An Enhanced Chicken Swarm Optimization Algorithm Using Gaussian and Tent Chaotic Map Functions

Olatunji, Babatunde Lekan., Olabiyisi, Stephen Olatunde., Oyeleye, Christopher Akinwale, and Omotade, Adedotun Lawrence

Department of Computer Science, Ladoke Akintola University of Technology, Ogbomoso, Nigeria

DOI: https://doi.org/10.51584/IJRIAS.2025.100700059

Received: 25 June 2025; Accepted: 04 July 2025; Published: 08 August 2025

ABSTRACT

Nature-inspired optimization algorithms have proven effective in addressing complex optimization problems, but they often suffer from premature convergence to local optima. Chicken Swarm Optimization (CSO), modeled after the hierarchical behavior of chickens, is one such algorithm that, despite its strengths, can stagnate due to poor exploration dynamics. This study proposes an Enhanced Chicken Swarm Optimization (ECSO) algorithm that integrates chaotic map functions, specifically Gaussian and Tent maps, to improve its exploration capabilities and mitigate premature convergence. The developed enhancements dynamically influence the movement updates of roosters and hens, significantly improving the algorithm's ability to discover globally optimal solutions. The ECSO is applied to optimise CNN in a forensic recognition task. Simulation results indicate that ECSO exhibits superior convergence behavior and search space coverage compared to the standard CNN and CSO optimized CNN. The developed algorithm demonstrates improved performance in both recognition accuracy and computational efficiency, validating its suitability for real-world forensic tasks.

Keywords: Enhanced Chicken Swarm Optimization, Chaotic Maps, Gauss Map, Tent Map, Nature-Inspired Algorithms, Optimization algorithm.

INTRODUCTION

Swarm intelligence-based optimization algorithms have shown promise across a broad range of machine learning applications, including feature selection, parameter tuning, and training deep learning models. Among them, Chicken Swarm Optimization (CSO) stands out due to its biologically inspired mechanics and adaptability. However, CSO remains vulnerable to premature convergence and stagnation in local optima, particularly when applied to high-dimensional and non-linear optimization problems such as hyperparameter tuning for Convolutional Neural Networks (CNNs).

This paper introduces an Enhanced Chicken Swarm Optimization (ECSO) algorithm, incorporating chaotic map functions to improve the global search capabilities of the CSO framework. The chaotic dynamics introduces ergodicity, randomness, and sensitivity to initial conditions thereby enhancing exploration without compromising convergence speed. Specifically, we integrate the Gauss map for rooster position updates and the Tent map for hen updates. These enhancements aim to tackle two key issues in traditional CSO: lack of diversity and early convergence. The ECSO is applied to tune critical CNN hyperparameters, contributing both to the theoretical field of bio-inspired computing and the practical domain of forensic document analysis.

Optimization plays a fundamental role across diverse domains in engineering, machine learning, and computational sciences. Traditional optimization techniques such as gradient descent or exhaustive search methods often become inefficient in solving non-convex, high-dimensional problems. Metaheuristic algorithms, inspired by natural systems, offer flexible and effective alternatives (Yang, 2014; Mirjalili, 2019).

Optimization is everywhere, and is thus an important paradigm itself with a wide range of applications. In almost all applications in engineering and industry, researchers are always trying to optimize something, whether to minimize the cost and energy consumption, or to maximize the profit, output, performance and efficiency. As

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

resources are finite and systems become increasingly complex, optimization is no longer optional but critical (Yang, 2015).

For any optimization problem, the integrated components of the optimization process are the optimization algorithm, an efficient numerical simulator and a realistic-representation of the physical processes we wish to model and optimize. This is often a time-consuming process, and in many cases, the computational costs are usually very high. Once there is a good model, the overall computation costs are determined by the optimization algorithms used for search and the numerical solver used for simulation (Yang, 2016).

Optimization problems can be formulated in many ways. For example, the commonly used method of least-squares is a special case of maximum-likelihood formulations.

By far the most widely formulation is to write a nonlinear optimization problem as

Minimize $f_i(x)$, (=1,2i , ..., M),

Subject to the constraints

$$h_j(x) = 0, \quad (j = 1,2,...,J),$$

$$g_k(x) \leq 0, \quad (k = 1,2,...,K),$$

where fi, hj and gk are in general nonlinear functions. Here the design vector x= (x x1, 2 ,...,xn) can be continuous, discrete or mixed in n -dimensional space. The functions fi are called objective or cost functions, and when M >1, the optimization is Multiobjective or Multicriteria. Metaheuristic algorithms are often nature-inspired, and they are now among the most widely used algorithms for optimization. They have many advantages over conventional algorithms. Nature-inspired algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Chicken Swarm Optimization (CSO) are prominent examples. These algorithms simulate biological or physical phenomena to guide search processes toward optimal or near-optimal solutions (Pham et al., 2014).

