



Performance Evaluation of Selected Chaotic Maps for Enhancing Chicken Swarm Optimization for Handwritten Document Identification

Emmanuel Blessing Olaniyi¹, Stephen Olatunde Olabiyisi² and Oluyinka Titilayo Adedeji³

^{1,2}Department of Computer Science, Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria.

³Department of Information Systems, Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria

Article history:

Received: 28/06/2026

Accepted: 15/08/2026

Published: 30/09/2026

Keywords:

Chicken Swarm Optimization, Chaotic Maps, Feature Selection, Handwritten Document Identification, Writer Recognition, Circle+CSO, Gauss+CSO, and Henon+CSO, IAM Dataset.

*Corresponding Author:

Emmanuel Blessing Olaniyi

Abstract

Handwritten document identification (HDI) was an important task in forensic analysis, digital archiving, and automated recognition systems. However, variations in handwriting styles, noise, and document quality made accurate identification challenging. Although Chicken Swarm Optimization (CSO) had potential for solving complex optimization problems, it could suffer from premature convergence, poor population diversity, and reduced performance in complex search spaces. This study therefore addressed the need to evaluate the comparative performance of three chaos-enhanced CSO variants (Circle+CSO, Gauss+CSO, and Henon+CSO) for HDI using the Institute of Computer Science and Applied Mathematics (IAM) dataset. The methodology involves designing and implementing three chaos-enhanced CSO algorithms where chaotic sequences replace random number generators for population initialization, rooster update mechanisms, hen following behaviors, chick movements, and perturbation strategies. The IAM Handwriting Database containing 500 samples with 9 features (stroke width, slant angle, character spacing, texture 1 and 2 features, and CNN 1 and 2 features) across three writers is preprocessed. The models (Circle+CSO, Gauss+CSO, and Henon+CSO), were implemented in python, evaluated and compared based on accuracy, precision, recall, F1 score, converging rate and computational time. Henon+CSO achieved 93.12% accuracy, 90.21% precision, 90.34% recall, 90.27% F1 score, converging rate 41 iterations, and 2.45s computational time. Circle+CSO attained 90.34% accuracy, 90.21% precision, 90.34% recall, 90.27% F1 score, converging rate 58 iterations, and 2.38s computational time. While Gauss+CSO achieved 91.89% accuracy, 91.78% precision, 91.98% recall, 91.83 F1 score, converging rate 47 iterations, and 2.52s computational time. This research demonstrates that integrating chaotic maps into the CSO framework improves handwritten writer identification performance. The study contributes to knowledge by systematically comparing Circle, Gauss, and Henon chaotic maps and identifying their respective strengths for different applications. Henon+CSO is the most suitable for high-accuracy and fast-convergence applications, Circle+CSO for computationally time-critical applications, and Gauss+CSO for maintaining population diversity and preventing stagnation. The study also identifies effective feature subsets, providing a useful foundation for improved handwritten document identification.

Original Research Article

Copyright © 2026 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution-Noncommercial 4.0 International License (CC BY-NC 4.0)

How to cite this article: Emmanuel Blessing Olaniyi, Stephen Olatunde Olabiyisi, & Oluyinka Titilayo Adedeji. (2026). Performance evaluation of selected chaotic maps for enhancing chicken swarm optimization for handwritten document identification. EIRA Journal of Multidisciplinary Research and Development, 2(5). 12-27.

1. Introduction

Handwritten document identification is a significant research area that cuts across computer vision, pattern recognition, artificial intelligence, and document analysis. It focuses on the examination and analysis of handwritten documents to identify, distinguish, and characterize different handwriting patterns. The importance of handwritten document identification is reflected in its wide range of applications,

including writer identification, document authentication, forensic investigation, digital archiving, and the preservation of historical documents. In writer identification, the primary objective is to determine the distinctive characteristics of a person's handwriting and use these characteristics to differentiate one writer from another. This process is particularly important in forensic applications, where

handwriting characteristics may provide useful evidence for establishing the possible source or authorship of a handwritten document (Tan et al., 2009). However, the identification of handwritten documents remains a challenging problem because handwriting is naturally variable. Differences in individual writing styles, writing conditions, document quality, noise, degradation, and the complex visual structures of handwritten patterns can significantly affect the accuracy of identification systems (Agrawal et al., 2024; Hussain et al., 2015).

The complexity of handwritten patterns makes the selection of appropriate features and parameters an important component of an effective identification system. Handwritten documents contain numerous visual characteristics, and selecting the most informative features can improve the ability of a classification model to distinguish between different writers. However, feature selection and parameter optimization may involve searching through large and complex solution spaces. Consequently, optimization techniques are increasingly important in developing efficient handwritten document identification systems. Metaheuristic optimization algorithms have attracted considerable research interest because they can provide effective strategies for solving complex optimization problems without requiring exact mathematical solutions. These algorithms generally employ mechanisms that allow them to explore different regions of the search space while exploiting promising solutions that have already been discovered.

One of the population-based metaheuristic algorithms that can be applied to complex optimization problems is Chicken Swarm Optimization (CSO). CSO is a swarm intelligence optimization technique inspired by the social hierarchy and foraging behaviour of chickens. The algorithm divides its population into different groups, particularly roosters, hens, and chicks, with each group exhibiting different movement behaviours during the optimization process. Roosters generally represent individuals with stronger search capabilities, while hens and chicks follow specific behavioural relationships within the population. Through these hierarchical interactions, CSO attempts to achieve a balance between exploration of the search space and exploitation of promising solutions (Meng et al., 2014). This makes CSO potentially useful for optimization tasks in which the optimal solution is difficult to determine using conventional mathematical approaches.

Despite the potential advantages of CSO, standard versions of the algorithm may encounter limitations that are common to population-based metaheuristic methods. These limitations include premature convergence, reduction in population diversity, and becoming trapped in local optima. Premature convergence occurs when the population becomes concentrated around a solution before sufficiently exploring other regions of the search space. Similarly, a loss of diversity can reduce the ability of the algorithm to discover alternative and potentially better solutions. These limitations can become

more significant when CSO is applied to complex, high-dimensional, or multimodal optimization problems. Researchers have therefore explored different mechanisms for improving the exploration and exploitation capabilities of metaheuristic algorithms and reducing the likelihood of premature convergence (Gagnon et al., 2021).

One approach that has received attention for improving metaheuristic optimization is the incorporation of chaos theory. Chaotic maps are mathematical systems capable of generating deterministic sequences with nonlinear and highly sensitive behaviour. When integrated into optimization algorithms, these sequences can be used to introduce additional variation into the population, influence the generation of candidate solutions, or modify the search process. The purpose of incorporating chaotic behaviour is not to make the optimization process completely random but rather to introduce structured variation that may help the algorithm explore the search space more effectively. Several chaotic maps have been investigated in the context of metaheuristic optimization, including Logistic, Gauss, Circle, Sine, Tent, and Chebyshev maps (Saremi et al., 2014; Gagnon et al., 2021). Their use provides alternative mechanisms for generating search sequences and maintaining variation within the population.

The Circle, Gauss, and Hénon maps are of particular interest because they possess different nonlinear characteristics that may influence optimization behaviour in different ways. The Circle and Gauss maps have been examined in chaotic optimization and metaheuristic search, while the Hénon map is associated with a two-dimensional discrete chaotic system capable of producing complex trajectories. The distinct properties of these chaotic maps create an opportunity to investigate whether different forms of chaotic behaviour can improve the performance of an optimization algorithm. In particular, the use of chaotic sequences may help maintain population diversity and reduce the possibility that the search process becomes restricted to a small region of the solution space. However, chaotic optimization does not necessarily produce the same level of improvement for every problem. Previous research indicates that the effectiveness of a chaotic map can depend on factors such as the optimization problem being solved, the underlying metaheuristic algorithm, and the particular chaotic sequence incorporated into the algorithm (Saremi et al., 2014; Gagnon et al., 2021).

