C. Ratanavilisagul, "Modified Particle Swarm Optimization for Solving Handwritten Digits," 2024 4th International Conference on Computer Communication and Information Systems (CCCIS), Phuket, Thailand, 2024, pp. 52-58, doi: 10.1109/CCCIS63483.2024.00017.

Modified Particle Swarm Optimization for Solving Handwritten Digits

Chiabwoot Ratanavilisagul
Department of Computer and Information Sciences, Faculty of Applied Science
King Mongkut's University of Technology North Bangkok (KMUTNB)
Bangkok, Thailand
chaibwoot.r@sci.kmutnb.ac.th

***Abstract*— The Handwriting Digits Recognition Problem is a classification problem from interpreting handwritten digit inputs from various sources, such as paper, photos, touch screens, and other devices. This problem is notably challenging and complex due to the uniqueness of everyone's handwriting, coupled with several crucial characteristics that affect the interpretation process. Many researchers have dedicated their efforts to addressing this challenge. Recent research has made significant strides by enhancing preprocessing and feature extraction techniques, effectively applies with particle swarm optimization to tackle the handwritten digit recognition problem. The results of these experiments have proven to be promising. However, it's important to note that the particle swarm optimization has its limitations, particularly problem of trapping into local optimum. If we can mitigate or resolve the issue problem of trapping into local optimum within the particle swarm optimization, it has the potential to create superior models. These models can yield higher Recognition Rates during both training and testing phases. Consequently, this paper proposes the incorporation of a mutation technique into the particle swarm optimization. This technique aims to reduce or eliminate problem of trapping into local optimum, ultimately leading to the creation of superior models. When compared to non-mutation techniques, the proposed approach demonstrates higher Recognition Rates when tested with the original MNIST datasets and individual handwritten digit datasets.**

Keywords*— *Handwritten Digits Recognition, Particle Swarm Optimization, Mutation Operation, Trapping in Local Optimum, K-Means component

I. Introduction

Handwriting recognition (HWR) is the computer's capacity to comprehend and interpret human handwriting, often found in sources like paper documents, photographs, touch screens, and more. Handwritten Digits Recognition (HDR) is a specific aspect of HWR, focusing solely on interpreting handwritten numbers, such as digits from 0 to 9 [1]. This problem poses a significant challenge in the field of pattern recognition due to the inherent variability in people's handwriting. Each individual's handwriting carries unique traits influenced by personal style and conditions at the time of writing. Consequently, even if the same individual writes the same number twice, the results might not match [2]. Notably, there are numerous defining characteristics in handwriting [3], including aspects like line quality, spacing, height, width, font size, stroke variations, stroke connections, slant, pen angle, and other writing conditions. These intricacies make the task of HDR particularly demanding. The Modified National Institute of Standards and Technology database (MNIST) [4], [5] is a well-known handwritten datasets and it is used to compare classification performance of algorithms. Over the years, the MNIST dataset has spurred extensive research, contributing to the development of more efficient HDR methods.

Predominantly, Handwritten Digits Recognition (HDR) is typically addressed using Deep Learning Algorithms (DP) [6-8]. Nevertheless, implementing DP in practical application is unsuitable due to its substantial data requirements, extensive computational resources, and time-intensive nature. Consequently, recent research efforts have been directed towards mitigating these time and resource constraints to enhance DP's effectiveness in tackling HDR, as outlined below: In their study, R. Dixit, R. Kushwah, and S. Pashine [6] conducted a performance comparison of models generated using Convolutional Neural Networks (CNN) and Deep Learning Algorithms (DP), Multi-Layer Perceptrons (MLP), and Support Vector Machines (SVM) with the MNIST datasets. The findings from their experiments revealed that DP achieved a high level of accuracy in modeling; however, it was associated with significantly longer training times. S. Bhattacharjee, M. B. U. Sifat, J. B. Kibria, N. S. Pathan, and N. Mohammad [7] made enhancements to Deep Learning Algorithms (DP). These improvements resulted in DP achieving higher accuracy, all while significantly reducing training times and using fewer computational resources compared to various other machine learning models. J. Subhedar and A. Mahajan [8] have summarized that a significant cons of deep learning must use the substantial volume of data that is needed for training the model. W. Liu, J. Wei, and Q. Meng [9] presented the fundamental principles and performance comparison of four common algorithms for recognizing handwritten digits, including K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Back-Propagation (BP), and Convolutional Neural Networks (CNN). The experimental results reveal that CNN achieves the highest recognition rate with MNIST datasets. However, it's important

to note that the performance of these methods can be influenced by the dataset size. When the dataset is small, it can impact the accuracy of the models. Conversely, with larger datasets, the training process becomes more resource-intensive, affecting runtime. In practical applications of HDR, adding new individuals with their unique handwriting styles need to be accommodated. Training models on data from new individuals requires significant amounts of character writing and entails substantial computational costs and time-consuming efforts on the part of the new individuals.