Chicken Swarm Optimization (CSO)

Chicken Swarm Optimization (CSO), introduced by Meng et al. (2014), is a nature-inspired algorithm that simulates the social behavior and foraging strategies of chickens. Despite its competitive performance, CSO's tendency to converge prematurely due to insufficient exploration limits its broader applicability. Integrating chaotic maps into CSO offers a solution to this challenge, leveraging the deterministic yet non-repetitive nature of chaotic sequences to enhance the diversity and adaptability of the swarm (Caponetto et al., 2003).

CSO divides the population into three hierarchical roles: roosters (leaders), hens (followers), and chicks (learners). These roles determine how agents explore the search space:

- Roosters explore locally around promising solutions.
- Hens follow roosters and interact with other hens.
- Chicks follow mother hens with limited autonomy.

While this structure balances exploration and exploitation, stagnation can occur, especially in static or complex fitness landscapes (Deb et al., 2020).

Initialization

A population of individuals is initialized with random positions in the solution space. Fitness is evaluated, and chickens are ranked to determine roles:

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

- The top individuals become roosters.
- The next set becomes hens, among which a subset are mother hens.
- The remaining individuals are chicks assigned to mothers.

This hierarchy is updated periodically based on fitness changes every GG generations (Wang et al., 2018).

Movement Update Rules

The update equations differ across roles:

- **Roosters** update based on fitness.
- **Hens** update based on movement toward both a rooster and another individual.
- **Chicks** follow their assigned mother hen with a learning factor.

However, without mechanisms to increase randomness or exploration, the swarm can become trapped in local optima (Li et al., 2020).

Chaos Theory

The chaos theory deals with dynamic chaotic systems that are potentially impossible to predict or control, like turbulence, weather, and stock market. These phenomena are often described by fractal mathematics, which captures the infinite complexity of nature. Many natural objects exhibit fractal properties, including landscapes, clouds, trees, organs, rivers, and many of the systems such as exhibit complex life and chaotic behavior. Chaos theory describes deterministic systems with unpredictable behavior due to sensitivity to initial conditions (Mojarrad and Ayubi, 2015). This sensitivity, called the "butterfly effect," can be modeled using chaotic maps like Gaussian and Tent maps to generate pseudo-random sequences for exploration.

The mathematical use of the word chaos does not align well with its more common usage to indicate lawlessness or the complete absence of order. On the contrary, mathematically chaotic systems are, in a sense, perfectly ordered, despite their apparent randomness. The famous feature of chaotic systems is their extreme sensitivity to initial conditions and this sensitivity is not linear but is exponential. The researchers seek to understand how to measure the size of the population in the next generation Xt+1, according to the population in the preceding generation Xt, such a relationship may be expressed in Equation 2.40 (Mojarrad and Ayubi, 2015).

$$X_{t+1} = \mathrm{F}(X_t)$$

Where X_{t+1} is the size of the population in the next generation, Xt is the population in the preceding generation and F represents a particular chaotic mapping function.

Related Works

A growing body of literature highlights various efforts aimed at improving metaheuristic optimization through hybrid strategies, swarm intelligence refinements, and chaos theory applications. Meng et al. (2014) introduced the original Chicken Swarm Optimization (CSO), modeling it after the hierarchical foraging and social dynamics of chickens. Although promising, the algorithm was shown to suffer from early convergence in complex optimization tasks.

To mitigate stagnation, Caponetto et al. (2003) demonstrated that integrating chaotic sequences into evolutionary algorithms could effectively enhance diversity in the population and improve global search performance. Alatas (2010) later applied similar concepts to the Bee Colony Algorithm with successful results.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

Subsequent modifications of CSO have emerged, such as the Modified CSO by Chen et al. (2015), incorporating adaptive factors and learning schemes to address convergence issues. Similarly, Li et al. (2020) proposed a hybrid CSO with adaptive differential mutation, improving robustness in handling global optimization problems.

In broader metaheuristics, Mirjalili (2019) emphasized the role of hybrid and nature-inspired algorithms in evolving neural network training, optimization, and problem-solving. Tian et al. (2017) specifically demonstrated the benefits of chaotic maps in refining particle swarm optimization.

The ECSO method proposed in this paper builds on this foundation by integrating both Gaussian and Tent chaotic maps to simultaneously govern the behaviors of roosters and hens. This approach is inspired by the findings of Mojarrad and Ayubi (2015), who highlighted the exponential sensitivity of chaotic dynamics and their capacity for robust search behavior.

Thus, ECSO represents an evolution in the lineage of chaos-integrated swarm optimization algorithms and contributes to the growing field of adaptive metaheuristic development.

METHODOLOGY

Enhanced Chicken Swarm Optimization (ECSO)

ECSO enhances the standard CSO by integrating chaotic map functions to diversify the search dynamics of roosters and hens. Chaotic systems, such as Gaussian and Tent maps, introduce controlled unpredictability, which promotes better coverage of the search space (Alatas, 2010).

