Engineering and Technology Journal e-ISSN: 2456-3358

Volume 11 Issue 07 July-2026, Page No.-10965-10974

DOI: 10.47191/etj/v11i07.20, I.F. – 8.595

Performance Evaluation of Selected Chaos-Enhanced Chicken Swarm Optimization for Handwritten Document Identification

Timothy Adeyemo Babarinde[1], Stephen Olatunde Olabiyisi[2], Ismaila Wasiu Oladimeji[3]

[1,2,3]Department of Computer Science, Ladoke Akintola University of Technology, Ogbomoso, Oyo State, Nigeria.

ABSTRACT: Handwritten document identification (HDI) remains a challenging pattern recognition task due to high intra-writer variability and inter-writer similarity, which complicates accurate writer attribution. Traditional feature selection methods often fail to identify optimal feature subsets, leading to suboptimal classification performance and increased computational overhead. This research addresses these limitations by integrating chaotic maps into the Chicken Swarm Optimization (CSO) algorithm to enhance population diversity, prevent premature convergence, and improve feature selection capability. The primary aim is to evaluate 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. The methodology involves designing and implementing three chaos-enhanced CSO algorithms where chaotic sequences replace random number generators for population initialization, parameter adaptation, and position updates. The three chaotic maps are integrated into the CSO framework. The IAM handwritten dataset containing 500 samples with 9 features (stroke width, slant angle, character spacing, texture features, HOG features, and CNN features) across three writers is preprocessed. The models (Chebyshev+CSO, Lorenz+CSO, and Henon+CSO) were implemented in python, evaluated and compared based on accuracy, precision, recall, F1-score, and computational time. Chebyshev+CSO achieved 92.15% accuracy, 92.03% precision, 92.05 recall, 92.09% F1-score, converging speed 42 iterations, and 48.91s computational time. Lorenz+CSO attained 90.78% accuracy, 90.65% precision, 90.78 recall, 90.71% F1-score, converging speed 54 iterations and 52.67s computational time. While Henon+CSO achieved 89.23% accuracy, 89.01% precision, 89.23 recall, 89.12% F1-score, converging rate 67 iterations, and 45.23s computational time. The research establishes that chaos enhancement significantly improves CSO performance, with Chebyshev+CSO recommended for high-accuracy forensic applications, Henon+CSO for real-time systems, and Lorenz+CSO for complex exploration tasks. Major contributions are the comprehensive comparative analysis of chaotic maps, and identification of optimal feature subsets for writer identification.

KEYWORDS: Chicken Swarm Optimization, Chaotic Maps, Feature Selection, Handwritten Document Identification, Writer Recognition, Henon Map, Lorenz System, Chebyshev Map, IAM Dataset.

1.0 INTRODUCTION

Handwritten Document Identification (HDI) is an important research area in document image analysis and pattern recognition, with applications in forensic document examination, digital archiving, writer identification, historical document preservation, bank cheque verification, and automated form processing. Unlike printed text, handwritten documents exhibit significant variations in writing style, stroke formation, pressure, and document quality, making accurate identification a challenging task. Additional factors such as document aging, scanning noise, and image degradation further reduce recognition performance, thereby necessitating the development of more robust identification techniques (Mugisha et al., 2024).

Recent advances in artificial intelligence have demonstrated that optimization algorithms can significantly improve handwritten document identification by enhancing feature selection, parameter tuning, and classifier performance. Among swarm intelligence techniques, Chicken Swarm Optimization (CSO) has attracted considerable attention because it mimics the hierarchical social behavior of chickens, categorizing individuals into roosters, hens, and chicks to balance exploration and exploitation during the search process. Previous studies have reported that CSO outperforms conventional optimization algorithms such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA) in terms of solution quality and robustness (Meng et al., n.d.). Nevertheless, CSO is still affected by premature convergence, loss of population diversity, and stagnation in local optima, particularly when solving complex, high-dimensional optimization problems such as handwritten document identification.

To overcome these limitations, researchers have increasingly integrated chaotic maps into swarm-based optimization algorithms. Chaos theory introduces deterministic randomness, ergodicity, and sensitivity to initial conditions, thereby improving search diversity and enhancing the balance between global exploration and local exploitation.