Recently, many researchers have suggested employing heuristic algorithms to address the challenges in HDR and related problems. Heuristic and Meta-heuristic algorithms like Particle Swarm Optimization (PSO) [10-13] and Ant Colony Optimization (ACO) [14], [15] have proven to be effective in addressing highly complex problems. PSO, in particular, has shown promise and is well-suited for solving HDR [11], [12]. PSO draws its inspiration from the behaviors of flying birds and their methods of communication [16]. It is a stochastic global optimization technique employed for solving optimization problems with the primary objective of finding the global optimum of a fitness function. PSO boasts several advantages, including rapid convergence, simplicity, and minimal parameters to adjust. However, it comes with the drawback of a heightened risk of becoming trapped in local optima [10], [17]. Notably, technologies like Pytesseract [18], Google Lens [19], and Apple Live Text [20] have evolved from Deep Learning (DP). In their study, J. Samleepant and C. Ratanavilisagul [12] experimented with these technologies using datasets of individual handwritten digits contributed by thirty-five volunteers. These datasets had never been subjected to prior training. Their experimental results revealed that, without training, DP exhibited subpar predictive performance. In comparison, when pitted against algorithms emphasizing swift learning and minimal resource consumption like PSO, the algorithm proposed in their paper, which employed PSO for training, yielded superior results, including higher Recognition Rates.

As previously mentioned, the advantages of PSO include swift convergence and simplicity. When applied to the task of HDR, PSO facilitates rapid model training. Therefore, PSO is a suitable choice for practical applications in HDR. Consequently, this paper centers its attention on the utilization of PSO for HDR problem-solving. In recent times, researchers use of PSO for HDR problem-solving, including: N. O. S. Ba-Karait and S. M. Shamsuddin [11] employed PSO to tackle the HDR. They encoded HDR as a multi-dimensional feature space and employed PSO to seek the best positions for each feature space's centroid. Their experiments using the MNIST dataset demonstrate that PSO is effective in addressing HDR challenges. J. Samleepant and C. Ratanavilisagul [12] made adjustments to the feature extraction process to facilitate the use of PSO in solving the HDR. Pre-processing methods were applied prior to PSO. This pre-processing involved steps such as noise removal, image segmentation, thinning processing, and checking the pixel chain code (TCCC) to extract features from the images. Subsequently, K-means clustering was used to categorize the data. PSO was then employed to search for the optimal positions of each feature space's centroids. This searching process served as training to create a new model. The proposed method is referred to as PSO with thinning processing and checked pixel chain code processing (PSOTCCC). PSOTCCC was evaluated using both MNIST datasets and datasets containing individual's handwritten digits, achieving a high recognition rate with reduced runtime.

As mentioned earlier, PSO is prone to becoming stuck in local optima [10], [17]. When PSO becomes trapped in a local optimum, the search process comes to a standstill. It's equivalent to reaching the end of the search process with PSO. To address this issue, the mutation operation technique is commonly employed to overcome local optima [10]. Mutation operations introduce diversity into the population, influencing the particles to move or evade local optima. This approach leads to improved search outcomes. Due to the issue of becoming stuck in local optima, the models generated during the training of PSOTCCC are seldom of quality. To obtain superior models, PSO must address and alleviate the problem of getting trapped in local optima. Consequently, this paper suggests an enhancement to PSOTCCC by incorporating a mutation operator into the PSO process (MPSOTCCC) to resolve or mitigate the issue of local optima entrapment and create improved models.

The remainder of this document is structured as outlined below: Section 2 explains the background (particle swarm optimization, Euclidean distance, K-means clustering, PSO with Data Clustering, mutation operation). Section 3 explains the related works (PSOTCCC). Section 4 explains the proposed method. Section 5 explains the experiment results. Section 6 and concludes this research and directions for development this research in future work.

II. BACKGROUNDS

A. Particle Swarm Optimization

The idea behind PSO draws inspiration from the collective behavior observed in natural phenomena, like the coordinated movement of bird flocks or fish schools. PSO belongs to the category of stochastic optimization algorithms utilized for the identification of optimal solutions. The foundational concept of PSO is based on the notion that these animal groups, in their quest for sustenance, communicate and cooperate to locate food sources and guide the entire group towards them. In the context of PSO, these food sources represent the target solutions that particles seek to discover.