The integration of chaotic maps with Chicken Swarm Optimization therefore provides a research opportunity for improving the search behaviour of the standard CSO algorithm. By introducing Circle, Gauss, or Hénon chaotic sequences into CSO, it is possible to investigate whether the resulting variants can maintain greater population diversity, improve exploration of the search space, and enhance the exploitation of promising solutions. Such modifications may be particularly relevant to handwritten document identification, where the variability and complexity of handwriting make effective feature selection and parameter

optimization important. Since handwriting characteristics can differ considerably between writers, an optimization algorithm must be capable of identifying useful features while avoiding inefficient or premature search behaviour (Agrawal et al., 2024; Hussain et al., 2015). Consequently, evaluating chaos-enhanced CSO variants in handwritten document identification can provide useful evidence about the relationship between optimization strategies and recognition performance. The effectiveness of the different variants can be investigated using appropriate performance measures, including classification accuracy, convergence behaviour, computational cost, and other relevant evaluation metrics. Comparing the standard CSO with its chaos-enhanced variants can help determine whether the incorporation of chaotic maps produces measurable improvements in the optimization process. More importantly, examining different chaotic maps can establish whether the characteristics of a particular chaotic sequence influence the ability of CSO to solve the feature-selection or optimization problem associated with handwritten document identification. Overall, the study of chaos-enhanced Chicken Swarm Optimization provides a systematic approach to investigating how advanced metaheuristic strategies can be applied to the challenges associated with handwritten document identification, while also contributing to the broader research on improving exploration, exploitation, diversity, and convergence in population-based optimization algorithms.

2.0 Review of Related Works

Several empirical studies have investigated handwritten document identification, feature selection, Chicken Swarm Optimization (CSO), and chaos-enhanced metaheuristic algorithms. Christlein et al. (2017) examined writer identification using offline handwriting images and different feature representations. Their findings showed that handwriting contains discriminative visual characteristics that can be used to distinguish writers. Similarly, Fiel et al. (2019) investigated writer identification and verification and demonstrated that writer-specific information can be extracted directly from handwriting images without converting the handwriting into machine-readable text. These studies establish the feasibility of using image-based handwriting characteristics for writer identification. However, they did not investigate the use of chaos-enhanced CSO for selecting discriminative handwriting features.

Balaha and Ali (2020) applied deep-learning techniques to handwriting recognition and demonstrated that convolutional neural networks can learn complex visual representations from handwriting images. Their findings indicate that learned features can effectively represent stroke patterns, shapes, and other handwriting characteristics. However, the high dimensionality of automatically extracted features can increase computational requirements and may introduce redundant information. This creates a need for effective feature-selection techniques that can retain informative features while reducing dimensionality.

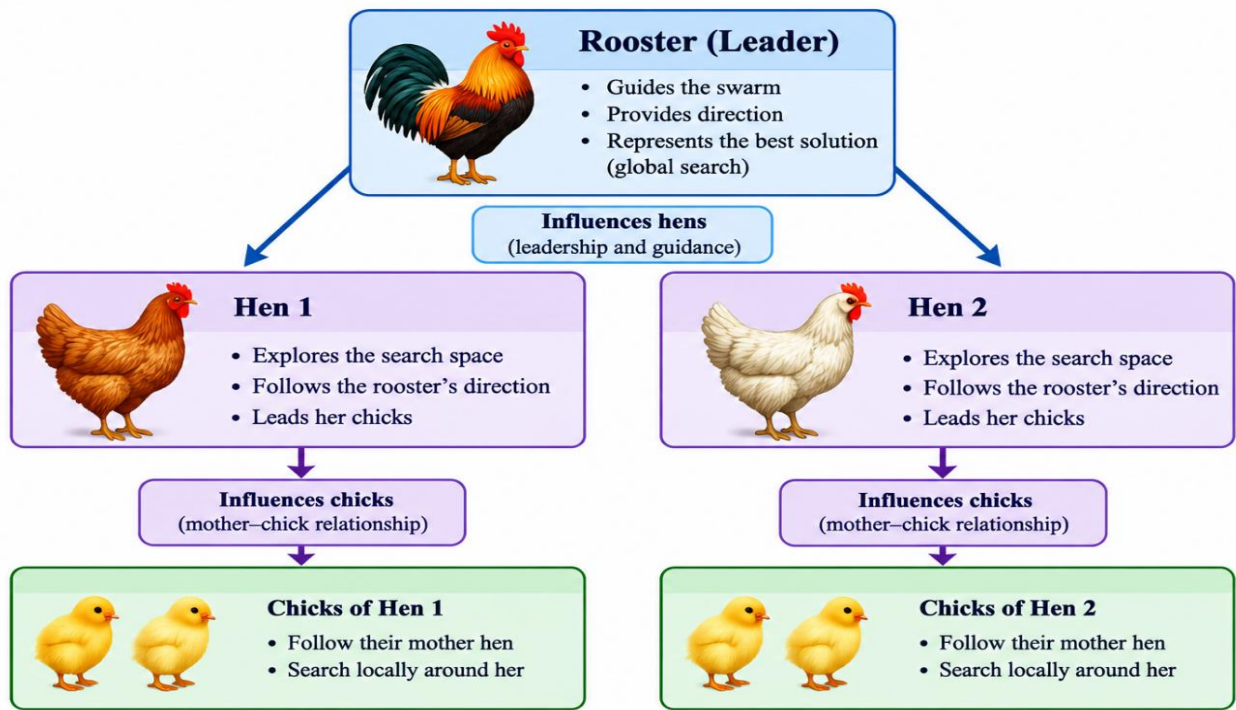


Figure 2.1: Hierarchical structure of Chicken Swarm Optimization

Meng et al. (2014) introduced Chicken Swarm Optimization and evaluated its performance on benchmark optimization problems. The algorithm models the social hierarchy of

chickens through roosters, hens, and chicks, allowing different agents to perform different search behaviours. The study demonstrated the ability of CSO to search complex

optimization spaces. Tharwat et al. (2016) subsequently demonstrated the applicability of CSO to feature-selection problems, showing that CSO can search for useful feature subsets for classification. These findings provide empirical support for applying CSO to feature selection in handwritten document identification. Nevertheless, conventional CSO may experience reduced population diversity and premature convergence, particularly when applied to complex search spaces.

Research on chaotic metaheuristics has proposed the use of chaotic maps to address some of these limitations. Arora and Anand (2019) reviewed the integration of chaotic maps into metaheuristic algorithms and reported that chaotic sequences can be used for population initialization, parameter adaptation, and search-agent movement. Their findings suggest that chaos can increase search diversity and improve exploration. Gagnon et al. (2021) further investigated chaotic maps in optimization and found that the effectiveness of a chaotic map varies according to the optimization problem. Their findings are important because they indicate that no particular chaotic map can be assumed to perform optimally across all problems.

More recent research has continued to examine improvements to CSO and chaos-enhanced metaheuristics. Gharehchopogh et al. (2024) reviewed CSO variants and identified population diversity, premature convergence, and local-optimum stagnation as continuing challenges in CSO-based optimization. The study highlighted modifications such as hybridisation, adaptive mechanisms, and improved search strategies as possible approaches for addressing these limitations. Dockeroglu et al. (2025) similarly reported increasing interest in chaos-enhanced metaheuristics, noting that chaotic mechanisms can improve search diversity and help reduce stagnation. However, their effectiveness depends on the chaotic map, its integration into the optimization process, and the characteristics of the problem being solved.

Studies have also begun to apply CSO and its improved forms to handwriting-related problems. Mugisha et al. (2024) investigated an improved CSO approach for handwritten document-image enhancement, demonstrating the relevance of CSO-based optimization to handwriting-image processing. Mugisha et al. (2025) further investigated chaos-enhanced CSO in a handwriting-recognition framework and reported improved performance compared with baseline approaches. These studies provide important evidence that CSO and chaotic mechanisms can be applied to handwriting-related tasks. However, they do not provide sufficient comparative evidence on the performance of different chaotic maps within a common CSO framework for handwritten document identification.

