CHAPTER 5

OPTIMIZED NEURAL NETWORK FOR INK SELECTION IN PRINTED ELETRONICS

5.1 INTRODUCTION

In the past few years, the landscape of electronics manufacturing has undergone dramatic changes. Traditionally, complex photo-chemical lithography processes were the cornerstone of circuit design. They mostly use materials like silicon or semiconductors. But there has been a paradigm shift with the emergence of PE, which offers a revolutionary approach to circuit design. Basically, PE uses a common photo-printing process to produce a variety of electronic devices to incorporate flow problems into organic and flexible substrates.

The heart of the PE concept is the use of conductive inks it is a unique material that can be printed and processed to conduct electricity. Because it is a basic component of circuit boards and other devices. Conductive inks are therefore considered as the basic component of PE these inks are very capable of producing antennas, contact electrode in the transistor, low resistance circuit connection, and other integrated features. As a result, there is a significant focus on the cost of conductive inks in the PE sector.

The complexities of PE require a comprehensive information of fabric homes, substrates, and other elements considering the fact that numerous method variables without delay impact product satisfactory. Traditionally, the values of those parameters were installed using physics-based totally methodologies, that are often hard, time-ingesting, and vulnerable to inaccuracies. To conquer those limitations and beautify product exceptional, ML models have been increasingly more embraced in the discipline of PE.

This phase of studies work intends to broaden an optimized model to pick out the ideal conductive ink for printing functions by way of thinking about selected input specs. The proposed method entails the construction of a MLPNN for ink choice, accompanied by means of the optimization of weight and bias values using PSO algorithm. Furthermore, the effectiveness of device is evaluated through the variant of network configurations and testing samples, accordingly supplying insights into its efficacy and ability for realistic implementation.

5.2 PARTICLE SWARM OPTIMIZATION

PSO is a stochastic optimization, stimulated by using the collective conduct of swarms in nature, insects, herd, birds, and fishes (Ecer et al.2020) (Tang et al. 2020), which become proposed by Eberhart and Kennedy (1995). These swarms exhibit cooperative behavior in trying to find meals, depending on each member's non-public and other participants' reports, changing their search behaviors for that reason (Wang et al. 2017). In PSO, debris navigate through a seek area, prompted by using their individual reports and the information shared by their buddies. The motion of every particle is encouraged via the positions of close by debris, main to the exploration of promising areas. As particles observe their first-class appearing pals, they collectively converge in the direction of highest quality answers (Li et al. 2021).

Consider a problem with an optimal solution existing in a D-dimensional space, where a swarm of size n is searching. The population can be represented as swarm = {x1, x2,…xn}, where xi(i=1,2,…,n) denotes a particle. Let T_{max} represents the total number of iterations required. The position of i^{th} particle in the t^{th} iteration can be denoted by a d-dimensional vector, $x_i^t = (x_{i1}^t, x_{i2}^t, \dots, x_{id}^t), i = 1,2, \dots, n$ and the velocity of each particle as, $v_i^t = (v_{i1}^t, v_{i2}^t, \dots, v_{id}^t), i = 1,2, \dots, n$. During each iteration, the positions and velocities of particles are adaptively adjusted based on historical optimal fitness values.

The calculation for the (t+1) [th] iteration of the i[th] particle in d-dimensional space can be expressed as:

$$V_{i,d}^{t+1} = \omega V_{i,d}^{t} + c_1 * r_1 * \left(X_{i,d}^{pbest,t} - X_{i,d}^{t}\right) + c_2 * r_2 * \left(G_d^{best,t} - X_{i,d}^{t}\right) \text{.....(5.1)}$$

The new position of each i[th] particle in every dimension is updated as:

$$X_{i,d}^{t+1} = V_{i,d}^{t+1} + X_{i,d}^{t} \text{ .....(5.2)}$$

Where,

ω: Inertia weight

c1and c2: Acceleration coefficients

r1, r2 : Random numbers [0,1]

$X_{i,d}^{pbest,t}$: Historical optimal fitness value of each particle in the optimization process

$G_d^{best,t}$: Global best position

d: Dimension of the search space

Equation (5.1) and Equation (5.2), detail the composition of the velocity in the PSO algorithm, consisting of three components : $V_{i,d}^{t}$ denotes the speed of the particle at the t[th] iteration, $c_1 * r_1 * \left(X_{i,d}^{pbest,t} - X_{i,d}^{t}\right)$ denotes the information of the particle itself, and $c_2 * r_2 * \left(G_d^{best,t} - X_{i,d}^{t}\right)$ is the portion of the particle in the population that is available for exchanging information and working together. In Figure 5.1, the fundamental steps of the PSO algorithm are shown. .

