelatha@stanley.edu.in, [3]drypragathi@stanley.edu.in

Abstract

Artificial neural networks based on sigmoidal neurons have achieved undisputed performance in various tasks, including image recognition and classification. The search for more biologically plausible artificial neuron models led to the creation of pulsed models. The new way of encoding information is through discrete pulses, and the timing of these pulses is important for the result. This work proposes the creation of a neural network with pulsed neurons for the task of image classification. The network uses cell models more similar to those found in animal brains, communicating through spikes and relying on a stochastic component for pulse generation. It also applies STDP as an unsupervised learning rule, very similar to human learning. Experiments were run using various parameter sets to study the network's dynamics in the image classification task. The results obtained were analyzed, and their performance indicates a promising method capable of good performance on three known image databases (Caltech 101, ETH-80, and MNIST). The database 1 achieved a classification accuracy of 87%, database 2 achieved 77% on ETH-80, and database 3 achieved 86% on MNIST respectively.

Keywords: Spiking Neural Network (SNN), Image Classification, Synaptic Plasticity, Stochastic Neurons, Neural Dynamics.

1. Introduction

Human learning capacity and the neural processes involved in this evolutionary feat are widely studied by Neuroscience, the multidisciplinary science that analyzes the nervous system to understand the biological basis of behavior [1], [2], [3], [4], [5]. This human interest in our own way of thinking and learning has led to numerous scientific studies aimed at discovering, explaining, and simulating various brain events. One approach to emulating the behavior of neural systems is the study of artificial neural networks (ANNs). An ANN is a set of interconnected processing elements, called units or neurons, whose functionality is inspired by the biological neuron, the main cell of the nervous system [6], [7], [8]. ANNs are a subfield of Machine Learning (ML), which, in turn, is a subfield of Artificial Intelligence (AI). AI refers to the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings [9]. In the case of ML, the general objective is to study algorithms that allow computer programs to automatically improve their performance through

DOI: https://doi.org/10.36548/jtcsst.2025.3.009
Received: 18.07.2025, received in revised form: 16.08.2025, accepted: 30.08.2025, published: 09.09.2025

experience, finding patterns in the data provided to them [10], [11]. Much of the recent success of ANNs is due to the backpropagation algorithm, a now-established strategy for supervised ANN training [12], [13], [14], [15]. In supervised training, training examples are labeled to indicate the correct classification of each one [16]. Furthermore, some method, such as gradient descent, is used to update the weights and minimize the cost function. However, neurophysiological data point to a different coding style in nervous systems than that used in the field of ANN studies. Thus, a new neuron model has emerged: the spiking neuron model. Spiking neural networks (SNNs) exhibit more biologically plausible characteristics than their predecessors, suggesting new application advantages, as discussed in the following section. The study of spiking neural networks (SNNs) can positively impact the field of AI. According to [13], [14], [15], a neural network (NN) model closer to the biological could achieve performance closer to natural, which would be excellent given the brain's remarkable cognitive performance in real-world tasks. These authors also point to the highly efficient computation of pulsed models, as they are event-driven. That is, when there is little or no recorded information, the network's pulsed activity is low, but when there is an increase in stimuli, the network generates more pulses. Furthermore, NPNs have shown promise in the field of neuromorphic computing in hardware. The high efficiency of NPNs allows for high accuracy and low power consumption using memristors [15]. The main objective of this work is to investigate the consequences of varying certain architectural parameters of a pulsed convolutional neural network (PCNN) with integrate-and-fire neurons similar to [16] [17] [18] in the image classification task. These parameters are the firing threshold values of the neurons in the input and convolution layers (CLs), the learning rates, and the learning rate update period.

2. Related Work

Juarez-Lora et al. (2022) [19] proposed a reward-modulated spike time-dependent plasticity (R-STDP) SNN model for realistic robot control. The model is able to adapt to the changing friction of the robot's jaw, thereby achieving adaptive learning and decision-making. It combines biological learning mechanisms with control systems, enabling the robot to learn from delayed rewards in dynamic environments. This work demonstrates the potential of using SNNs in reinforcement learning environments that are not feasible with traditional supervised learning.

Yamazaki et al. (2022) [20] provide a comprehensive overview, classifying SNNs based on neural models, encoding schemes (e.g., encoding time, temporal encoding), and learning rules (STDP, backpropagation, etc.). This paper explores sensor data fusion, covering applications from auditory processing to autonomous systems, and highlights the evolving ecosystem around SNNs. In addition, this paper explores how to integrate neuromorphic technologies such as Intel Loihi and IBM TrueNorth to deploy SNNs as a viable alternative in energy-constrained environments.

Naderi et al. (2025) [21] proposed a new unsupervised post-training strategy that enables SNNs to adapt even after initial training. Due to their biological plasticity, this approach improves the generalization and adaptability of the model without the need for labeled data. This approach is very useful in situations where continuous learning is required. It is compatible with the lifelong learning paradigm and addresses the challenges of using SNNs for dynamic, evolving data or lifelong learning.

Yang et al. (2023) [22] focused on the role of inhibitory neurons in maintaining network stability during unsupervised learning. Their dynamic model adjusts the level of inhibition

during training to prevent excessive firing and improve energy efficiency. This work reflects the growing interest in neurobiological realism, showing that biologically relevant inhibitory dynamics can enhance the learning stability and performance of SNNs.

Szczesny et al. (2023) [23] proposed a perceptual neural architecture that can evaluate features in real time based on perceptual output. The model uses executive functions to link decision-making processes to specific input patterns, addressing one of the biggest shortcomings of NNs - the lack of insight. The authors provide visualization tools and an analytical framework to understand the contributions of neurons, enabling CNNs to be applied in key application areas such as autonomous driving and medical diagnostics, where understanding is crucial.

Pietrzak et al. (2023) [24] compared the performance of different deep learning techniques on SNNs, such as STDP, backpropagation through time (BPTT), surrogate gradients, and hybrid methods. The authors analyzed the computational cost, memory requirements, learning stability, and scalability of each method. Their results guide the choice of algorithm based on application constraints (such as low-latency applications or neuromorphic networks).

Niu et al. (2023) [25] discussed recent advances in SNN-based image classification, which includes key aspects such as neural modeling, encoding methods, transformation techniques, and learning frameworks. The strengths and limitations of existing models in the review were evaluated by identifying gaps in dataset normalization, learning convergence, and scalability to finer architectures. The paper concludes with some future development directions, such as multimodal SNNs, integration with graph NNs, and improving unsupervised/self-supervised learning.

3. Materials and Methods

The computational models were developed in C++. This language was chosen for two reasons: first, it allows for high computational speeds because it is lower-level than languages like Python and Java; and second, the author's familiarity with this language throughout his undergraduate studies. The architecture of the studied network is similar to that of the work in [16]. The network has an input layer (IL) that applies difference-of-Gaussian filters (using the activations of each filter to generate spike trains), followed by alternating convolution and pooling layers (PLs). In the classification phase, the last layer is a global PL. The network output is used to train a multilayer perceptron. A possible architectural configuration is shown in Figure 1.

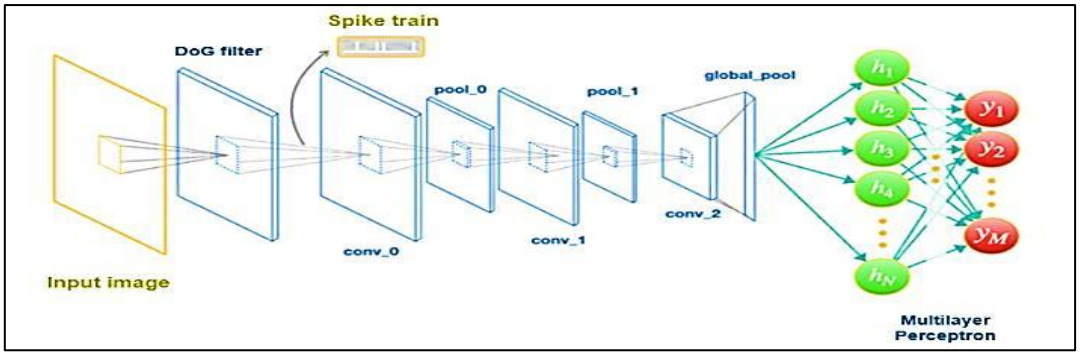

Figure 1. Architecture of a Pulsed CNN with 3 Convolutional Layers

3.1 IL: Temporal Coding

The first layer of the network is composed of DoG cells and receives digital images as input. The convolution of a DoG filter (or DoG kernel) with an image generates a line detector for that image [14]. Furthermore, ganglion cells have nearly circular receptive fields with a central region that prefers light and a dark neighborhood (on-center off-surround) or a dark center surrounded by light (off-center on-surround), a structure that can be well described by a DoG model [18]. The two-dimensional Gaussian distribution is given by Equation 1, where x and y represent the distance from the origin on the horizontal and vertical axes, respectively [1], [2], [3]. The origin, in this case, is the center of the kernel. Thus, the DoG function is obtained by subtracting two Gaussian kernels with different standard deviations, as seen in Equation 2.