Formulation of ECSO for the system

In standard CSO, there is variation in the food searching capacity of different members of the group. In the update step, the fitness values are updated depending on the food searching capacity of the different members of the group. Food searching capacity of rooster depends on their fitness values and their update formula is captured in the equation below.

$$x_i^{t+1}{}_{,j} = x_i^t{}_{,j} * (1 + Randn(0, \sigma^2))$$

$$\sigma^2 = \begin{cases} 1, & \text{if } f_i \leq f_k \\ e^{\left(\frac{f_k - f_i}{|f_i| + \varepsilon}\right)}, & \text{otherwise}, \mathrm{k} \in [1, \mathrm{N}], \mathrm{k} \neq \mathrm{i} \end{cases}$$

Where $Ran(0, \sigma^2)$ is a gaussian distribution with mean 0 and standard deviation σ^2. ε is used to avoid zero-division-error. k is a rooster's index, f is the fitness value of the corresponding $\boldsymbol{x}$.

Hens follow their group mate roosters in their quest for food. Moreover, there is also a tendency among the chickens to steal the food found by other chickens. The mathematical representation of their update formula is defined in Equation 3.3.

$$x_{i,\mathrm{j}}{}^{t+1} = x_{i,\mathrm{j}}{}^t + S1 \times Rand(x_{r1,\mathrm{j}}{}^t - x_{i,\mathrm{j}}{}^t) + S2 \times Rand(x_{r2,\mathrm{j}}{}^t - x_{i,\mathrm{j}}{}^t)$$

$$S1 = e^{\left(\frac{f_k - f_i}{|f_i| + \varepsilon}\right)}, S2 = {}^{(fr2 - fi)}$$

$$f$$

Where Rand is a uniform random number over [0, 1]. $r1 \in [1, \dots, N]$ is an index of the rooster, $r2 \in [1, \dots, N]$ is an index of the chicken (rooster or hen). Algorithm 3.1 describes the enhanced Chicken Swarm Optimization.

In the enhanced CSO, this study combined two chaotic map functions by using gauss map for rooster updates as shown in Equations 3.4, 3.5 and 3.6, while tent map was used for hen updates as shown in Equations 3.7, 3.8

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

and 3.9 and were used to select optimal weights for CNN. This is to prevent the roosters and the hens from fallen into local optima which could result in premature convergence.

$$Cx_{old} = \frac{\text{mod}(\text{abs}(\text{ini } x_{i,j}^{t+1}, \text{rand}))}{\text{rand}}$$

$$Cx_{new} = \exp(-\alpha * Cx_{old}^2) + \beta$$

$$x_{i,j}{}^{t+1} = sign\ (ini\ x_{i,j}{}^{t+1}) \times Cx_{new} \times rand$$

Where $rand$ is the random value between 0 and 1, $ini\ xit,+j\ 1$ is the primary rooster update calculated with Equation 3.2; $Cxnew$ is the chaotic gauss mapping, where the study considered $\alpha = 4.9\ and\ \beta = -0.58$ and Cx_{new} as x_{k+1} and Cx_{ol} as x_k. Cx_{old} was calculated to transform $ini\ x_i{}^{t,+}{}_j{}^1$ to range [0, 1]. For the hen phase, Equations 3.7, 3.8 and 3.9 were established

$$Chenx_{old} = \frac{\text{mod}(\text{abs}(\text{o}x_{i,j}^{t+1}, (x_{r1,j}^t - x_{i,j}^t)))}{x_{r1,j}^t - x_{i,j}^t}$$

$$Chenx_{new} = \mu \times min\ (Chenx_{old}, 1 - Chenx_{old})$$

$$x_{i,j}^{t+1} = \text{sign}(\text{o}x_{i,j}^t) \times Chenx_{new} \times x_{r1,j}^t - x_{i,j}^t$$

Where $rand$ is the random value between 0 and 1, $ox_{i,j}^{t+1}1$ is the primary hen update calculated with Equation 3.3; $Chenx_{new}$ is the chaotic tent mapping, where the study considered $\mu = 2$ and $Chenx_{new}$ as x_{k+1} and $Chenx_{old}$ as x_k. $Chenx_{old}$ was calculated to transform $ox_{i,j}^{t+1}$ to range [0, 1]. Algorithm 3.1 described the formulated enhanced Chicken Swarm Optimization where step 7 shows the rooster updates and step 8 shows the hen updates.

