# CHAPTER 3

# METHODOLOGY IN DATA COLLECTION

In PE, the selection of input and output variables for the success and effectiveness of the printing process and fluency is greatly influenced by the selection of ink.

## 3.1 INPUT VARIABLES

The rationale behind selecting specific input and output variables is based on their direct impact on the operations and performance of the PE. The input variables are as follows.

- ♣ Product life
- ♣ Product quality
- ♣ Product usage and handling
- ♣ Grams per Square Meter (GSM)
- ♣ Caliper/Thickness
- ♣ Brightness
- ♣ Tear resistance
- ♣ Moisture content

***Product life***

The lifespan of PE is an important factor that determines the durability and longevity of the conductive ink used. Longer product life often requires inks that are more resistant to environmental factors such as humidity.

***Product quality***

The quality standard of the final printed product is directly affected by the properties of the conductive ink. Factors such as the electrical conductivity of the

ink adhesion to the surface and resistance to abrasion affect the quality of the product.

***Product usage and handling***

Understanding how to use and handle printed products helps in selecting inks that can withstand specific conditions such as bending or exposure to chemicals.

***GSM***

GSM indicates the weight of the substrate per square meter and affects ink absorption and drying time. It is important to optimize ink coverage and avoid problems such as smearing.

***Caliper/Thickness***

The thickness of the substrate impacts ink penetration and layer formation. Controlling caliper ensures uniform printing and prevents issues like uneven surface or ink pooling

***Brightness***

For applications that require a beautiful appearance or clarity. Ink light plays a role in achieving the desired beauty and contrast.

***Tear resistance***

Where the printed material is subjected to mechanical or controlled stress that may cause it to tear.

***Moisture content***

Humidity control is important to ensure ink stability, prevents problems such as clogging of the print head or deterioration of ink properties over time.

These variables directly affect the performance and durability of PE, making it necessary to consider them in the ink selection process. The goal is to increase printing efficiency by combining these material characteristics as input features. Selecting the right ink based on these parameters ensures that the PE meets quality standards. A long service life and work reliably under various operating conditions. The input variables and their range are given in Table.3.1.

**Table 3.1 Input variables and their range**

| Sl. No. | Input Variables | Range | Units |
|---|---|---|---|
| 1. | Product life- | (1 to 5)<br>(Low to High) | 1 denotes-2-3 months<br>2-3-5 months<br>3-5-7 months<br>4-7-9 months<br>5-9-12 months |
| 2. | Product quality- | (1 to 5)<br>(Low to High) | 1 denotes-poor<br>2-bad<br>3-moderate<br>4-good<br>5-excellent |
| 3. | Product handling & usage | (1 to 5)<br>(Low to High) | 1-denotes-worst<br>2-poor<br>3-moderate<br>4-good<br>5-effiecient |
| 4. | GSM | 0-200 gsm | - |

| | | | |
|---|---|---|---|
| 5. | Thickness | 0-500 mm | - |
| 6. | Brightness | 0-100 % | - |
| 7. | Tear resistance | 0-200mN | - |
| 8. | Moisture content | 0-150gm/m2 | - |

## 3.2 TARGET VARIBALES

The selected target variables are listed below:

- ♣ Carbon ink
- ♣ Copper ink and
- ♣ Silver ink

**Carbon ink**

As it is affordable and works well with a variety of substrates, carbon ink is used extensively in the printed manufacturing sector. It is suitable for applications where moderate conductivity is sufficient.

**Copper ink**

Copper ink offers higher conductivity compared to carbon ink and is often chosen for applications requiring better electrical performance but with considerations for cost and compatibility.

**Silver ink**

Silver ink provides the highest conductivity among the three types and is favoured for high-performance applications where superior electrical properties are paramount, despite being more expensive.

By considering these specific input variables related to material characteristics and the output variables represented by different types of conductive inks, developing a reliable ANN model with the ability to forecast

and optimize conductive ink choices for a range of printing applications is the aim. This data-driven approach ensures informed decision-making, enhances printing efficiency, and contributes to achieving desired printing outcomes.

The companies that have been referenced for their dataset properties are given below:

1.Keertronics (India) Pvt. Ltd., Pune

2.RVL Scientific & Engineering Pvt. Ltd., Lucknow and

3. Printed Electronics House, Bangalore.

## 3.3 EXPERIMENTAL SETUP

The proposed research works have been implemented using image processing toolbox, neural network toolbox, and deep learning toolbox of MATLAB 2022a, released by Mathworks, Inc. The computational tasks were efficiently handled by an Intel Core i5 processor operating at 3.6GHz with 16GB RAM. To ensure robustness and accuracy, extensive validation was conducted using a substantial collection of material characteristics, contributing to the credibility and reliability of the findings.

## 3.4 PROPOSED METHODOLOGY

The proposed methodologies for conductive ink selection in PE are depicted in Figure 3.1. Three approaches have been introduced such as

- ✓ Conductive ink selection using ANN model
- ✓ Conductive ink selection using ANN and metaheuristic algorithm and
- ✓ Conductive ink selection using deep learning model

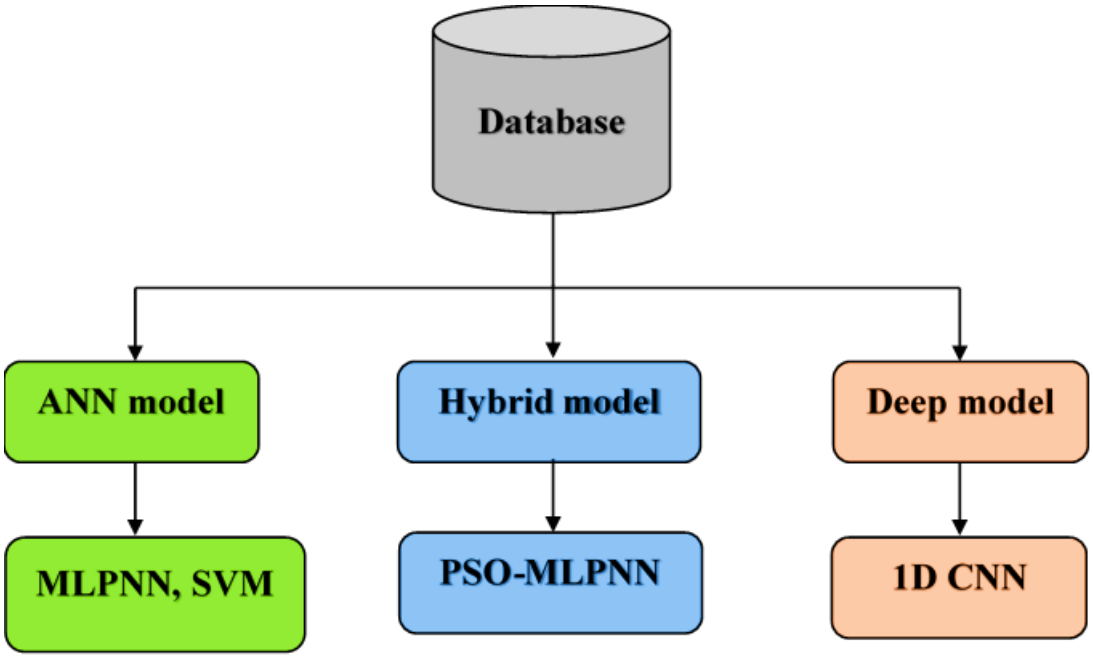


**Figure 3.1 Proposed methodologies for conductive ink selection**

In the proposed methodologies for conductive ink selection in PE, a systematic and innovative approach using ANN, metaheuristic algorithm, and deep learning technique is used to increase the process of selecting ink's precision and efficiency.

In the ANN model, input data is normalized to ensure consistency and optimal performance during training and testing phase. Normalized data is divided into samples for testing and training. This facilitates the evaluation of the model's generalizability. A multilayer artificial neural network (MLPNN) is designed and trained using the algorithm. Levenberg-Marquart (LM) algorithm to select conductive inks for printing purposes. The algorithm is selected based on its ability to efficiently optimize the network parameters. This increases convergence and model optimization. MLPNN is specifically designed to capture the complex relationships between input factors related to ink flow characteristics and desired printing results.

Data preparation method is used within the 2nd approach to enhance the satisfactory of the statistics before it's far divided into datasets for trying out and

training. This is carried out to growth the general great of the records. The MLPNN on this method is educated using PSO set of rules. PSO algorithm, being a metaheuristic optimization method, aids in exploring the answer space successfully and coming across choicest parameter configurations for the MLPNN. This technique makes use of the strengths of PSO and MLPNN to version and expect the complicated relationships inherent in conductive ink residences and printing packages.

The third method introduces a 1D neural network (1D CNN) to select the right ink for a printing application. Unlike MLPNN, CNN excels at capturing patterns. spatiotemporal within the input data this makes it suitable for 1D data in PE. 1D CNNs are trained to identify patterns and features related to ink selection. It is therefore a novel and potentially more suitable approach compared to standard MLPNN.

These proposed methods demonstrate a comprehensive and novel method for conductive irrigation in PE, utilizing the capabilities of deep learning MLPNN. and metaheuristic algorithms. These approaches aim to increase efficiency. increase accuracy and improve efficiency in the conductive irrigation process. This has led to advances in PE technology.

## 3.5 CHAPTER SUMMARY

This Chapter delved right into a comprehensive dialogue regarding the input and target variables taken into consideration. Additionally, it elaborated on the simulation platform hired and placed forth numerous methodologies geared toward facilitating the process of conductive ink selection.

# CHAPTER 4

# AUTOMATED INK SELECTION SYSTEM EMPLOYING ARTIFICIAL NEURAL NETWORK FOR APPLICATIONS IN PRINTED ELETRONICS

## 4.1 INTRODUCTION

Printing methods such as inkjet, offset, spin coating. and screen printing (SP) may be used to fabricate electrophotonic devices. Printed electronics is a subfield of electronics manufacturing and the physical sciences that make this possible. Due to PE's capacity to create flexible electronic devices that are both affordable and huge using traditional printing methods, there has been a lot of interest in the material recently (Secor et al. 2014). According to Reese and colleagues (2004), this technology has several advantages over silicon-based technologies, such as low resource consumption increased workload and easier invention methods. Applications of PE include the use of inkjet and screen printing techniques to produce RFID tags, displays, sensors, and transistors (Aliaga et al. 2011) (Bonnassieux et al.2021).