Each member of the particle population is referred to as a "particle" or "X". Each particle possesses its unique position and velocity. These individual particles explore the search space based on their velocities, the best position found among all particles (referred to as GBEST), and their own best position (PBEST). PSO commences by initializing particle positions and velocities randomly, and the evaluation of each particle's position is determined through the optimization problem's objective function. During each generation, each particle updates both its position and velocity according to the formula below:

$$V'_{i,d} = \varpi V_{i,d} + \eta_1 \text{rand}()(P_{i,d} - X_{i,d}) + \eta_2 \text{rand}()(G_{i,d} - X_{i,d}) \quad (1)$$

$$X'_{i,d} = X_{i,d} + V'_{i,d} \quad (2)$$

Where $X_{i,d}$ represents the previous positions of the i-th particle in the d-th dimension, $X'_{i,d}$ denotes the current positions of the i-th particle in the d-th dimension, $V_{i,d}$ is indicative of the previous velocity of the i-th particle in the d-th dimension, $V'_{i,d}$ represents the current velocity of the i-th particle in the d-th dimension, $P_{i,d}$ stands for the individual best (PBEST) position of the i-th particle in the d-th dimension, and $G_{i,d}$ signifies the global best (GBEST) position in the d-th dimension. ϖ corresponds to the inertia weight, η_1 and η_2 are acceleration constants, and rand() is a uniformly generated random number within the range of [0, 1]. There's a defined maximum velocity (V_{max}), and if the computed velocity of a particle exceeds this limit, it is replaced with the value of V_{max}.

B. Euclidean Distance

The Euclidean Distance method calculates the spatial separation between two points within a coordinate system. It relies on the principles of the Pythagorean Theorem. The K-Nearest Neighbors (KNN) algorithm extensively utilizes this method as a fundamental component, primarily because it leverages all pertinent data to classify items based on their proximity in a feature space [21], [22]. Its definition is as follows:

$$D = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \tag{3}$$

In the context of Euclidean Distance, D represents the spatial separation between two points. When dealing with two-dimensional space, this distance is essentially the measure between two sets of values (x_i, y_i), where i range from 1 to k, and k denotes the data points involved.

C. K-Means Clustering

The K-means algorithm is a widely used unsupervised learning technique employed to cluster unlabelled data by grouping features. Its primary objective is to minimize the average squared distance between points within the same cluster. In conventional K-means, individual data points are referred to as "objects." A collection of objects forms a "class" or "centroid vector," and both classes and objects possess specific attributes. The classification is achieved by measuring the Euclidean distance between the attributes of objects and classes. The fundamental idea behind K-means can be summarized as follows: an object is assigned to a particular class based on the smallest Euclidean distance between its attributes and those of the class. Standard K-means initiates by randomizing attributes of classes, and the evaluation of each class is determined using the following expression:

$$d(z_p, m_j) = \sqrt{\sum_{k=1}^{N_d} (z_{p,k} - m_{j,k})^2} \tag{4}$$

$$J_e = \frac{\sum_{j=1}^{N_c} [\sum_{\forall Z \in C_{i,j}} d(z_p, m_j) / |C_{i,j}|]}{N_c} \tag{5}$$

In this context, N_d represents the total number of attributes, N_c signifies the total number of classes, Z_p is an object denoted as p, $Z_{p,k}$ corresponds to attribute k within object p, m_j represents class j, $m_{j,k}$ refers to attribute k within class j, $|C_{i,j}|$ is the count of objects belonging to class j, and $d(z_p, m_j)$ indicates the distance between object p and class j. The algorithm can be terminated when the maximum allowable number of iterations is reached. Its practical appeal lies in its simplicity and speed. The algorithm operates as follows: Step 1, determine the number of clusters by designating them as *k* reference points (representing the number of neighboring points). Step 2, allocate each data point to the nearest *k* reference point. Step 3, update the positions of these *k* reference points. Step 4, Continue to iterate through steps 2 and 3 using the updated cluster centers until there is no further movement.

D. Particle Swarm Optimization with Data Clustering

The classic PSO algorithm was originally designed to address benchmark function problems. To adapt PSO for data clustering tasks, modifications are made to the objective function and particle positions. In this context, an individual particle corresponds to a specific class [13]. The fitness of a particle is evaluated using equations (4) and (5). The PSO procedure for data clustering is identical to the standard PSO process. Except, every particle consists of the centroids of all the classes, resulting in a two-dimensional array for X_i, where $X_{i,d}$ can be expressed as $X_{i,j,k}$. Here, 'j' represents the class of each centroid, and 'k' denotes the attributes of each class. Likewise, the velocity of each particle is composed of the centroids of all classes, yielding a two-dimensional array for V_i, with $V_{i,d}$ being rewritten as $V_{i,j,k}$.

