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.

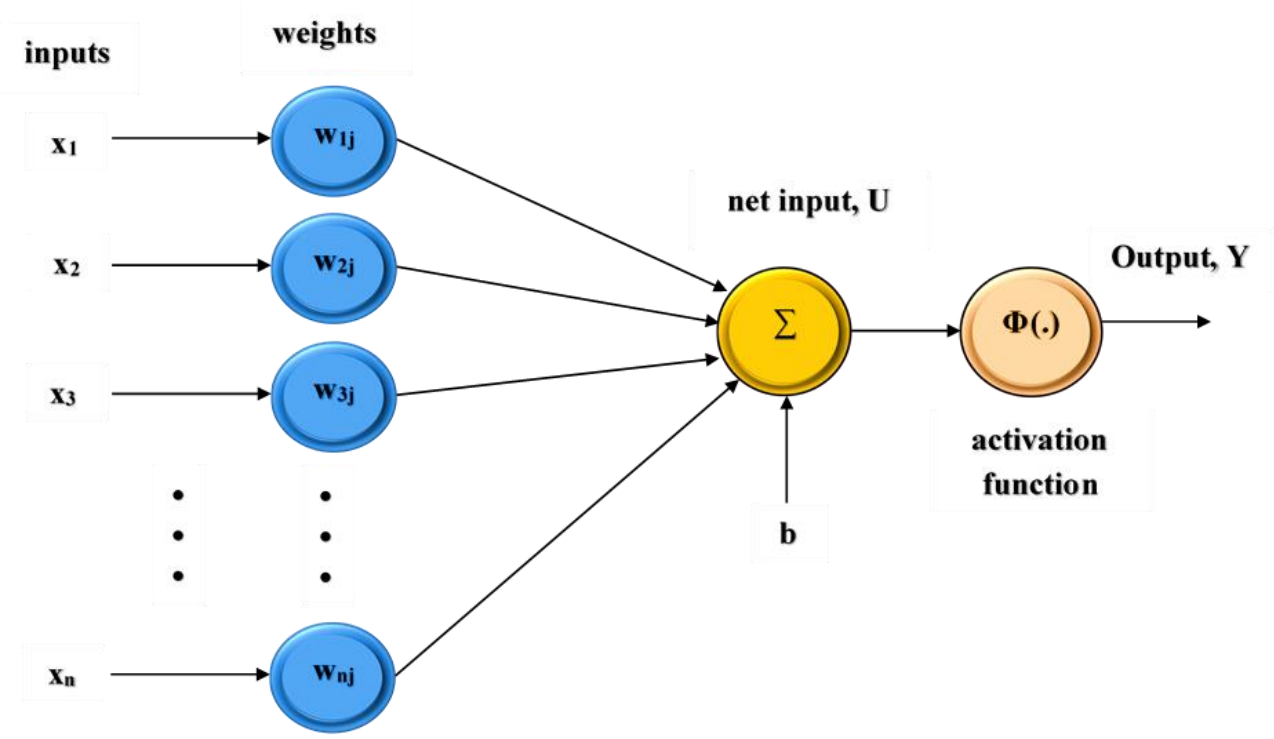

Figure 4.1 Artificial neural network

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

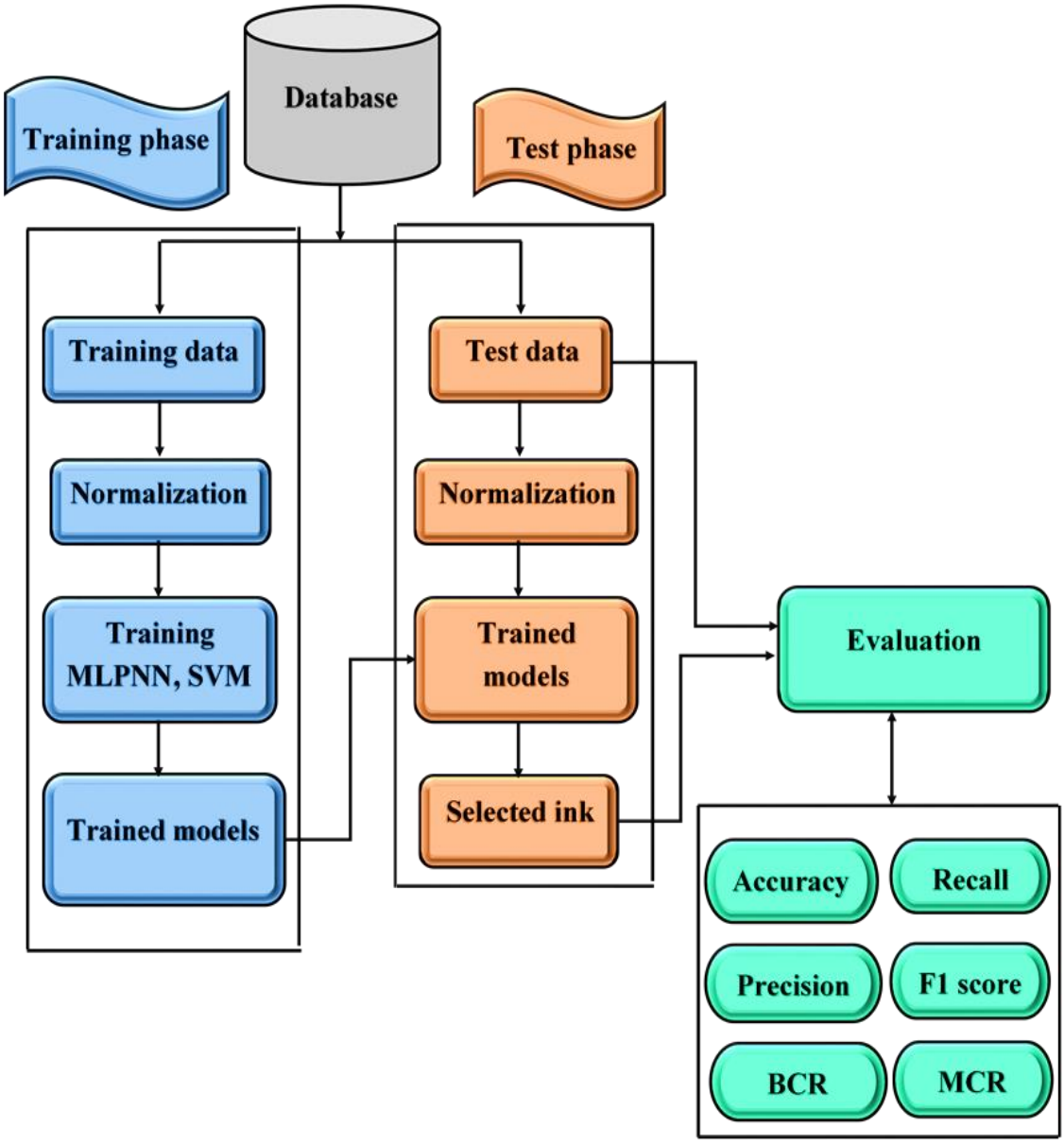