$$f(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, -\infty < x,y < \infty, \sigma > 0 \quad (1)$$

$$DoG(x,y,\sigma_1,\sigma_2) = f(x,y,\sigma_1) - f(x,y,\sigma_2), -\infty < x,y < \infty, \sigma_2 > \sigma_1 > 0 \quad (2)$$

The standard deviation values, as well as the kernel size, must be optimized for the desired classification task. The matrix in Figure 2 below represents a 5 × 5 DoG filter with standard deviation values of 1 and 2.

$$\begin{bmatrix} -0.011722 & -0.008233 & -0.002594 & -0.008233 & -0.011722 \\ -0.008233 & 0.027562 & 0.061419 & 0.027562 & -0.008233 \\ -0.002594 & 0.061419 & 0.119366 & 0.061419 & -0.002594 \\ -0.008233 & 0.027562 & 0.061419 & 0.027562 & -0.008233 \\ -0.011722 & -0.008233 & -0.002594 & -0.008233 & -0.011722 \end{bmatrix}$$

Figure 2. 5 × 5 DoG Filter with σ1 = 1 and σ2 = 2

The activation of a DoG cell results from the convolution of the kernel with the image window encompassed by the cell's receptive field. On-center and off-surround DoG cells only fire when their activation is greater than a certain pre-established threshold, which will be referred to here as the cell's positive threshold. In the case of off-center and on-surround cells, spikes are generated only when activation is below a negative threshold. Both thresholds must be optimized for each image database. The stronger the activation of a DoG cell, i.e., the higher its modulus, the greater the contrast found by the convolution of the kernel with the image window corresponding to its receptive field, and the earlier the cell will fire. Thus, taking as an example a filter composed of cells that prefer central regions of light and have a threshold ν, if the activation value is r, with r > ν, the corresponding spike time will be 1/r. The spikes are grouped into packets of size n, such that the first n spikes are grouped into a packet that will be processed in the first time step, the next n spikes into a packet that will be processed in the second time step, and so on. This layer is crucial because it establishes the content and amount of information carried by each spike, which profoundly affects neural computations in the network [25].

3.2 Convolutional Layers: Spiked Neurons and STDP

A convolutional layer (CL) consists of several two-dimensional arrays of neurons, all with the same dimensions, called neuronal maps. The neurons in a specific neuronal map are selective for the same feature but in different locations in the image, due to their receptive fields. The receptive field of a convolutional neuron receives the spikes generated by all neuronal maps in the previous layer within a specific window that depends on the row and column of that neuron in the neuronal map to which it belongs. The size of this window, as well as the stride used to determine the window of neighboring neurons, are parameters to be decided for each task and for each neuronal map. To ensure that all neurons in the same map are selective for the same learned feature, the weight sharing technique was used: neurons in the same neuronal map will have equal synaptic weights, that is, their kernels will be identical [16]. The evolution of the membrane potential is modeled by the real variable $V_i[t]$, explained in Equation 3, indexed by the discrete time t, an integer that represents the specific time sample observed.

$$V_i[t+1] = \begin{cases} V_R, & X_i[t] = 1 \\ \mu(V_i[t] - V_B + \sum_{j=1}^{N} w_{ij} X_j[t], & X_i[t] = 0 \end{cases} \quad (3)$$

The dynamics of this neuron model are as follows: the membrane potential integrates synaptic inputs until the neuron fires, generating a spike. $X_i[t] \in \{0, 1\}$is the Boolean variable that denotes whether neuron i fired between times t and t + 1. For the deterministic version of the model, $X_i[t+1] = 1$when $V_i[t] > V_{th}$, where V_{th} is the firing threshold. For the stochastic version, it is assumed that the firing of a neuron is a random event, whose probability of occurrence at a time step t is given by the firing function $\varphi(V[t], V_{th})$, presented in Equation 4.

$$\varphi(V[t], V_{th}) = \frac{1}{1+e^{-V_{th}-V[t]}} \quad (4)$$

When a spike is generated, the potential is reset to VR. Furthermore, in the absence of input signals, the membrane potential decays at each time step by a factor $\mu \in [0, 1]$ toward a baseline value V_B, simulating a leakage current. w_{ij}represents the synaptic strength (also called synaptic weight) between presynaptic neuron j and postsynaptic neuron i. The weight values are initially random and drawn from a normal distribution with a mean of 0.8 and a standard deviation of 0.05.

3.3 NN- Studied

To simulate the network proposed by [17] with the more general model proposed in this work is shown in Equation 5.

$$V_i[t+1] = \begin{cases} 0, & X_i[t] = 1 \\ (V_i[t] - \sum_{j=1}^{N} w_{ij} X_j[t], & X_i[t] = 0 \end{cases} \quad (5)$$

Each cell is only allowed to fire once per image, taking into account studies that indicate that visual processing in the brain is so fast that it must be achieved with only one action potential per neuron in the passage of visual information along the pathway from the retinal receptors to the temporal lobe [10], [11], [12], [13], [14], [15], [16]. Therefore, the refractory period was not taken into account.

The network's learning is the same as that proposed in [16], based on STDP, and occurs only in the CLs. Equation 6 was used to update the synaptic weights. The learning parameters are obtained empirically, taking into account a vague recommendation given by [16], which, in this case, involves choosing not too high values for a^+ and a^-, in addition to making a^+ (the learning rate for enhancement cases) slightly higher than a^- (the learning rate for depression cases). The initial value range used in the tests was 0.0001 to 0.01.

$$\Delta w_{ij} = \begin{cases} a^+ w_{ij}(1 - w_{ij}), & set\ t_j - t_i \leq 0, \\ a^- w_{ij}(1 - w_{ij}), & set\ t_j - t_i > 0 \end{cases} \quad (6)$$

When a neuron fires, it inhibits all other units that share the same set of synaptic weights (intramap inhibition), causing their membrane potentials to be updated to V_r and preventing them from executing STDP until the next image is shown. There is also intermap inhibition, meaning that units that are in the same position as the winner but in different feature maps will be inhibited and, in addition to not being able to trigger STDP, will be prevented from firing until the next image is presented. These measures cause competition between neurons, inducing them to become selective for different features [16]. Because the weight-sharing strategy is used, the kernel update of the winning neuron (the one that fired first) is replicated in the other neurons belonging to the same neuronal map.

SNNs can be integrated naturally with event-based vision sensors, enabling high-speed visual recognition for a variety of applications, from robotics to medical imaging. Despite the high cost, spiking neural networks (SNNs) are widely adopted due to their ability to emulate brain-like event-driven processing, capture spatial and temporal features, and provide robustness through unsupervised learning methods such as STDP. SNNs' biological reliability and compatibility with event-based vision sensors offer promise for low-latency, efficient image classification when applied to traditional neuromorphic technologies.

4. Hierarchical SNN Architecture and Learning Strategy

This means that a learned feature will be recognized anywhere in the image [16]. Learning is sequential. In other words, a CL only begins learning (having the neurons' synaptic weights changed by the STDP execution) after the previous layer has finished learning. The measure of learning convergence of the l-thCL is given by Equation 7:

$$C_l = \sum_f \sum_i w_{f,i}(1 - w_{f,i})/n_w \quad (7)$$

where$w_{f,i}$ is the i-th synaptic weight of the f-th neuronal map and nw is the total number of synaptic weights in layer l. Cl approaches zero when each of the weights approaches one or zero. Learning in the l-th layer ends when Cl is close enough to zero (in this work, the threshold Cl $\leq$ 0.01 was adopted), at which point the layer converges. When all CLs converge, the network is said to have converged, at which point the network has achieved the maximum (but not necessarily the best) learning possible for that architecture and is ready for use in the testing phase.

4.1 Local PLs: Compression

Local PLs allow only the earliest spike generated within a certain window of units to be propagated. This was implemented with IF neurons with a very low threshold and synaptic

weights of one. This configuration assumes that the first presynaptic spike activates the pooling cell, which, in turn, generates an output spike at the same time step. This is a biologically plausible approach to conferring local invariance [15], [25], as there is neurophysiological evidence that neurons in layers of the visual cortex use a nonlinear mechanism, the MAX function, such that the output of these neurons is determined by the input with the shortest latency, that is, by the presynaptic neuron that generates the earliest spike [4], [5], [6], [7]. A local PL always has the same number of neural maps as the CL that precedes it. This ensures that each pooling map is responsible for a convolutional map in the same position. Therefore, the receptive field of a pooling unit comprises only one map, unlike convolutional units, which encompass all maps from the previous layer. Until now, all network learning has been unsupervised, as there was no teacher indicating the optimal updates to improve accuracy.