Squeegee, mesh, and ink are used in the SP method to press a stenciled pattern onto a flat surface. To transfer ink through a fine mesh screen and imprint a pattern onto the surface below, the procedure involves first making a stencil on the screen. Included with the SE are a printed information stencil, a frame, and a screen (Kipphan, 2001). While paper and fabric are common surface for SP, other materials like wood, plastic, and metal can also be used with specialized inks. Screen frames are constructed from aluminum, wood, and steel. The fabric's stencil defines the print image itself. An instrument called a squeegee is used to force ink the stencil is generated on the screen. Usually, it has a wooden handle and is constructed of rubber or polymer. SP finds use in many domains such as posters, plastic bottles, printed circuit boards (PCBs), product displays, and many more.

Conductive inks play a vital role in SP. Several conductive inks are used depending on the application. The quality of the ink has a huge impact on the final product. Therefore, they must be selected carefully according to their intended use. Choosing the right ink is very important in achieving high quality printing results. This is because material properties significantly affect the quality of the final product. This section of research focuses on the key decisions in selecting conductive inks for printing purposes to ensure a flawless printed sheet. If ink is chosen wrongly, an error will occur in the printed sheet. The result can be a deterioration in the quality and appearance of the printed card. For this reason, it is important to create an automated method for selecting the appropriate ink for printing.

The aim of this phase of research is creating a model for ink selection using ANN in printing and to evaluate the performance of the designed model.

## 4.2 ARTIFICIAL NEURAL NETWORK

Machine learning (ML) models that simulate the composition and functioning of real neural networks in the human brain. It has been described as an artificial neural network or ANN (MCCulloch & Pitts, 1943). An artificial neural network (ANN) layer consists of nodes that connect the network. These networks are powerful tools for tasks such as classification, pattern recognition, regression, and decision making. Because these networks are able to identify complex patterns and relationships in data (Jain et al. 1996) (Basheer & Hameer, 2000), ANNs are usually organized into layers. with a large number of neurons per layer, the most common type of ANN is a feed-forward neural network (FFNN), which uses one or more hidden layers to transmit information in one direction from the input layer to the output layer (Ibrahim et al.2022) The general structure of an ANN is shown in Figure 4.1.

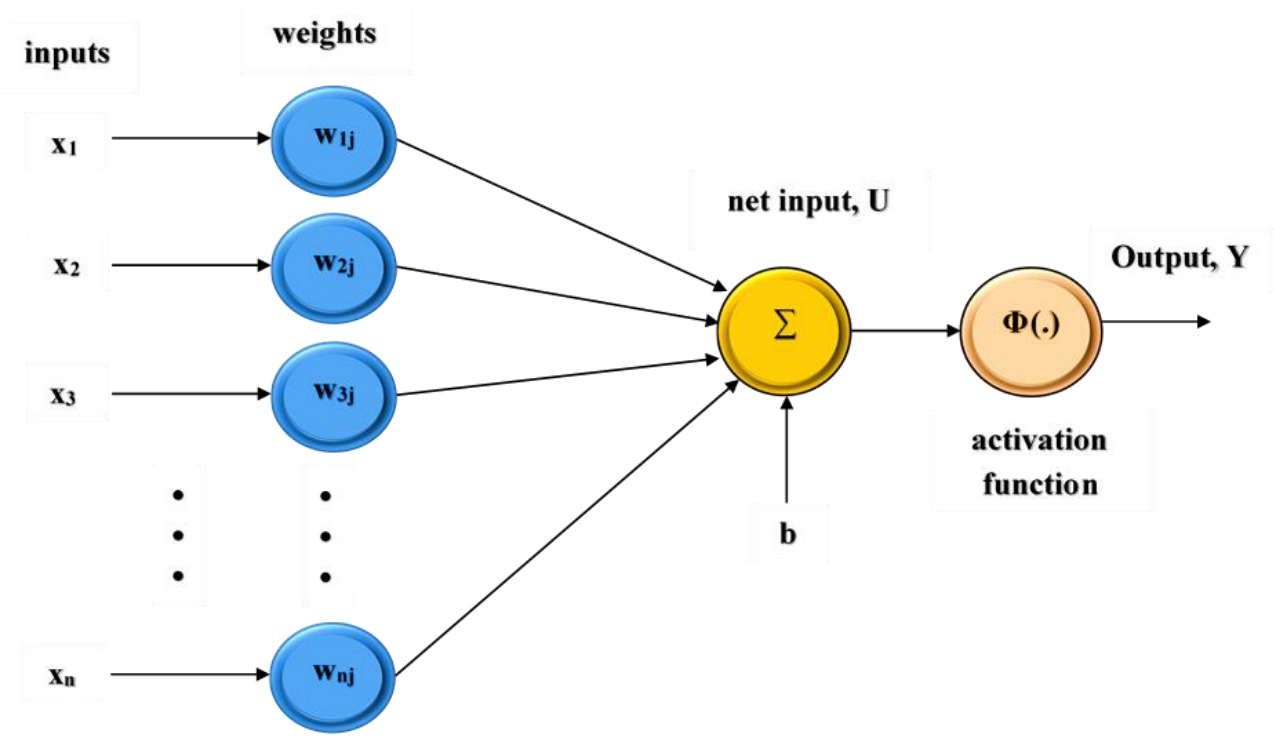


**Figure 4.1 Artificial neural network**

Mathematical functions that take many inputs summarizing and applying weighted activation functions to achieve output are the basic elements of ANN. The weight associated with each input determine the strength of its contribution to the neuron's output, the activation function gives the model more non-linearity, which makes it better able to identify complex relationships in the data. The neuron sums up the inputs multiplied by the weights using Equation (4.1)

$$U = \sum_{i=1}^{n} x_i w_i + b \quad \text{.....(4.1)}$$

$$Y = \varphi(U) = \varphi(\sum_{i=1}^{n} x_i w_i + b) \quad \text{.....(4.2)}$$

Where,

$x_1, x_2, \ldots x_n$ : Input feature

$w_1, w_2, \ldots w_n$ : Weight

b : weight

U : Net input

Φ : Activation function

Y : Output

The process of training an ANN entails using training algorithms to modify the synaptic weights and biases are adjusted to minimize a predetermined loss function. To make the network perform better on the job at hand, this procedure often entails giving it labelled training data and repeatedly modifying the model parameters. ANNs have seen widespread adoption in many areas, including weather forecasting, computer vision, financial modelling, printing applications, and medical diagnosis. They are useful tools for resolving complicated issues in a variety of disciplines because of their capacity to automatically learn from data and generalize to new, unknown data.

The key layers of the ANN are as follows:

**Input layer:** In an ANN, in the network, the input layer functions as the first layer through which data is introduced. Each neuron in this layer corresponds to a distinct element of the data.

**Hidden layers:** Hidden layers are intermediary layers located between the layers in the input and output system. Each hidden layer consists of many neurons it plays an important role in finding complex patterns from incoming data.

**Output layer:** The network generates its outputs, or predictions, the output layer is the final layer of an artificial neural network. In the output layer, the number of neurons is determined by task-specific programming. A single neuron may represent the probability of a single class in binary classification, but in multi-class classification, one neuron would represent each class.

**Neurons:** Before passing the result to the neuron in the next layer, the input data for each APNN neuron is received from the neurons in the layer above it,

transforms it, and then passes it through an activation function. The sigmoid, tangent, and linear activation functions are examples of common functions.

**Weights and biases:** The degree of neuronal connectivity in adjacent layers and the weight assigned to each connection determines the degree of connectivity. Additionally, the term bias is included in almost every neuron. This allows the network to capture more complex relationships between elements.

**Activation function:** The network gains the ability to realize complex mappings between inputs and outputs due to the nonlinearity generated by this function. The expressive capacity of the network is limited. If there is a non-linear activation function because it needs to be organized in a linear model.

**Training:** The process of minimizing a loss function, which calculates the difference between what was anticipated and the labels produced by the training data, is the objective of training neural networks (NNs). This is achieved by using training methods such as gradient descent, Levenberg-Marquart (LM), and Back Propagation (BP).

**Hyperparameters:** The activation function that is used, the learning rate, the number of layers and neurons that are included within each layer, and the training algorithm are only a few examples of the parameters that are chosen before training starts and control many parts of the learning process.

## 4.3 PROPOSED METHODOLGY

The creation of an effective automated system is the main objective of this work. for choosing conductive inks in printed electronics applications, using ANN. Figure 4.2 depicts the proposed system's conceptual layout. There are two main stages to this system:

- Training phase and

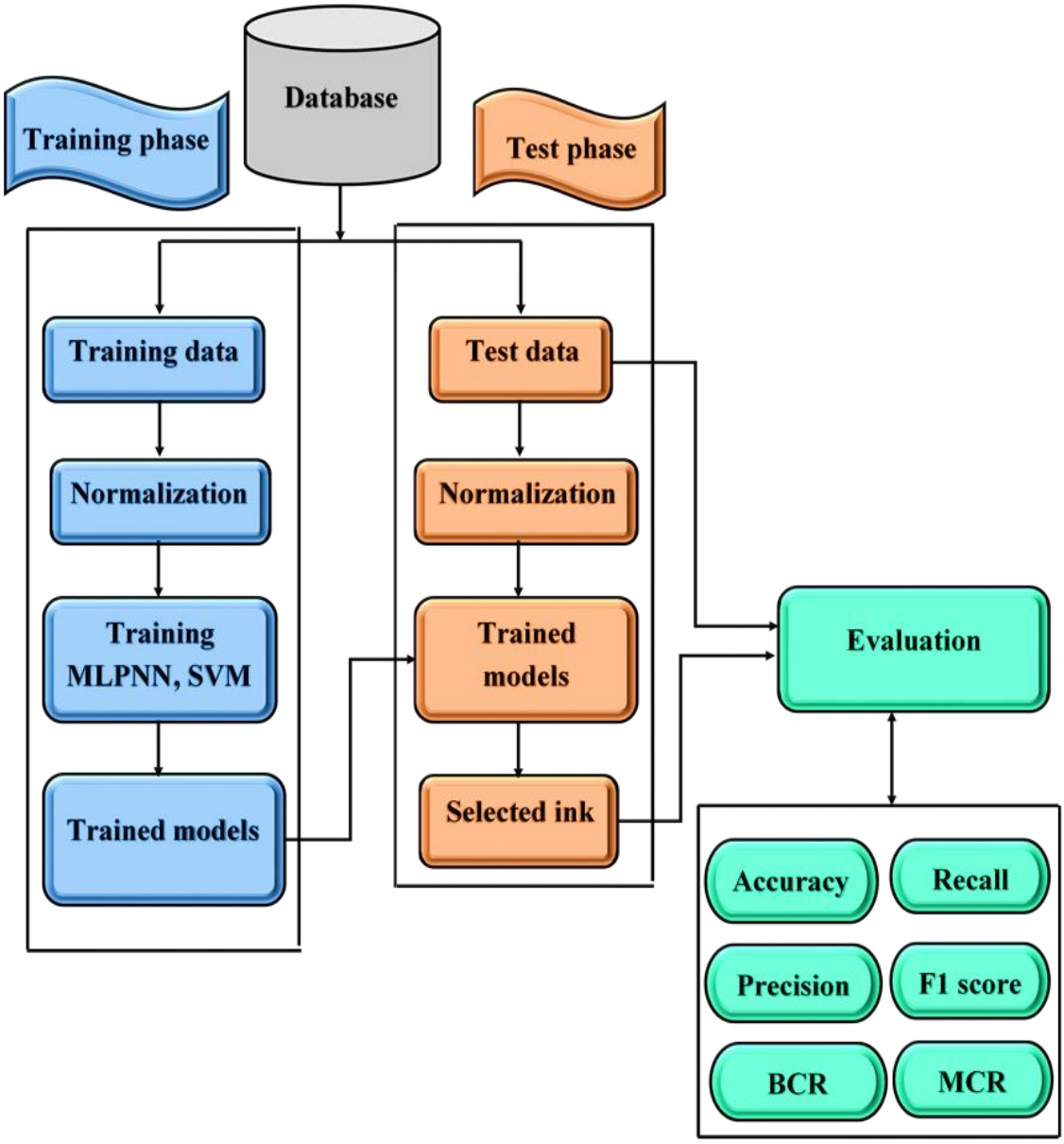


**Figure 4.2 Overall process of the developed ink selection system**

- Testing phase.

In the training phase, the input data is preprocessed to make them fit for analysis. These preprocessed data are then utilized as inputs for two ML models such as SVM and MLPNN. Through the application of the Levenberg-Marquart Algorithm (LMA), the MLPNN is trained to discern and internalize the intricate relationships existing between input and output variables. Similarly, SVM is trained using training data.

During the testing process, the trained model is then used individually to select the appropriate conductive ink using unseen data. This step is critical in evaluating the performance and reliability of the developed system. The performance and robustness of the model were verified by calculating the precision, precision, recall, balanced classification rate (BCR), misclassification rate (MCR), and F1 score, providing insights. There is value in its practicality and ability to be used in the real world.

### 4.3.1 Data Collection

Data collection is critical in designing ANNs for printing applications. ANN models designed specifically for printing applications should pass this important step. Data collection in this research was done through two main methods: material properties and conductive ink.

Material properties act as an important factor in selecting the right ink for a printing application. This is a task facilitated by MLPNN taking into account various material characteristics. To ensure that the printing results are successful. These include:

- Product life
- Product quality
- Product usage and handling
- Grams per square Meter (GSM)
- Calliper/thickness
- Brightness
- Tear resistance and
- Moisture conten

**Table 4.1 Input features and target used in this work**

| INPUT FEATURES | | | | | | | | TARGETS |
|---|---|---|---|---|---|---|---|---|
| **Product life (1 to 5)** | **Product Quality (1 to 5)** | **Product usage and handling (1 to 5)** | **Grammage (gsm)** | **Calliper/ Thickness (mm)** | **Brightness (%)** | **Tear resistance (mN)** | **Moisture content (g/m2)** | **Conductive ink** |
| 2 | 2 | 3 | 78 | 103 | 93 | 63 | 41 | Carbon Ink |
| 5 | 5 | 5 | 101 | 112 | 95 | 68 | 40 | Silver ink |
| 4 | 2 | 1 | 83 | 109 | 92 | 65 | 42 | Silver ink |
| 3 | 3 | 5 | 200 | 250 | 93 | 101 | 18 | Silver ink |
| 4 | 2 | 1 | 110 | 108 | 94 | 72 | 20 | Carbon ink |
| 3 | 3 | 4 | 280 | 261 | 92 | 151 | 19 | Silver ink |
| 3 | 3 | 2 | 98 | 115 | 93 | 121 | 38 | Carbon ink |
| 4 | 2 | 4 | 250 | 255 | 92 | 131 | 39 | Silver ink |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 4 | 2 | 3 | 68 | 115 | 89 | 125 | 37 | Carbon ink |
| 3 | 2 | 4 | 68 | 115 | 89 | 125 | 36 | Carbon ink |
| 3 | 2 | 2 | 62 | 110 | 91 | 84 | 35 | Carbon ink |
| 3 | 5 | 3 | 170 | 100 | 90 | 78 | 40 | Silver Ink |
| 3 | 2 | 3 | 180 | 190 | 50 | 56 | 43 | Carbon ink |
| 3 | 2 | 4 | 100 | 105 | 80 | 86 | 46 | Copper Ink |
| 3 | 2 | 2 | 80 | 102 | 50 | 83 | 41 | Copper ink |
| 2 | 3 | 2 | 100 | 110 | 60 | 92 | 36 | Carbon ink |
| 2 | 2 | 2 | 160 | 171 | 65 | 87 | 38 | Carbon ink |
| 3 | 4 | 3 | 200 | 250 | 98 | 142 | 12 | Silver ink |
| 2 | 4 | 2 | 190 | 208 | 96 | 125 | 15 | Silver ink |
| 2 | 3 | 2 | 100 | 132 | 94 | 115 | 12 | Copper ink |

The study considers three distinct types of conductive inks:

- Carbon ink
- Copper ink an
- Silver ink

A comprehensive list of these important material characteristics is provided in Table 4.1, highlighting their importance in informing the ink selection process and ultimately optimizing print performance.

### 4.3.2 Data Normalization

Considering the sensitivity of ANN to input data, normalizing the input properties and target values is important to ensure consistent and efficient model training (Bielecki et al. 2008) (Lee et al. 2021) (Asesh, 2022). Normalization process of the min-max method obtained by using this mathematically converts the data to a range from 0 to 1 normalization can be expressed as follows.

$$z_{norm} = \frac{z - \min(z)}{\max(z) - \min(z)} \quad \text{..... (4.3)}$$

Where,

Z : Original value

$z_{norm}$ : Normalized value

min(z) : Minimum value and

Max(z) : Maximum value

By using this normalization technique, the data is converted to a standard scale. This allows for more efficient training and convergence within the ANN model. This normalization ensures that all input features and target values are consistently represented. Reduce the possibility of bias or disproportionate influence of the model on learning dynamics.

### 4.3.3 Multilayer Perceptron Neural Network

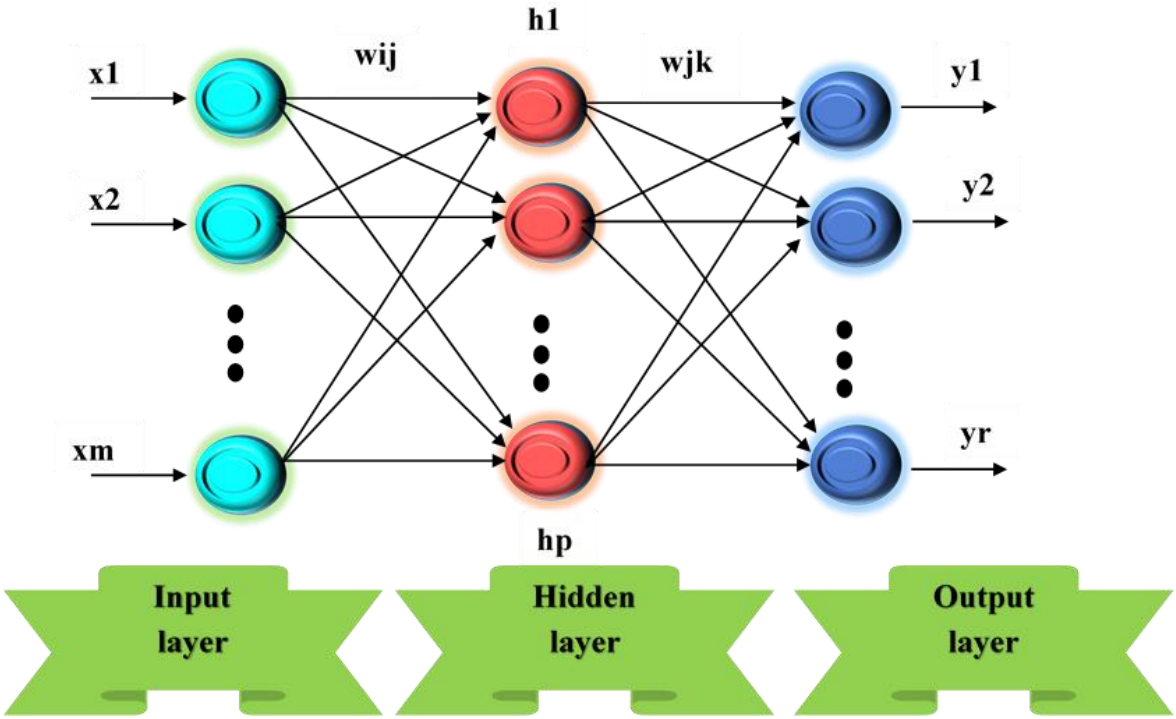


**Figure 4.3 Developed MLPNN**

In the suggested system, the most robust neural network is taken into consideration. Regarding its capacity to map inputs to desired outputs, a single perceptron is limited. This restriction stems from the fact that, despite the type activation function investigated, it only has one neuron for each variable synaptic weight and bias, making it only capable of supporting ridge-like function ((Isabona & Ojuh, 2020) (Isabona et al. 2022)). In using MLPNN with additional source nodes that have hidden layers positioned between data input and output layers, the aforementioned restriction may be addressed. The MLPNN's many neuron layers improve the capacity to map inputs to desired outputs. Hence, this work employs MLPNN to develop an ink selection system.

A MLPNN is a kind of ANN commonly used in ML for supervised learning tasks. A feed-forward neural network is explained in an elaborate manner (Park & LEk, 2016; Vieira et al., 2022; Liu et al., 2023). Due to the

feedforward arrangement of its numerous layers of neurons, the MLPNN prevents cycles from forming in the connections between its nodes. All neurons inside a layer are linked to all neurons within the layer below it, and each connection has a weight link.