Figure 4.2 Overall process of the developed ink selection system

- 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

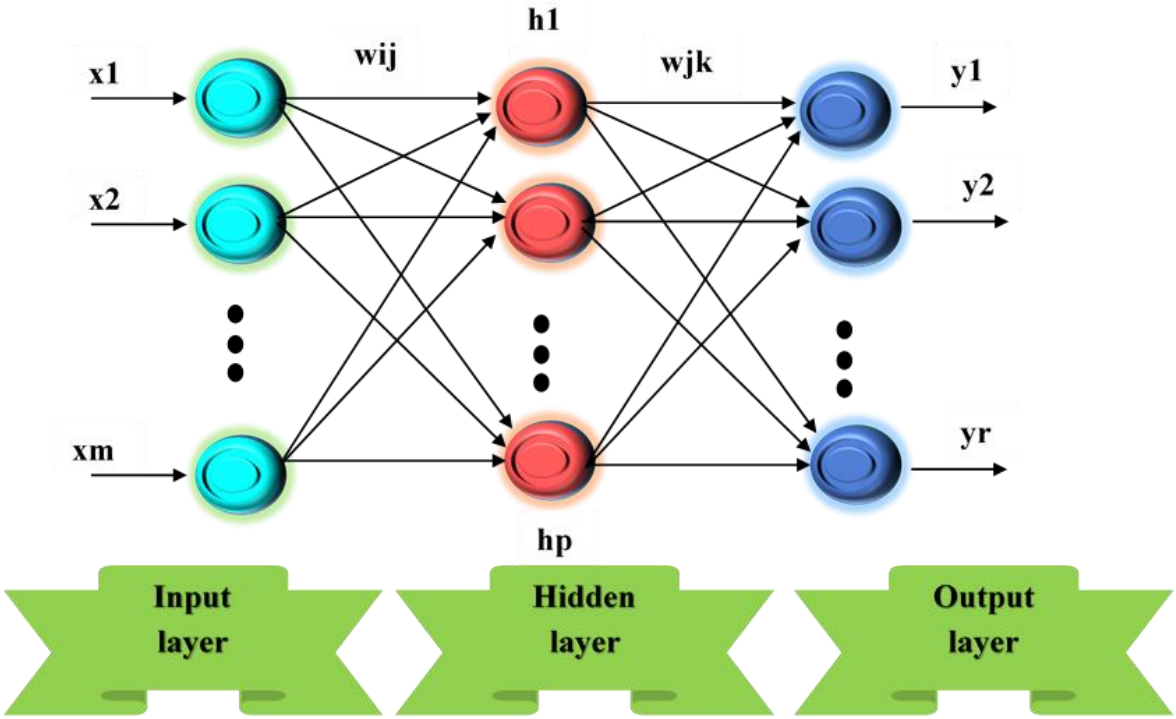

Figure 4.3 Developed MLPNN

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.