Fitness Function Formulation

The general formulation of an optimal weight determination problem used in this study is as follows:

Min $\emptyset(y(W_f^d))$

$rosterbest, henbest, chickbest, Wfd$

Subject to: C1: $0 \leq (wt, Fit, rooster, hen, chick, W^d) \leq 1 \qquad W^d \in W_e$

C2: $\text{Fit} = \begin{cases} 1, & \text{if Fit} \leq \overline{\overline{\text{Fit}}} \\ 0 & \text{otherwise Fit} > \overline{\overline{\text{Fit}}} \end{cases}$

Where $wt \in Rn$is the vectors of randomized weights. $\overline{Fit}$ is the mean square error for Fit. The entire state vector is denoted as $y = [wt]$, where wt is the set of the weights of CNN at input, hidden and output layer. The problem was defined on the weight's horizon $W_e = [W_o{}^d W_f{}^d]$. Where W_e consists of original weights of $W_o{}^d$ of y and final weight $W_f{}^d$ selected which is equivalent to optimal weight that was achieved.

The local and global best position weight-dependent control variables $rosterbes$, $henbest$, $chickbest \in R^n$ and possibly the final feature $W_f{}^d$ are decision variables for optimization. The goal of the optimization is to find the optimal set of decision variables to minimize the fitness function$\emptyset$, that is, $\emptyset(y(W_f^d))$.

The search space for finding the optimum is restricted by constraints (C1 and C2), which described an appropriate error fitness and weight parameter requirements respectively, to be fulfilled during determination of optimal weight. The research considered weight constraint C1 and fitness constraint C2. C1 ensured that the values range between 0 and 1. C2 certified that the fitness value for optimal weights was tagged as 1 and other weights was tagged as 0.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

Chaotic Map Integration

- **Rooster Update (Gaussian Map):**

For i = 1:N

If i = rooster Update its solution/location

$$Cx_{old} = \frac{\text{mod}(\text{abs}(\text{ini}\, x_{i,j}^{t+1}, \text{rand}))}{\text{rand}}$$

$$Cx_{new} = \exp(-\alpha * Cx_{old}^2) + \beta$$

$$x_{i,j}^{t+1} = sign\,(ini\, x_{i,j}^{t+1}) \times Cx_{new} \times rand$$

End if

- **Hen Update (Tent Map):**

If *i = hen* Update its solution/location;

$$Chenx_{old} = \frac{\text{mod}(\text{abs}(\text{ox}_{i,j}^{t+1}, (x_{r1,j}^{t} - x_{i,j}^{t})))}{x_{r1,j}^{t} - x_{i,j}^{t}}$$

$$Chenx_{new} = \mu \times min\,(Chenx_{old}, 1 - Chenx_{old})$$

$$x_{i,j}^{t+1} = \text{sign}(\text{ox}_{i,j}^{t}) \times Chenx_{new} \times x_{r1,j}^{t} - x_{i,j}^{t}$$

End if

ECSO Algorithm Overview

1. Initialize population and assign roles.
2. Evaluate fitness and sort population.
3. For each generation:
    - Update roosters using Gaussian chaotic map.
    - Update hens using Tent chaotic map.
    - Update chicks using standard CSO rules.
4. Re-rank and reassign roles every GG generations.
5. Store and update the best solution found.

Algorithm 3.1: Enhanced Chicken Swarm Optimization

Step 1: Input: Set of initial weight parameters W= $\{w_1, , \ldots w_p\}$

Predefined swarm size: $\boldsymbol{N_c}$

Number of dimensions of a chicken: $\boldsymbol{D} = \boldsymbol{q}$

Step 2: Output: Optimal weight parameters $\{wopt_I, wopt_H, wopt_c\}$

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

Step 3: Initialize chickens Ck= [RN=CN =MN=HN] $\forall \boldsymbol{i}, \boldsymbol{j}, \mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{N_c}, \quad \mathbf{1} \leq \boldsymbol{j} \leq \boldsymbol{D} =$

$\boldsymbol{q}$, number of CHs, G (maximum generation)

$\boldsymbol{x_{i,j}}(\mathbf{0}) = (\boldsymbol{x_{i,j}}(\mathbf{0}), \boldsymbol{y_{i,j}}(\mathbf{0}))$ */* position of the weights */*

Step 4: Evaluate the N chickens' fitness values (Ck).

Step 5: t=0;

Step 6: ***While*** (t < G)

i. ***If*** $(t \ mod \ G = 0)$

c. Rank the chickens' fitness values and establish a hierarchal order in the swarm;

Fitness values = $(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} \Delta(W_{i^{m,}{}^{n}}) ((x_i) - (x_j))$

Where ^{t} represent the s at i=1,2, …, n and k=2,3, …, m

Where $(W_{i^{m,j^{n}}})((x_i) - (x_j)$ is the change in weight of input, hidden and output layers x along the row *n* and column *m*

d. Divide the swarm into different groups, and determine the relationship between the chicks and mother hens in a group;