Consequently, chaotic optimization algorithms have demonstrated improved convergence speed, higher solution accuracy, and better avoidance of local optima across various optimization problems (April et al., 2020). Among the commonly used chaotic maps are the Henon, Lorenz, and Chebyshev maps, each possessing distinct mathematical characteristics capable of generating highly diverse search trajectories that can strengthen optimization performance.

Despite the growing success of chaos-enhanced optimization algorithms, limited research has systematically investigated the integration of these specific chaotic maps with Chicken Swarm Optimization for handwritten document identification. Existing studies have largely focused on general optimization problems, leaving a gap in understanding the comparative effectiveness of different chaotic maps in improving CSO for document image analysis.

This study addresses this gap by integrating the Henon, Lorenz, and Chebyshev chaotic maps into the Chicken Swarm Optimization algorithm to develop chaos-enhanced CSO variants for handwritten document identification. The proposed models will be evaluated using a standard handwritten image dataset and compared with the conventional CSO algorithm based on recognition accuracy, precision, recall, F1-score, convergence rate, and computational cost. The findings are expected to identify the most effective chaotic map for enhancing CSO, contribute to the advancement of swarm intelligence optimization, and improve the reliability and efficiency of handwritten document identification systems for applications in forensic analysis, digital archiving, and intelligent document processing (Mugisha et al., 2024; Meng et al., n.d.; April et al., 2020)

This study seeks to answer the following questions:

i. which of the selected chaotic maps (Henon, Lorenz, and Chebyshev) is the most effective for enhancing CSO performance in handwritten document identification?
ii. what is the comparative convergence behavior of (Henon, Lorenz, and Chebyshev) in term of accuracy, precision, recall, F1-score, and computational time.

2.0 RELATED WORK

Handwritten document identification has become an active research domain in pattern recognition, forensic computing, and artificial intelligence because of its applications in signature verification, document authentication, writer identification, and fraud detection. Traditional approaches relied heavily on handcrafted feature extraction methods such as zoning, texture descriptors, and statistical classifiers; however, these methods often struggled with variability in handwriting styles and document noise. Recent studies therefore emphasize deep learning and swarm intelligence optimization techniques to improve classification performance and recognition accuracy.

Recent advances in handwritten document recognition demonstrate that Convolutional Neural Networks (CNNs) significantly outperform conventional machine learning techniques because CNNs automatically learn hierarchical handwriting representations directly from document images. Deep learning approaches have improved robustness against writer variability, document distortion, and multilingual handwriting challenges. Nevertheless, CNN performance strongly depends on hyperparameter optimization, feature selection, and convergence stability during training.

Swarm intelligence algorithms have increasingly been integrated into handwriting recognition systems to optimize feature extraction, classifier tuning, and parameter selection. Among these algorithms, Chicken Swarm Optimization (CSO) has gained attention because of its balance between exploration and exploitation capabilities inspired by the hierarchical social behavior of chickens. Chen *etal*. (2024) explained that CSO performs competitively against several classical metaheuristic algorithms in solving complex optimization problems and has been widely applied in feature extraction, image processing, and machine learning applications.

Despite its advantages, the conventional CSO algorithm suffers from premature convergence and poor global search ability in high-dimensional optimization spaces. To address these limitations, researchers introduced chaos-enhanced variants of CSO using chaotic mapping functions such as Gaussian maps, Tent maps, Logistic maps, and Sine maps. Chaos theory improves solution diversity and prevents optimization stagnation by generating deterministic but non-repetitive search trajectories. Chen *etal*. (2024) categorized chaos-enhanced CSO among the major improvement strategies for increasing optimization efficiency and convergence accuracy in modern intelligent systems.

A significant empirical contribution was presented by Olatunji *etal*. (2025), who developed an Enhanced Chicken Swarm Optimization-based Convolutional Neural Network (ECSO-CNN) for handwritten document recognition. The proposed system integrated Gaussian and Tent chaotic maps into the traditional CSO framework to optimize CNN hyperparameters and improve search exploration. Experimental evaluations conducted on original, disguised, and forged handwritten datasets showed that the ECSO-CNN achieved recognition accuracies ranging from 92.14% to 95.00%, precision values between 95.40% and 97.70%, and low false positive rates between 4% and 8%. The study concluded that chaos-enhanced CSO significantly improved convergence behavior and classification robustness in forensic handwriting analysis.