Figure 4.3 illustrates how the MLPNN's m input neurons, which represent input characteristics, are linked to the hidden layer by weight (w) for each input ($x_i$). Following the multiplication of input characteristics by initial weights in a weighted sum, the activation function is implemented, and the results are then passed on to the subsequent layer. The expression for the net input that occurs at the jth hidden neuron is:

$$net_{inj} = \sum_{i=1}^{m} x_i w_{ij} + b_j \quad \text{.....(4.4)}$$

Where,

x: input features

$w_{ij}$: the ratio of the hidden neuron, j, to the input neuron, i.

b: Bias

m: Number of input features

The activation function, g, is required to activate the net and cause the $j^{th}$ hidden neuron to produce an output.

$$h_j = g(net_{inj}) = g\left(\sum_{i=1}^{m} x_i w_{ij} + b_j\right) \quad \text{.....(4.5)}$$

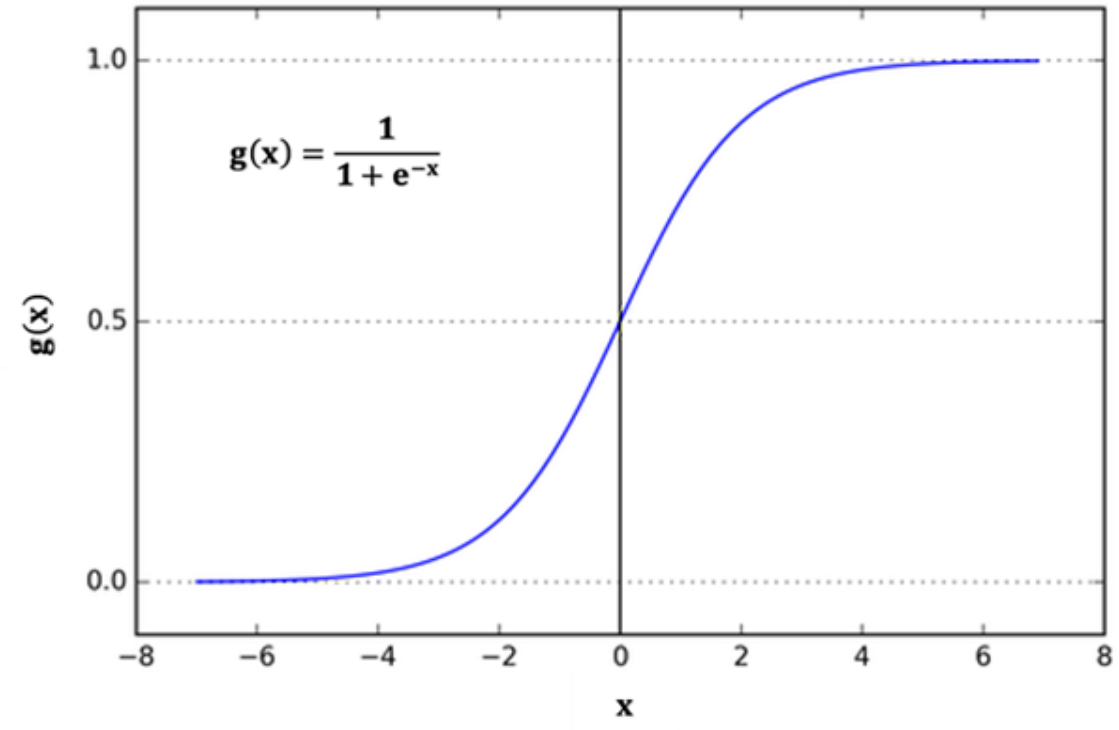


**Figure 4.4 Sigmoidal activation function**

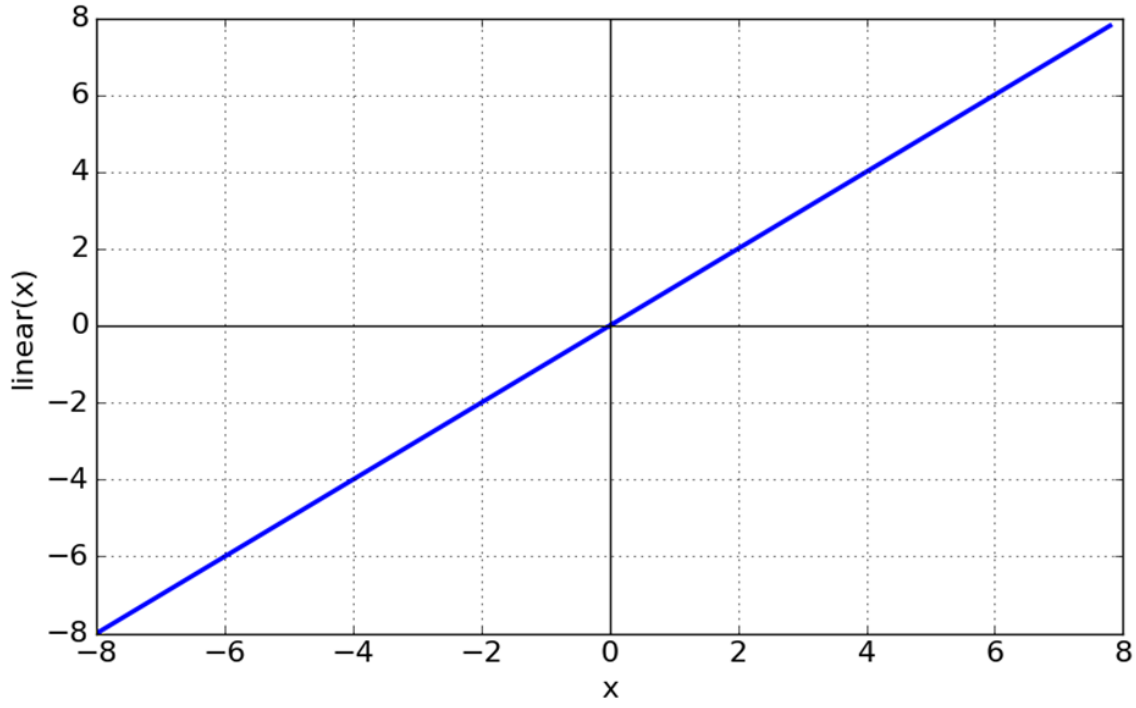


**Figure 4.5 Linear activation function**

Where,

g: Hidden layer activation function

hj: Hidden layer output at $j^{th}$ hidden neuron

The hidden layer of the representation has a sigmoidal activation function. The function that activates the hidden layer, represented by the letter g, is equivalent to

$$g(x) = \frac{1}{1+e^{-x}} \text{ .....(4.6)}$$

Sigmoidal activation function is shown in Figure 4.4.

The output of $k^{th}$ neuron at output layer can be computed as,

$$y_{ink} = \sum_{j=1}^{p} h_j w_{jk} + b_k = \sum_{j=1}^{p} w_{jk}\, g\left(\sum_{i=1}^{m} x_i w_{ij} + b_j\right) + b_k \text{.....(4.7)}$$

Pure linear activation is used to generate output value at the output layer. The expression for the pure linear activation function, f, is as follows:

$$f(x) = x \text{.....(4.8)}$$

Illustrated in Figure 4.5 is the linear activation function.

Equation (4.7) can be rewritten as,

$$Y_k = f(y_{ink}) = f\left(\sum_{j=1}^{p} h_j w_{jk} + b_k = \sum_{j=1}^{p} w_{jk}\, g\left(\sum_{i=1}^{m} x_i w_{ij} + b_j\right) + b_k\right) \text{....(4.9)}$$

Where,

f : Output layer activation function

Y : output

$v_{jk}$ : Weight between $j^{th}$ hidden neuron and $k^{th}$ output neuron

The following equation is used to calculate an error in updating the weights assigned to the output and hidden layers:

$$Error, \delta_k = \frac{1}{r}\sum_{r=1}^{r}(a_k - y_k)^2 \quad \text{.....(4.10)}$$

$$w_{new} = w_{old} + \Delta w \quad \text{.....(4.11)}$$

$$\Delta w = \eta y \delta_k \quad \text{.....(4.12)}$$

Where,

$w_{old}$ : Old weight

$w_{new}$ : New weight

$\Delta w$: Change in weight

$\eta$: Learning rate

The developed MLPNN is trained with LMA. The LMA algorithm is an optimization algorithm primary used for solving nonlinear least squares problems (Kisi & Uncuoglu,2005) (Mun et al. 2022). The basic concept is to use a combination of gradient descent and the Gauss-Newton technique to repeatedly update the parameter vector to reduce the residuals' sum squared. The Hessian approximation can be computed as,

$$H = J^T J \quad \text{.....(4.13)}$$

Gradient, g can be computed as,

$$g = J^T e \quad \text{.....(4.14)}$$

Where,

J : Jacobian matrix

e : Error

To update weight, this LMA makes use of the following approximation to the Hessian matrix:

$$\Delta w = w_{old} - [J^T J + \mu I]^{-1} J^T \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.15)$$

This is just Newton's approach, using the estimated Hessian matrix, where the scalar $\mu$ is zero. A short step-size gradient descent is what happens when $\mu$ is large. It is desirable to switch as soon as feasible to Newton's approach since when it comes to minimal errors, it is faster and more accurate. Gauss-Newton's methods and gradient descent are alternated by the LMA during the training stage. Because of this, $\mu$ is only raised when a tentative step will improve the performance function, and it is dropped after every successful step. Table 4.2 presents the algorithmic stages for training the MLPNN.