End if

Step 7: ***For*** i = 1:N

If i = rooster Update its solution/location

$$Cx_{old} = \frac{\text{mod}(\text{abs}(\text{ini } x_{i,j}^{t+1}, \text{rand}))}{\text{rand}}$$

$$Cx_{new} = \exp(-\alpha * Cx_{old}{}^{2}) + \beta$$

$$x_{i,j}{}^{t+1} = sign\ (ini\ x_{i,j}{}^{t+1}) \times Cx_{new} \times rand$$

End if

Step 8: ***If*** *i = hen* Update its solution/location;

$$Chenx_{old} = \frac{\text{mod}(\text{abs}(\text{o}x_{i,j}^{t+1}, (x_{r1,j}^{t} - x_{i,j}^{t})))}{x_{r1,j}^{t} - x_{i,j}^{t}}$$

$$Chenx_{new} = \mu \times min\ (Chenx_{old}, 1 - Chenx_{old})$$

$$x_{i,j}^{t+1} = \text{sign}(\text{o}x_{i,j}^{t}) \times Chenx_{new} \times x_{r1,j}^{t} - x_{i,j}^{t}$$

End if

Step 9: ***If*** *i = chick* Update its solution/location

$$x_{i,j}^{t+1} = x_{i,j}^{t} \times FL(x_{m,j}^{t} - x_{i,j}^{t})$$

Where $x_{m,j}^{t}$, stands for the position of the *ith* chick's mother ($m \in [1, N]$). *F*L($FL \in (0, 2)$) is a parameter *End if*

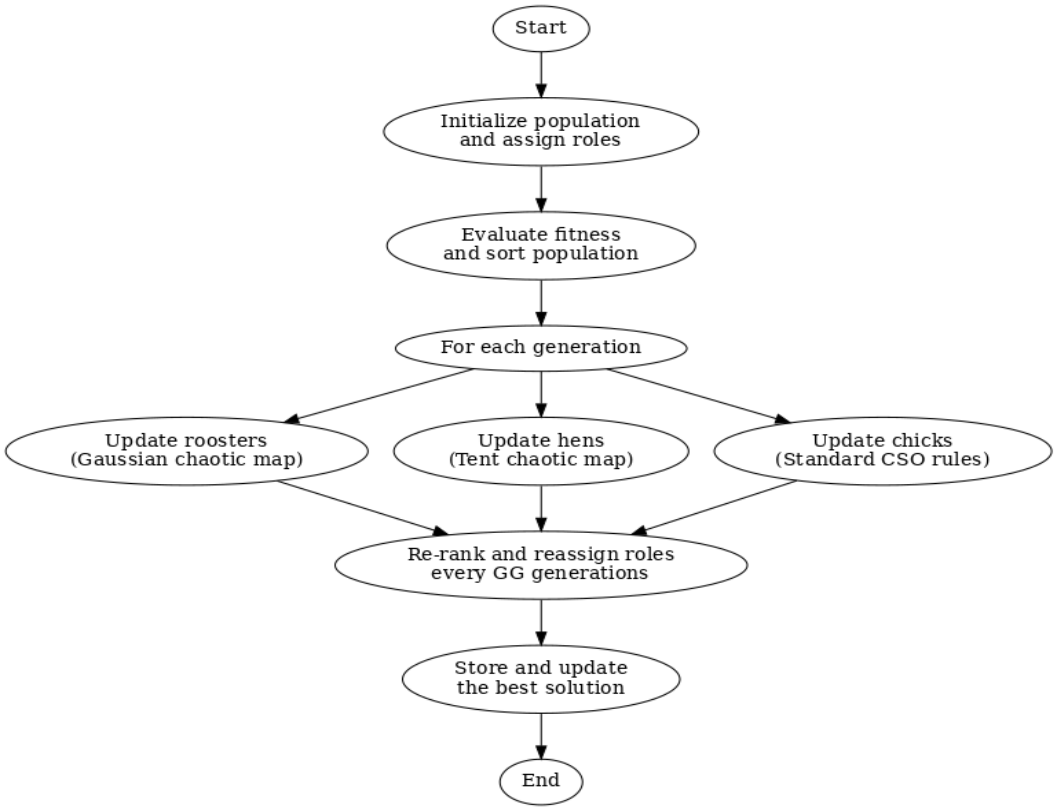
ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

Step 10: Evaluate the new solution;

Step 11: If the new solution is better than it's previous one, update it;

End for End while

Flowchart 3.1 : Enhanced Chicken Swarm Optimization



RESULTS AND DISCUSSION

This section presents the comparative performance analysis of the developed Enhanced Chicken Swarm Optimization (ECSO) algorithm, augmented with Gaussian and Tent chaotic maps, against both the baseline Convolutional Neural Network (CNN) and the standard Chicken Swarm Optimized CNN (CSO-CNN). The evaluation was conducted on two categories of forensic handwriting datasets: original and forged samples. Performance metrics include classification accuracy and recognition time, implemented using MATLAB R2020a.

A. Performance on Original Handwritten Documents

As shown in Table I and Figure 1, the baseline CNN model achieved an accuracy of 82.14% on the original handwritten dataset, with a recognition time of 77.47 seconds. The CSO-CNN variant improved upon this baseline by reaching an accuracy of 86.43% with a reduced processing time of 52.21 seconds. However, the proposed ECSO-CNN model delivered the best performance, achieving an accuracy of 92.14% and the lowest recognition time of 26.18 seconds.