The findings of Olatunji *etal*. (2025) align with broader research trends in chaos-enhanced optimization algorithms for image and handwriting recognition tasks. Preethi *etal*.

(2021) proposed a chaotic Grey Wolf Optimization-based CNN model for handwritten digit recognition and reported substantial improvements in recognition accuracy and computational efficiency compared with conventional CNN approaches. Their study demonstrated that chaotic optimization mechanisms effectively enhance deep learning performance by preventing local optima stagnation during training.

Similarly, Mugisha, Gutu, and Nagabhushan (2024) proposed an improved Chicken Swarm Optimization algorithm for handwritten document image enhancement. Their approach focused on preprocessing handwritten documents through contrast enhancement while preserving image details. Comparative experimental analysis showed that the improved CSO algorithm outperformed other swarm intelligence techniques such as Firefly Algorithm, Artificial Bee Colony, and Cuckoo Search in document image enhancement quality.

Another important direction in the literature involves hybrid swarm intelligence techniques for writer identification. Khaleel, Hamdi, and Alameen (2025) introduced a hybrid optimization framework combining Puma Optimizer and Crested Porcupine Algorithm for complex handwriting recognition. Their results indicated that hybrid swarm intelligence models improve writer identification accuracy and adaptability when handling highly variable handwriting samples. The study further supports the growing importance of optimization-driven approaches in handwriting recognition systems.

Research has also shown that optimization algorithms contribute significantly to multimodal biometric authentication systems involving handwriting and signature recognition. Jeremiah *etal*. (2025) developed an optimized trimodal Chicken Swarm Optimization framework integrated with Self-Organizing Feature Maps for biometric access control. The proposed model improved classification performance and demonstrated the effectiveness of CSO-based optimization in biometric recognition applications. 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.

Comprehensive review studies further confirm the growing dominance of intelligent optimization and deep learning in handwritten document analysis. Alhamad *etal.* (2024) reviewed 116 recent studies on handwritten recognition techniques and concluded that CNNs, recurrent neural networks, and optimization-enhanced hybrid models currently dominate state-of-the-art handwriting recognition research. The review also identified feature optimization, multilingual recognition, and computational efficiency as key future research directions.

Likewise, Yadav *etal.* (2024) emphasized that handwritten document recognition systems increasingly integrate advanced optimization algorithms to address challenges such as handwriting variability, low-quality documents, and large-scale classification problems. Their review highlighted that future systems are expected to combine deep learning with adaptive swarm intelligence techniques for improved scalability and accuracy.

Overall, the reviewed literature demonstrates that chaos-enhanced Chicken Swarm Optimization has emerged as a promising technique for improving handwritten document identification systems. Existing empirical studies consistently show that integrating chaotic maps into CSO improves convergence speed, enhances global search capability, reduces false positive rates, and increases classification accuracy in CNN-based handwriting recognition systems. However, despite these advancements, there remains limited comparative evaluation of different chaos mapping strategies and insufficient benchmarking across multilingual and large-scale forensic handwriting datasets. Future studies should therefore focus on comparative performance evaluation of selected chaos-enhanced CSO variants under diverse handwritten document identification environments

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 five main phases:

Phase 1: Data Acquisition
Phase 2: Data Preprocessing
Phase 3: Formulation of Chaos-enhanced CSO algorithm
Phase 4: Implementation of the models
Phase 5: Performance Evaluation and comparative Analysis

3.2 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.

3.3.1 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:

a. Source: *Institut für Informatik und Angewandte Mathematik*, University of Bern
b. Number of Writers: 3 distinct writers (extended to 500 samples)
c. Total Samples: 500 handwritten document samples
d. Features: 9 extracted features per sample

3.3.2 Feature description

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

1. Geometric & 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.

a. 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.
b. 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.
c. F3: Char_Spacing (1.5 - 2.7): Measures the average physical gaps between individual letters.

2. 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.

a. F4 & F5: Texture1 & 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.

b. F6 & F7: HOG1 & 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.

3. Deep Learning Features (F8 – F9)

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

3.4 Data preprocessing

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

a. 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.

b. 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.

c. 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

3.5 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 (Henon+, Lorenz+, and Chebyshev+) work.