**Table 4.2 Algorithmic process of the MLPNN training**

| |
|---|
| **Step 1:** Set the bias (b) and weights (w) to small, random initial values. |
| **Step 2:** Select the maximum iteration number and the learning rate, $t_{max}$ |
| **Step 3:** Select each layer's activation function, each hidden layer has neurons, and there are multiple. |
| **Step 4:** Check for stop condition. If stop condition is false, do steps 4 to 8. |
| **Step 5:** For each training samples, $x_i$, where i=1, 2, …m, do steps 5 to 8 |
| **Step 6:** For each hidden layer, compute output using Equation (4.4) |
| **Step 7:** Apply activation function using Equation (4.6) to get the hidden layer output |
| **Step 8:** Compute the output layer output |
| **Step 9:** Apply activation function using Equation (4.8) to obtain output |

**Step 10:** Compute the error in the output layer and back propagate the error through the network

**Step 11:** Update the network's biases and weights

**Step 12:**Repeat steps 4 to 11 until convergence or for fixed number of iterations.

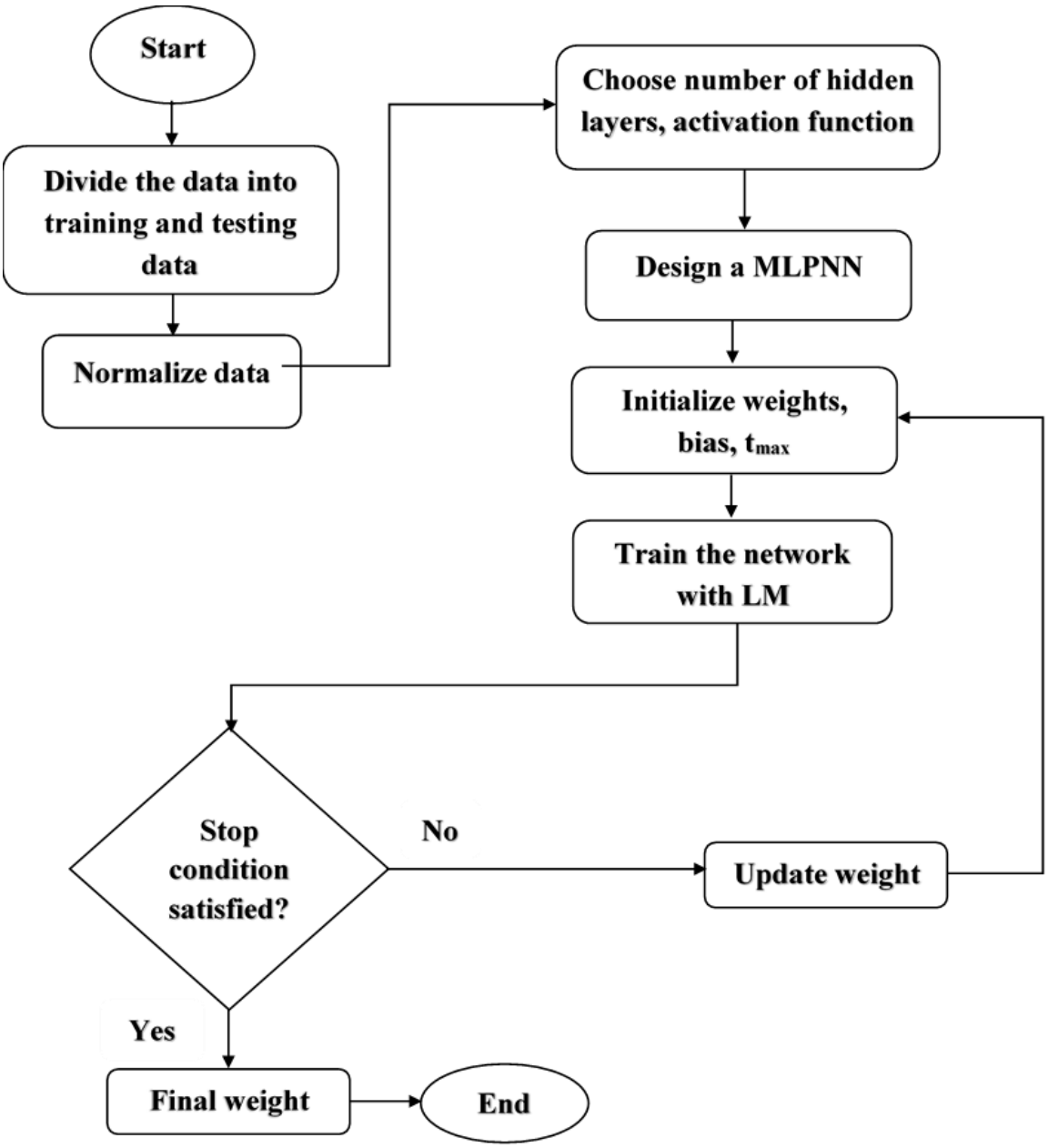


**Figure 4.6 Flowchart of the proposed methodology**

Flowchart of the developed ink selection system is depicted in Figure 4.6. Data is collected and then normalized. Separated from this data are a training set and a testing set. The MLPNN is developed by the inputs and targets. The developed network is trained to capture the intricate relationships between inputs and targets using LMA. Finally, using testing data, the trained network is used to determine which conductive ink to use.

### 4.3.4 Support Vector Machine

Trained ML algorithms include SVM with applications in both regression and classification problems (Cortes & Vapnik,1995). It operates by establishing an optimal boundary, named as hyperplane that effectively separates between different classes within the data space. This optimal boundary is determined based on complex data transformations facilitated by a selected kernel function, aiming to maximize the margin between data points belonging to different classes (Burges,1998).

Consider a problem with multiple classes denoted as $C_1$, $C_2$, $C_3$, …$C_M$. Each of these classes is associated with a set of support vectors:

For class $C_1$: $C_1=s_1,s_2,s_3,\ldots s_n$

For class $C_2$: $C_2=s_1,s_2,s_3,\ldots s_n$

For class $C_M$: $C_M=s_1,s_2,s_3,\ldots s_n$

These support vectors play a significant role in defining the separation boundaries between classes. Specifically, for the $i^{th}$ class, $C_i$, it comprises the sum of support vector from all previous classes ($C_1$ to $C_{i-1}$):

$$C_i = \sum_{k=1}^{M-1} \sum_{j=1}^{n} C_k s_j \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.16)$$

Where,

$C_i$ : A set of support vectors separates $i^{th}$ class form all other (i-1)classes

$C_k$ : $k^{th}$ class

$s_j$ : $j^{th}$ support vector within the class

## 4.4 EXPERIMENTAL RESULTS

This section details the numerical results of printing-specific ink selection methods. It delves into the complexity and uniqueness of the results obtained through testing and analysis. This section attempts to provide important insights into the performance and reliability of ink selection methods by providing a detailed review of the numerical results. This facilitates informed decision making for printing applications.

### 4.4.1 Evaluation Criteria

The performance of the implemented system is evaluated by evaluating various indicators including recall, precision, F1 score, BCR, MCR and precision system performance is measured using a confusion matrix. It consists of four different factors: positive class is correctly identified, while True Negative (TN) indicates cases where negative class is correctly classified. When the system misidentifies the positive class, it produces a False Positive (FP), and when it misclassifies the negative class, it produces a False Negative (FN). It is essential to note that this study involves multiclass classification. The TP class serves as the focal point for the calculation, while the negative encompasses the remaining labels. The ensuring metrics are enumerated below:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{.....(4.17)}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

.....(4.18)

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad \text{.....(4.19)}$$

$$\text{F1} - \text{score} = 2\ \text{X}\ \frac{\text{Precision X Recall}}{\text{Precision+Recall}} \quad \text{.....(4.20)}$$

$$\text{BCR} = 1/2\ \text{X}(\text{Recall} + \text{Specificity}) \quad \text{.....(4.21)}$$

$$\text{Specificity} = \frac{\text{TN}}{\text{FP+TN}} \quad \text{.....(4.22)}$$

$$\text{MCR} = 1 - \text{Accuracy} \quad \text{.....(4.23)}$$

### 4.4.2 Experiment Analysis

Experimental research determined the critical parameters needed to construct a dependable and accurate MLPNN. To decide on the MLPNN's ultimate settings, many trials were conducted. Table 4.3 lists the simulation's settings.

**Table 4.3 Simulation parameters**

| 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 |
| Activation function of hidden layer | Sigmoidal |
| Activation function of the output layer | Linear |
| Epoch | 1000 |
| M | 0.01 |
| Training algorithm | LMA |

The basic design of the MLPNN mostly consisted of one input layer, one hidden layer, and an output layer. A function of linear activation generated output values, while a sigmoidal activation function at the buried layer transformed inputs nonlinearly.

During training phase, the MLPNN was trained with the LMA, a popular training algorithm for NN. The objective of this optimization was to reduce the amount of variance that existed between the actual and anticipated output values, thereby improving the MLPNN's ability to accurately model complex relationships within the data. Subsequently, using unique test data that weren't utilized during training, the performance of the trained MLPNN was evaluated while it was still in the training phase. This division of training and testing data aided in determining the MLPNN's generalization capacity. ensuring that it could predict outputs for unseen data.

The effectiveness and robustness of the designed MLPNN was assessed through a series of three distinctive experiments:

**Experiment 1:** Changing the amount of neurons that are hidden allows one to assess the system's performance.

**Experiment 2:** To evaluate the efficiency of the system, it is advisable to change the quantity of hidden layers.

**Experiment 3:** By changing the quantity of training samples, you may evaluate the system's efficacy.

**Experiment 1**

Experiment 1 involved the systematic analysis of the designed model's performance by adjusting the amount of neurons that are concealed inside the hidden layer. Examining the MLPNN's capacity to represent and grasp intricate correlations in the data by varying the number of hidden neurons shedding light on the optimal configuration for achieving superior performance.

The first experiment used 80% of normalized data for training and 20% for testing. There were differences between 10 and 15 buried neurons. In training, the MLPNN analyses training data repeatedly to determine the connection between the variables that are input and output. Weight is being updated by the data. To choose the ink, the test data was sent into the trained network. A summary of the first experiment's results is shown in Table 4.4. In Table 4.4, the least accurate MLPNN, with 70.01 percent accuracy, 82.50% recall, 75.01% precision, and 78.57% F1 score, the model in issue has a single hidden layer and a total of 10 hidden neurons. The developed system has 86.67% accuracy, 95% recall, 86.36% precision, and 90.48% F1 score for 12 hidden neurons. Using 15 hidden neurons, the algorithm scored 85% F1 score, 85.5% recall, 80% accuracy, and 85% precision. When compared to alternative configurations, the model with 12 hidden neurons had the lowest MCR of 13.33%. The acquired result shows that the system that was built with 12 hidden neurons performed better. Thus, 12 is the value assigned to the hidden neuron. Figure 4.7 shows the pictorial representation of Table 4.4. Accuracy was lowest for the created method while using 10 and 11 hidden neurons. When the number of placed neurons exceeds 12, the system's functioning becomes unstable. To improve the results, hidden neurons were set to 12.