This progression indicates a 4.29% accuracy gain from CNN to CSO-CNN, and an additional 5.71% gain from CSO-CNN to ECSO-CNN. In terms of efficiency, CSO-CNN reduced the recognition time by 32.6% compared to CNN, while ECSO-CNN achieved a further 49.8% reduction relative to CSO-CNN. The improvement is largely attributed to the use of chaotic maps in ECSO, which enhance the optimizer's ability to explore the search space, avoid local optima, and converge to better CNN hyperparameters.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

B. Performance on Forged Handwritten Documents

For the forged handwriting dataset, which contains deceptive alterations, Table I and Figure 2 below shows the CNN baseline achieved 83.57% accuracy with a recognition time of 72.24 seconds. The CSO-CNN model improved this to 89.29% accuracy with a processing time of 47.27 seconds. The ECSO-CNN once again outperformed both, attaining an accuracy of 93.57% and reducing the recognition time to 26.86 seconds.

These results represent a 4.28% increase in accuracy from CSO-CNN to ECSO-CNN and a 10% improvement over the baseline CNN. In terms of computational time, ECSO-CNN reduced processing time by 43.2% compared to CSO-CNN and by 62.8% relative to CNN. This confirms ECSO-CNN's robustness in handling high intra-class variability and its effectiveness in recognizing forged signatures.

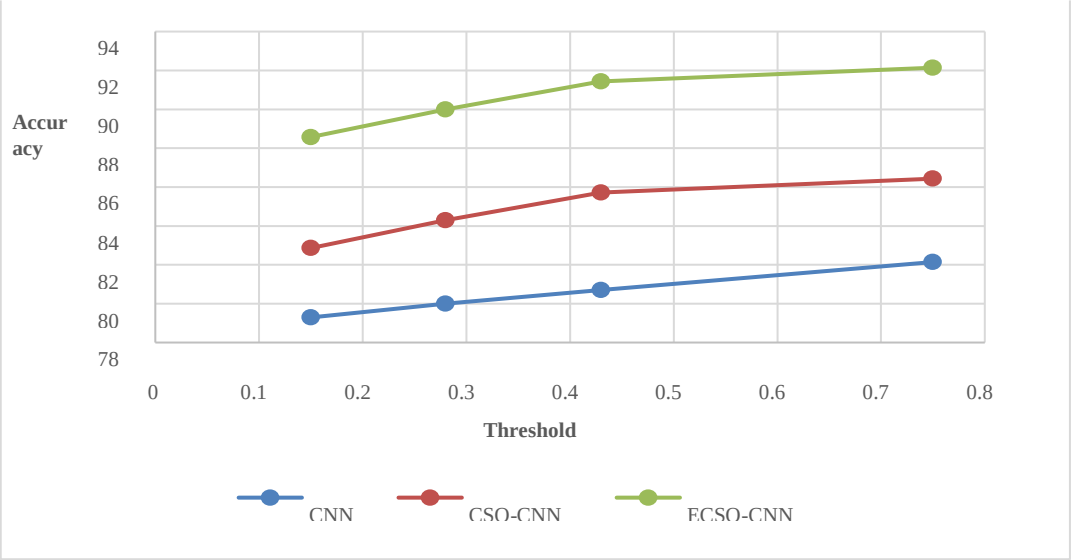
C. Summary of Findings

Across both dataset categories, the empirical results demonstrate a consistent performance ranking: ECSO-CNN outperforms both CSO-CNN and the baseline CNN in terms of classification accuracy and recognition time.

The empirical results from both datasets confirm that by using the developed ECSO as optimizer for CNN framework, the optimized CNN consistently outperforms the conventional CNN in terms of both accuracy and efficiency**.** The use of chaotic dynamics within the CSO algorithm significantly improves search diversity, mitigates premature convergence, and yields better-tuned CNN architectures.