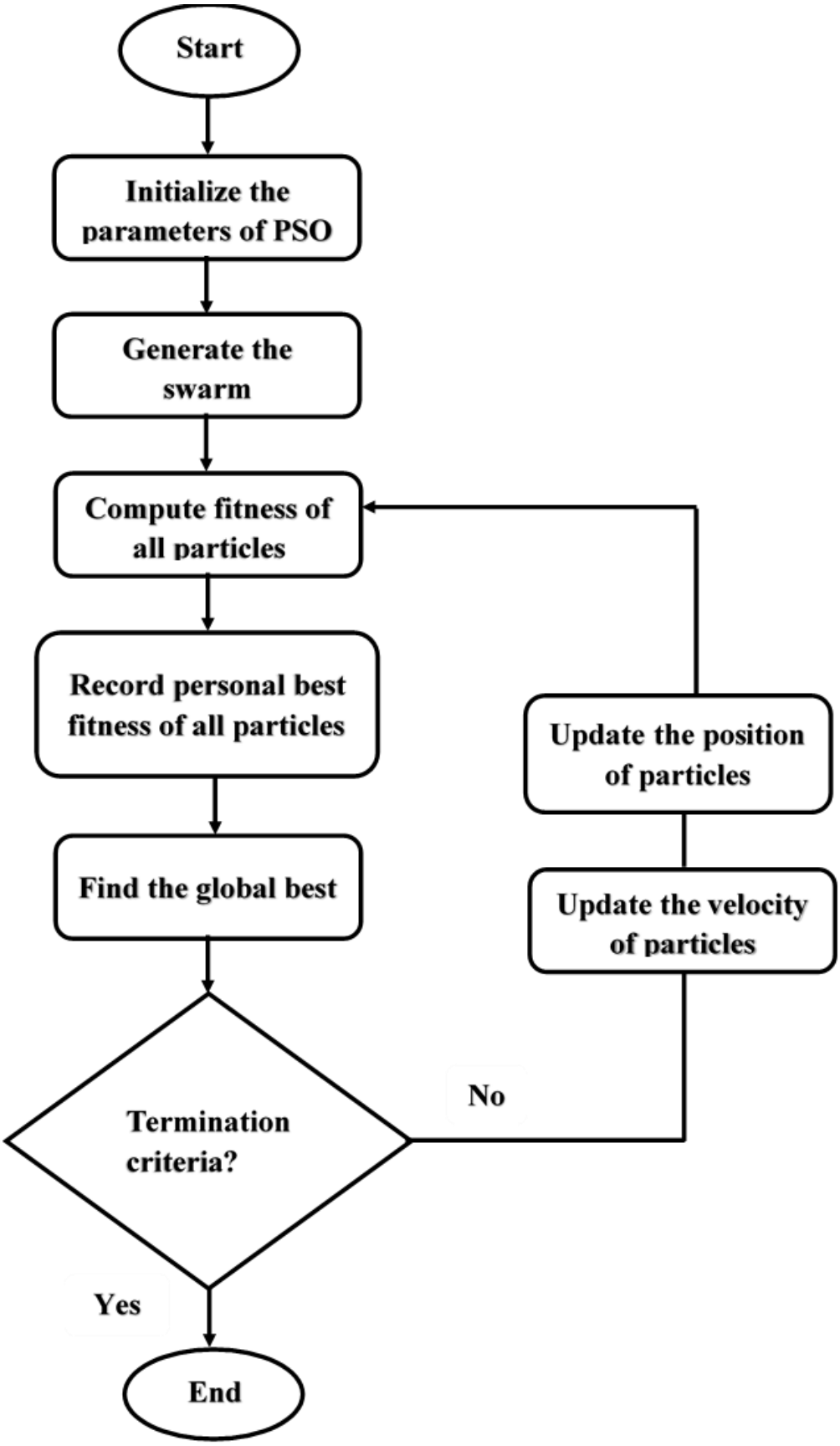

Figure 5.1 PSO algorithm

5.3 INK SELECTION METHOD

This research phase's primary objective is to create an optimal model for choosing conductive ink in printed electronics using NN and PSO algorithm. The schematic representation of the proposed optimized ink selection model is shown in Figure 5.2. The proposed model encompasses four components, which are as follows:

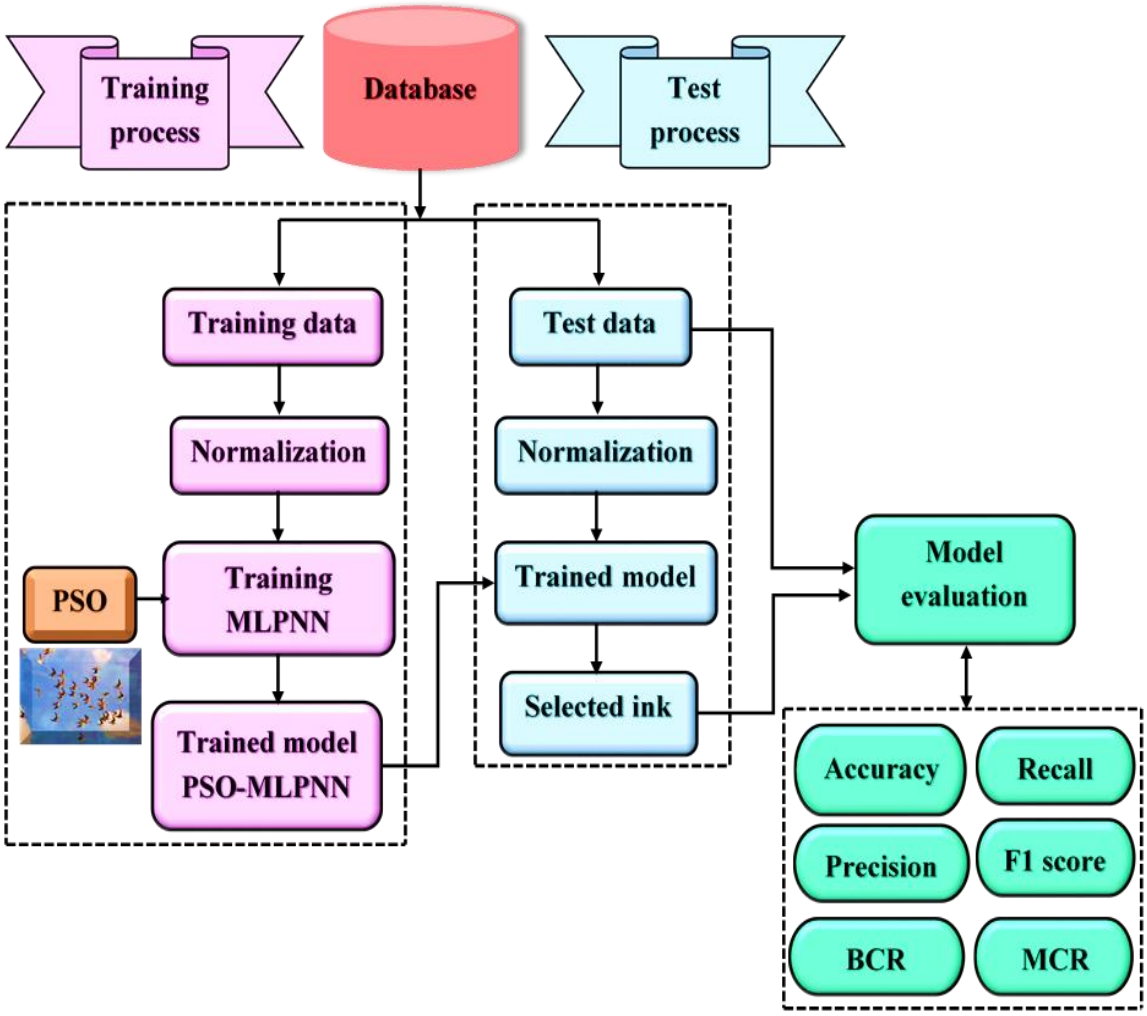

Figure 5.2 Workflow of the proposed system for selecting conductive ink

- ♣ Data collection
- ♣ Data segmentation
- ♣ Data normalization and
- ♣ Development of PSO-MLPNN

Data collection module plays a pivotal role in efficiently gathering all requisite input and output variables essential for subsequent analysis. Following data collection, the data is partitioned into two distinct subsets during the data segmentation phase. The first set, the training dataset serves as the muse for schooling the model, at the same time as the second subset, the testing dataset, is reserved particularly for comparing the model's overall performance. Once segmented, the records undergo preprocessing through statistics normalization, which standardizes the values to a uniform scale ranging between zero and 1, facilitating premiere version overall performance.

During the training phase, MLPNN is used to learn and build complex relationships between input and output variables. This step is critical in providing a model with the necessary capabilities to accurately predict outcomes based on the provided input data. The trained model is then evaluated in the testing phase the performance and prediction accuracy will be tested using a test dataset. Ultimately, the reliability and effectiveness of the model in real-world applications will be determined by this critical evaluation this provides insight into the model's performance in the real world.

5.3.1 Data Gathering

In the field of PE applications, the data collection process serves as the basic step of ANN modeling. Research at this important stage involves focusing on two areas: physical characterization and finding ink with high rates. Low flow MLPNN takes advantage of these material properties to find the most suitable ink for printing. especially The investigation delves into three main ink flow modes: carbon, copper and silver ink, evaluating the characteristics of the various

materials. Comprehensive coverage is essential to ensure the effectiveness of the printing operation. Print performance depends on lifespan, quality, handling and use, GSM, caliper/thickness, tear resistance, brightness, and moisture content.

5.3.2 Data Division

The process of partitioning data is one of the most important steps in modeling the data is divided into two different subsets. Training and testing the Dataset MLPNN is trained over the training dataset and was evaluated using the test data set. Therefore, it provides insights into the predictability.

5.3.3 Data Preprocessing

For MLPNN, the need for data normalization is obvious this is due to the sensitivity of handling input data. This important pre-processing step involves scaling the data to a standard range. It typically lies between 0 and 1. Achieving this normalization is facilitated by using the min-max method. Mathematically, this normalization process can be represented as follows.

$$x_{norm} = \frac{x - \min(x)}{\max(x) - \min(x)} \quad \text{.....(5.3)}$$

Where,

x: Original value

x_{norm}: Normalized value

min(x): Minimum value and

Max(x): Maximum value

This normalization process guarantees uniformity in data display, thus increasing the efficiency and reliability of the MLPNN model.