**Table 4.4 Performance of the developed MLPNN for varying hidden neurons**

| Number of hidden neurons | Accuracy (%) | Recall (%) | Precision (%) | F1 score (%) | BCR (%) | MCR (%) |
|---|---|---|---|---|---|---|
| 10 | 70.01 | 82.50 | 75.01 | 78.57 | 64.25 | 29.99 |
| 11 | 73.33 | 82.50 | 78.57 | 80.49 | 68.77 | 26.67 |
| **12** | **86.67** | **95.00** | **86.36** | **90.48** | **82.51** | **13.33** |
| 13 | 81.67 | 90.00 | 83.72 | 86.75 | 77.95 | 18.33 |
| 14 | 85.00 | 87.50 | 89.74 | 88.61 | 83.85 | 15.00 |
| 15 | 80.00 | 85.00 | 85.00 | 85.00 | 77.50 | 20.00 |

**Experiment 2**

When changing the MLPNN structure's number of hidden layers, the experiment's system's performance was assessed. The ability of the system to learn hierarchical representations and extract complex characteristics from the input data was explored in detail by manipulating the network's depth by including more hidden layers. The objective of this experiment was to clarify how network depth affects the MLPNN's overall performance and capability for generalization.

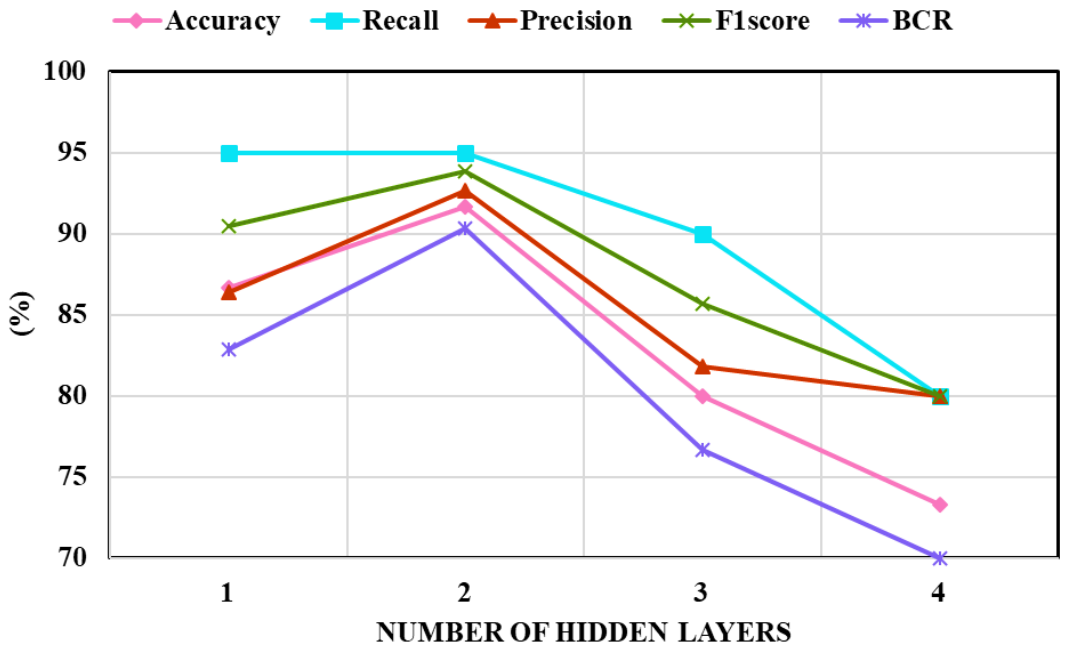


**Figure 4.8 Performance of the designed system for varying hidden layers**

Secret layers were expanded from one to four in the second experiment condition. There were a fixed twelve hidden neurons. Figure 4.8 shows the developed system's performance for a range of hidden layer counts. As shown in Figure 4.8, the system's accuracy for the first, second, third, and fourth hidden layers was 86.67%, 91.67%, 80%, and 73.33%, respectively. The systems with one and two hidden layers had the same recall value of 95%. However, MLPNN with two hidden layers has higher accuracy, F1 score, and precision than single hidden layer architecture. Furthermore, the model yielded MCR values of 13.33%, 8.33%,20%, and 26.67% for hidden layers 1,2,3, and 4 respectively. Additionally, the designed model exhibited MCR value of 13.33% with 1 hidden layer, 8.33% with 2 hidden layers, 20% with 3 hidden layers, 26.67% with 4 hidden layers. The system's performance degraded if the hidden layers were raised to three or four. As a result, two hidden layers are the ideal amount for printing applications.

**Experiment 3**

Experiment 3 focused on evaluating variations in the number of training and testing samples are used to assess how efficient the system. By exploring the influence of varying training dataset sizes on the MLPNN's learning dynamics and predictive accuracy, this experiment provided valuable insights into the scalability and robustness of the MLPNN. Understanding how the MLPNN's performance evolves with changes in the volume of training data is crucial for assessing its practical applicability and reliability across diverse datasets and real-world scenarios.

The MLPNN parameters were modified to 12 for the first testing, then to 2 for the second testing, along with the hidden neurons. The third scenario changes the quantity of training samples to test the algorithm. Training samples ranged from 50% to 90%. For different training data, Table.5 shows system performance. According to Table 5.5, increasing training samples from 50% to 80% enhanced system performance. For 50% training data, the system gave accuracy of 85.23% with recall of 88.89%, precision of 88.89%, and F1score of 88.89%. For 90% training data, the system attained accuracy, recall, precision, and F1 score values are 90%, 95%,90.48%, and 92.68% respectively. The system achieved a higher accuracy of 91.67%, recall of 95.01%, precision of 92.68%, F1 score of 93.83%, BCR of 90.15%, and reduced MCR of 8.33% for 80% of the training data, which indicated exceptional performance. From the empirical findings, the system that was built that had the topology of 8-12-12-3 was found to perform better than the systems that had the topologies of 8-12-3, 8-12-12-12-3, and 8-12-12-12-12-3. In Figure 4.9, the optimal MLPNN structure for ink selection is shown.

**Table 4.5 Performance of the developed system for varying training data**

| Data ratio (%) | Accuracy (%) | Recall (%) | Precision (%) | F1 score (%) | BCR (%) | MCR (%) |
|---|---|---|---|---|---|---|
| 50:50 | 85.23 | 88.89 | 88.89 | 88.89 | 83.70 | 14.77 |
| 60:40 | 86.67 | 91.67 | 88.71 | 90.16 | 84.17 | 13.33 |
| 70:30 | 88.24 | 91.14 | 91.14 | 91.14 | 86.82 | 11.76 |
| **80:20** | **91.67** | **95.00** | **92.68** | **93.83** | **90.15** | **8.33** |
| 90:10 | 90.00 | 95.00 | 90.48 | 92.68 | 87.50 | 10.00 |

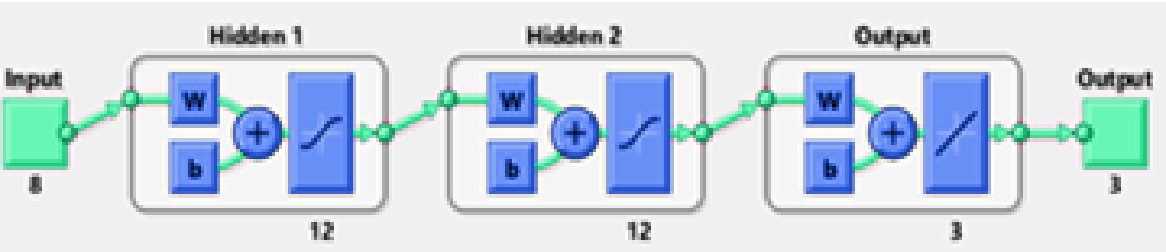


**Figure 4.9 The best MLPNN structure for printing applications**

These experiments served to comprehensively evaluate the MLPNN's performance under varying architectural configurations and data conditions, ultimately contributing to a deeper understanding of its capabilities in real-world applications.

Figure 4.10 illustrates a comprehensive comparison of the efficacy between SVM and MLPNN. As shown in Figure.4.10, it is quite clear that the MLPNN