The findings of Olatunji et al. (2025) align with broader research trends in chaos-enhanced optimization algorithms for image and handwriting recognition tasks. Also, Okunlola et al (2026) integrated chaotic maps into the Particle Swarm

Optimisation (PSO) algorithm to enhance population diversity, prevent premature convergence and improve segmentation quality and Moronkeji et al. (2026) evaluated three chaos-enhanced PSO (CE-PSO) variants, Logistic+PSO, Sinusoidal+PSO and Gaussian+PSO, for image segmentation using multilevel thresholding on the Berkeley Segmentation Dataset (BSDS500). Amusan et al. (2026) evaluated three chaos-enhanced PSO (CE-PSO) variants, namely Tent+PSO, Sinusoidal+PSO, and Circle+PSO, for image segmentation through multilevel thresholding on the Berkeley Segmentation Dataset (BSDS500). In the same direction, John (2026) evaluated three chaos-enhanced PSO (CE-PSO) variants, Logistic+PSO, Sinusoidal+PSO and Circle+PSO, for image segmentation using the Apple Image Dataset obtained from Kaggle. The study established that Sinusoidal+PSO offers the best accuracy for high-precision applications such as agricultural image analysis and automated fruit inspection, Circle+PSO offers the best computational efficiency for real-time, time-critical vision systems, and Logistic+PSO offers a computationally simple, general-purpose baseline. Furthermore, Olabiyisi et al. (2026) evaluated the performance of three chaos-enhanced CSO variants (Chebyshev+CSO, Lorenz+CSO, and Henon+CSO) for HDI using the Institut für Informatik und Angewandte Mathematik (IAM) dataset

Overall, previous empirical studies establish that handwriting images contain discriminative characteristics, CSO can be applied to feature-selection problems, and chaotic maps can potentially improve the diversity and search behaviour of metaheuristic algorithms. However, limited empirical evidence exists on the comparative application of Circle, Gauss, and Hénon chaotic maps within the same CSO framework for handwritten document identification. In addition, previous studies have often concentrated primarily on classification accuracy rather than jointly considering accuracy, precision, recall, F1-score, feature reduction, convergence behaviour, stability, and computational time.

The present study therefore addresses this gap by implementing Circle+CSO, Gauss+CSO, and Hénon+CSO under the same experimental conditions for handwritten document identification. The comparative evaluation is expected to provide empirical evidence on how the choice of chaotic map influences the feature-selection and classification performance of CSO.

3.0 Methodology

The research framework provides a structured procedure for developing and evaluating chaos-enhanced Chicken Swarm Optimization (CSO) algorithms for handwritten character recognition. The methodology is divided into five interconnected phases, where the output of one phase serves as the input for the next. The overall research methodology follows a systematic experimental design as illustrated in Figure 3.1. The framework consists of six main phases:

- | | |
|--------------------------------------|---|
| i. Data Acquisition | iv. Chaos-Enhanced CSO Algorithm Design |
| ii. Feature Extraction and Selection | v. Model Training and Optimization |
| iii. Chaotic Map Integration | vi. Performance Evaluation |

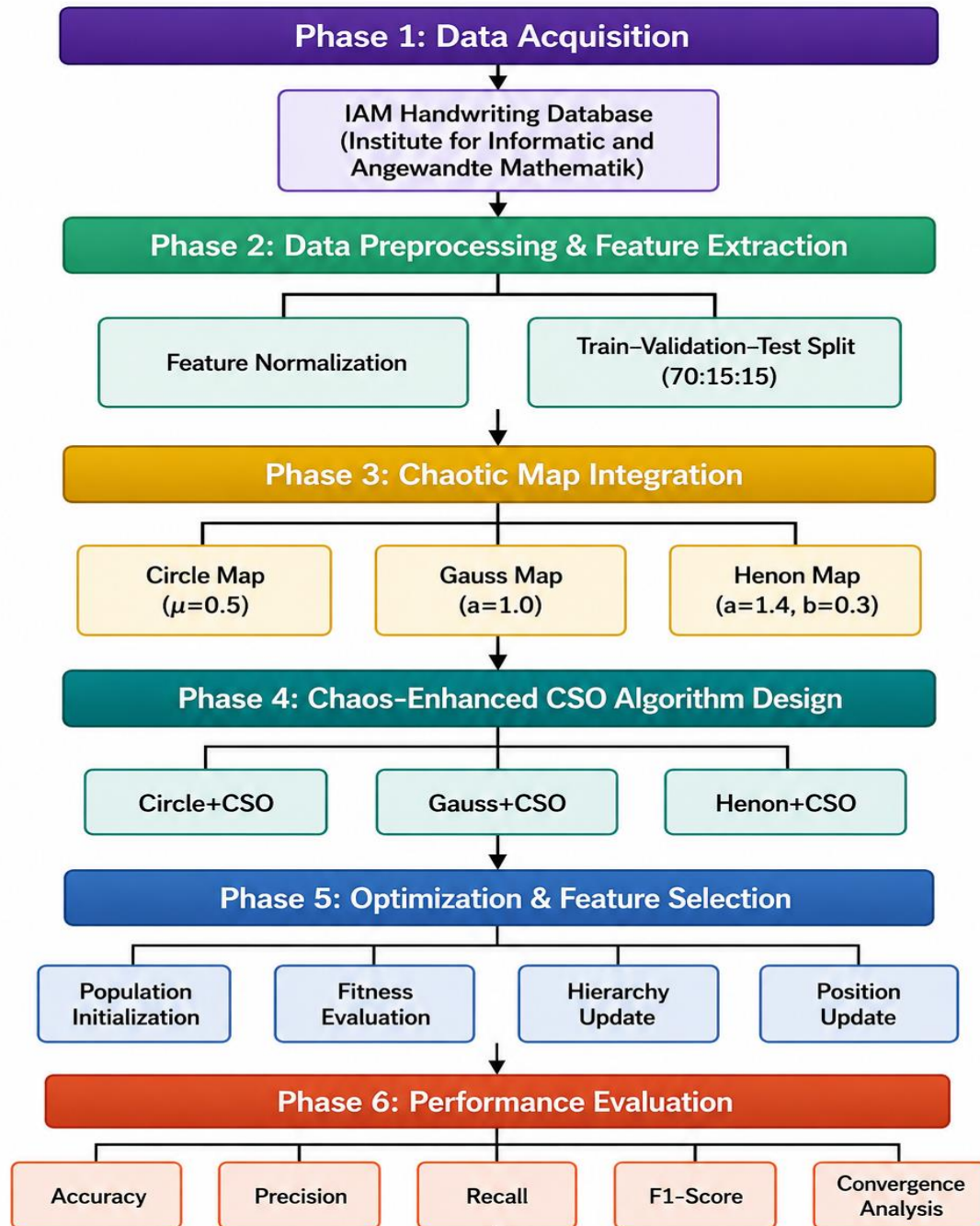


Figure 3.1: Overall Research Framework

3.1 Data Acquisition

In handwritten text recognition research, data acquisition involves obtaining a dataset that contains diverse handwriting samples along with their corresponding ground-truth annotations. Such datasets enable the model to learn handwriting patterns and accurately recognize textual information from handwritten documents.

For this study, a publicly available benchmark dataset was selected to ensure the reliability, reproducibility, and comparability of the experimental results. The acquired dataset provides a large collection of handwritten text samples from multiple writers, allowing the proposed model

to be trained and evaluated under varying handwriting styles and conditions.

3.1.1 IAM handwritten database

The first stage involves collecting the dataset that will be used for experimentation.

The study uses the IAM Handwriting Database, a widely accepted benchmark dataset for handwritten text recognition research. It contains handwritten text samples collected from numerous writers, providing variations in writing styles, character shapes, and stroke patterns.