5.3.4 Design of PSO-MLPNN

In this investigation, the MLPNN consists of an input layer, hidden layers, and output layer. In this framework, the sigmoid activation function is applied at the hidden layer this facilitates non-linear changes in inputs. Conversely, a linear activation function is employed at the output layer, serving to generate output values. The training of the MLPNN is done using the PSO algorithm, as depicted in Figure 5.3. Initially, the weights and bias of the MLPNN are randomly initialized. The training data is sent to the network during the training phase, and the PSO algorithm is deployed to iteratively fine-tune its parameters.

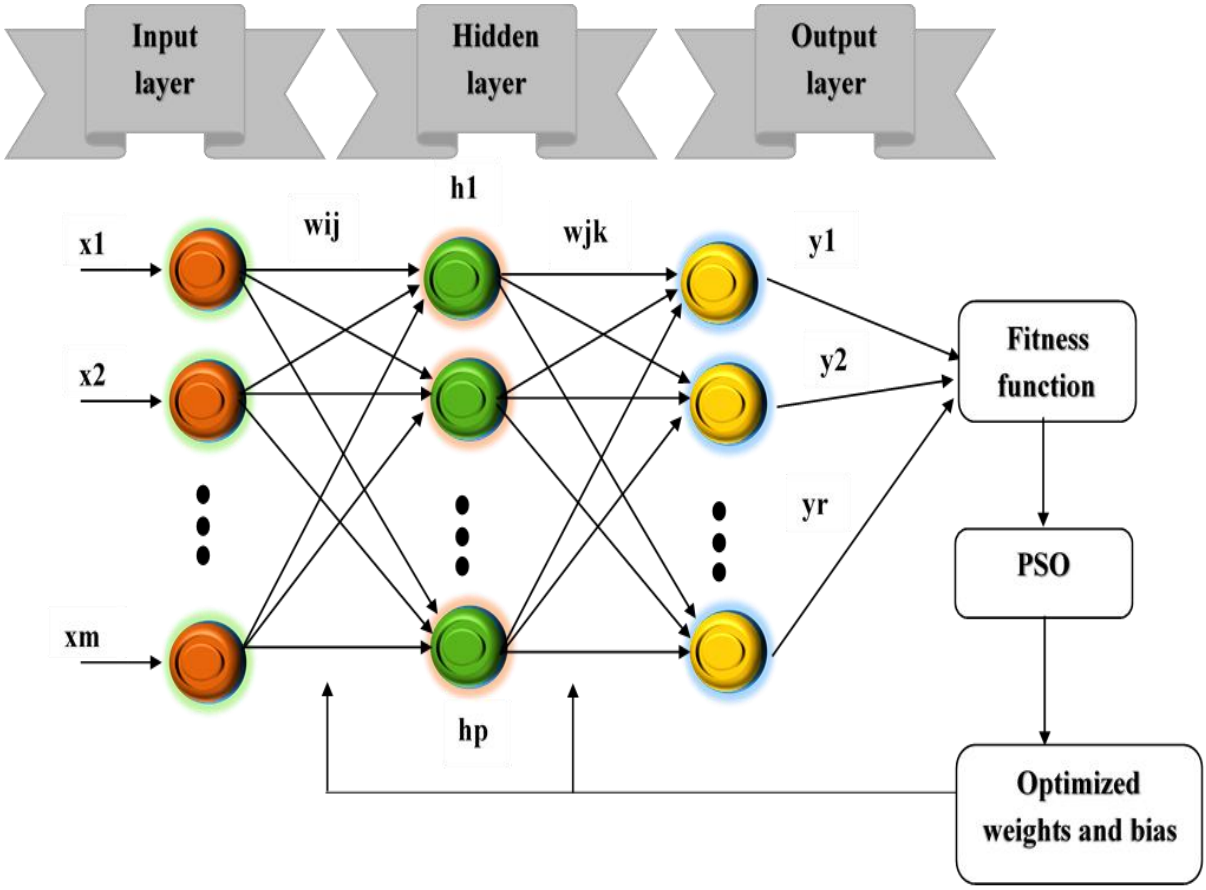

Figure 5.3 Developed PSO-MLPNN model

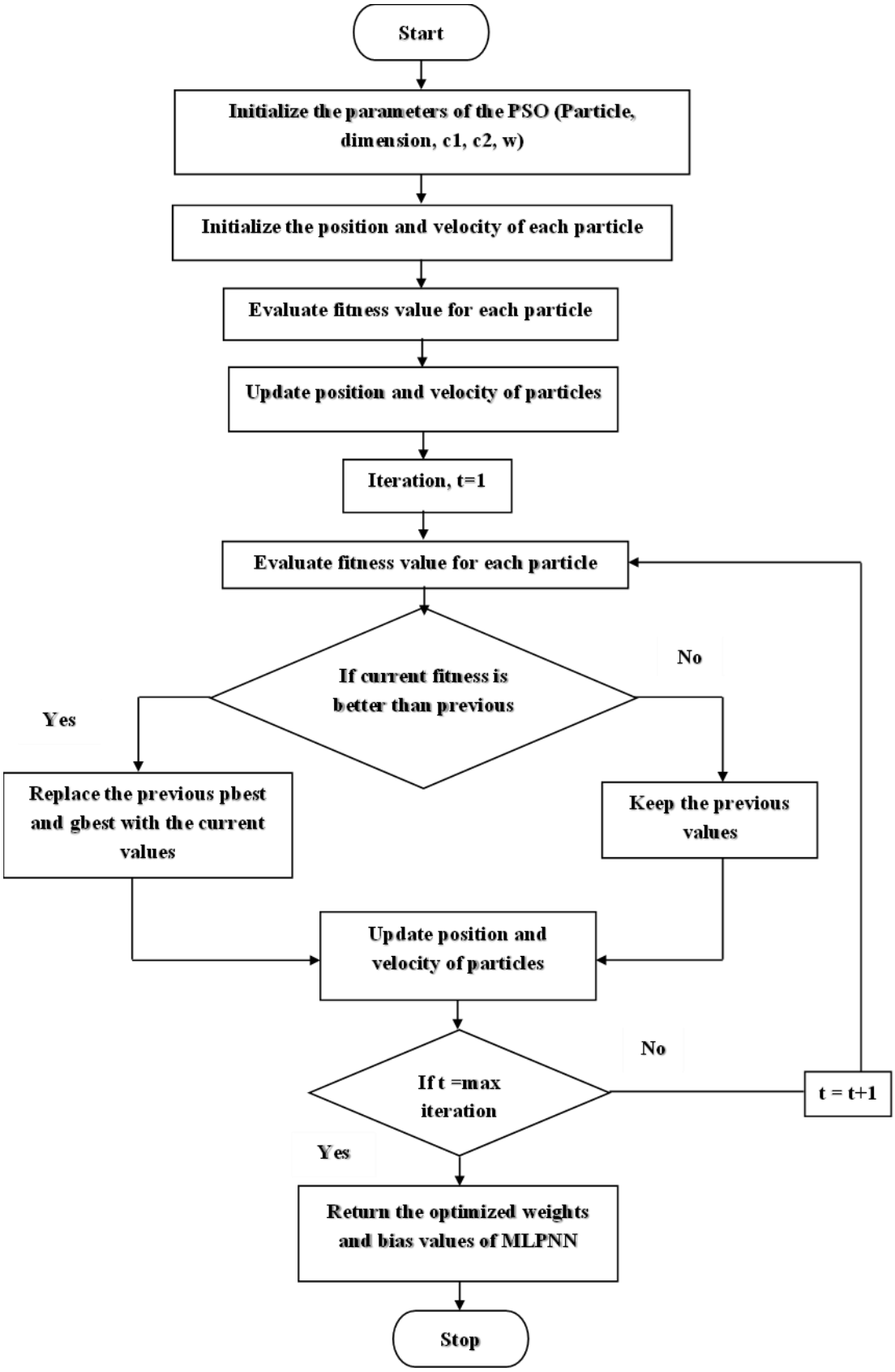

Figure. 5.4 The PSO-MLPNN for ink selection

In MLPNN, optimization focuses on network weights and biases during training. Some training methods minimize an evaluation of the difference between expected and actual labels using a loss function. To optimize the MLPNN's weights and bias, the PSO method is used in this study. Input data moves across the network layer by layer during the training process. with each layer applying activation function to generate outputs. A loss function is used to compare the actual objective values and the model's projections. Based on the error function, the MLPNN parameters such as weights and biases are computed using PSO algorithm.

Throughout the training process, the characteristics of the network are continuously updated based on the input-output values. This repetition continues until the network performance shows a noticeable improvement. Finally, the output from the output layer is compared with the actual value. The reduced difference between the output and the actual value is the ultimate implementation of the model. This improves the model's efficiency in capturing and reproducing complex patterns in the dataset. Figure 5.4 shows a flowchart of PSO-MLPNN.