E. Mutation Operation

The mutation operation is a concept borrowed from genetic algorithms (GA). It involves randomly adjusting the positions of particles in some dimensions within the population. The PSO algorithms tend to converge towards local optima due to limited population diversity. To counter this issue, a mutation operation is introduced into the PSO process. This addition serves to enhance the diversity of the population, expanding the exploration of the search space and preventing convergence towards local optima. It also heightens the probability of escaping local optima, ultimately enhancing the effectiveness of the PSO, as highlighted in references [23], [24].

III. Relate Works

J. Samleepant and C. Ratanavilisagul modified the feature extraction for applying PSO to create model for solving HDR. The proposed feature extraction used a novel black pixel-by-pixel counting methods. Their proposal method is called PSOTCCC [12]. The processes of PSOTCCC have a total of 4 steps and can summarize as follows:

Step 1: Fig. 1 displays a sample of handwritten digits from an individual. Initially, the original image is cropped to isolate the handwritten numbers within a single frame. Subsequently, the Discrete Wavelet Transform (DWT) technique is employed to extract crucial features, effectively eliminating unwanted artifacts that can interfere with image processing, such as specks and scanning-related dust. Furthermore, image segmentation is used to trim the surrounding white space, ensuring that the numbers are centered. This is important because some input datasets may include images with off-center numbers, located in various positions (top-left, top-right, bottom-left, bottom-right, etc.), which could lead to recognition

errors. Finally, the image is resized to 10x10 pixels, allowing for straightforward analysis of the black pixel count, representing the digital number.

Step 2: Thinning is a technique that simplifies the input data by removing any dots or the outer layer of a number until it becomes a single pixel in width. In other words, it transforms all the strokes or arcs of each input digit into thin objects. The outcome is an array composed of binary values, where 1 indicates a black pixel and 0 signifies a white pixel. The subsequent step involves tallying the black pixels in the resulting array, a method referred to as TCCC, which draws inspiration from the Freeman chain code approach [25]. The process of counting the black pixels is categorized into four types: First, counting 10x2 pixels in each row yields 5 attributes. Second, counting 5x5 pixels in the up-left, up-right, bottom-left, and bottom-right regions yields 4 attributes. Third, counting 10x5 pixels in the upward and downward directions yields 2 attributes. Finally, counting 5x10 pixels in the left and right directions yields 2 attributes. In total, there are 13 attributes in this approach.

Step 3: The outcomes from earlier stages were incorporated into the PSO method for the training phase. This led to the generation of the output model, which represents the optimal position of each individual feature.

Step 4: The model derived from PSO is utilized during the testing phase to assess the recognition accuracy of the model.

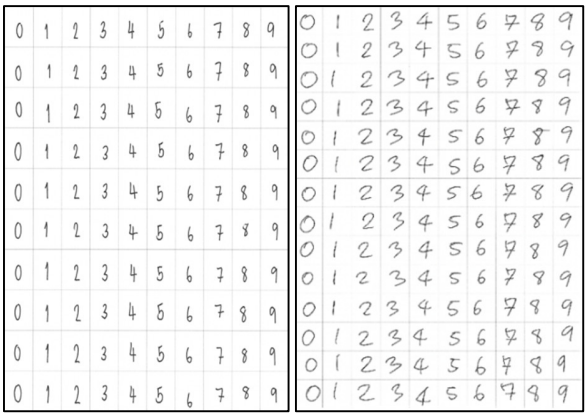

Fig. 1. the individual's handwritten digits dataset example from volunteers.

IV. Proposal works

The PSOTCCC method achieves superior recognition rates when compared to other techniques employing PSO for training. However, PSOTCCC is susceptible to getting stuck in local optima. When the swarm becomes trapped in a local optimum, the solution remains stagnant until the search is completed. It's equivalent to reaching the end of the search process with PSO. PSOTCCC lacks a mechanism for addressing or mitigating issues related to being trapped in a local optimum. To resolve this problem, this research introduces a common approach, the mutation operation, which is integrated into the PSO process to counteract trapping in local optima in the PSOTCCC method.

The introduction of the mutation operation serves to enhance diversity within the population and influences particle movements, thereby enabling them to escape local optima. Consequently, the search outcomes yield improved results. The mutation operation technique has the potential to produce a model with higher recognition rates during the training phase. Such a model, characterized by a high recognition rate during training, is likely to maintain a high recognition rate in the testing phase as well. Therefore, incorporating the mutation operation technique into PSOTCCC leads to the creation of a superior model and achieves higher recognition rates in both the training and testing phases.