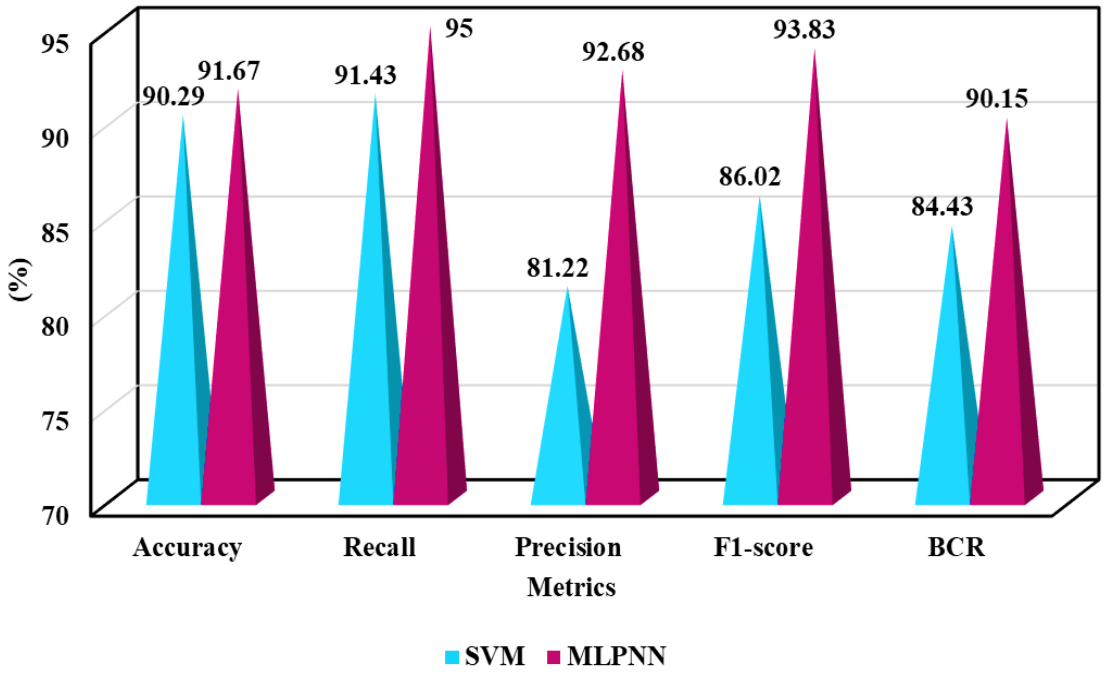


**Figure 4.10 Performance comparison between SVM and MLPNN**

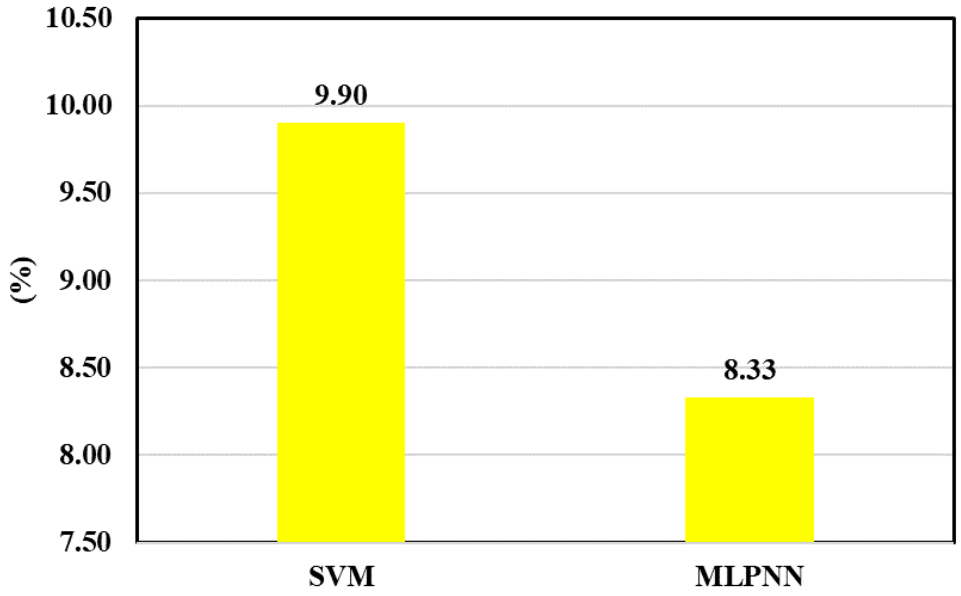


**Figure 4.11 Performance comparison between SVM and MLPNN in terms of MCR**

Performed consistently better than SVM on many assessment criteria including accuracy, precision, recall, F1-score, and BCR. This indicated that the MLPNN attained superior performance in terms of all metrics. Comparing the performance of the SVM and MLPNN about MCR is shown in Figure 4.11. A lower MCR value indicates more effective model. Upon examination of Figure 4.10, it becomes evident that the MLPNN exhibited excellent results by reporting MCR value of 8.33% which was lower than MCR value of SVM (9.90%). Therefore, MLPNN appeared to be the preferred model for choosing conductive ink for printing based on the performance outcomes.

## 4.5 CHAPTER SUMMARY

The primary aim of this study endeavour was to develop a model for selecting a conductive ink suitable for printing applications. In this Chapter, brief introduction about artificial neural network was provided. This Chapter detailed the data that were used for experimentation. Data was preprocessed using min-max method. Training and testing sets were created from the preprocessed data. Two models namely SVM and MLPNN were designed and trained using training data. Performance of the both the models were evaluated using testing data. The MLPNN's effectiveness was evaluated by adjusting the amount of training and testing samples, hidden layers, and hidden neurons. Based on the experimental findings, it was found that the planned system with an 8-12-12-3 structure performed better. Furthermore, effectiveness of the MLPNN was contrasted against SVM to find out suitable one for conductive ink selection in PE applications.

# 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,

$\omega$: 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. .

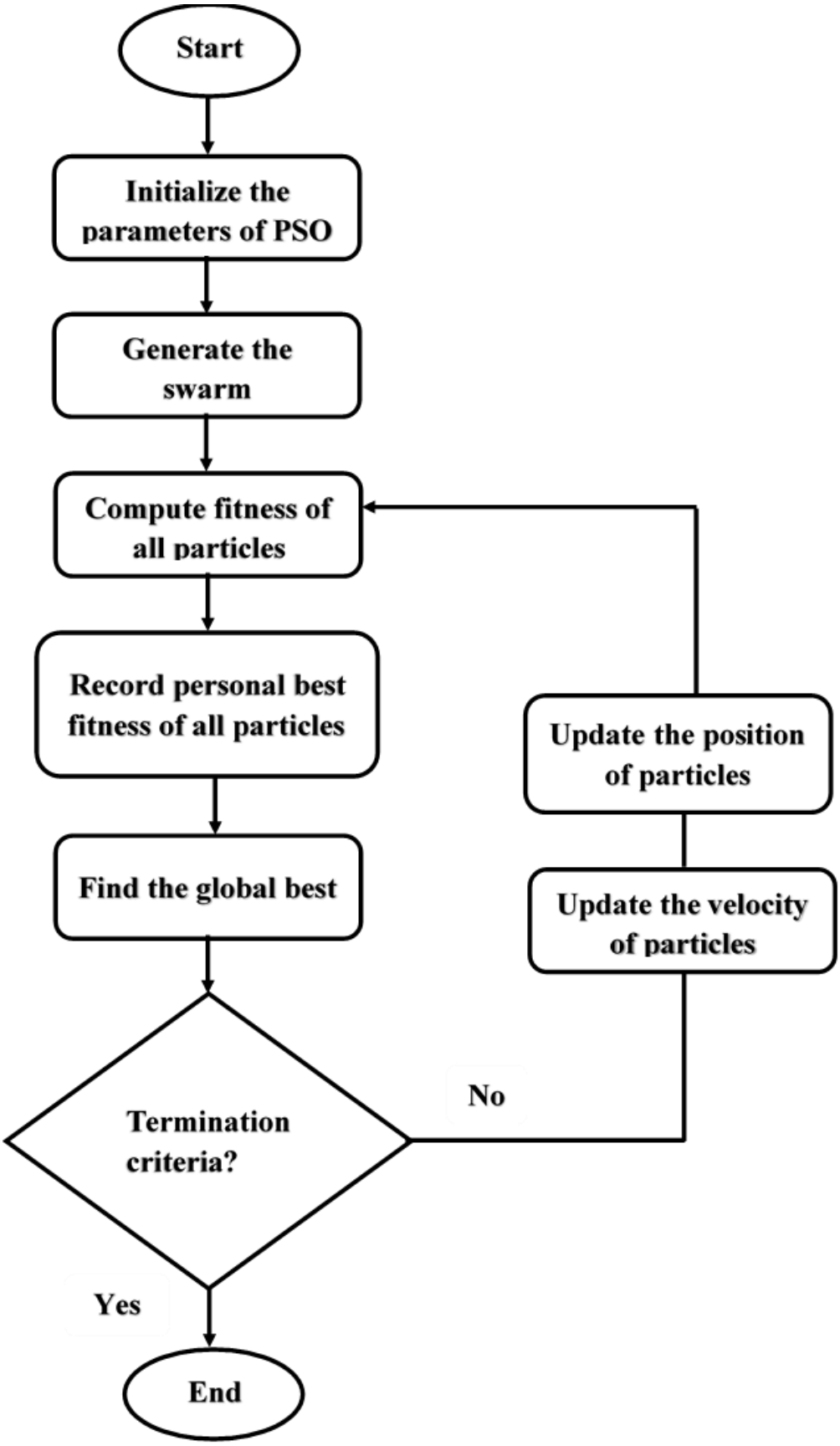


**Figure 5.1 PSO algorithm**

## 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:

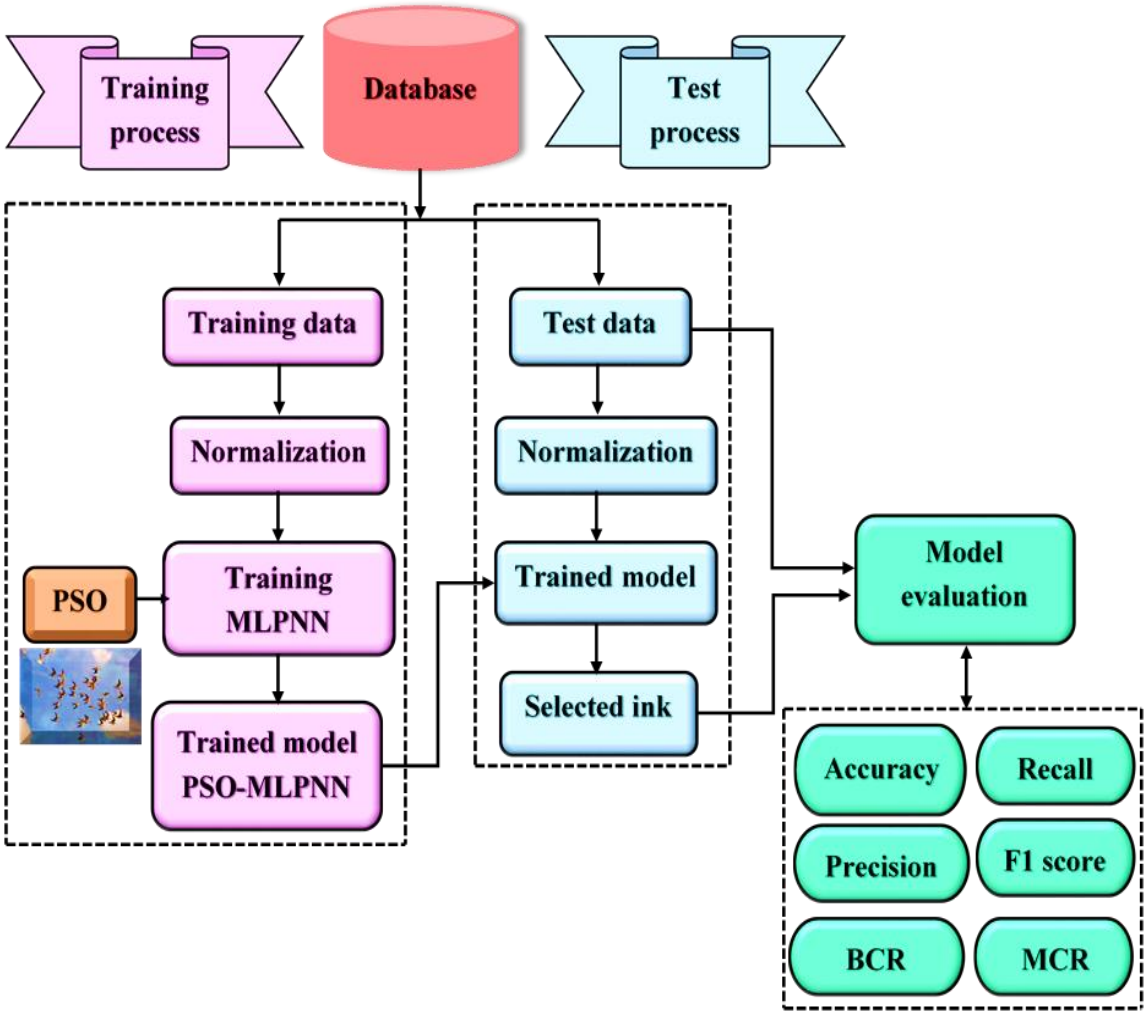


**Figure 5.2 Workflow of the proposed system for selecting conductive ink**

- ♣ 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.