5.4 EXPERIMENTAL RESULTS

The proposed system is implemented on the MATLAB2022a platform. This section presents a detailed review of the quantitative research results generated by an ink selection system designed for printing. It explores the details and details of the results obtained through experimental testing. From examining the numerical results This section is intended to provide a valuable understanding of the performance and reliability of ink selection systems. This helps in making informed decisions for printing applications.

5.4.1 Evaluation Metrics

The efficiency of the developed model is evaluated using the following variables.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{.....(5.4)}$$

$$\text{Precision} = \frac{TP}{TP+FP} \quad \text{.....(5.5)}$$

$$\text{Recall} = \frac{TP}{TP+FN} \quad \text{.....(5.6)}$$

$$F1 - \text{score} = 2\ X\ \frac{\text{Precision X Recall}}{\text{Precision+Recall}} \quad \text{.....(5.7)}$$

$$BCR = 1/2\ X(\text{Recall} + \text{Specificity}) \quad \text{.....(5.8)}$$

$$\text{Specificity} = \frac{TN}{FP+TN} \quad \text{.....(5.9)}$$

$$MCR = 1 - \text{Accuracy} \quad \text{.....(5.10)}$$

5.4.2 Performance Analysis

The sigmoid activation function is used to create hidden layers and a linear activation function is used to create the output layer, which results in the MLPNN having three layers: an input layer; Hidden layers and output layer The performance of the trained network is evaluated during the testing phase using the test data. For analysis, the mean data from each experiment were recorded after being run many times. To determine the primary design parameters for the PSO-MLPNN system, several tests were conducted. The simulation variable utilized in these experiments are outlined in Table. 5.1 for MLPNN.