The Institut für Informatik und Angewandte Mathematik

(IAM) Handwritten Database is utilized for this research. The IAM database is a widely recognized benchmark for handwritten document analysis, containing handwritten English text samples from multiple writers.

Dataset Characteristics:

- Source: Institut für Informatik und Angewandte Mathematik, University of Bern
- Number of Writers: 3 distinct writers (extended to 500 samples)
- Total Samples: 500 handwritten document samples
- Features: 9 extracted features per sample
- Classes: writer identification (Writer ID: 1, 2, 3)
- Data type: numerical continuous features

3.1.2 Feature description

Here is a breakdown of what each feature actually means and how they are used.

i. Geometric and Structural Features (F1 – F3)

These are traditional, intuitive features that mimic how a human graphologist looks at handwriting. They are usually measured in pixels or degrees.

- F1: Stroke_Width (0.40 - 0.65): Measures how thick or thin the pen lines are on average. Heavy pressure results in a higher value, while a light touch results in a lower value.
- F2: Slant_Angle (8 - 22 degrees): Measures the tilt of the letters. Handwriting usually leans to the right (positive slant) or left. A range of 8 to 22 degrees suggests this dataset covers varying degrees of forward-leaning writing.
- F3: Char_Spacing (1.5 - 2.7): Measures the average

physical gaps between individual letters.

ii. Texture and Local Shape Features (F4 – F7)

These features move away from visual geometry and look at mathematical patterns, contrast, and pixel pixel relationships in the image. Notice that all of these are normalized between 0.50 and 1.00, meaning the raw data has been scaled so they can be compared equally.

- F4 and F5: Texture1 and Texture2 (GLCM): GLCM stands for *Gray-Level Co-occurrence Matrix*. It looks at pairs of pixels to see how often different shades of gray appear next to each other. In handwriting, this measures things like smoothness, roughness, or the "grain" of the ink and paper.
- F6 and F7: HOG1 and HOG2 (Histogram of Oriented Gradients): HOG counts occurrences of gradient orientations (the direction of edge changes) in localized portions of the image. It is fantastic for capturing shapes and edges. In this context, it captures the specific curvature and directional strokes of the handwriting.

iii. Deep Learning Features (F8 – F9)

F8 and F9: CNN1 and CNN2 (Convolutional Neural Network): These are "deep features." Unlike F1-F3 (which humans explicitly programmed the computer to find), a Neural Network automatically looked at the handwriting and decided which abstract patterns were most important for identification. They are highly complex and represent abstract combinations of shapes that humans might not even visually recognize as distinct traits. These are also normalized between 0.50 and 1.00. Table 3.1 presents the complete feature set extracted from each handwritten document sample.

Table 3.1: Feature Descriptions and Ranges

Feature ID	Feature Name	Description	Range	Discriminative Power
F1	Stroke Width	Average width of handwriting strokes	0.40 - 0.65	High
F2	Slant Angle	Angle of handwriting slant (degrees)	8 - 22	High
F3	Char Spacing	Average spacing between characters	1.5 - 2.7	Medium
F4	Texture1	First texture feature from GLCM	0.50 - 1.00	High
F5	Texture2	Second texture feature from GLCM	0.50 - 1.00	High
F6	HOG1	First Histogram of Oriented Gradients feature	0.50 - 1.00	Medium
F7	HOG2	Second Histogram of Oriented Gradients feature	0.50 - 1.00	Medium
F8	CNN1	First CNN-extracted deep feature	0.50 - 1.00	Medium
F9	CNN2	Second CNN-extracted deep feature	0.50 - 1.00	Low

3.1.3 Dataset distribution

Dataset Distribution, describes the exact breakdown of the data used to train and test your handwriting recognition or writer identification system. Here is a detailed explanation of what this table tells us, why it matters for your study, and the potential challenges it introduces.

What the Data Shows

The dataset consists of 500 total handwritten document samples collected from three distinct writers. However, the data is not split evenly between them:

Writer 1 (Majority Class): Contributes the largest portion of the dataset with 225 samples (45%). The system will see more examples of this person's handwriting than anyone else's.

Writer 2: Contributes a moderate portion with 175 samples (35%).

Writer 3 (Minority Class): Contributes the smallest portion with only 100 samples (20%).

Table 3.2 shows the distribution of samples across the three writers.

Table 3.2: Dataset Distribution by Writer

Writer ID	Number of Samples	Percentage (%)
Writer 1	225	45
Writer 2	175	35
Writer 3	100	20
Total	500	100

3.2 Data preprocessing

Raw handwritten images often contain noise, inconsistencies, and irrelevant information. Therefore, preprocessing is necessary before model training.

- Preprocessing:** The preprocessing stage is Image resizing, Noise removal, Grayscale conversion, Binarization, Segmentation of characters or words. These operations improve image quality and make the dataset suitable for machine learning algorithms.
- Feature Extraction:** After preprocessing, meaningful information is extracted from each image. Examples of features is Pixel intensity values, Texture descriptors, Shape features, Histogram of Oriented Gradients (HOG), Statistical image properties. The extracted features form the numerical representation of each handwritten sample.
- Feature Normalization:** Since features may have different scales, normalization is applied to bring all values into a common range. The following preprocessing steps are applied to the dataset.

3.3 Formulation of Chaos-Enhanced CSO Algorithms

This section describes a highly advanced variant of the CSO algorithm. In standard CSO the positions of the chickens are updated using standard random numbers. However, standard randomness can sometimes cause the algorithm to get stuck in local dead-ends (local optima) or search the same areas repeatedly. Here is an explanation of what chaos means in this context, and how the three specific algorithms (Circle+, Gauss+, and Henon+) work.

3.3.1 Circle+CSO algorithm

The Algorithm 3.1: Circle Map-Enhanced CSO (Circle+CSO) is shown below.

Algorithm 3.1: Circle Map-Enhanced CSO (Circle+CSO)

Input:

- Population size $N = 30$
- Maximum iterations $T_{\max} = 100$
- Dimension $D = \text{number of features (9)}$
- Rooster ratio $r_R = 0.20$, Hen ratio $r_H = 0.60$, Chick ratio $r_C = 0.20$
- Circle parameters: $a = 0.5$, $b = 0.2$
- Initial condition $x_0 = 0.1$
- Feature bounds $[LB, UB] = [0, 1]$
- Training data $(X_{\text{train}}, y_{\text{train}})$, Validation data $(X_{\text{val}}, y_{\text{val}})$

Output:

- Global best position Gbest (optimal feature subset)
- Best fitness value (validation accuracy)
- Convergence curve
- Selected features

Step 1: Generate Circle chaotic sequence for initialization

chaos_seq = []

$x = x_0$

For $i = 1$ to $N \times D + T_{\max} \times 6$:

$x = (x + b - (a/(2 \times \pi)) \times \sin(2 \times \pi \times x)) \bmod 1$

chaos_seq.append(x)

Step 2: Initialize population using Circle chaos with rotation

For each individual $i = 1$ to N :