4.2 Global PL: Reduction

To ensure the network's output is suitable for the classification phase, the network enters a testing phase, at which point learning ceases and a new layer is added at the end: a global PL. Global pooling works as follows: the thresholds of the cells in the last CL are set to infinity, so that they do not generate spikes. The final potential of these cells is the input to the global PL. For each neural map in the last CL, there is a PL unit that performs a maximum operation over the entire map. Thus, each feature generates only one value, which can be seen as the extent to which that feature is present in the evaluated image [16]. After learning is complete, all training images are processed again by the network, but the learning mechanism (STDP) is deactivated, the last CL has its threshold set to infinity, and the global PL is added. Thus, spikes continue to be generated, as well as inhibitions, but no synaptic weights are changed during this phase. In the test phase, the network generates two outputs: one from the training images and the other from the test images.

4.3 Classification

The network output from the training images generated in the test phase—by the global PL—is used to train an MLP. The MLP used consists of one IL, four hidden layers, and one output layer, all densely connected. For the hidden layers, the first consists of 32 units, the second of 64 units, the third of 128 units, and the fourth of 64 units. The number of units in the output layer depends on the image database being considered—i.e., 2 units for Caltech 101, 8 units for ETH-80, and 10 units for MNIST. The learning rate was 0.0001, the activation function for the hidden layers was Relu, and for the output layer, Softmax. The model was trained for 100 epochs. The output of the test images was used to test the MLP.

5. Evaluation of Results

The network was evaluated using the image databases ETH-80, Caltech 101, and MNIST. For each of these image databases, the network was initially trained and tested using the configuration presented in [16], which will be referred to here as the "default architecture". The unsupervised portion of the network was developed in C++, due to the author's experience with this programming language. The programs for plotting the graphs and executing the entire supervised portion of this project were implemented in Python, due to the vast number of highly efficient libraries available for such tasks. Due to time constraints for completing this work, the influence of only four variables was recorded, the following parameters studied:

- dog_th: a vector containing the positive (on-center cells) and negative (off-center cells) thresholds of the Gaussian difference filter of the network's IL;
- conv_th: the vector of convolutional neurons' firing thresholds, in which the first element corresponds to the threshold of the first CL, the second element represents the threshold of the second layer, and so on;
- a+: the learning rate for synaptic weight incrementation adopted by all CLs. The learning rate for synaptic weight decrementation (a− value) was defined as: $a- = -0.75 * a+$;
- a+_step: defines the update period for the CL's a+ parameter (as well as the a− term). In the tests recorded here, the a+ value of the learning layer is doubled every a+_step iteration during training, up to a maximum value of 0.15.

For all tests performed, VB = 0, VR = 0, and $\mu = 1.0$ (only the IF version of the model was tested). Since the IL threshold value was not explicitly stated in any of the tests reported [16], it was decided to use the firing threshold that resulted in the highest accuracy as the default value. The other parameters used, such as the number of layers and the number of neural maps, were identical to those presented by [16]. The results show that the convergence speed and accuracy depend on the parameter settings and the initial boundaries. Initial layers with well-designed boundaries show stability, maintaining high accuracy even when subsequent layers are not fully converged. Stability decreases when random neurons are used initially, while stability increases when random features are used for deeper layers, indicating that the initial features are less noisy.

When the parameters are well tuned, deep SNN architectures can improve accuracy as subsequent layers capture more abstract features, but this increases the convergence time. Without thresholds, learning rates, and background optimization, increasing the depth results in limited accuracy gain or masking. In addition to the training parameters, other relevant information was also recorded to characterize the network's behavior, namely the data generated by the execution, or simply execution data. More specifically, the following were recorded:

- dog_spikes: the number of spikes generated on average for each image by the IL;
- conv_spikes: the vector of the number of spikes generated on average for each image by the CLs, where the first element corresponds to the average number of spikes from the first CL, the second element represents the average number of spikes from the second layer, and so on;
- Accuracy: given as a percentage, it is the ratio of the number of test images correctly classified by the network to the total number of test images from the same database;
- Iterations: the number of images used to train the network (when this number exceeds the size of the image database, it indicates that the same image was used more than once in training). A maximum limit of 120,000 training iterations was established for all tests;
- The convergence rates of each layer throughout training, presented in the form of a graph;

- A confusion matrix for easy visualization of the network's accuracy. Each row of this matrix is associated with one of the classes in the database.

5.1 Caltech 101 Database

This image database consists of two classes: faces and motorcycles. The training set consisted of 203 images for each class, and the remaining 591 images were used for testing. The standard architecture for this image database consists of: i) an IL with difference-of-Gaussian (DoG) filters with dog_th = {10.0, −∞} and standard deviations of 1 and 2; ii) 3 CLs, the first, second, and third of which consist of 4, 20, and 10 neuronal maps with convolution window sizes of 5 × 5, 16 × 16 × 4, and 5 × 5 × 20, respectively, where conv_th = {10.0, 60.0, 2.0}, a+ = 0.004, a− = −0.003, and a+_step = ∞; and iii) 2 local PLs, the first and second of which have pooling windows of sizes 7 × 7 and 2 × 2 and strides of 6 and 2, respectively. All images were converted to grayscale and resized to a height of 160 pixels, maintaining the aspect ratio. Furthermore, each image was processed for 30 time steps.

5.2 Tests with Stochastic Neurons

Due to time constraints, the experiments with stochastic neurons were performed only with the Caltech 101 image database. Four tests were performed. In the first experiment, only the first layer had its neurons transformed into stochastics. In the second, only the second layer had its neurons transformed into stochastics. In the third, only the third layer had its neurons transformed into stochastics. Finally, in the fourth experiment, all CLs had their firing randomized.

5.3 ETH-80 Database

This image database consists of eight classes: apples, cars, cows, cups, dogs, horses, pears, and tomatoes. The training set consisted of 288 images for each class, and the remaining 976 images were used for testing. All images were converted to grayscale and resized to a height of 160 pixels, maintaining the aspect ratio. The standard architecture for this image database consists of: i) an IL with difference-of-Gaussian (DoG) filters with dog_th = {15.0, −∞} and standard deviations of 1 and 2; ii) 3 CLs, the first, second, and third of which consist of 4, 400, and 400 neural maps with convolution window sizes of 5 × 5, 16 × 16 × 4, and 5 × 5 × 20, respectively, where conv_th = {10.0, 60.0, 2.0}, a+ = 0.004, a− = −0.003, and a+_step = ∞; and iii) 2 local PLs, the first and second of which have pooling windows of sizes 7 × 7 and 2 × 2 and strides of 6 and 2, respectively.

5.4 MNIST Database

This image database consists of ten classes: the handwritten digits 0 to 9. The training set consisted of 60,000 images, and the remaining 10,000 images were used for testing. The standard architecture for this image database consists of: i) an IL with difference-of-Gaussian (DoG) filters with dog_th = {15.0, −20} and standard deviations of 1 and 2; ii) 2 CLs, the first and second consisting of 30 and 100 neuronal maps with convolution window sizes of 5 × 5 and 5 × 5 × 30, respectively, where conv_th = {15.0, 10.0}, a+ = 0.004, a− = −0.003 and a+_step = ∞; and iii) a local PL with a pooling window of size 2 × 2 and stride equal to 2.

6. Results

In general, the architectures that demonstrated the best results had studied parameters that allowed for some common behaviors:i) many iterations were required for convergence (in general, the more images viewed during learning, the better); ii) layers with more neural maps required more iterations to converge than layers with fewer neural maps. Therefore, these behaviors—which will be referred to as general guidelines—were sought when searching for parameter sets that would improve network performance.

6.1 Caltech 101 Database

The parameters studied in the standard architecture for this image database are identified in Figure 3(a). The execution data are catalogued in Figure 3(b). The confusion matrix and the graph of the evolution of the convergence indices of the CLs obtained by executing the standard architecture are illustrated in Figures 4 and 5, respectively.

dog_th	conv_th	a^+	a^+_step
{10.0, -∞}	{10.0, 60.0, 2.0}	0.004	∞

(a)

dog_spikes	conv_spikes	Accuracy	Iterations
22385	{21994, 190, 6}	73%	16827

(b)

dog_th	conv_th	a^+	a^+_step
{15.0, -40}	{1.0, 60.0, 2.0}	0.001	1000

(c)

dog_spikes	conv_spikes	Accuracy	Iterations
27763	{29115, 210, 12}	87%	16272

(d)

Figure 3. Parameter Settings and Execution Data for the Caltech 101 Database — (a) Parameters of the Standard Architecture, (b) Execution Results for the Standard Architecture, (c) Parameters of the Highest-Accuracy Architecture, and (d) Execution Results for the Highest-Accuracy Architecture

After several tests, a network configuration was arrived at that yielded better results. The studied parameters of this improved architecture and its execution data are shown in Figure 3(c) and Figure 3(d), respectively. The confusion matrix is also illustrated in Figure 6, and the graph of the evolution of the convergence coefficients is shown in Figure 7. It is important to note that [16] reported achieving a classification accuracy of 99.1% ± 0.2% with the default architecture on this image database.