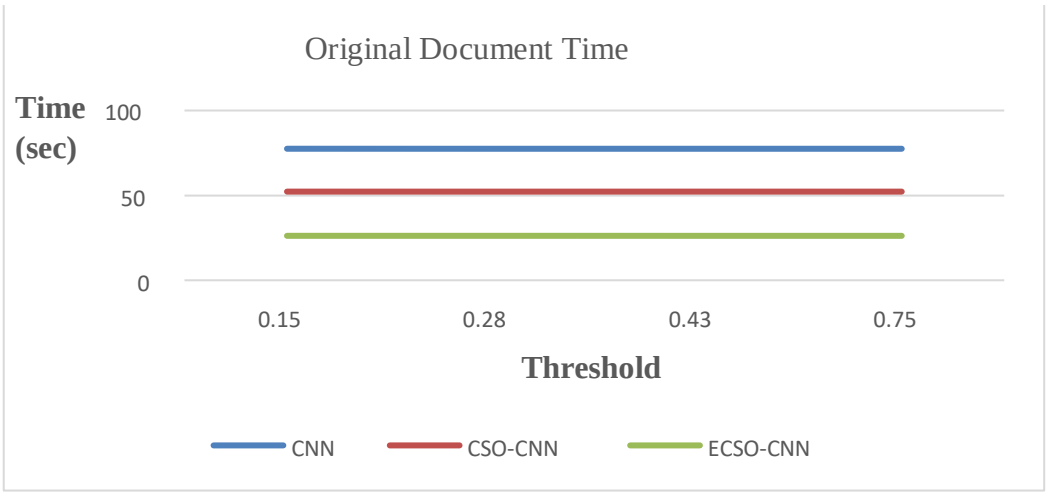
Table I: Comparison of CNN, CNN and ECSO-CNN on Forensic Handwritten Datasets

Dataset Type	Algorithm	Accuracy (%)	Time (sec)
Original	CNN	82.14	77.47
	CSO-CNN	86.43	52.21
	ECSO-CNN	92.14	26.18
Forged	CNN	83.57	72.24
	CSO-CNN	89.29	47.27
	ECSO-CNN	93.57	26.86



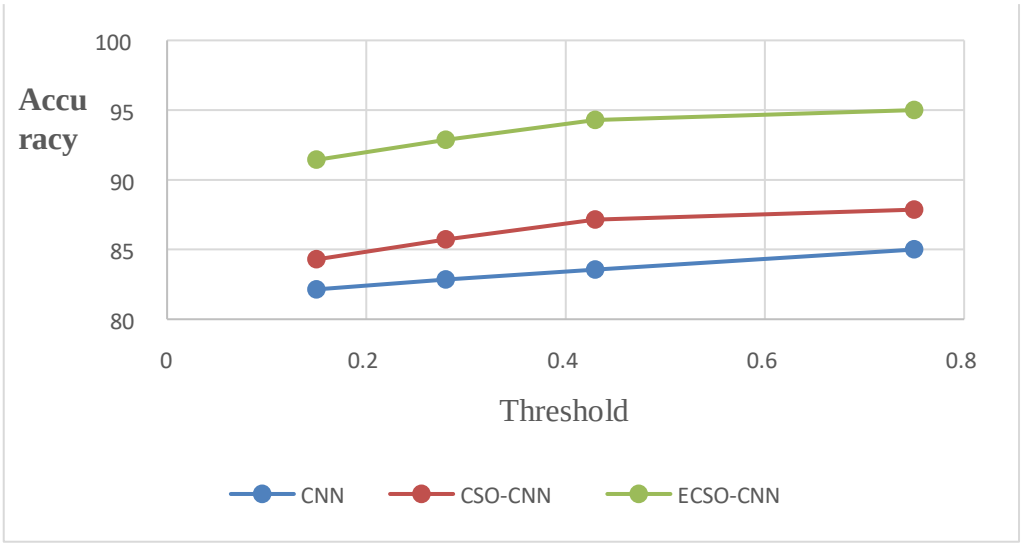
(a) Accuracy for Original Dataset

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

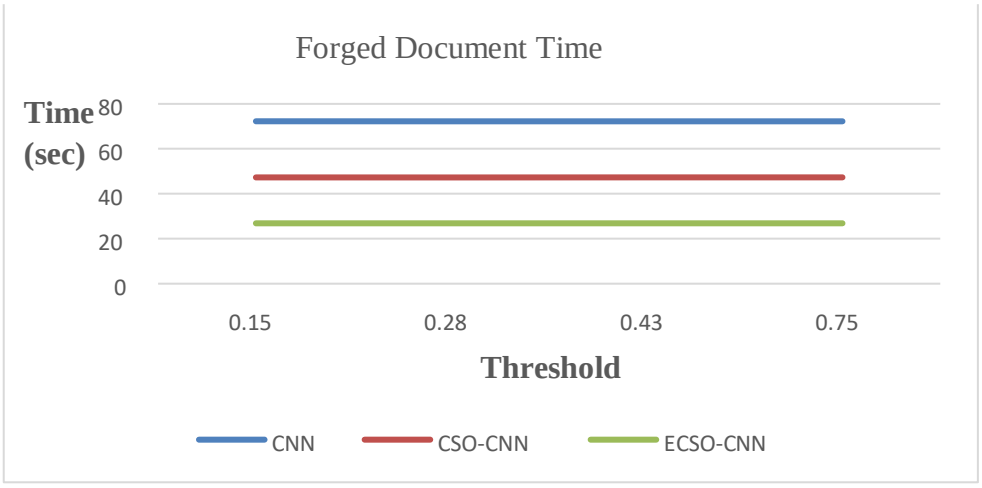


(b) Recognition Time for Original Dataset

Figure 1 : Graph showing Accuracy and Recognition Time for Original Dataset



(a) Accuracy for Forged Dataset



(b) Recognition Time for Forged Dataset

Figure 2: Graph showing Accuracy and Recognition time for Forged Datasets

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

CONCLUSION

This paper presented an Enhanced Chicken Swarm Optimization (ECSO) algorithm that integrates Gaussian and Tent chaotic map functions to improve global search diversity and mitigate premature convergence. The ECSO was employed to optimize a Convolutional Neural Network (CNN) and evaluated on forensic handwriting datasets. specifically, original and forged samples. Performance was benchmarked against both the baseline CNN and the standard CSO-optimized CNN (CSO-CNN).