The fundamental idea behind the mutation operation is to introduce random adjustments to particle positions, thereby increased diversity within the population [24]. The specific process of each mutation operation varies depending on the behavior of the PSO search. Some problems necessitate more substantial mutation operations because PSO easily becomes trapped in local optima. Conversely, some problems require minor mutations to prevent PSO from getting stuck in local optima [9], [12]. Therefore, the use of the mutation technique is determined by the state of the swarm, particularly when it is either trapped in or trending toward a local optimum. An indicative sign of being trapped in a local optimum is a consistent number of consecutive GBEST values remaining unchanged. Consequently, this paper proposes the introduction of a mutation period (MP) based on the consecutive GBEST values that remain unaltered. Setting a very small MP would cause particles to be reset prematurely, even before they've converged. Conversely, an excessively large mutation period would result in particles remaining stuck in local optima instead of exploring other areas. Therefore, it's crucial to find the right balance for the mutation period, favoring a somewhat larger value because setting it too small would disrupt the PSO search process. In addition, we have the mutation rate (MR), which denotes the number of particles subjected to mutation, expressed as a percentage. A higher mutation rate is favorable because it helps particles break free from local optima. Consequently, both MP and MR are easily adjustable and straightforward to conceptualize when fine-tuning them.

In the case of PSOTCCC, it shares similarities with the challenge of solving data clustering problems. Solving data clustering problems differs from addressing benchmark functions. Typically, PSO encounters issues related to premature convergence and getting stuck in local optima. However, when it comes to solving data clustering problems, PSO faces these challenges to a greater extent than it does with benchmark functions. Therefore, the introduction of mutation in combination with PSO for solving data clustering problems introduces new techniques, including the resetting of GBEST and PBEST, as well as adjustments to the initial velocity and position of particles to encourage better distribution. Additionally, the parameters MR and MP are defined, with MR set to a small value for frequent resets and MP set to a larger value to facilitate wider distribution. Both MR and MP are easily adjustable and have straightforward concepts for tuning. Hence, this paper presents the application of the mutation operation in Particle Swarm Optimization, specifically in the context of thinned processing and checked pixel Chain Code

processing (MPSOTCCC). The core concept of this technique can be explained as follows: In each generation, the PSO process is executed, and if the consecutive number of unchanged GBEST values exceeds MP, the mutation operation is applied to escape from local optima. The pseudo code for MPSOTCCC is provided as follows.

```
Initial position and velocity of each particle according to formula (6) and (8)
Initial PBEST, GBEST, RBEST
For gen = 0 to MAXGEN
    For i = 0 to N
        Evaluate particle fitness according to formula (5)
        If F(Xi) < F(PBESTi)
            PBEST = Xi
        End If
        If F(Xi) < F(GBEST)
            GBEST = Xi
        End If
        If F(Xi) < F(RBEST)
            RBEST = Xi
        End If
        Update particle position according to formula (1) and (2)
    End For
    If F(GBEST) = TGEBST
        CGBEST++
    Else
        CGBEST = 0
        TGEBST = F(GBEST)
    End If
    If CGBEST ≥ MP
            CGBEST = 0
            Reset GBEST, PBEST
            For p = 1 to N
                For j =1 to Nc
                    For k =1 to Na
                        If MR < rand() then
                            Initial position of p particle according to formula (6)
                            Initial velocity of p particle according to formula (8)
                        End If
                    Next k
                Next j
            Next p
    End If
Next gen
```

In the context of these parameters: N_a stands for the number of attributes, N_c represents the number of classes, N_{ch} signifies the count of input data or characters used for training, N denotes the total number of particles, RBEST refers to the best answer that hasn't been reset, $F(X_i)$ corresponds to the fitness of the 'i' particle, $F(PBEST_i)$ is the fitness derived from PBEST for the 'i' particle, $PBEST_i$ designates the PBEST of the 'i' particle, F(GBEST) signifies the fitness of the global best (GBEST), CGBEST denotes the consecutive number of unchanged GBEST values, TGEBST is the temporary fitness used to monitor changes in GBEST, MAXGEN is the maximum number of generations, and $a_{p,j,k}$ represents the value of attribute 'p' for class 'j' and character 'k'.