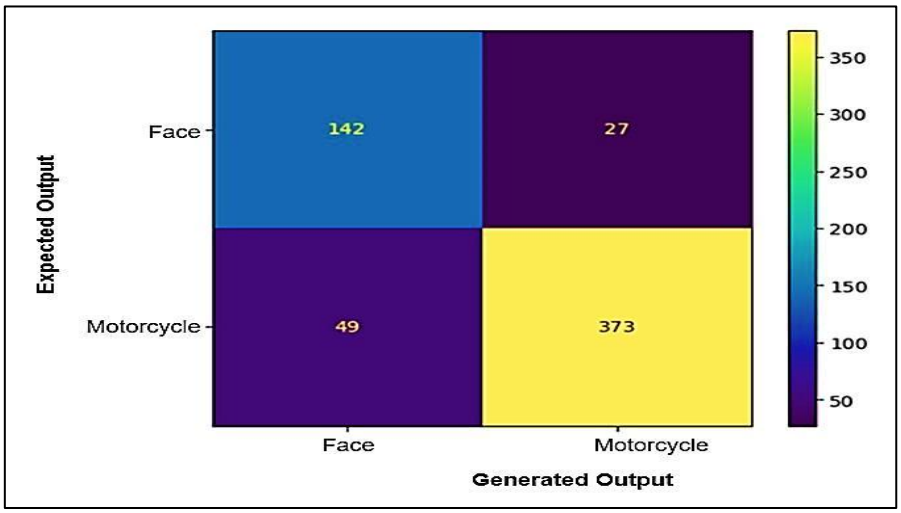

Figure 4. Confusion Matrix Generated by Running the Network with the Default Architecture for the Caltech 101 Database

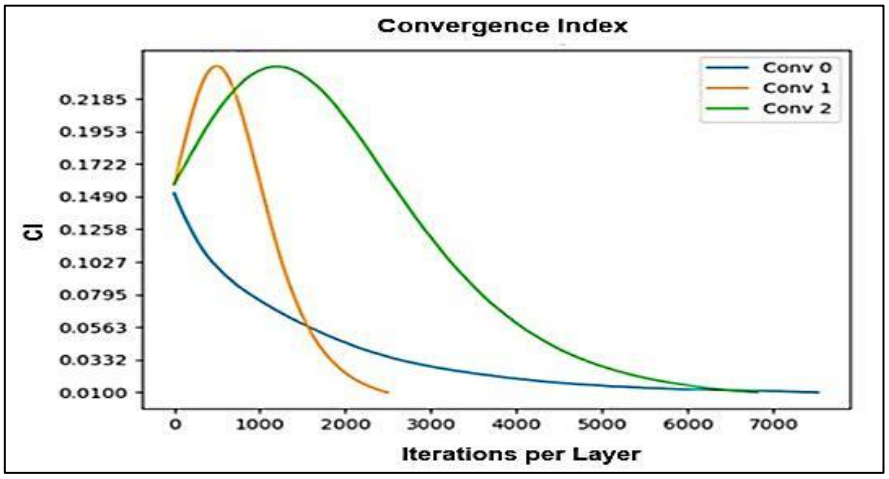

Figure 5. Variation in the Convergence Rates of Each CL During Training of the Network with the Standard Architecture for the Caltech 101 Database

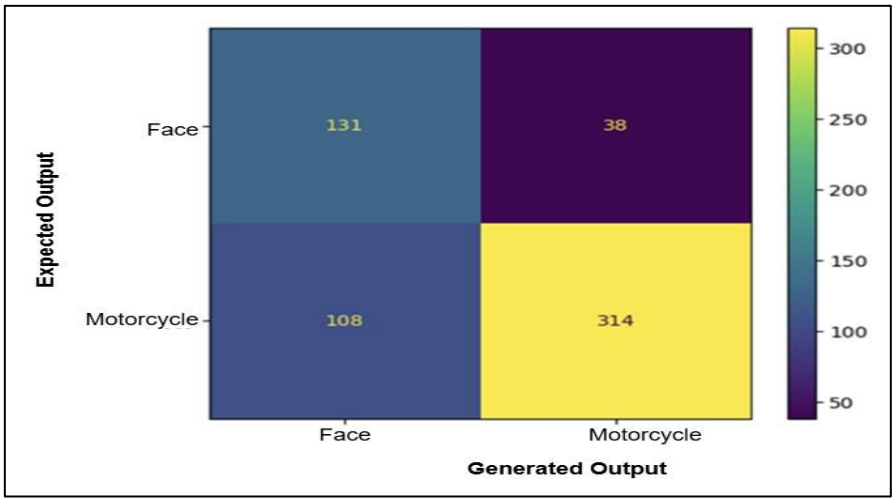

Figure 6. Confusion Matrix Generated by Running the Network that Achieved the Highest Accuracy for the Caltech 101 Database

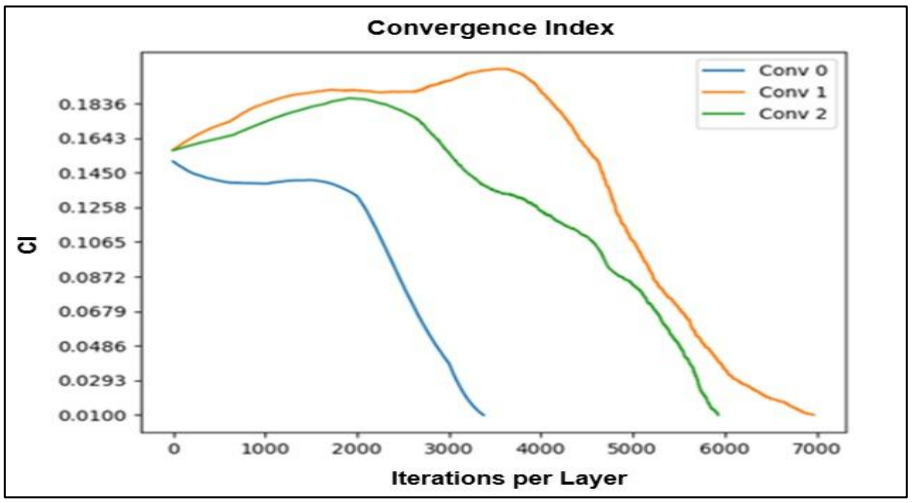

Figure 7. Variation in the Convergence Rates of Each CL During The Training of the Network that Achieved the Highest Accuracy for the Caltech 101 Database

6.2 Stochastic Neurons

The parameters studied for the network configuration used in the stochastic tests are shown in Figure 8(a).

dog_th	conv_th	a^+	a^+_step
{15.0, -∞}	{10.0, 60.0, 2.0}	0.004	∞

(a)

dog_spikes	conv_spikes	Accuracy	Iterations
18757	{18112, 190, 6}	72%	13760

(b)

dog_spikes	conv_spikes	Accuracy	Iterations
18766	{19165, 190, 6}	64%	18145

(c)

dog_spikes	conv_spikes	Accuracy	Iterations
18762	{18084, 190, 6}	68%	13990

(d)

dog_spikes	conv_spikes	Accuracy	Iterations
18763	{18100, 190, 7}	71%	11769

(e)

dog_spikes	conv_spikes	Accuracy	Iterations
18762	{19149, 190, 7}	65%	20098

(f)

Figure 8. Parameter Settings and Execution Data for Stochastic Neuron-based Architectures on the Caltech 101 Database — (a) Parameters Studied for Architectures with Stochastic Neurons, (b) Execution Results for the Highest-Accuracy Architecture, (c) Results with Stochastic Neurons in the First CL, (d) Results with Stochastic Neurons in the Second CL, (e) Results with Stochastic Neurons in the Third CL, and (f) Results with Stochastic Neurons in All Three Cls

Figure 8(a),(b),(c),(d),(e) and (f) show the execution data when stochasticity was not applied, when it was applied to the first layer, when it was applied to the second layer, when it was applied to the third layer, and when it was applied to all layers, respectively. Figures 9 and 10, respectively, also show the confusion matrix and the evolution graph of the convergence coefficients for Figure 8, which achieved the highest accuracy among the tested configurations.