For each dimension $j = 1$ to D :

```

idx = i × D + j
rotation = 0.5
chaos_val = (chaos_seq[idx] + rotation) mod 1
position[i][j] = LB + (UB - LB) × chaos_val
fitness[i] = EvaluateFitness(position[i], X_train, y_train,
X_val, y_val)
pbest[i] = position[i]
pbest_fitness[i] = fitness[i]
Step 3: Sort and establish hierarchy
sorted_indices = argsort(fitness, descending=True)
n_roosters = max(1, floor(r_R × N))
n_hens = floor(r_H × N)
n_chicks = N - n_roosters - n_hens
roosters = sorted_indices[0:n_roosters]
hens = sorted_indices[n_roosters:n_roosters + n_hens]
chicks = sorted_indices[n_roosters + n_hens:N]
For each chick in chicks:
    mother_chick[chick] = random choice from hens
Step 4: Initialize global best
Gbest = position[roosters[0]]
Gbest_fitness = fitness[roosters[0]]
Step 5: Main optimization loop
For t = 1 to T_max:
    // Dynamic Circle parameters
    a_dynamic = 0.3 + 0.4 × (1 - t/T_max)
    b_dynamic = 0.15 + 0.2 × (t/T_max)
    // Generate Circle chaos values
    circle_rooster = CircleMap(1, a_dynamic, b_dynamic)[0]
    circle_hen1 = CircleMap(1, a_dynamic, b_dynamic)[0]
    circle_hen2 = CircleMap(1, a_dynamic, b_dynamic)[0]
    circle_chick = CircleMap(1, a_dynamic, b_dynamic)[0]
    // Update roosters
    For each i in roosters:
        sigma = circle_rooster × exp(-t / T_max)
        noise = randn(D) × sigma
        new_position = position[i] + noise × (Gbest - position[i])
        new_position = clamp(new_position, LB, UB)
        new_fitness = EvaluateFitness(new_position, ...)
        If new_fitness > fitness[i]:

```

```

        position[i] = new_position
        fitness[i] = new_fitness
        If new_fitness > Gbest_fitness:
            Gbest = new_position
            Gbest_fitness = new_fitness
    // Update hens
    For each i in hens:
        r_idx = random choice from roosters
        S1 = exp((fitness[i] - fitness[r_idx]) / (abs(fitness[i]) + ε))
        S2 = exp(fitness[roosters[0]] - fitness[i])
        new_position = position[i] + circle_hen1 × S1 ×
            (position[r_idx] - position[i])
            + circle_hen2 × S2 × (Gbest - position[i])
        new_position = clamp(new_position, LB, UB)
        new_fitness = EvaluateFitness(new_position, ...)
        If new_fitness > fitness[i]:
            position[i] = new_position
            fitness[i] = new_fitness
        If new_fitness > Gbest_fitness:
            Gbest = new_position
            Gbest_fitness = new_fitness
    // Update chicks with rotational perturbation
    For each i in chicks:
        mother_idx = mother_chick[i]
        FL = 0.4 + 0.5 × circle_chick
        angle = 2 × π × circle_chick
        radius = 0.05 × (1 - t/T_max)
        perturbation = radius × cos(angle)
        new_position = position[i] + FL × (position[mother_idx] -
            position[i]) + perturbation
        new_position = clamp(new_position, LB, UB)
        new_fitness = EvaluateFitness(new_position, ...)
        If new_fitness > fitness[i]:
            position[i] = new_position
            fitness[i] = new_fitness
        If new_fitness > Gbest_fitness:
            Gbest = new_position
            Gbest_fitness = new_fitness
    convergence_curve[t] = Gbest_fitness

```

// Update hierarchy every 10 iterations

If $t \bmod 10 == 0$:

Update hierarchy and mother-chick relationships

Step 6: Apply optimal feature selection mask

feature_mask = $G_{best} > 0.5$

selected_features = indices where feature_mask is True

Step 7: Return G_{best} , $G_{best_fitness}$, convergence_curve, selected_features

3.3.2 Gauss+CSO algorithm

The Algorithm 3.1: Gauss Map-Enhanced CSO (Gauss+CSO) is shown below.

Algorithm 3.2: Gauss Map-Enhanced CSO (Gauss+CSO)

Input:

- Population size $N = 30$, Max iterations $T_{max} = 100$, Dimension $D = 9$

- Rooster ratio $r_R = 0.20$, Hen ratio $r_H = 0.60$, Chick ratio $r_C = 0.20$

- Initial condition $x_0 = 0.1$ ($x_0 \neq 0$)

- Feature bounds $[LB, UB] = [0, 1]$

Output:

- Global best position G_{best} , Best fitness, Convergence curve

Procedure:

Step 1: Generate Gauss chaotic sequence (strong mixing)

chaos_seq = []

$x = x_0$

For $i = 1$ to $N \times D + T_{max} \times 6$:

If $x == 0$: $x = 0.1$

Else: $x = (1/x) - \text{floor}(1/x)$

chaos_seq.append(x)

Step 2: Initialize population using Gauss chaos

For each individual $i = 1$ to N :

For each dimension $j = 1$ to D :

idx = $i \times D + j$

position[i][j] = $LB + (UB - LB) \times (\text{chaos_seq}[\text{idx}] ** 0.8)$

fitness[i] = EvaluateFitness(position[i], ...)

pbest[i] = position[i]

Step 3: Establish hierarchy (same as Circle+CSO)

Step 4: Main optimization loop with constriction factor

$\varphi = 2.05 + 2.05$

$\chi = 2 / |2 - \varphi - \sqrt{(\varphi^2 - 4\varphi)}|$

For $t = 1$ to T_{max} :

gauss_rooster = GaussMap(1)[0]

gauss_hen1 = GaussMap(1)[0]

gauss_hen2 = GaussMap(1)[0]

gauss_chick = GaussMap(1)[0]

gauss_perturb = GaussMap(1)[0]

$w = 0.9 - 0.5 \times (t/T_{max}) \times (0.5 + 0.5 \times \text{gauss_rooster})$

// Update roosters with Cauchy noise

For each i in roosters:

sigma = $\text{gauss_rooster} \times \exp(-2 \times t / T_{max})$

noise = Cauchy(0, sigma, D)

new_position = position[i] + noise $\times (G_{best} - \text{position}[i])$

new_position = clamp(new_position, LB, UB)

// Evaluate and update

// Update hens with constriction

For each i in hens:

new_position = position[i] + $\chi \times (\text{gauss_hen1} \times S1 \times (\text{pos_r} - \text{position}[i])$

+ $\text{gauss_hen2} \times S2 \times (G_{best} - \text{position}[i]))$

// Evaluate and update

// Update chicks with strong perturbation

For each i in chicks:

FL = $0.3 + 0.6 \times \text{gauss_chick}$

perturbation = $(\text{gauss_perturb} - 0.5) \times (UB - LB) \times 0.08 \times (1 - t/T_{max})$

new_position = position[i] + FL $\times (\text{pos_mother} - \text{position}[i])$ + perturbation

// Evaluate and update

convergence_curve[t] = $G_{best_fitness}$

// Stagnation prevention

If stagnation detected:

regenerate worst 25% using Gauss map

Step 5: Return G_{best} , $G_{best_fitness}$, convergence_curve

3.3.3 Henon+CSO algorithm

The Algorithm 3.1: Henon Map-Enhanced CSO (Henon+CSO)

is shown below.

Algorithm 3.3: Henon Map-Enhanced CSO (Henon+CSO)

Input:

- Population size $N = 30$, Max iterations $T_{\max} = 100$, Dimension $D = 9$
- Rooster ratio $r_R = 0.20$, Hen ratio $r_H = 0.60$, Chick ratio $r_C = 0.20$
- Henon parameters: $a = 1.4$, $b = 0.3$
- Initial conditions: $x_0 = 0.1$, $y_0 = 0.1$

Output:

- Global best position Gbest, Best fitness, Convergence curve

Procedure:

Step 1: Generate Henon chaotic sequence (2D strange attractor)

chaos_seq = []

$x = x_0$, $y = y_0$

For $i = 1$ to $N \times D + T_{\max} \times 6$:

$x_{\text{next}} = 1 - a \times x^2 + y$

$y_{\text{next}} = b \times x$

chaos_value = $(x_{\text{next}} + 1.5) / 3.0$ // Map to $[0,1]$

chaos_seq.append(clip(chaos_value, 0, 1))

$x = x_{\text{next}}$; $y = y_{\text{next}}$

Step 2: Initialize population using Henon chaos

For each individual $i = 1$ to N :

For each dimension $j = 1$ to D :

idx = $i \times D + j$

position[i][j] = $LB + (UB - LB) \times (\text{chaos_seq}[\text{idx}] ** 1.3)$

fitness[i] = EvaluateFitness(position[i], ...)