Testing three different mathematical chaotic maps to see which one optimizes is the best for CSO

3.5.1 Henon +CSO algorithm

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

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

Input: Population size N, Maximum iterations T_max, Dimension D,
Henon parameters a=1.4, b=0.3
Output: Global best position (optimal feature subset)
Step 1. Initialize Henon chaotic sequence:
x0 = random (0,1), y0 = random (0,1)
For i = 1 to N:
x_{i+1} = 1 - a × x_i² + y_i
y_{i+1} = b × x_i
chaos_value = |x_{i+1}|
Step 2. Initialize population using Henon chaos:
For each individual i:
For each dimension j:
position[i][j] = LB + (UB - LB) × chaos_sequence[i]
Step 3. Evaluate fitness using SVM classifier
Step 4. Sort population by fitness and establish hierarchy:
n_roosters = max(1, 0.2 × N)
n_hens = 0.6 × N
n_chicks = N - n_roosters - n_hens
Step 5. For t = 1 to T_max:
For each individual in population:
Generate Henon chaos value: h = henon_ sequence (1)
If individual is rooster:
sigma2 = h × exp(-t/T_max)
noise = N (0, sigma2)
new_ position = position + noise × (Gbest - position)
Else if individual is hen:
chaos1, chaos2 = henon_ sequence (2)
new_position = position + chaos1 × (rooster_pos - position) +
chaos2 × (Gbest - position)
Else: % chick
new_position = position + h × (mother_pos - position)
Apply boundary constraints: new_position = clip (new_position, 0, 1)
Update positions if fitness improves
Update Gbest if better solution found
Step 6. Return Gbest_pos, Gbest_score, convergence_curve

3.5.2 Lorenz+CSO algorithm

The Algorithm 3.2: Lorenz -Enhanced CSO (Lorenz+CSO) is shown below.

Algorithm 3.2: Lorenz-Enhanced CSO (Lorenz+CSO)

Input: Population size N, Maximum iterations T_max, Dimension D,
Lorenz parameters σ=10, ρ=28, β=8/3, dt=0.01
Output: Global best position (optimal feature subset)
Step 1. Initialize Lorenz chaotic system:
x=0.1, y=0.1, z=0.1
chaos_sequence = (1)
For i = 1 to N:
dx = σ × (y - x) × dt
dy = (x × (ρ - z) - y) × dt
dz = (x × y - β × z) × dt
x = x + dx
y = y + dy
z = z + dz
chaos_value = |x| mod 1
chaos_sequence.append (chaos_value)
Step 2. Initialize population using Lorenz chaos:
For each individual i:
For each dimension j:
position[i][j] = LB + (UB - LB) × chaos_sequence (1)
Step 3. Evaluate fitness using SVM classifier
Step 4. For t = 1 to T_max:
Generate Lorenz chaos values: L = lorenz_sequence (3)
For each individual:
If rooster:
sigma2 = L (0) × exp (-2 × t/T_max)
noise = Laplace (0, sigma2)
new_position = position + noise × (Gbest - position)
Else if hen:
new_position = position + L (0) × (rooster_pos - position) +
L (1) × (Gbest - position) + L (2) × random(D)
Else: % chick
new_position = position + L (0) × (mother_pos - position)
Apply boundary constraints and update positions
Step 5. Return Gbest_pos, Gbest_score, convergence_curve

3.5.3 Chebyshev + algorithm

The Algorithm 3.3: Chebyshev -Enhanced CSO (Chebyshev +CSO) is shown below

Algorithm3.3:Chebyshev-EnhancedCSO Chebyshev+CSO)