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

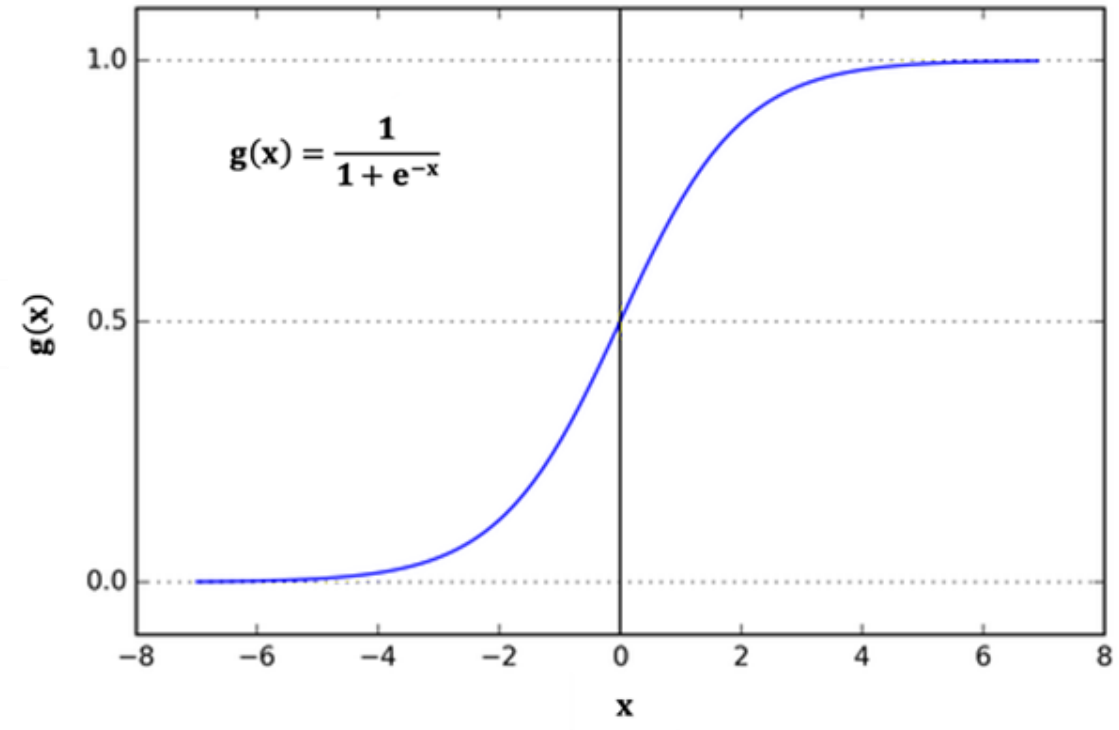

Figure 4.4 Sigmoidal activation function

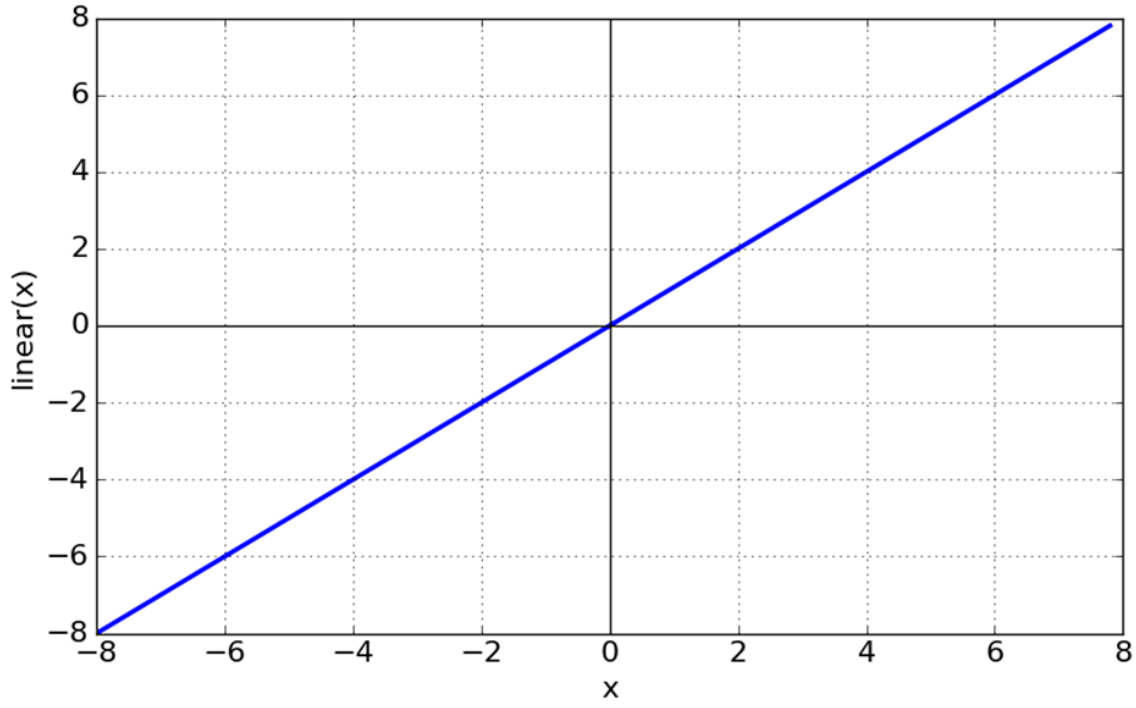

Figure 4.5 Linear activation function

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

Δw: Change in weight

η: 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 μ is zero. A short step-size gradient descent is what happens when μ 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, μ 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.