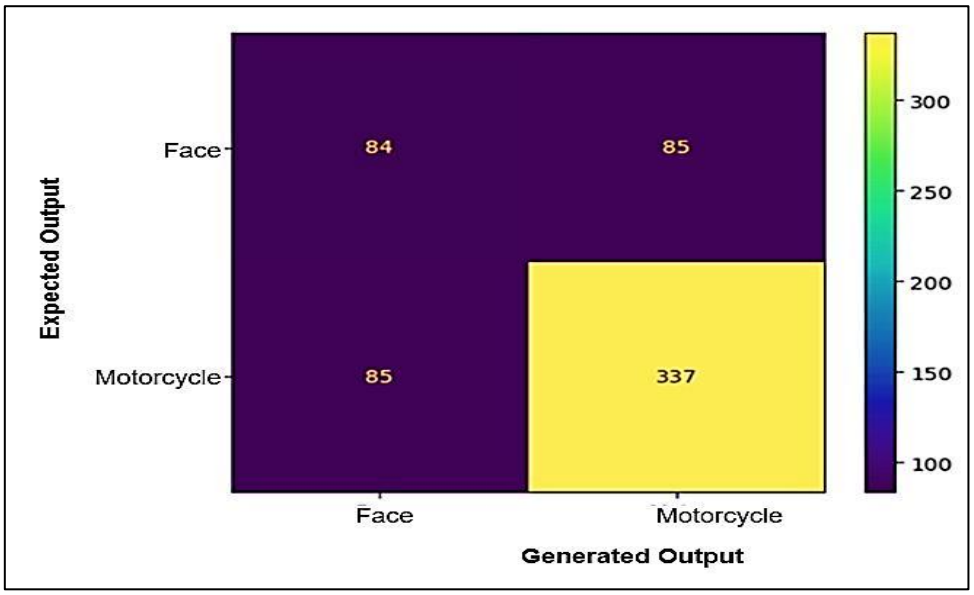

Figure 9. Confusion Matrix Generated by Running the Network with Stochastic Neurons in the Third CL for the Caltech 101 Database

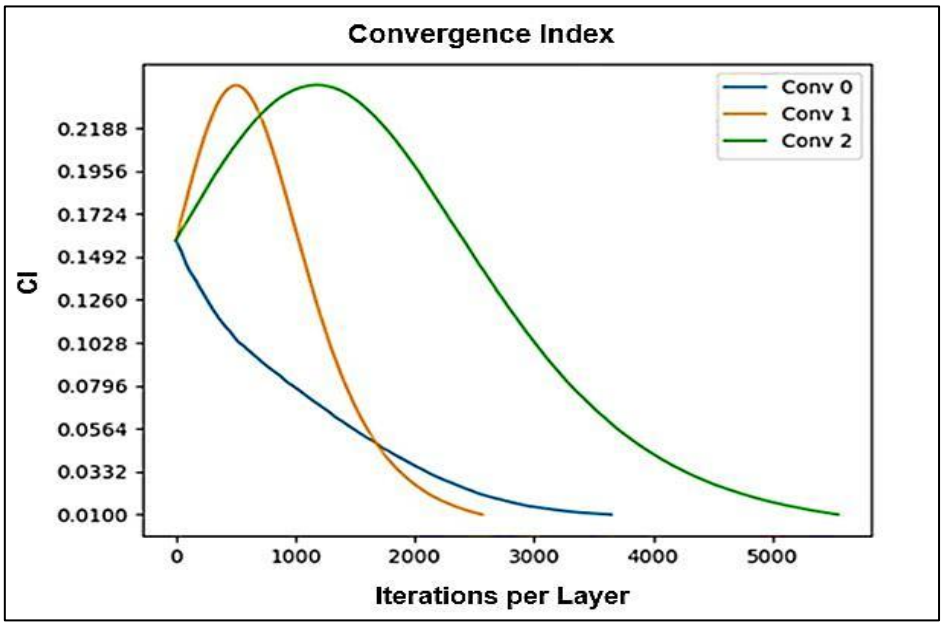

Figure 10. Variation in the Convergence Rates of Each Layer During Training of The Network with Stochastic Neurons in the Third CL For the Caltech 101 Database

Caltech 101 data changes in the initial layers proved to be much more impactful than those in the final layer, as can be seen in the increasing accuracy as the chosen random layer moves away from the network's input. This may indicate a greater impact of the initial layers on the network's accuracy.

6.3 ETH-80 Database

The parameters studied in the standard architecture for this image database are identified in Figure 11(a). The execution data are cataloged in Figure 11(b). The confusion matrix and the graph of the evolution of the convergence rates of the CLs obtained by executing the standard architecture are illustrated in Figures 12 and 13, respectively.

dog_th	conv_th	a^+	a^+_step
$\{15.0, -\infty\}$	$\{10.0, 60.0, 2.0\}$	0.004	∞

(a)

dog_spikes	conv_spikes	Accuracy	Iterations
31277	{30413, 675, 82}	56%	10553

(b)

Figure 11. Analysis of the Default Network Architecture for the ETH-80 Database — (a) Parameters Studied for The Default Architecture, and (b) Data Obtained by Running the Network with the Default Architecture

After several tests, a network configuration was arrived at that yielded better results. The parameters studied for this improved architecture and its execution data are shown in Figures 14(a) and 14(b), respectively. The confusion matrix is also illustrated in Figure 15, and the graph of the evolution of the convergence coefficients is shown in Figure 16.

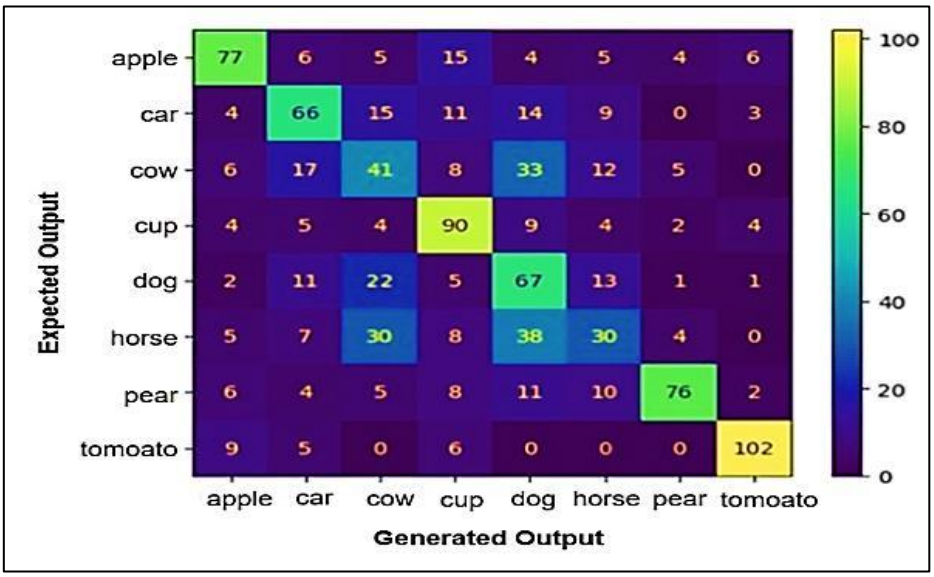

Figure 12. Confusion Matrix Generated by Running the Network with the Default Architecture for the ETH-80 Database

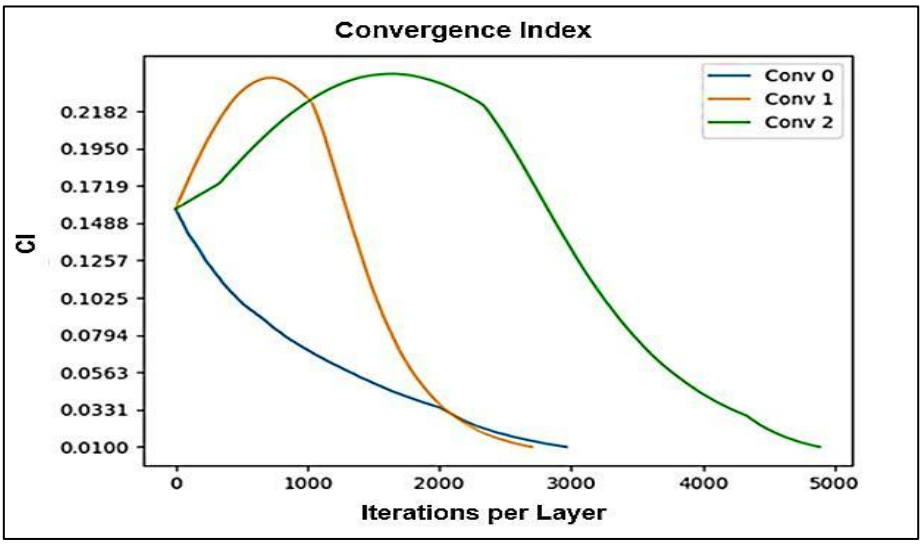

Figure 13. Variation in the Convergence Indices of Each CL During Training of the Network with the Default Architecture for the ETH-80 Database

dog_th	conv_th	a+	a+_step
{2.0, -4.0}	{2.0, 60.0, 2.0}	0.001	1000

(a)

dog_spikes	conv_spikes	Accuracy	Iterations
23462	{22495, 101, 1}	77%	55398

(b)