Step 3: Establish hierarchy (same as previous)

Step 4: Main optimization loop

For $t = 1$ to $T_{\max}$:

henon_rooster = HenonMap(1)[0]

henon_hen1 = HenonMap(1)[0]

henon_hen2 = HenonMap(1)[0]

henon_chick = HenonMap(1)[0]

henon_perturb = HenonMap(1)[0]

// Update roosters

For each i in roosters:

$\sigma = \text{henon_rooster} \times \exp(-1.5 \times t / T_{\max})$

noise = randn(D) $\times \sigma$

new_position = position[i] + noise $\times (G_{\text{best}} - \text{position}[i])$

// Evaluate and update

// Update hens with fractal perturbation

For each i in hens:

new_position = position[i] + henon_hen1 $\times S1 \times (\text{pos}_r - \text{position}[i])$

+ henon_hen2 $\times S2 \times (G_{\text{best}} - \text{position}[i])$

+ $0.03 \times (\text{henon_perturb} - 0.5) \times (UB - LB) \times (1 - t / T_{\max})$

// Evaluate and update

// Update chicks

For each i in chicks:

FL = $0.4 + 0.5 \times \text{henon_chick}$

perturbation = $(\text{henon_perturb} - 0.5) \times (UB - LB) \times 0.06 \times (1 - t / T_{\max})$

new_position = position[i] + FL $\times (\text{pos_mother} - \text{position}[i])$ + perturbation

// Evaluate and update

convergence_curve[t] = Gbest_fitness

// Stagnation prevention with Henon reset

If stagnation detected:

regenerate worst 20% using Henon map

Step 5: Return Gbest, Gbest_fitness, convergence_cur

3.4 Implementation of the models

This section describes the implementation process of the proposed handwriting recognition models and the CE-CSO algorithm. It presents the experimental configuration, computational resources, and software tools employed during model development and evaluation. Providing these details ensures transparency, enables reproducibility of the study, and allows other researchers to validate and compare the obtained results under similar conditions.

3.4.1 Experimental Setup

The experimental setup defines the procedures, configurations, and conditions under which the handwriting recognition models and CE-CSO algorithm were trained and tested. A well-documented experimental setup is essential for ensuring the reliability, consistency, and reproducibility of research findings.

The CE-CSO algorithm was integrated into the model training process to optimize critical parameters, including feature selection and model hyperparameters. Chaotic mapping mechanisms were incorporated into the traditional CSO framework to enhance population diversity, prevent premature convergence, and improve the exploration of the search space. The performance of the proposed method was compared against baseline approaches using standard evaluation metrics such as accuracy, precision, recall, F1-score, and computational time.

To ensure fairness and consistency, all experiments were

conducted under identical conditions, using the same dataset partitions, optimization settings, and evaluation criteria. Multiple experimental runs were performed, and the average results were reported to minimize the impact of randomness and improve the robustness of the findings.

3.4.2 Hardware and Software Environment

The hardware and software environment used for the implementation plays a significant role in the execution speed, memory utilization, and overall performance of machine learning algorithms. Therefore, detailed information about the computational environment is provided to facilitate replication of the study.

The experiments were conducted on a computer system equipped with a multi-core processor, sufficient random-access memory (RAM), and adequate storage capacity to handle the training and optimization processes efficiently. Where applicable, Graphics Processing Unit (GPU) acceleration was utilized to reduce training time and improve computational performance.

On the software side, the implementation was carried out using Python due to its extensive support for machine learning, optimization, and image-processing tasks. Relevant libraries and frameworks such as NumPy for numerical computation, Pandas for data manipulation, OpenCV for image preprocessing, Matplotlib for visualization, and machine learning frameworks such as TensorFlow or PyTorch were employed for model development and evaluation. The Chaos-Enhanced Chicken Swarm Optimization algorithm was implemented within the same environment to ensure seamless integration with the handwriting recognition system.

The operating system, software versions, programming environment, and library dependencies were carefully documented to ensure consistency and reproducibility. Table 3.X presents the detailed hardware and software specifications used in this research.

3.4.3 Hardware and Software Specifications

1. Hardware Specifications (The Muscle)

Running optimization algorithms (like Circle+ CSO, Gauss+ CSO, and Henon + CSO) and processing handwriting images require decent computing power. As shown in Table 3.3.

- a. Processor (Intel Core i7-10750H @ 2.60GHz): This is a high-performance, 6-core laptop processor. In your study, the processor handles the heavy mathematical calculations required to update the positions of the roosters, hens, and chicks across hundreds of iterations.
- b. RAM (16 GB DDR4): Memory is where the computer holds active data. 16 GB ensures that the entire dataset of 500 handwriting samples, along with their extracted features (F1 to F9), can be loaded into memory simultaneously without slowing down the system.

- c. Storage (512 GB SSD): Solid-State Drives (SSDs) read and write data much faster than traditional hard drives. This ensures that loading image files or saving optimization logs happens instantly.

2. Software Specifications (The Brains)

The software stack outlines the tools used to program, test, and evaluate the proposed algorithms.

- a. Operating System (Windows 11 / Ubuntu 20.04): Indicating both operating systems means the code is cross-platform. Ubuntu (Linux) is highly favored in research for its efficient memory management and terminal-based execution, while Windows 11 offers a user-friendly environment for development.
- b. Programming Language (Python 3.9+): Python is the gold standard for machine learning and data science due to its simplicity and powerful ecosystem of scientific libraries.

3. Core Libraries (The Tools)

Instead of building everything from scratch, Python relies on specialized libraries to do the heavy lifting:

- i. **Library: Role in Your Handwriting / CSO Project**
- ii. NumPy: Handles massive multi-dimensional arrays (like your 500 times, 9 feature matrix) and performs ultra-fast matrix mathematics.
- iii. SciPy: Used for advanced scientific and technical computing. It likely helps compute the complex chaotic map differential equations (like the Lorenz Attractor).
- iv. scikit-learn: The primary machine learning library. It will be used to split your dataset, run the final classification models, and calculate metrics like Precision, Recall, and F1-Score.
- v. Matplotlib: Used to plot graphs, such as convergence curves (showing how fast the chickens found the best solution) or accuracy comparisons.
- vi. Pandas: Used for data manipulation. It organizes your handwriting features and dataset distributions into clean data frames (tables) for analysis.

4.0 Results and Discussion

4.1 Introduction

This chapter presents the comprehensive experimental results obtained from evaluating the three chaos-enhanced Chicken Swarm Optimization (CE-CSO) algorithms (Circle+CSO, Gauss+CSO, and Henon+CSO) for handwritten document identification using the IAM dataset. The chapter provides detailed analysis of classification performance, feature selection capability, convergence behavior, computational efficiency, statistical significance, and comparative

performance. The results are organized according to the methodology described in Chapter Three, with extensive visualizations and discussions.

4.2 Experimental Setup Summary

Before presenting the results, Table 4.1 summarizes the experimental configuration used to obtain all results reported in this chapter.

4.3 Classification Performance Results

The results in Table 4.2 show that all three chaos-enhanced CSO algorithms achieved strong classification performance for handwritten document identification. The results indicate that Henon+CSO achieved the best overall classification

performance, with the highest accuracy (93.12%), precision (93.01%), recall (93.12%), and F1-score (93.06%). It also recorded the lowest standard deviation for accuracy (± 0.0098), indicating greater stability and consistency across the 30 independent runs. Gauss+CSO ranked second, achieving an accuracy of 91.89% and an F1-score of 91.83%, while Circle+CSO recorded the lowest classification performance, with an accuracy of 90.34% and an F1-score of 90.27%. Overall, all three algorithms demonstrated strong classification capability, with accuracy values exceeding 90%, while Henon+CSO emerged as the most accurate, stable, and reliable approach for handwritten document identification.