$$x_{i,j,k} = \frac{\sum_{p}^{N_{ch}} a_{p.j.k}}{N_{ch}} \tag{6}$$

$$V_{max,d} = 0.1 \times (UL_d - LL_d) \tag{7}$$

$$V_{i,d} = -\mathrm{V}_{max,d} + \mathrm{rand}() \times 2 \times V_{max,d} \tag{8}$$

Equations (6), (7), (8) serve as mutation equations and are used with the initial position and velocity of each particle, in both MPSOTCCC and PSOTCCC, as well as in the mutation operation within MPSOTCCC. Equation (6) is employed to determine the initial position of each particle, and this is computed based on the average value of a characteristic of class 'j' and attribute 'k'. Equation (7) is utilized to calculate V_{max} for attribute 'd', with UL_d representing the maximum value for attribute 'd', LL_d representing the minimum value for attribute 'd', and $V_{max,d}$ indicating the maximum velocity for attribute 'd'. Equation (8) is employed to establish the initial velocity of particles.

V. Experiments and Results

A. Datasets

MPSOTCCC and PSOTCCC were subjected to testing using the MNIST datasets, as listed in Table 1, and datasets of handwritten digits contributed by thirty-five volunteers. The MNIST dataset comprises 60,000 training samples and 10,000 test samples, and it can be downloaded from [26]. In the preprocessing stage, the images within the MNIST dataset are converted into binary numeral bitmaps, centered, and size-normalized to 28 x 28 pixels. The dataset containing handwritten digits from individuals consists of 35 datasets, with each dataset corresponding to the work of a different volunteer who filled out an A4-sized paper using a 0.7 mm black gel pen, resulting in numbers ranging from 0 to 9, as shown in Fig. 1. The scanned papers were processed using a clean, dust-free scanner that ensured minimal interference with the model's recognition. The scanned files are in ".BMP" format, and the paper size is 1600x2240 pixels, with each paper containing 100 characters. For training, 90 characters from each row, spanning rows 1 to 9, were utilized, while the remaining 10 characters in row 10 were reserved for testing. The MNIST dataset is employed as a standard benchmark for assessing algorithm performance, whereas the dataset of handwritten digits from individuals is utilized to demonstrate the practical applicability of the results.

B. The Measures of Algorithm Performance

For evaluating the algorithm's performance, I rely on the average results obtained from multiple runs, as both MPSOTCCC and PSOTCCC are stochastic algorithms. The performance metrics used in the experiments are described as follows: The Average Recognition Rate (ARR) Percentage is calculated as the average percentage of correctly identified predictions divided by the total number of data across all runs. ARR serves as an indicator of the algorithm's classification performance. Higher ARR values correspond to better classification performance. SD, which stands for "Standard Deviation," serves as an indicator of the reliability of a clustering algorithm. A smaller SD value signifies a higher level of reliability in the clustering results.

C. Parameters Setting

The parameters used in all experiments are as follows: both acceleration constants, η_1 and η_2, are set to 1.45, ϖ is set to 0.75, and the population size is 5000. When working with the MNIST datasets, 10 runs of experiments are conducted. In the case of the dataset comprising individual's handwritten digits, 10 runs are performed for each paper. The maximum number of iterations is limited to 500. The parameters for the compared algorithms and the proposed algorithm are configured based on the recommendations outlined in the original paper. For the proposed algorithm, MP is initially set to 1, and MR is set to 0.1. These specific values for MP and MR are determined through experimentation.

TABLE I. Detail of MINST dataset

Category	Data for Training	Data for Testing
0	5,923	980
1	6,742	1,135
2	5,958	1,032

3	6,131	1,010
4	5,842	982
5	5,421	892
6	5,918	958
7	6,265	1,028
8	5,851	974
9	5,949	1,009
Total	60,000	10,000

The experimentation process involves starting with MP set at 1 and then incrementing it gradually up to 20, ultimately selecting the values of MP that yield the best results. Similarly, the experiments commence with MR set at 0.01 and are incremented in increments of 0.01 up to 0.2, and the values of MR that generate the best results are chosen. This research is conducted by a personal computer of AMD FX-8320 with 3.5 GHz CPU, 8 GB RAM and Visual C++ 2010 as the programming language.