Hyperparameters are those that are predetermined and not learned during the training procedure. The learning rate, layer count, and activation functions are examples of hyperparameters. To achieve optimum model performance, hyperparameters optimization aims to identify the ideal set of hyperparameters to use. In this method, the model's performance is assessed while varying the values

of each hyperparameters. In this work, hyperparameters are fixed by experimentation.

Table 5.1 Parameters of the MLPNN

Parameters	Value
No. of input neurons	8
No. of hidden layers	1 to 4
No. of hidden neurons in each hidden layer	10 to 15
No. of output neurons	3
Hidden layer activation function	Sigmoidal
Output layer activation function	Linear
Training algorithm	PSO

The PSO parameters and the values that correspond to them that must be determined to train the proposed PSO-MLPNN model are explained in detail in Table 5.2. In this model, Cross entropy is used as the objective function to perform tuning of weights and bias employing the PSO algorithm. Through this algorithm, the MLPNN iteratively adjusts its weight and bias values during training, aiming to minimize the objective function. Sample fitness value generated over the iterations are tabulated in Table 5.3. Analysis of Table 5.3 revealed that by the 500^{th} iteration, the fitness value has notably decreased to a minimal value of 0.0843, indicating significant optimization progress. Furthermore, the fitness value's convergence plot compared to the PSO algorithm's iteration count for optimizing the weight and bias values of the MLPNN network is shown in Figure 5.5.

Table 5.2 Simulation parameters of PSO algorithm

Variables	Value
No. of particles	25
c1	1
c2	2.5
W	0.1
r1 and r2	0 to 1
Maximum iteration	500

Table 5.3 Fitness function evolved during iterations

Iteration	Fitness value
1	0.9325
50	0.1406
100	0.1299
150	0.1182
200	0.1110
250	0.1036
300	0.0985
350	0.0928
400	0.0883
450	0.0859
500	0.0843

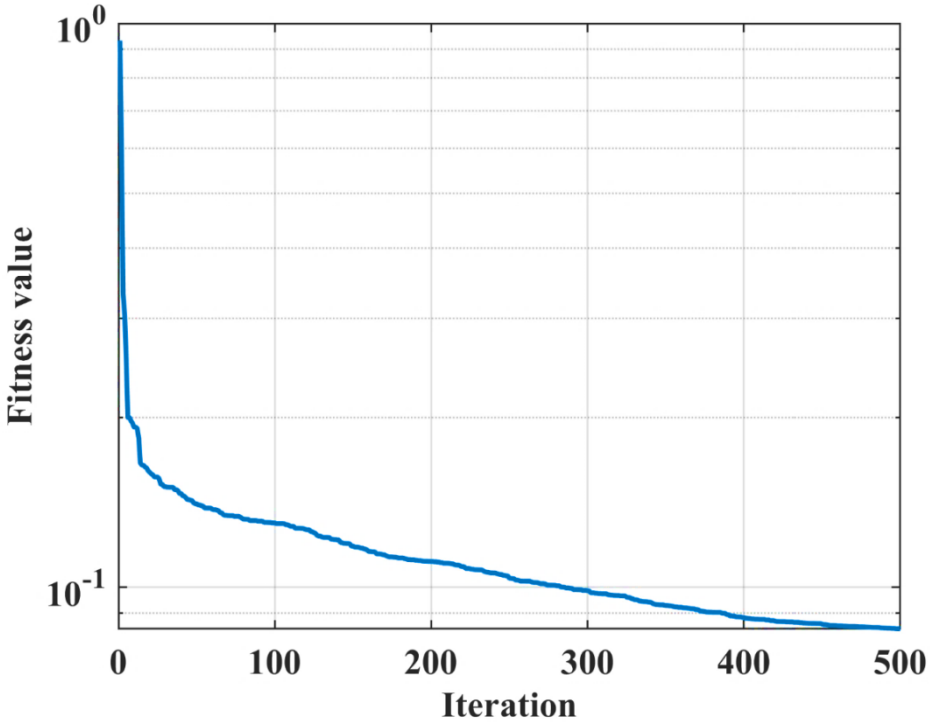

Figure 5.5 Convergence plot with respect to number of iterations

Three different situations were taken into consideration to assess the efficacy of the PSO-MLPNN system:

- **Case 1:** Variations in the number of hidden neurons were used to analyse the PSO-MLPNN's performance.

- **Case 2:** The system's effectiveness was assessed by changing the quantity of concealed layers.

- **Case 3:** The PSO-MLPNN's efficacy was evaluated using varying quantities of training and testing data.