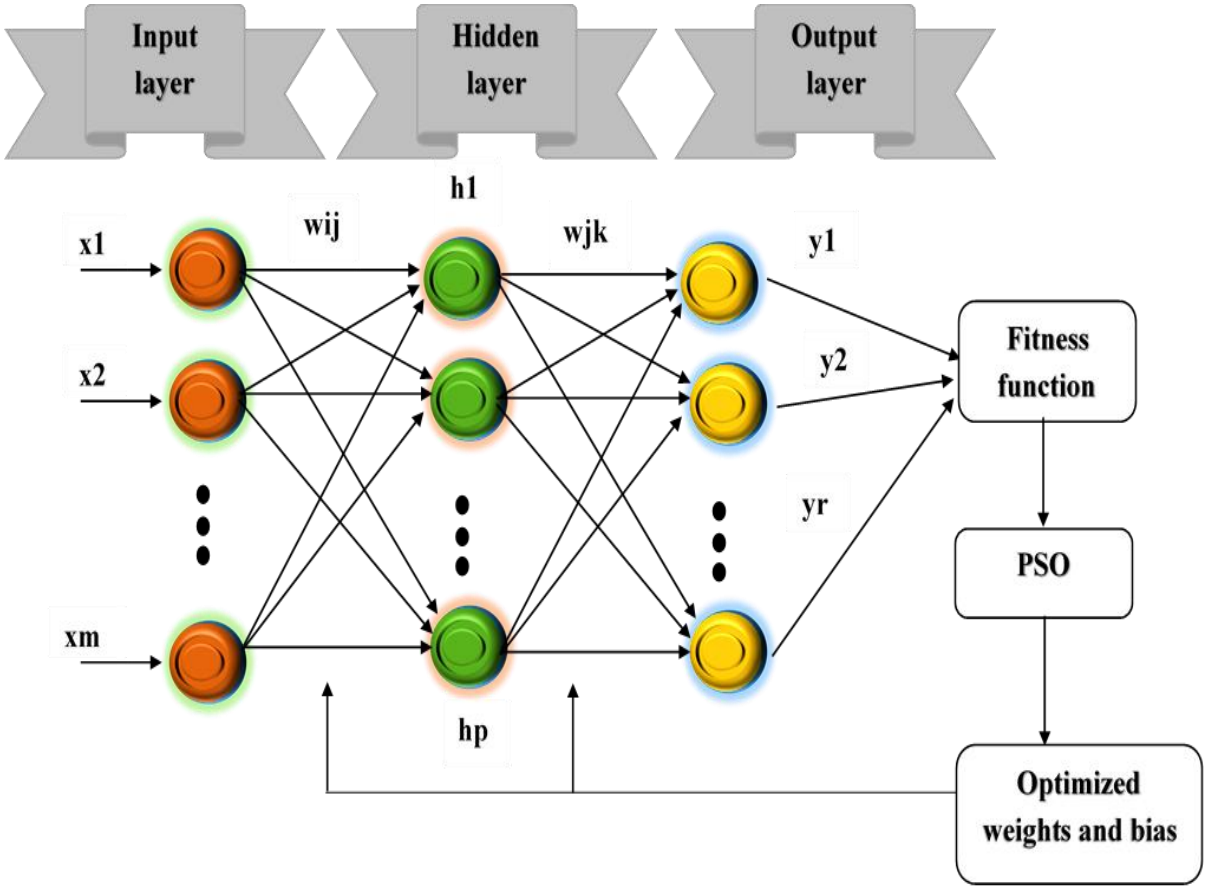


**Figure 5.3 Developed PSO-MLPNN model**

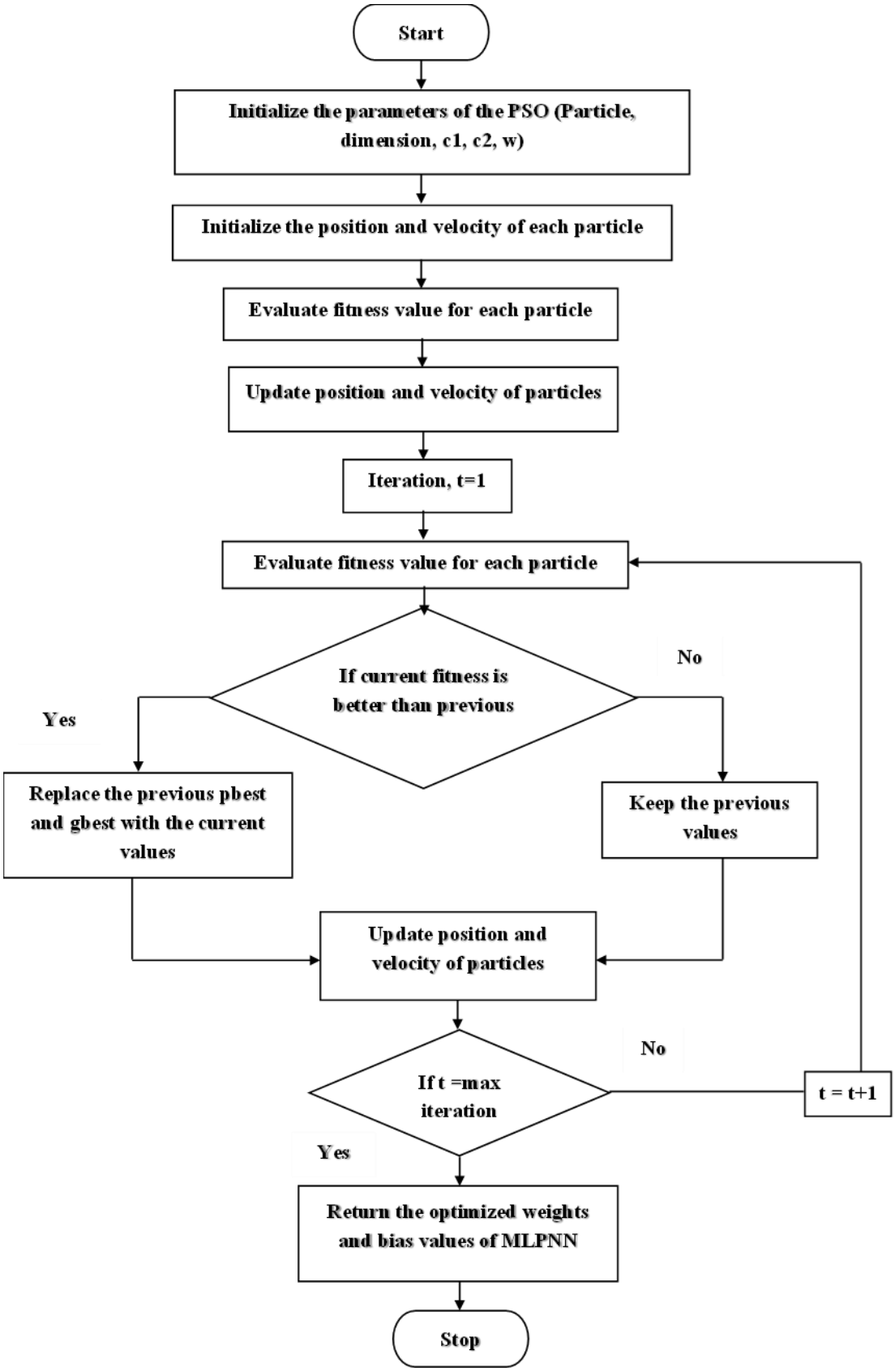


**Figure. 5.4 The PSO-MLPNN for ink selection**

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 |

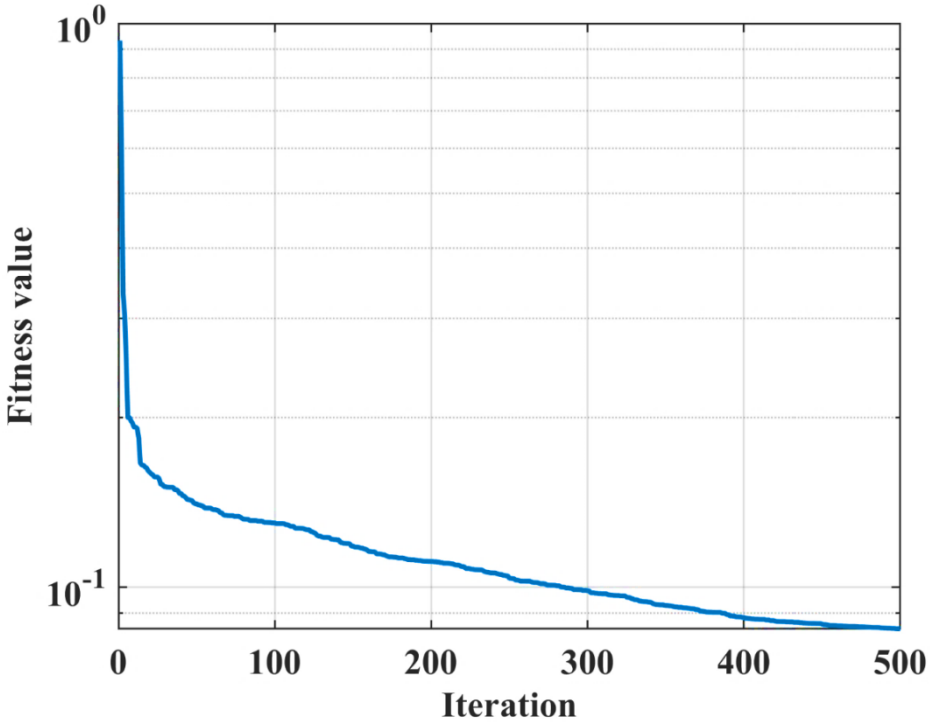


**Figure 5.5 Convergence plot with respect to number of iterations**

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.