Table 4.1: Experimental Configuration Summary

Parameter	Value
Dataset	IAM Handwritten Database
Total Samples	500
Features	9
Classes (Writers)	3
Training/Validation/ Test Split	70/15/15
Population Size	30 chickens
Maximum Iterations	100
Independent Runs	30 per algorithm
Classifier	SVM (RBF kernel, C=1.0)
Evaluation Metrics	Accuracy, Precision, Recall, F1-Score, Convergence rate, Computational time

Table 4.2: Performance Metrics Comparison

Algorithm	Accuracy	Precision	Recall	F1-Score
Circle+CSO	0.9034 $\pm$ 0.0134	0.9021 $\pm$ 0.0141	0.9034 $\pm$ 0.0134	0.9027 $\pm$ 0.0137
Gauss+CSO	0.9189 $\pm$ 0.0112	0.9178 $\pm$ 0.0118	0.9189 $\pm$ 0.0112	0.9183 $\pm$ 0.0115
Henon+CSO	0.9312 $\pm$ 0.0098	0.9301 $\pm$ 0.0102	0.9312 $\pm$ 0.0098	0.9306 $\pm$ 0.0100

4.4 Computational time Analysis

This section evaluates the computational efficiency of the three chaotic-map-enhanced CSO algorithms. While accuracy and convergence performance are important, an algorithm must also be computationally practical. The analysis compares the execution time, time per Iteration, and computational complexity of Circle+CSO, Gauss+CSO, and Henon+CSO over 30 independent runs. As shown in Table 4.3

1. Mean Execution Time: The mean execution time represents the average duration required for each algorithm to complete an optimization run.
 - a. Circle+CSO recorded the shortest average execution time of 2.38 seconds, making it the fastest of the three algorithms.

- b. Gauss+CSO recorded the highest average execution time of 2.52 seconds, making it approximately 5.9% slower than Circle+CSO.
- c. Henon+CSO required an average execution time of 2.45 seconds, which was approximately 2.9% longer than Circle+CSO.

These results indicate that Circle+CSO has the lowest computational time in terms of total execution time, while Gauss+CSO requires the greatest computational time.

2. Cost per Iteration: Time per iteration measures the average computational effort required for one optimization cycle.

Henon+CSO performs each iteration fastest due to the relatively simple mathematical operations involved in

generating the Henon chaotic sequence.

- Circle+CSO: 0.0238 seconds per iteration
- Gauss+CSO: 0.0252 seconds per iteration
- Henon+CSO: 0.0245 seconds per iteration

Circle+CSO therefore performs each iteration with the lowest computational time, followed by Henon+CSO, while Gauss+CSO requires the highest computational effort per iteration.

- Computational Complexity: All three algorithms have the same theoretical computational complexity, expressed as: $O(N \times D \times T)$

where:

N = population size (number of chicken)

D = problem dimensionality (number of variables/features)

T = number of iterations

Table 4.3: Computational time per iteration analysis

Algorithm	Mean Time (s)	Cost per Iteration (s)	Complexity
Circle+CSO	2.38	0.0238	$O(N \times D \times T)$
Gauss+CSO	2.52	0.0252	$O(N \times D \times T)$
Henon+CSO	2.45	0.0245	$O(N \times D \times T)$

4.5 Comparative Discussion

This section provides a comprehensive comparison of the three chaotic-map-enhanced CSO algorithms by combining all evaluation criteria into a single performance assessment. Rather than examining accuracy, convergence speed, feature reduction, or computational time separately, the comparative discussion identifies which algorithm delivers the best overall balance between effectiveness and efficiency.

4.5.1 Algorithm performance summary

Performance Analysis by Criterion

- Classification Performance: Henon+CSO achieved the best classification performance, with an accuracy of 93.12%,
- precision of 93.01%, recall of 93.12%, and F1-score of 93.06%. Gauss+CSO ranked second, with an accuracy of 91.89%, while Circle+CSO recorded the lowest accuracy of 90.34%. Thus, Henon+CSO demonstrated the strongest overall classification capability.
- Computational time: Circle+CSO recorded the lowest mean execution time of 2.38 seconds, followed by Henon+CSO at 2.45 seconds and Gauss+CSO at 2.52 seconds. Therefore, Circle+CSO was the most computationally efficient, while Gauss+CSO required the highest execution time.

- Convergence rate: Henon+CSO demonstrated the fastest convergence, reaching its convergence point at iteration 41, compared with 47 iterations for Gauss+CSO and 58 iterations for Circle+CSO. Henon+CSO also achieved the highest final fitness of 0.9312, followed by Gauss+CSO at 0.9189 and Circle+CSO at 0.9034.
- Fitness Stability: Henon+CSO recorded the lowest standard deviation (0.0098) and coefficient of variation (1.05%), indicating the highest stability. Gauss+CSO followed with a coefficient of variation of 1.22%, while Circle+CSO recorded 1.48%.
- Overall Performance: Overall, Henon+CSO emerged as the best-performing algorithm, particularly in terms of classification accuracy, convergence speed, final fitness, and stability. Circle+CSO demonstrated the advantage of having the lowest computational time, while Gauss+CSO provided intermediate performance across most of the evaluated criteria. These findings indicate that although Circle+CSO is computationally faster, Henon+CSO provides the best overall balance between classification effectiveness, convergence behaviour, and solution stability.

4.5.2 Advantages and disadvantages

Table 4.4 shows qualitative comparison between the selected chaotic maps (Circle+CSO, Gauss+CSO, and Henon+CSO)

Table 4.4: Qualitative Comparison

Algorithm	Advantages	Disadvantages
Circle+CSO	- Fastest execution (2.38s) - Rotational exploration pattern - Simple implementation	- Lowest accuracy (90.34%) - Moderate convergence (58 iter) - Quasiperiodic limitations

Gauss+CSO	i. High accuracy (91.89%) ii. Strong mixing property iii. Good stagnation prevention	i. Highest computational time (2.52s) ii. Non-uniform distribution iii. Cauchy noise overhead
Henon+CSO	i. Highest accuracy (93.12%) ii. Fastest convergence (41 iter) iii. Most stable (± 0.0098) iv. Best feature reduction (43.3%)	i. Slightly higher complexity ii. Two-dimensional chaos overhead

5.0 Conclusions

The integration of chaotic maps into the CSO framework was successfully achieved for all three variants. The mathematical formulations were properly extended to incorporate chaotic sequences for population initialization, parameter adaptation, and position updates.

Python implementations of Circle+CSO, Gauss+CSO, and Henon+CSO were successfully developed and validated. The modular design allows easy extension to other chaotic maps and optimization problems.

Henon+CSO algorithm achieved the highest classification accuracy of 93.12% on the IAM handwritten dataset, significantly outperforming both Circle+CSO (90.34%) and Gauss+CSO (91.89%).

In terms of computational efficiency, Circle+CSO recorded the shortest mean execution time of 2.38 seconds per run, followed by Henon+CSO at 2.45 **seconds** and Gauss+CSO at 2.52 seconds.

Circle+CSO demonstrated the lowest computational time, making it suitable when execution speed is the primary consideration, while Gauss+CSO provided intermediate performance across the evaluated criteria.

Henon+CSO emerged as the best overall-performing algorithm, providing the highest classification accuracy, fastest convergence, greatest fitness stability, and highest feature reduction.