TABLE II. COMPARATIVE SD AND ARR OF PSOTCCC AND MPSOTCCC ON MNIST DATASETS

Technique	PSOTCCC		MPSOTCCC
Train	ARR	79.20%	87.08%
	SD	1.20	0.84
Test	ARR	72.24%	80.92%
	SD	2.51	1.53

TABLE III. COMPARATIVE SD AND ARR OF PSOTCCC AND MPSOTCCC ON THE INDIVIDUAL'S HANDWRITTEN DIGITS DATASETS

Technique	PSOTCCC		MPSOTCCC
Train	ARR	92.08%	95.06%
	SD	0.97	0.28
Test	ARR	80.03%	84.86%
	SD	7.3	4.73

VI. EXPERIMENT OF PROPOSED ALGORITHM

The results presented in this paper illustrate the performance of MPSOTCCC and PSOTCCC in recognizing handwritten digits, focusing on their classification performance and reliability during both training and testing. The experimental results in Table 2 and Table 3 clearly indicate that MPSOTCCC surpasses PSOTCCC in terms of classification performance, as evidenced by its superior ARR across all datasets, including both the MNIST datasets and the dataset of individual's handwritten digits. Furthermore, MPSOTCCC exhibits greater reliability, as indicated by its consistently lower SD across all datasets. Therefore, MPSOTCCC outperforms PSOTCCC in classification performance and reliability. The experiments validate that the inclusion of the mutation operation can enhance the classification performance of PSOTCCC when dealing with handwritten digit recognition. PSOTCCC tends to encounter the problem of being trapped in local optima, which can occasionally result in poor classification performance. In contrast, MPSOTCCC incorporates a mechanism for addressing or mitigating issues related to being trapped in local optima. Hence, the mutation technique is effective in elevating the classification performance. The data presented in two tables indicate that achieving a high classification performance during the training stage has a significant influence on obtaining high classification performance in the testing stage as well. As a result, MPSOTCCC can create a good model with a high recognition rate in the training stage, which subsequently leads to excellent classification performance in the testing stage.

VII. CONCLUSION

PSOTCCC effectively addresses HDR, yielding promising results. The experimental outcomes highlight that the solutions produced by PSOTCCC outperform those obtained through comparison with other algorithms. Nevertheless, a primary drawback of PSOTCCC is its susceptibility to getting trapped in local optima. This issue negatively impacts the Recognition Rate during the training phase, leading to less-than-desirable model performance and lower Recognition Rates when the model is employed for prediction in the testing stage or practical applications. To address this local optimum trapping problem, this research introduces a mutation operation that is applied alongside PSOTCCC when the population is ensnared in local optima. This new approach is referred to as MPSOTCCC. MPSOTCCC effectively handles and mitigates the local optimum trapping issue, resulting in enhanced classification performance for PSOTCCC. This improvement positively influences the Recognition Rate during the training phase and yields higher-performing models when employed for classification in the testing phase. The experimental results clearly demonstrate that the proposed MPSOTCCC outperforms PSOTCCC in terms of reliability and classification performance across various datasets, encompassing standard datasets and those reflective of practical usage. Looking ahead, there are plans to further develop MPSOTCCC to achieve even higher Recognition Rates in both training and testing stages. Future endeavors will involve the introduction of novel Feature Extraction techniques into MPSOTCCC. Additionally, novel strategies such as resetting or reinitializing will be explored to alleviate or resolve the local optimum trapping issues in PSO, with the ultimate goal of further enhancing the recognition rates.

ACKNOWLEDGMENT

This research was funded by King Mongkut's University of Technology North Bangkok, Contract no. KMUTNB-67-NEW-07

REFERENCES

[1] M. A. Mohamad, H. Hassan, D. Nasien and H. Haron, "A Review on Feature Extraction and Feature Selection for Handwritten Character Recognition," International Journal of Advanced Computer Science and Applications(ijacsa), 6(2), 2015.

[2] H. H. Harralson, E. Waites, and E. J. Will, Handwriting Examination: Theory, Proficiency, and Methods. A Survey of Forensic Handwriting Examination Research in Response to the Nas Report. Retrieved 2013.

[3] M. H. Alkawaz, C. C. Seong and H. Razalli, "Handwriting Detection and Recognition Improvements Based on Hidden Markov Model and Deep Learning," 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA), 2020, pp. 106-110.

[4] Y. LeCun, C. Cortes e J. C. B. Christopher. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[5] L. Deng, "The mnist database of handwritten digit images for machine learning research," IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141-142, Nov 2012.

[6] S. Pashine, R. Dixit, R. Kushwah, "Handwritten Digit Recognition using Machine and Deep Learning Algorithms," International Journal of Computer Applications, Volume 176 , Number 42, 2020.

[7] S. Bhattacharjee, M. B. Uddin Sifat, J. B. Kibria, N. S. Pathan and N. Mohammad, "Recognition of Bengali Handwritten Digits Using Spiking Neural Network Architecture," 2023 International Conference on Electrical, Computer and Communication Engineering (ECCE), Chittagong, Bangladesh, 2023, pp. 1-5.