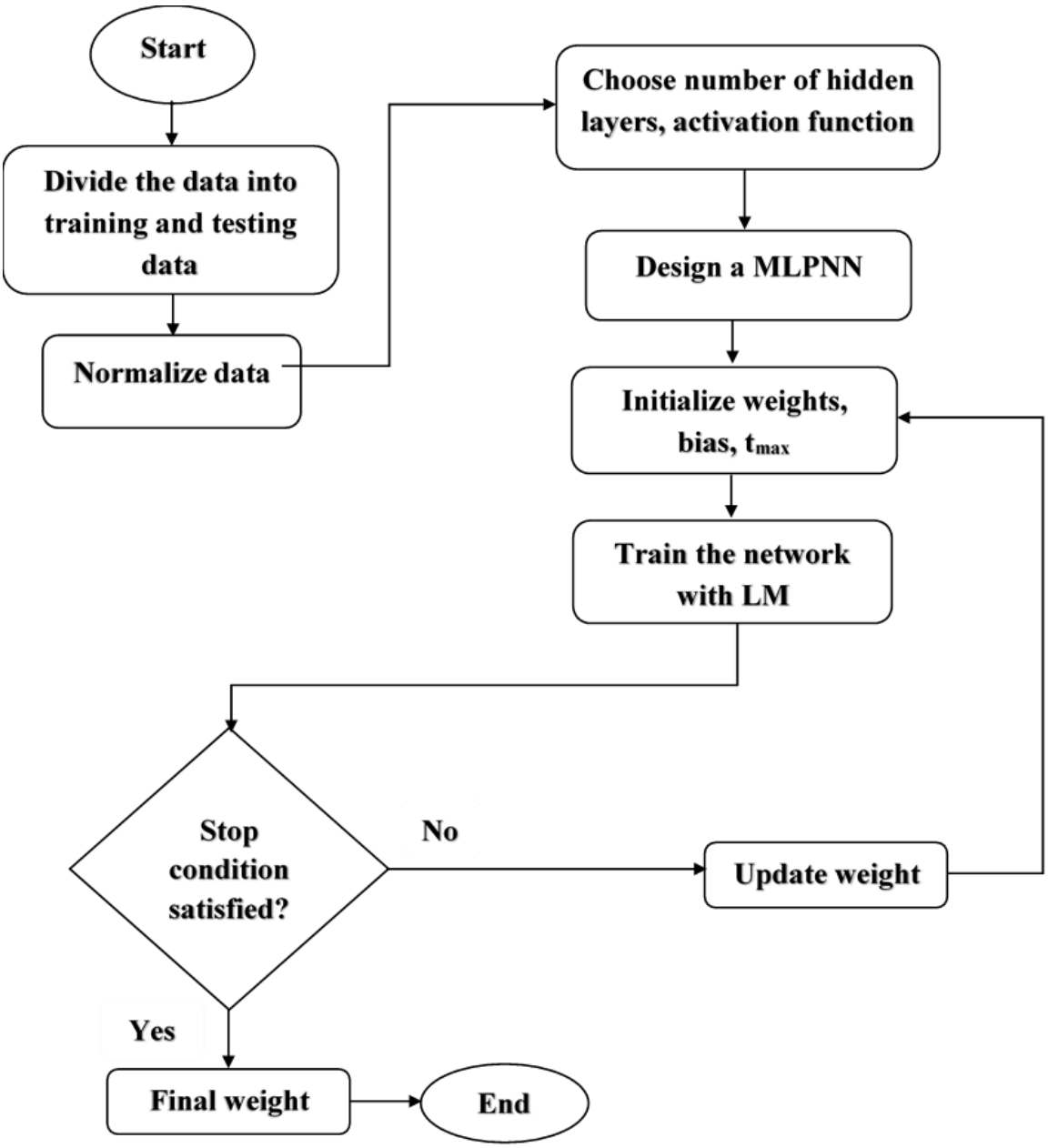

Figure 4.6 Flowchart of the proposed methodology

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.