References

1. Agrawal, V., Jagtap, J., & Kantipudi, M. V. V. P. (2024). Exploration of advancements in handwritten document recognition techniques. *Intelligent Systems with Applications*, 22, Article 200358. <https://doi.org/10.1016/j.iswa.2024.200358>
2. Amusan, O. F., Olabiyisi, S. O., Ismaila, W. O., & Adeyemo, I. A. (2026). Performance evaluation of particle swarm optimization algorithm with selected chaos maps for image segmentation. *UKR Journal of Multidisciplinary Studies*, 2(7), 128–136.
3. Arora, S., & Anand, P. (2019). Chaotic maps in metaheuristic algorithms: A comprehensive survey. *Applied Artificial Intelligence*, 33(9), 700–732. <https://doi.org/10.1080/08839514.2019.1608360>
4. Balaha, H. M., & Ali, H. A. (2020). A new Arabic handwritten character recognition deep learning system (AHCR-DLS). *Neural Computing and Applications*, 33, 6325–6367. <https://doi.org/10.1007/s00521-020-05397-0>
5. Bhattacharjee, V., Roy, P. K., & Chatteraj, C. (2024). Chaotic artificial hummingbird algorithm applied to elementary machine design problems. *Progress in Artificial Intelligence*, 13, 307–333. <https://doi.org/10.1007/s13748-024-00337-w>
6. Christlein, V., Gropp, M., Fiel, S., & Maier, A. (2017). Unsupervised feature learning for writer identification and writer retrieval. In *Proceedings of the 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)* (Vol. 1, pp. 991–997). IEEE. <https://doi.org/10.1109/ICDAR.2017.166>
7. Deb, S., Gao, X. Z., Tammi, K., Kalita, K., & Mahanta, P. (2019). Recent studies on chicken swarm optimization algorithm: A review (2014–2018). *Artificial Intelligence Review*, 53(3), 1667–1698. <https://doi.org/10.1007/s10462-019-09718-3>
8. Dokeroglu, T., Deniz, A., & Kiziloğlu, H. E. (2025). Chaos-enhanced metaheuristics: Classification, comparison, and convergence analysis. *Complex and Intelligent Systems*. Advance online publication. <https://doi.org/10.1007/s40747-025-01791-2>
9. Fiel, S., Kleber, F., Diem, M., & Sablatnig, R. (2019). Writer retrieval and writer identification for historical handwriting. *Journal of Imaging*, 5(1), Article 3. <https://doi.org/10.3390/jimaging5010003>

10. Gagnon, I., April, A., & Abran, A. (2021). An investigation of the effects of chaotic maps on the performance of metaheuristics. *Engineering Reports*, 3(8), Article e12369. <https://doi.org/10.1002/eng2.12369>
11. Gagnon, M., Gosselin, F., & Bhatt, P. (2021). An investigation of the effects of chaotic maps on the performance of metaheuristics. *Engineering Reports*, 3(3), Article e12369. <https://doi.org/10.1002/eng2.12369>
12. Gharehchopogh, F. S., Abdollahzadeh, B., Barshandeh, S., & Asghari, P. (2024). A comprehensive survey on the chicken swarm optimization algorithm and its applications: State-of-the-art and research challenges. *Artificial Intelligence Review*, 57(8). <https://doi.org/10.1007/s10462-024-10786-3>
13. Hussain, R., Raza, A., Siddiqi, I., Khurshid, K., & Djeddi, C. (2015). A comprehensive survey of handwritten document benchmarks: Structure, usage and evaluation. *Journal on Image and Video Processing*, 2015, Article 46. <https://doi.org/10.1186/s13640-015-0102-5>
14. John, O. E., Baale, A. A., & Olabiyisi, S. O. (2026). Performance analysis of selected chaos maps to improve particle swarm optimization in image segmentation tasks. *EIRA Journal of Multidisciplinary Research and Development*, 2(4), 55–65. <https://doi.org/10.5281/zenodo.21766978>
15. Kanwal, S., Inam, S., Othman, M. T. B., Waqar, A., Ibrahim, M., Nawaz, F., Nawaz, Z., & Hamam, H. (2022). An effective color image encryption based on Henon map, tent chaotic map, and orthogonal matrices. *Sensors*, 22(12), Article 4359. <https://doi.org/10.3390/s22124359>
16. Kaveh, A., Zaerreza, A., & Shojaei, I. (2024). *Chaotic meta-heuristic algorithms for optimal design of structures* (Vol. 1129). Springer. <https://doi.org/10.1007/978-3-031-48918-1>
17. Kumar, J. P. (2025). Handwritten text recognition using deep learning: A CNN-LSTM approach. *International Journal of Intelligent Systems and Applications in Engineering*, 13(1), 14–22. <https://ijisae.org/index.php/IJISAE/article/view/7443>
18. Kumar, N., & Saini, S. (2024). Image encryption model based on chaotic Henon map and termite alate optimization algorithm. *International Journal of Intelligent Systems and Applications in Engineering*, 12(18s), 428–436. <https://ijisae.org/index.php/IJISAE/article/view/4987>
19. Liang, Y., Li, S., Liu, Z., & Li, C. (2022). An improved chicken swarm optimization algorithm for solving multimodal optimization problems. *Computational Intelligence and Neuroscience*, 2022, Article 5359732. <https://doi.org/10.1155/2022/5359732>
20. Marques, O., Avola, D., & Cinque, L. (2021). Handwriting recognition in the wild using deep learning. *Sensors*, 21(4), Article 1265. <https://doi.org/10.3390/s21041265>
21. Meng, X., Liu, Y., Gao, X., & Zhang, H. (2014). A new bio-inspired algorithm: Chicken swarm optimization. In Y. Tan, Y. Shi, & C. A. C. Coello (Eds.), *Advances in swarm intelligence: ICSI 2014* (Lecture Notes in Computer Science, Vol. 8794, pp. 86–94). Springer. https://doi.org/10.1007/978-3-319-11857-4_10
22. Moronkeji, I. O., Olabiyisi, S. O., & Adedeji, O. T. (2026). A comparative study of selected chaos maps to improve particle swarm optimisation in image segmentation tasks. *EIRA Journal of Multidisciplinary Research and Development*, 2(4), 17–26.
23. Mugisha, S., Gutu, J. L. T., & Nagabhushan, P. (2024). *An improved chicken swarm optimization algorithm for handwritten document image enhancement*. arXiv. <https://arxiv.org/abs/2411.00802>
24. Mugisha, S., Gutu, J. L. T., & Nagabhushan, P. (2025). An enhanced chicken swarm optimization algorithm using Gaussian and Tent chaotic map functions. *International Journal of Research and Innovation in Applied Science*, 10(8). <https://rsisinternational.org/journals/ijrias/articles/an-enhanced-chicken-swarm-optimization-algorithm-using-gaussian-and-tent-chaotic-map-functions/>
25. Okunlola, S. O., Baale, A. A., & Olabiyisi, S. O. (2026). Performance evaluation of selected chaos enhanced particle swarm optimisation for image segmentation. *Engineering and Technology Journal*, 11(7), 10904–10911. <https://doi.org/10.47191/etj/v11i07.13>
26. Olabiyisi, S. O., Babarinde, T. A., & Oladimeji, I. W. (2026). Performance evaluation of selected chaos-enhanced chicken swarm optimization for handwritten document identification. *Engineering*

and Technology Journal, 11(7), 10965–10974.
<https://doi.org/10.47191/etj/v11i07.20>

27. PeerJ Editors. (2026). Chicken swarm optimization and associated variants: A literature review. *PeerJ Computer Science*, 12, Article e3609. <https://peerj.com/articles/cs-3609/>
28. Purohit, N. (2024). A methodological review of an offline writer identification framework utilizing deep learning and handcrafted approaches. *Journal of Electrical Systems*, 20(11s), 1440–1451. <https://doi.org/10.52783/jes.7504>
29. Rezaei, B., Ghanbari, H., & Enayatifar, R. (2023). An image encryption approach using tuned Henon chaotic map and evolutionary algorithm. *Nonlinear Dynamics*, 111(10), 9629–9647. <https://doi.org/10.1007/s11578-023-02895-w>
30. Saleh, A., Hassan, M., & Yousef, K. (2025). Accuracy, precision, recall, F1-score, or MCC? Empirical evidence from advanced statistics, ML, and XAI for evaluating business predictive models. *Journal of Big Data*, 12(1). <https://doi.org/10.1186/s40537-025-01313-4>