Case 1: Performance analysis by varying the number of hidden neurons

In the Experiment 1, the PSO-MLPNN's efficacy through systematic variations in the number of hidden neurons within the hidden layer. This exploration involved adjusting the quantity of hidden neurons to gauge the PSO-MLPNN's capacity in capturing and modelling intricate relationships within the dataset, thereby elucidating the optimal configuration to superior performance. Furthermore, efficacy of the PSO-MLPNN was compared against MLPNN model.

In this experiment, the remaining twenty percent of the samples were used for testing, while eighty percent of the samples were designated for training purposes. The MLPNN was constructed using an input layer, a single hidden layer, and an output layer. All three layers were included in the creation process. The investigation spanned hidden neurons ranging from 10 to 15. The designed MLPNN underwent separate training phases utilizing both LM and PSO algorithms. The first experiment's key results were compiled in Table 5.4. Analysis of Table 5.4 revealed that the PSO-MLPNN's superior performance compared to the standard MLPNN model.

Table 5.4, the results indicate that the single hidden layer with 10 hidden neurons had the lowest accuracy of 70.01%, recall of 82.50%, precision of 75.01%, F1-score of 78.57%, BCR of 64.25%, and MCR of 29.99% for the MLPNN. In contrast, the performance measures of the developed PSO-MLPNN were notably enhanced, achieving an accuracy of 88.19%, recall of 88.57%, precision of 76.68%, F1-score of 83.33%, and BCR of 88.29% in comparison to its MLPNN equivalent. Additionally, PSO-MLPNN attained lower MCR value of 11.81%. Improvements in the classification accuracy were obtained with subsequent increases in the number of neurons, notably from 10 to 12. Using 12 hidden neurons, the MLPNN and PSO-MLPNN performed very well, achieving

86.67% and 94.48% accuracy, 95% and 96% recall, 86.36% and 88.42% precision, and 90.48% and 92.05% F1-score, respectively.

However, further increases beyond 12 hidden neurons, specifically to 13,14, or 15, precipitated a decline in the classification accuracy. Notably, the PSO-MLPNN attained an accuracy of 90.86%, whereas the MLPNN achieved an accuracy of 80% with a single hidden layer with 15 hidden neurons. This comparative analysis clearly highlights the superior performance of the system with 12 hidden neurons. Consequently, the perfect amount of hidden neurons was determined to be 12. Figure 5.5 visually represents the data from Table 5.4, highlighting the noteworthy observation that the model's accuracy was at its lowest when utilizing 10 hidden neurons, and further increments beyond 12 hidden neurons was deemed essential to ensure consistent and improved performance.

Table 5.4 Performance comparison between PSO-MLPNN and MLPNN for varying number of hidden neurons.

No. of hidden neurons	Models	Accuracy (%)	Recall (%)	Precision (%)	F1 score (%)	BCR (%)	MCR (%)
10	**MLPNN**	70.01	82.5	75.01	78.57	64.25	29.99
	PSO-MLPNN	88.19	88.57	78.68	83.33	88.29	11.81
11	**MLPNN**	73.33	82.5	78.57	80.49	68.77	26.67
	PSO-MLPNN	89.14	89.71	80.1	84.64	89.29	10.86
12	**MLPNN**	86.67	95	86.36	90.48	82.51	13.33

	PSO-MLPNN	94.48	96	88.42	92.05	94.86	5.52
13	**MLPNN**	81.67	90	83.72	86.75	77.95	18.33
	PSO-MLPNN	92.19	94.29	84.18	88.95	92.71	7.81
14	**MLPNN**	85	87.5	89.74	88.61	83.85	15
	PSO-MLPNN	91.43	93.14	83.16	87.87	91.86	8.57
15	**MLPNN**	80	85	85	85	77.50	20.00
	PSO-MLPNN	90.86	92.57	82.23	87.1	91.29	9.14

Case 2: Performance analysis by varying the number of hidden layers

Second experiment focused on assessing the performance of the PSO-MLPNN by altering the number of hidden layers within the network architecture. By manipulating the network's depth through the incorporation of additional hidden layers, this phase aimed to investigate the PSO-MLPNN's capability to acquire hierarchical representations and extract intricate features from the input data. The objective was to discern the influence of network depth on the system's overall performance and its ability to generalize effectively.

In this phase of experiment, while maintaining 12 hidden neurons, the number of hidden layers was systematically adjusted from 1 to 4. Each setup was tested to determine model performance, as shown in Table 5.5.