Experimental results demonstrated that the ECSO-optimized CNN consistently outperformed both the baseline CNN and CSO-optimized CNN models in terms of classification accuracy and computational efficiency. On original handwriting data, ECSO-CNN achieved an accuracy improvement of 5.71% over CSO-optimized CNN and 12.2% over the baseline, while recognition time was reduced by nearly 50% relative to CSO-optimized CNN and over 66% compared to the CNN. Similar performance gains were observed on forged datasets, affirming the robustness and adaptability of ECSO in complex classification scenarios.

These results underscore the effectiveness of integrating chaotic dynamics within swarm intelligence frameworks for optimizing deep learning models. The ECSO algorithm offers a significant advancement over standard CSO by enhancing convergence behavior, expanding solution space exploration, and yielding superior CNN architectures.

The findings affirm the practical value of ECSO-CNN in real-world forensic handwriting identification and related security-sensitive applications. Future research may explore the adaptation of ECSO to other deep learning architectures as well as its deployment in broader biometric and document authentication domains.

REFERENCES

1. B. Alatas, “Chaotic bee colony algorithms for global numerical optimization,” *Expert Syst. Appl.*, vol. 37, no. 8, pp. 5682–5687, 2010.
2. M. Bulacu and L. Schomaker, “Text-independent writer identification and verification using textural and allographic features,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 4, pp. 701–717, Apr. 2007.
3. R. Caponetto, L. Fortuna, S. Fazzino, and M. G. Xibilia, “Chaotic sequences to improve the performance of evolutionary algorithms,” *IEEE Trans. Evol. Comput.*, vol. 7, no. 3, pp. 289–304, Jun. 2003.
4. H. Chen, X. Guo, and Y. Hu, “Modified Chicken Swarm Optimization Algorithm for function optimization problems,” *Math. Probl. Eng.*, vol. 2015, Article ID 409627, pp. 1–12, 2015.
5. S. Deb, M. C. Sahoo, D. Mishra, and S. Sharma, “A review on Chicken Swarm Optimization and its applications,” *Appl. Soft Comput.*, vol. 92, p. 106281, 2020.
6. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.
7. Z. He, H. Wang, S. Wang, and B. Li, “Handwritten signature verification based on deep convolutional neural networks,” *Neural Comput. Appl.*, vol. 32, pp. 14813–14822, 2020.
8. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
9. S. Li, Z. Wang, and B. Xue, “A hybrid Chicken Swarm Optimization with adaptive differential mutation,” *Inf. Sci.*, vol. 512, pp. 896–922, 2020.
10. X. Meng, L. Liu, X. Gao, and H. Zhang, “A new bio-inspired algorithm: Chicken Swarm Optimization,” *Int. J. Bio-Inspired Comput.*, vol. 6, no. 4, pp. 200–211, 2014.
11. M. H. Mojarrad and P. Ayubi, “Particle swarm optimisation with chaotic velocity clamping (CVC-PSO),” in *Proc. 2015 7th Conf. Inf. Knowl. Technol. (IKT)*, Urmia, Iran, 2015, pp. 1–5.
12. S. Mirjalili, *Evolutionary Algorithms and Neural Networks*, Springer, Cham, 2019.
13. S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014.
14. Y. Shi, J. Li, and C. Liu, “Chicken swarm optimization algorithm and its application in PID controller tuning,” *Appl. Soft Comput.*, vol. 17, pp. 419–430, 2014.
15. R. Storn and K. Price, “Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces,” *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume X Issue VII July 2025

16. Y. Tian, L. Wang, and Y. Chen, "An improved chaotic map-based particle swarm optimization algorithm," *Soft Comput.*, vol. 21, no. 15, pp. 4451–4463, 2017.
17. J. Wang, H. Zhao, and X. Wang, "Improved chicken swarm optimization algorithm for solving global optimization problems," *J. Comput. Sci.*, vol. 26, pp. 21–31, 2018.
18. J. Xue and B. Shen, "A novel swarm intelligence optimization approach: Human learning optimization algorithm," *Soft Comput.*, vol. 20, no. 10, pp. 3797–3815, 2016.
19. X. S. Yang, *Nature-Inspired Optimization Algorithms*, Elsevier, London, 2014.
20. J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, 1995, pp. 1942–1948.