[8] J. Subhedar and A. Mahajan, "A Review on Recent Work On OCT Image Classification for Disease Detection," 2022 OPJU International Technology Conference on Emerging Technologies for Sustainable Development (OTCON), Raigarh, Chhattisgarh, India, 2023, pp. 1-6.

[9] W. Liu, J. Wei and Q. Meng, "Comparisions on KNN, SVM, BP and the CNN for Handwritten Digit Recognition," 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), 2020, pp. 587-590.

[10] C. Ratanavilisagul and B. Kruatrachue, " A modified particle swarm optimization with mutation and reposition," in International Journal of Innovative Computing, Information and Control Volume 10, Number 6, December 2014, pp. 2127-2142.

[11] N. O. S. Ba-Karait and S. M. Shamsuddin, "Handwritten Digits Recognition Using Particle Swarm Optimization," 2008 Second Asia International Conference on Modelling & Simulation (AMS), 2008, pp. 615-619.

[12] J. Samleepant and C. Ratanavilisagul, "Modified Feature Extraction to apply PSO for solving Handwritten Digits," 2023 20th International Joint Conference on Computer Science and Software Engineering (JCSSE), Phitsanulok, Thailand, 2023, pp. 356-361.

[13] C. Ratanavilisagul, "A novel modified particle swarm optimization algorithm with mutation for data clustering problem," 2020 5th International Conference on Computational Intelligence and Applications (ICCIA), 2020, pp. 55-59.

[14] C. Ratanavilisagul, "Modified Ant Colony Optimization for Vehicle Routing Problem with Time Windows using Limited Search Space and Novel Updating Pheromone and Re-initialization Pheromone Techniques," ICIC Express Letters, Part B: Applications (ICIC-ELB), Vol. 1, pp. 571-580, 2022.

[15] C. Ratanavilisagul, "Modified Ant Colony Optimization with route elimination and pheromone reset for Multiple Pickup and Multiple Delivery Vehicle Routing Problem with Time Window", Journal of Advanced Computational Intelligent and Intelligent Informatics Vol. 26, No. 6, 2022, pp. 959 - 964.

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, 1995, Vol. 4, pp. 1942-1948.

[17] Q. Bai, "Analysis of Particle Swarm Optimization Algorithm," Computer and Information Science, pp.180–184, 2002.

[18] How to OCR with Tesseract, OpenCV and Python, Nanonets [Online]. Available: https://nanonets.com/blog/ocr-withtesseract/

[19] Google Lens, Wikipedia [Online]. Available: https://en.wikipedia.org /wiki/Google_Lens

[20] Handwriting Recognition: More than Just MNIST, [Online]. Available: https://medium.com/nerd-for-tech/handwriting-recognition-more-than-just-mnist-36e68832de73

[21] U. R. Babu, Y. Venkateswarlu and A. K. Chintha, "Handwritten Digit Recognition Using K-Nearest Neighbour Classifier," 2014 World Congress on Computing and Communication Technologies, 2014, pp. 60-65.

[22] T. Makkar, Y. Kumar, A. K. Dubey, Á. Rocha and A. Goyal, "Analogizing time complexity of KNN and CNN in recognizing handwritten digits," 2017 Fourth International Conference on Image Information Processing (ICIIP), 2017, pp. 1-6.

[23] H. Choi, S. Ohmori, K. Yoshimoto, H. Ohtake, "Improvement of Particle Swarm Optimization Application of the Mutation Concept for the Escape from Local Minima," 8th International Conference on Supply Chain Management and Information Systems, 2010, pp. 1-5.

[24] C. Ratanavilisagul and B. Kruatrachue, "A modified particle swarm optimization with dynamic mutation period," 2014 11th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Nakhon Ratchasima, Thailand, 2014, pp. 1-6.

[25] D. Nasien, H. Haron and S. S. Yuhaniz, "The Heuristic extraction algorithms for freeman chain code of handwritten character," 2011 International Journal of Experimental Algorithms (IJEA), 1 (1). pp. 1-20. ISSN 2180-1.

[26] MNIST Dataset, [Online]. Available: https://git-disl.github.io /GTDLBench/datasets/mnist_datasets/

Your Name	Title*	Affiliation	Research Field	Personal website
Chiabwoot Ratanavilisagul	assistant professor Dr.	Department of Computer and Information Sciences, Faculty of Applied Science King Mongkut's University of Technology North Bangkok (KMUTNB) Bangkok, Thailand	Artificial Intelligence, Natural language processing, Machine learning, Searching Technique, Optimization	http://chaibwoot.no-ip.org/Chiabwoot.jsp