No. of hidden neurons	Models	Accuracy (%)	Recall (%)	Precision (%)	F1 score (%)	BCR (%)	MCR (%)
1	**MLPNN**	86.67	95	86.36	90.48	82.85	13.33
	PSO-MLPNN	94.48	96	88.42	92.05	94.86	5.51
2	**MLPNN**	91.67	95	92.68	93.83	90.30	8.33
	PSO-MLPNN	97.71	95.26	92.93	96.76	97	3.24
3	**MLPNN**	80	90	81.82	85.71	76.70	20
	PSO-MLPNN	92.38	93.71	84.97	89.13	92.71	7.62
4	**MLPNN**	73.33	80	80	80	70	26.67
	PSO-MLPNN	90.48	92	81.73	86.56	90.86	9.52

Table 5.5 Performance comparison between MLPNN and PSO-MLPNN for different numbers of hidden layers

Analysis of the results from Table 5.5. indicated that both the MLPNN and PSO-MLPNN attained their peak accuracy, recall, precision, F1-score, BCR, and MCR values of 91.67% and 97.71%, 95% and 95.26%, 92.68% and 92.93%, and 93.83% and 96.76%, 90.30% and 97%, 8.33% and 3.24%, respectively, while making use of 12 hidden neurons in two hidden layers.

As hidden layers expanded to three or four, model performance measures declined. As a result, it may be inferred that two hidden layers are the optimal number for the PSO-MLPNN. This experiment underscores the importance of carefully considering the depth of the MLPNN when designing model, as excessively increasing the number of hidden layers beyond a certain threshold may lead to diminishing results in performance. The findings advocate for a balanced approach to network depth to optimize both performance and computational efficiency.

Case 3: Performance analysis by varying the number of training and testing samples

Experiment three was dedicated to assessing the effectiveness of the PSO-MLPNN through variations in the number of training and testing samples. By delving the impact of fluctuating training dataset sizes on the learning dynamics and predictive accuracy of the PSO-MLPNN, this experiment provided invaluable insights into the scalability and robustness of the model. Understanding how PSO-MLPNN's performance evolves in response to changes in training data volume is important to measure its practical usefulness and reliability across different datasets. and situations in the real world.

Based on the outcomes of the preceding experiments, wherein the parameters of the developed PSO-MLPNN, such as the number of hidden neurons and hidden layers, were established as 12 and 2, respectively. The third experiment shifted its focus to analyse the efficacy of the system through variations in The number of training models.

Table 5.6. Performance comparison between MLPNN and PSO-MLPNN for different numbers of training and testing samples

Data ratio (%)	Models	Accuracy (%)	Recall (%)	Precision (%)	F1 score (%)	BCR (%)	MCR (%)
50:50	**MLPNN**	85.23	88.89	88.89	88.89	83.70	14.77
	PSO-MLPNN	90.29	92	81.31	86.33	90.71	9.71
60:40	**MLPNN**	86.67	91.67	88.71	90.16	84.17	13.33
	PSO-MLPNN	92.19	93.71	84.54	88.89	92.57	7.81
70:30	**MLPNN**	88.24	91.14	91.14	91.14	86.82	11.76
	PSO-MLPNN	94.67	96.57	88.48	92.35	95.14	5.33
80:20	**MLPNN**	91.67	95	92.68	93.83	90.15	8.33
	PSO-MLPNN	97.71	95.26	92.93	96.76	97	3.24
90:10	**MLPNN**	90	95	90.48	92.68	87.50	10.01
	PSO-MLPNN	93.14	95.43	85.64	90.27	93.71	6.86

By altering the quantity of training and test samples in this experiment, the efficacy of the MLPNN and PSO-MLPNN was evaluated. Between 50% and 90% of the dataset was made up of training samples. Table 5.6 presents the performance parameters of the PSO-MLPNN for various ratios of training and testing data.

The number of training samples increased from 50% to 80%, as Table 5.6 illustrates, the intended system's performance considerably improved. For instance, the MLPNN and PSO-MLPNN showed accuracies of 85.23% and 90.29%, respectively, while using 50% of the test data. Conversely, with 10% testing data, both models exhibited increased accuracy, with the MLPNN and PSO-MLPNN attaining 90% and 93.14% accuracy, respectively. Remarkably, exceptional outcomes were observed when utilizing an 80% training and 20% testing data split, where PSO-MLPNN demonstrated superior performance, achieving higher accuracy of 97.71%, recall of 95.26, precision of 92.93%, F1-score of 96.76%, and BCR of 97%, and lower MCR of 3.24%.

The empirical findings strongly support the superiority of the PSO-MLPNN with a topology of 8-12-12-3, outperforming alternative topologies such as 8-12-3,8-12-12-12-3, and 8-12-12-12-12-3.

5.5 CHAPTER SUMMARY

Creating an ideal model for selecting a conductive ink for printing applications was the primary objective of this research study. In this Chapter, a concise introduction to the PSO algorithm was provided. This Chapter elaborated the data utilized for experimentation, which underwent preprocessing with the min-max method. Segmentation of the preprocessed data was performed to create training and testing sets. To capture the connection between input and output variables, the MLPNN was designed and developed. The network parameters were fine-tuned using PSO algorithm. Experimental data was used to evaluate the performance of the PSO-MLPNN. The number of hidden neurons, hidden

layers, and training, testing samples were all changed to assess the PSO-MLPNN's effectiveness. Empirical evidence shows that the specifically developed system with 8-12-12-3 configuration achieves this. Moreover, the performance of PSO-MLPNN is compared with standard MLPNN to demonstrate the superiority of the model PSO-MLPNN developed.