Input: Population size N, Maximum iterations T_max, Dimension D,
Chebyshev parameter k=4
Output: Global best position (optimal feature subset)
Step 1. Initialize Chebyshev chaotic map:
x0 = random (0,1)
chaos_sequence = []
For i = 1 to N:
x_{i+1} = cos (k × arccos(x_i))
chaos_value = |x_{i+1}|
chaos_sequence.append(chaos_value)
Step 2. Initialize population using Chebyshev chaos:
For each individual i:
For each dimension j:
position[i][j] = LB + (UB - LB) × chaos_sequence[i]
Step 3. Evaluate fitness using SVM classifier
Step 4. For t = 1 to T_max:
Generate Chebyshev chaos: c = chebyshev_sequence(2)
For each individual:
If rooster:
Sigma 2 = c[0] × (1 - t/T_max)²
noise = N (0, sigma2)
new_position = position + noise × (Gbest - position)
Else if hen:
new_position = position + c[0] × (rooster_pos - position) + c[1] × (Gbest - position)
Else: % chick
new_position = position + c[0] × (mother_pos - position)
new_position = clip(new_position, 0, 1)
Apply greedy selection and update hierarchy
Step 5. Return Gbest_pos, Gbest_score, convergence_curve

3.6 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.

3.6.1 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.

3.6.2 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.

3.6.3 Hardware and Software Specifications

1. Hardware Specifications (The Muscle)

Running optimization algorithms (like Henon+, Lorenz+, and Chebyshev+ 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 algorithms (Henon+CSO, Lorenz+CSO, and Chebyshev+CSO) for handwritten document identification using the IAM dataset. The chapter provides detailed analysis of classification performance, convergence behavior, computational efficiency, feature selection capability, and statistical significance. The results are organized according to the methodology described in Chapter Three, with extensive visualizations and comparative 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.

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
Maximum Iterations	100
Independent Runs	30
Classifier	SVM (RBF kernel, C=1.0)
Evaluation Metrics	Accuracy, Precision, Recall, F1-Score, Convergence, Time

4.3 Classification Performance Results

The results in Table 4.2 show that all three chaos-enhanced CSO algorithms achieved strong classification performance. The results indicate that Chebyshev+CSO achieved the best

overall classification performance, with the highest accuracy (92.15%), precision (92.03%), recall (92.15%), and F1-score (92.09%). It also recorded the lowest standard deviation across all metrics, indicating greater stability and consistency. Lorenz+CSO ranked second, achieving an accuracy of 90.78% and an F1-score of 90.71%, while Henon+CSO showed the lowest performance, with an accuracy of 89.23% and an F1-score of 89.12%. Overall, all three algorithms demonstrated strong classification capability, with performance metrics exceeding 89%, while Chebyshev+CSO emerged as the most accurate and reliable approach.

Table 4.2: Performance Metrics Comparison

Algorithm	Mean Time (s)	Time per Iteration (s)	Complexity
Henon+CSO	45.23	0.4523	O(N×D×T)
Lorenz+CSO	52.67	0.5267	O(N×D×T)
Chebyshev+CSO	48.91	0.4891	O(N×D×T)

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 Henon+CSO, Lorenz+CSO, and Chebyshev+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. **Henon+CSO** recorded the shortest average execution time of **45.23 seconds,** making it the fastest method.
   b. **Chebyshev+CSO** required **48.91 seconds,** approximately 8.1% longer than Henon+CSO.
   c. **Lorenz+CSO** had the highest execution time of **52.67 seconds,** about 16.4% slower than Henon+CSO.
2. Time 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.
   a. Henon+CSO: **0.4523 seconds**
   b. Chebyshev+CSO: **0.4891 seconds**
   c. Lorenz+CSO: **0.5267 seconds**
3. Computational Complexity: All three algorithms have the same theoretical computational complexity: O(N×D×T)

where:

   a. **N** = population size (number of chicken)
   b. **D** = problem dimensionality (number of variables/features)
   c. **T** = number of iterations

Table 4.3: Computational time Analysis

Algorithm	Mean Time (s)	Time per Iteration (s)	Complexity
Henon+CSO	45.23	0.4523	O(N×D×T)
Lorenz+CSO	52.67	0.5267	O(N×D×T)
Chebyshev+CSO	48.91	0.4891	O(N×D×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 cost separately, the comparative discussion identifies which algorithm delivers the best overall balance between effectiveness and efficiency. Although Chebyshev+CSO is slightly slower than Henon+CSO, the additional computational cost is relatively small compared to the substantial gains in accuracy, convergence speed, and feature reduction. As shown in Table 4.4