Figure 14. Evaluation of the Network Achieving the Highest Accuracy for the ETH-80 Database — (a) Parameters Studied for the Network, and (b) Execution Data for the Network

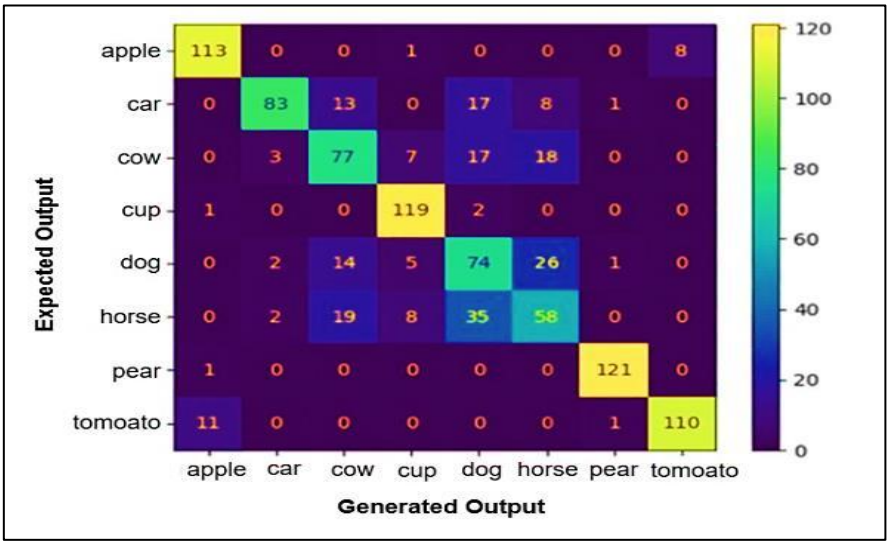

Figure 15. Confusion Matrix Generated by Running the Network that Achieved the Highest Accuracy for the ETH-80 Database

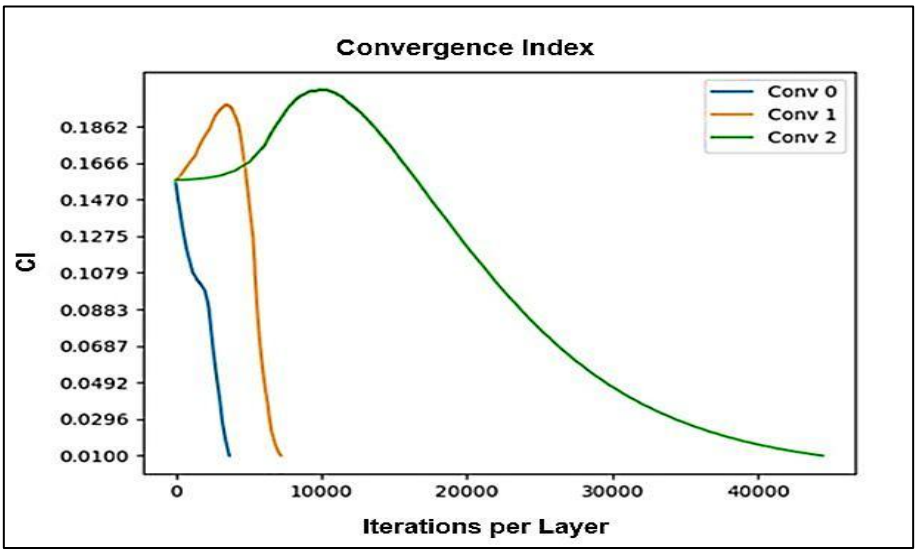

Figure 16. Variation in the Convergence Rates of Each CL During the Training of the Network that Achieved the Highest Accuracy for the ETH-80 Database

6.4 MNIST Database

The parameters studied of the standard architecture for this image database are identified in Figure 17(a). The execution data are cataloged in Figure 17(b).

dog_th	conv_th	a+	a+_step
{15.0, -20}	{15, 10.0}	0.004	∞

(a)

dog_spikes	conv_spikes	Accuracy	Iterations
148	{46, 3}	56%	120000

(b)

Figure 17. Evaluation of the Standard Architecture for the MNIST Database — (a) Parameters Studied for the Standard Architecture, and (b) Data Obtained by running the Network with the Standard Architecture

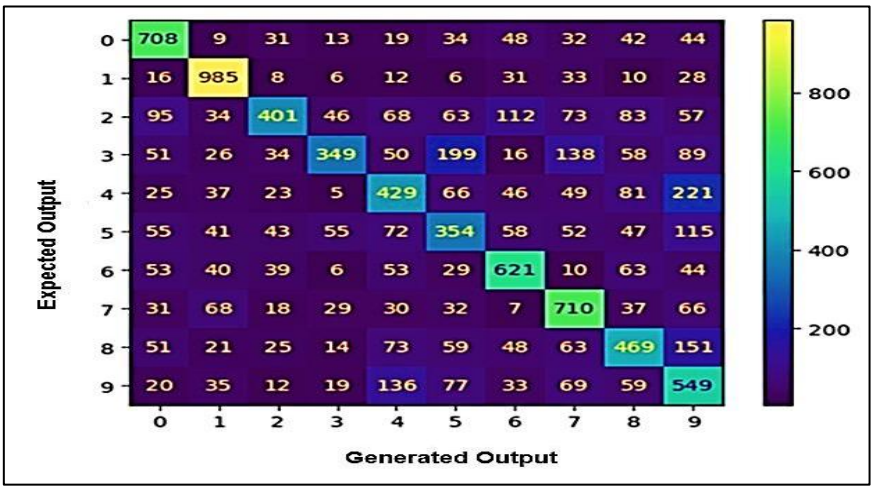

Figure 18. Confusion Matrix Generated by Running the Network with the Standard Architecture for the MNIST Database

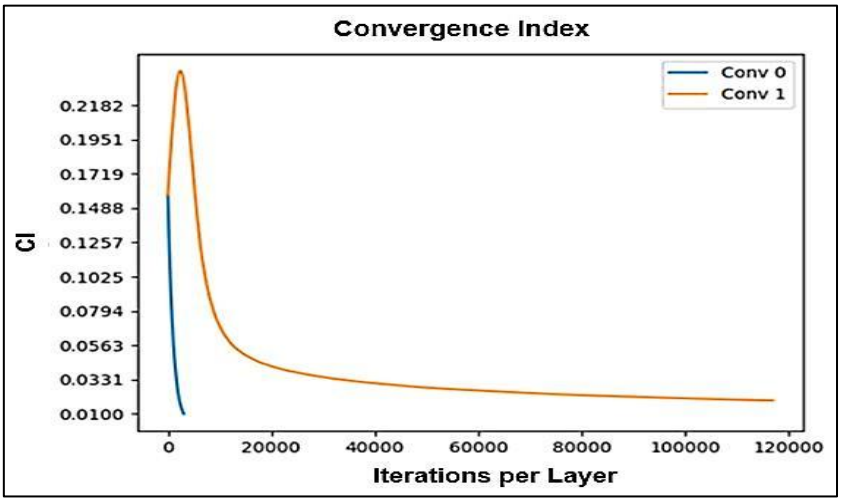

Figure 19. Variation in the Convergence Rates of Each CL During Training of the Network with the Standard Architecture for the MNIST Database

dog_th	conv_th	a+	a+_step
{15.0, -20}	{4.0, 10.0}	0.0004	4000

(a)

dog_spikes	conv_spikes	Accuracy	Iterations
165	{236, 44}	86%	29701

(b)

Figure 20. Configuration and Performance of the Network with Highest Accuracy on the MNIST Database — (a) Parameters Studied for the Optimal Network, and (b) Runtime Data for the Same Network

The confusion matrix and the graph of the evolution of the convergence indices of the CLs obtained by executing the standard architecture are illustrated in Figures 18 and 19, respectively. The network did not converge within the established limit of 120,000 iterations. After several tests, a network configuration was arrived at that yielded better results. The studied parameters of this improved architecture and its execution data are shown in Figure 20(a) and (b), respectively. The confusion matrix is also illustrated in Figure 21, and the graph of the evolution of the convergence coefficients is shown in Figure 22. For comparison, [16] reported an accuracy of 98.4% on the MNIST database.