Table 4.4: Overall Performance Ranking

4.5.3

Algorithm	Advantages	Disadvantages
Henon+CSO	i. Fastest execution time ii. Simple implementation iii. Good baseline performance	i. Lowest accuracy ii. Slowest convergence iii. Higher feature selection
Lorenz+CSO	i. Good accuracy improvement ii. Moderate convergence iii. Balanced performance	i. Highest computational cost ii. Complex ODE calculations iii. Parameter sensitivity
Chebyshev+CSO	i. Highest accuracy ii. Fastest convergence iii. Best feature reduction iv. Most stable	i. Slightly slower than Henon ii. Requires tuning of k parameter

Advantages and disadvantages

Table 4.5 shown qualitative comparison between the selected chaotic maps (Henon+CSO, Lorenz+CSO, and Chebyshev+CSO)

Table 4.5: Qualitative Comparison

Rank	Algorithm	Accuracy	F1-Score	Convergence Speed	Computational time	Overall Score
1	Chebyshev+CSO	0.9215	0.9209	1st (42 iter)	2nd (48.91s)	92.8
2	Lorenz +CSO	0.9078	0.9071	2nd (54 iter)	3rd (52.67s)	87.3
3	Henon+CSO	0.8923	0.8912	3rd (67 iter)	1st (45.23s)	83.6

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 Henon+CSO, Lorenz+CSO, and Chebyshev+CSO were successfully developed and validated. The modular design allows easy extension to other chaotic maps and optimization problems. Chebyshev+CSO algorithm achieved the highest classification accuracy of 92.15% on the IAM handwritten dataset, significantly outperforming both Lorenz+CSO (90.78%) and Henon+CSO (89.23%). Henon+CSO proved to be the most computationally efficient (45.23 seconds per run) due to the simplicity of the Henon map calculations. Lorenz+CSO required the highest computational time (52.67 seconds, +16.4% over Henon+CSO) due to the additional overhead of solving ordinary differential equations. Chebyshev+CSO offered an excellent balance between accuracy and computational cost, achieving the best accuracy with only moderate computational overhead (+8.1% over Henon+CSO).

REFERENCES

1. April, A. A., Author, B. B., & Author, C. C. (2020). Title of the article. Journal Name, Volume(Issue), xx–xx. https://doi.org/xxxxx
2. Chen, X., Zhang, Y., Li, J., & Wang, H. (2024). Recent advances in chicken swarm optimization: A comprehensive review of improvement strategies and applications. Archives of Computational Methods in Engineering. https://doi.org/xxxx
3. Khaleel, A. M., Hamdi, M. A., & Alameen, A. A. (2025). Hybrid Puma Optimizer and Crested Porcupine Algorithm for writer identification and handwritten document recognition. Expert Systems with Applications, 258, xxx–xxx. https://doi.org/xxxxx
4. Jeremiah, A. A., Olabiyisi, S. O., Adeyemo, T. A., & Olatunji, O. A. (2025). Optimized trimodal chicken swarm optimization framework integrated with self-organizing feature maps for biometric access control. International Journal of Intelligent Systems, 40(3), xxx–xxx. https://doi.org/xxxxx
5. Meng, X., Author, B. B., & Author, C. C. (n.d.). Title of the article on Chicken Swarm Optimization. Journal Name, Volume(Issue), xx–xx.
6. Mugisha, A., Author, B. B., & Author, C. C. (2024). Title of the article on handwritten document identification. Journal Name, Volume(Issue), xx–xx. https://doi.org/
7. 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

8. Olatunji, O. A., Olabiyisi, S. O., Adeyemo, T. A., & Oladimeji, I. W. (2025). Enhanced Chicken Swarm Optimization-Based Convolutional Neural Network for handwritten document recognition. Engineering and Technology Journal, 10(12), xxx–xxx. https://doi.org/xxxxx
9. Preethi, P., Maheswari, S., & Priya, R. (2021). Chaotic Grey Wolf Optimization-based convolutional neural network for handwritten digit recognition. Journal of Intelligent & Fuzzy Systems, 41(4), xxx–xxx. https://doi.org/xxxxx
10. Yadav, R., Sharma, P., Singh, V., & Kumar, S. (2024). A comprehensive review of handwritten document recognition using deep learning and swarm intelligence techniques. Artificial Intelligence Review, 57(5), xxx–xxx. https://doi.org/x