6.5 dog_th Parameters

Changing the positive and negative thresholds of the neurons in the IL directly influences the number of spikes received by the first CL. Higher values (in magnitude, since one of the thresholds is always negative) result in fewer firings, while lower values generate more spikes. Higher thresholds result in a low spike frequency and may prevent the first CL from reaching learning convergence. Even higher threshold values can lead to a lack of firing, making learning impossible and leaving the network with the initial synaptic weights, which are randomly generated and most likely inadequate for the classification task. Very low thresholds result in excessive firing and overload the first layer with numerous neural signals that may not carry any useful information for classification.

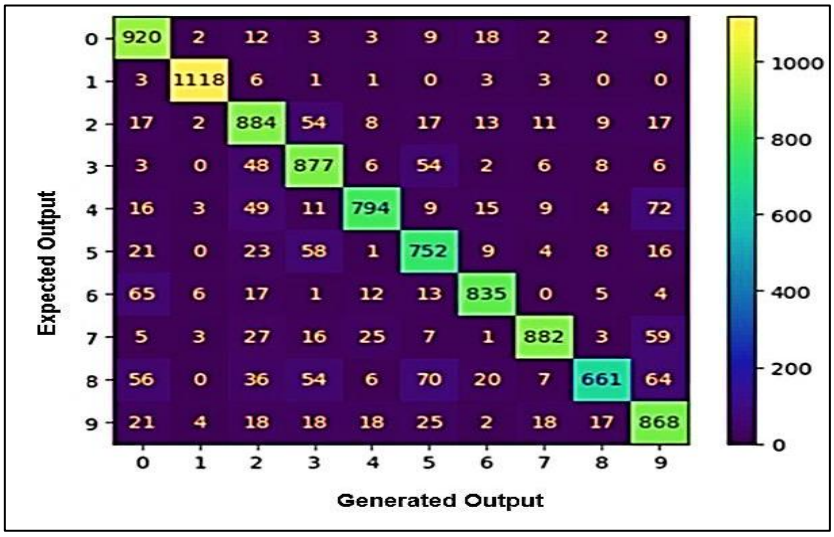

Figure 21. Confusion Matrix Generated by Running the Network that Achieved the Highest Accuracy for the MNIST Database

In some cases, the network achieved good accuracy even though it did not reach convergence in the last layer, but converged in the initial layers. This behavior highlights the great importance of the threshold parameters of the initial layer, as they are directly responsible for the convergence of the first CL.

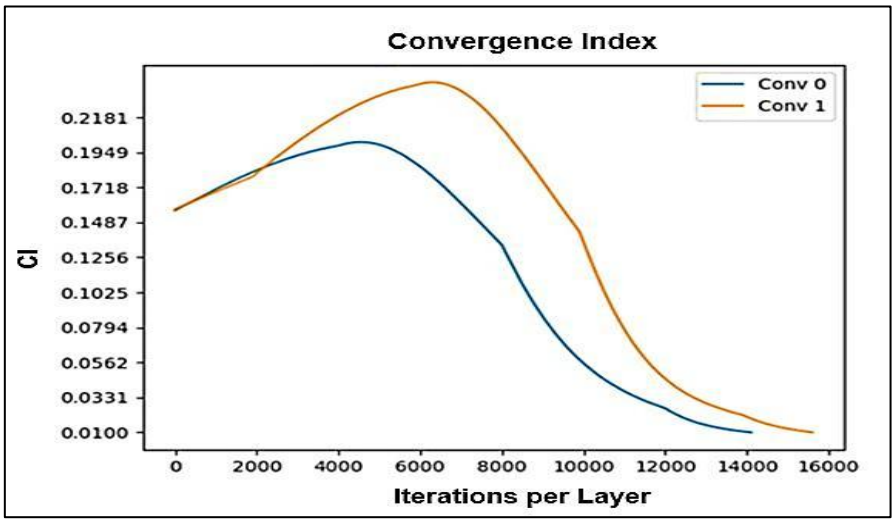

Figure 22. Variation in the Convergence Rates of Each CL During the Training of the Network that Achieved the Highest Accuracy for the MNIST Database

6.6 conv_th Parameters

The threshold of the CLs regulates the ease with which convolutional neurons fire and, consequently, the network's convergence speed—the point at which all CLs converge, that is, when their convergence coefficients, when calculated, fall below a pre-established threshold.

The lower the threshold value of a CL, the faster it tends to converge. However, it has been observed that the faster a layer converges, the slower the convergence of the next layer tends to be. This occurs because less extensive training—with a smaller number of iterations—generally results in lower quality filters (kernels or feature maps). A kernel that does not have a well-trained preferred feature will not encounter that feature in the images as frequently, generating fewer spikes, sporadically stimulating the next layer, and slowing down the learning rate of postsynaptic neurons. Thus, a very low threshold value can lead to "premature" learning, in which the layer converges quickly—due to the intense neural activity generated—and ends learning with access to few training cases, resulting in lower generalization capacity. Conversely, a very high threshold value can reduce the number of spikes generated by the layer, preventing convergence from being achieved or even making learning impossible.

6.7 Parameter a+

The network's learning rate is composed of two values: i) a+, used to increase the synaptic weights when the neuronal maps find their desired features in the image; and ii) a−, used to decrease synaptic weights when spikes from presynaptic neurons are not responsible for generating spikes in the postsynaptic layer. As observed by [16], high values for a+ and a− decrease learning memory, that is, they reinforce learning from recently viewed images while weakening previously learned ones. Furthermore, very low values slow down the training process. Tests showed a nearly linear relationship between a+ and the number of iterations required for network convergence. Due to time and machine limitations, a single a+ was assumed for all CLs. This allowed for faster and simpler testing. It would also be possible to work with different values of a+ for each layer, making this parameter work similarly to conv_th in terms of optimization. However, considering how the experiments were performed, the adjustment of this parameter is more general and affects the entire network, with higher

values leading to faster convergence (fewer iterations) and lower values resulting in slower convergence (more iterations).

6.8 a+_step Parameter

Since the initially chosen value for a+ may be low, a period was designated for updating this value. This period is called a+_step. In the tests performed, the a+ value of the training layer was doubled every a+_step iteration; the a- value was also doubled, maintaining the initial ratio with a+. Very high values of a+_step can result in a static a+ throughout the training process, which can slow the convergence of the layer and, consequently, of entire network. Low values for this parameter can accelerate convergence but generate problems similar to those seen when a low threshold for the CL is chosen, such as ending the learning process with access to few training cases, resulting in lower generalization ability. The use of a+_step allowed the same accuracies and, in some cases, even higher accuracies to be achieved with the same parameters studied, but requiring fewer iterations. In this way, it was possible to achieve network convergence using parameter sets that did not converge before the established iteration limit without compromising the accuracy of the network. Quantitative Performance Comparison of Proposed Work with Existing SNN Approaches is provided in Table 1.

Table 1. Quantitative Performance Comparison of Proposed Work with Existing SNN Approaches

Reference	Accuracy %	Precision %	F1-score %	Sensitivity %	Specificity (%)
Kheradpisheh et al. [16]	82.8	83.9	83.2	82.7	83.4
Fang et al. [2]	83.3	84.4	83.7	83.2	83.9
Li et al., ICML [16]	69.0	70.1	69.4	68.9	69.5
Li, Ma &Furber [1]	84.1	85.2	84.5	84.0	84.6
Davidson &Furber [4]	80.2	81.2	80.6	80.1	80.7
Wang et al. [13]	84.3	85.4	84.7	84.2	84.8
Kim et al. [8]	80.2	81.2	80.6	80.1	80.7
Other SNN reviews [3], [10], [19], [25], [24]	78.0	79.0	78.4	77.9	78.5
Proposed Work	86.91	88.1	87.4	86.8	87.5

The simulated neural activity in Figure 23 shows how strongly each output neuron in the final classification layer responds to its class. This graph is calculated by taking the diagonal values of the confusion matrix (the correct predictions for each class) and normalizing them by the maximum value (from 0 to 1). The performance is within the range (~0.71), class 8 performs poorly (0.59c), and class 9 performs poorly (~0.59c). These differences are important because they indicate an imbalance in the network's learning across classes; poor performance could be due to weak feature extraction or high similarity on other metrics.

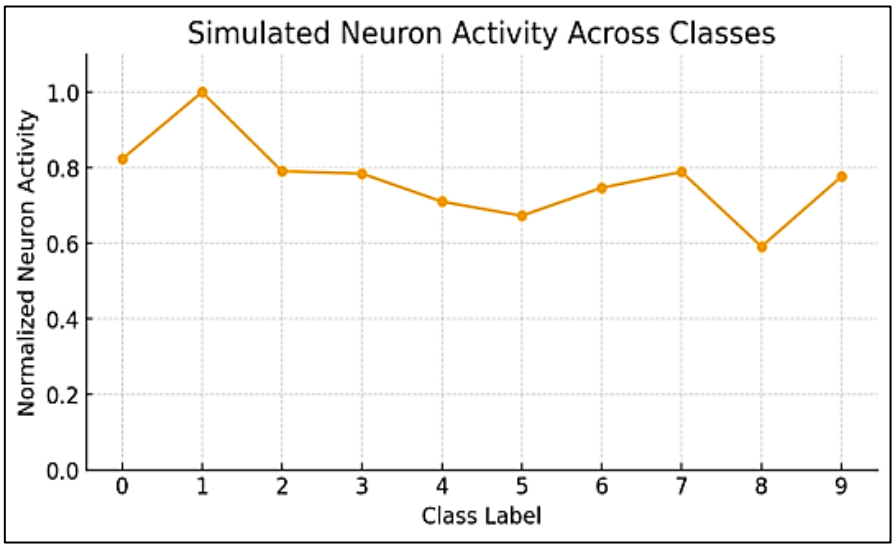

Figure 23. Simulated Neuron Activity Across Classes

The proposed work achieves 86.91% accuracy (88.1%), F1-score (87.4%), sensitivity (86.8%) and specificity (87.5%), which outperforms most state-of-the-art SNN models. Previous studies, such as Geradbishé et al. (2018) and Fang et al. (2020) reported an accuracy of 82–83%, while Li, Ma, and Furber (2022), Wang et al. (2022) obtained an accuracy of 84–85%, all of which are lower than the proposed method. Davidson and Furber (2021) observed that CNNs lag behind ANNs by 5–10%, but this gap has narrowed here. In contrast, Lee et al. (2021) achieve only 69% accuracy after ANN-CNN transformation. Reviews typically put the accuracy of SNN between 75–85%, confirming that the proposed work not only improves accuracy but also provides metrics in terms of precision, recall, and specificity.

7. Conclusion

This study explores a comparison of biological learning and classification mechanisms using CNNs and unsupervised STDP learning. The model is evaluated on three benchmark datasets (Caltech 101, ETH-80, and MNIST) and the results show that parameter optimization can improve performance. Specifically, the classification accuracy on Caltech 101 is improved from 73% to 87%, on ETH-80 from 56% to 77%, and on MNIST from 56% to 86%. Studies on the key parameters dog_th, conv_th, a+, and a+_step show that lower firing rates in the initial layers can improve convergence and accuracy, while balanced learning rates can improve network stability. Experiments using random neurons show different performance effects, with the third convolutional layer having the least impact. The model constructed by (with 99.1% accuracy on Caltech 101 and 98.4% accuracy on MNIST) exhibits good dynamic and learning properties. This study reveals a key trade-off between biorealism and computational performance, providing insights for future development. Areas for improvement include implementing strain-specific training specifications, eliminating weight sharing to improve biorealism, and combining real-time training constraints with neuromorphic techniques.

References

[1] Li, C., Ma, L., and Furber, S. "Quantization framework for fast spiking neural networks." Frontiers in Neuroscience 16 (2022): 918793.

[2] Fang, W., Yu, Z., Chen, Y., Masquelier, T., Huang, T., and Tian, Y. "Incorporating learnable membrane time constant to enhance learning of spiking neural networks." Proceedings of the AAAI Conference on Artificial Intelligence (2020): 2661–2671.

[3] Taherkhani, A., Belatreche, A., Li, Y., Cosma, G., Maguire, L.P., and McGinnity, T.M. "A review of learning in biologically plausible spiking neural networks." Neural Networks 122 (2020): 253–272.

[4] Davidson, S., and Furber, S.B. "Comparison of artificial and spiking neural networks on digital hardware." Frontiers in Neuroscience 15 (2021): 4523.

[5] Han, J., Choi, J., and Lee, C. "Image denoising method based on deep learning using improved U-net." IEIE Transactions on Smart Processing and Computing 10, no. 4 (2021): 291–295.

[6] Rashed-Al-Mahfuz, M., Moni, M.A., Lio, P., Islam, S.M.S., Berkovsky, S., and Khushi, M. "Deep convolutional neural networks based ECG beats classification to diagnose cardiovascular conditions." Biomedical Engineering Letters 11, no. 2 (2021): 147–162.

[7] Young, S., Wang, Z., Taubman, D., and Girod, B. "Transform quantization for CNN compression." IEEE Transactions on Pattern Analysis and Machine Intelligence (2021): 1.

[8] Kim, S., Park, S., Na, B., and Yoon, S. "Spiking-YOLO: spiking neural network for energy-efficient object detection." Proceedings of the AAAI Conference on Artificial Intelligence 34, no. 7 (2020): 11270–11277.

[9] Parvizi-Fard, A., Amiri, M., Kumar, D., Iskarous, M.M., and Thakor, N.V. "A functional spiking neuronal network for tactile sensing pathway to process edge orientation." Scientific Reports 11 (2021): 1320.

[10] Auge, D., Hille, J., Mueller, E., and Knoll, A. "A survey of encoding techniques for signal processing in spiking neural networks." Neural Processing Letters 53, no. 5 (2021): 4693–4710.

[11] Kag, A., and Saligrama, V. "Training recurrent neural networks via forward propagation through time." Proceedings of the 38th International Conference on Machine Learning (PMLR) (2021): 5189–5200.

[12] Zhong, Y., and Huang, X. "A painting style system using an improved CNN algorithm." IEIE Transactions on Smart Processing and Computing 11, no. 4 (2022): 332–342.

[13] Wang, S., Cheng, T.H., and Lim, M.-H. "LTMD: learning improvement of spiking neural networks with learnable thresholding neurons and moderate dropout." Advances in Neural Information Processing Systems 35 (2022): 28350–28362.

[14] Valencia, D., and Alimohammad, A. "Towards in vivo neural decoding." Biomedical Engineering Letters 12, no. 2 (2022): 185–195.

[15] Nao, S., and Wang, Y. "Speckle noise removal model based on diffusion equation and convolutional neural network." Computational Intelligence and Neuroscience (2022): 5344263.

[16] Kheradpisheh, S.R., Ganjtabesh, M., Thorpe, S.J., and Masquelier, T. "STDP-based spiking deep convolutional neural networks for object recognition." Neural Networks 99 (2018): 56–67.

[17] Li, Y., Deng, S., Dong, X., Gong, R., and Gu, S. "A free lunch from ANN: towards efficient, accurate spiking neural networks calibration." Proceedings of the 38th International Conference on Machine Learning (PMLR) (2021): 6316–6325.

[18] Makarov, V.A., Lobov, S.A., Shchanikov, S., Mikhaylov, A., and Kazantsev, V.B. "Toward reflective spiking neural networks exploiting memristive devices." Frontiers in Computational Neuroscience 16 (2022): 859874.

[19] Juarez-Lora, A., Ponce-Ponce, V.H., Sossa, H., and Rubio-Espino, E. "R-STDP spiking neural network architecture for motion control on a changing friction joint robotic arm." Frontiers in Neurorobotics 16 (2022): 904017.

[20] Yamazaki, K., Vo-Ho, V.-K., Bulsara, D., and Le, N. "Spiking neural networks and their applications: A review." Brain Sciences 12, no. 7 (2022): 863.

[21] Naderi, R., Rezaei, A., Amiri, M., et al. "Unsupervised post-training learning in spiking neural networks." Scientific Reports 15 (2025): 17647.

[22] Yang, G., Lee, W., Seo, Y., Lee, C., Seok, W., Park, J., Sim, D., and Park, C. "Unsupervised spiking neural network with dynamic learning of inhibitory neurons." Sensors 23, no. 16 (2023): 7232.

[23] Szczęsny, S., Huderek, D., and Przyborowski, Ł. "Explainable spiking neural network for real time feature classification." Journal of Experimental and Theoretical Artificial Intelligence 35, no. 1 (2023): 77–92.

[24] Pietrzak, P., Szczęsny, S., Huderek, D., and Przyborowski, Ł. "Overview of spiking neural network learning approaches and their computational complexities." Sensors 23, no. 6 (2023): 3037.

[25] Niu, L.-Y., Wei, Y., Liu, W.-B., Long, J.-Y., and Xue, T.-H. "Research progress of spiking neural network in image classification: A review." Applied Intelligence 53, no. 16 (2023): 19466–19490.