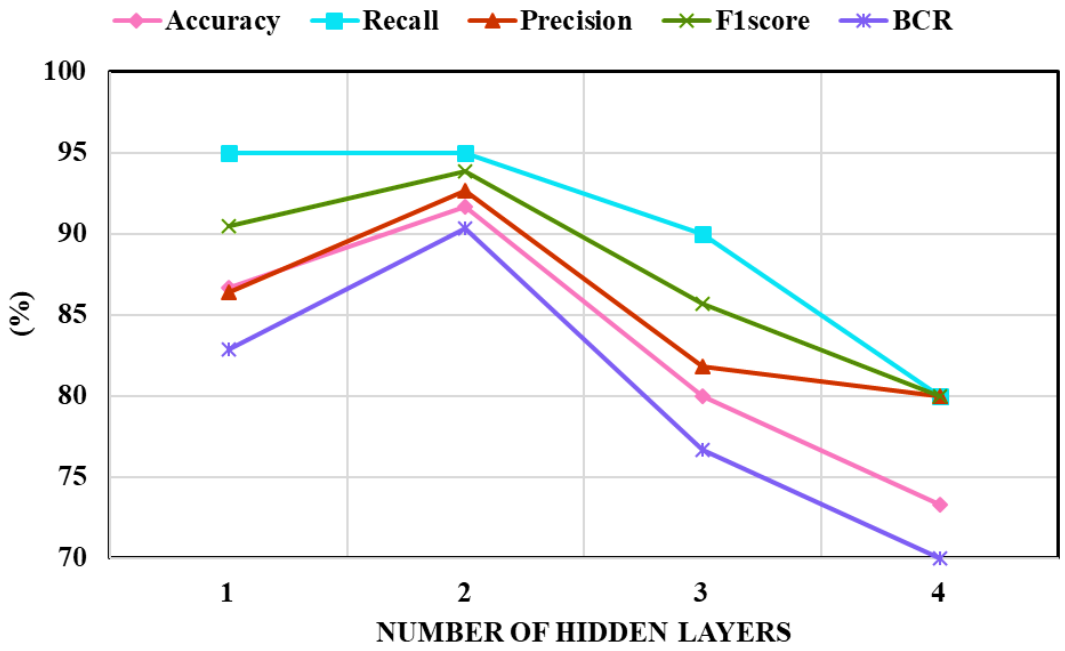

Figure 4.8 Performance of the designed system for varying hidden layers

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

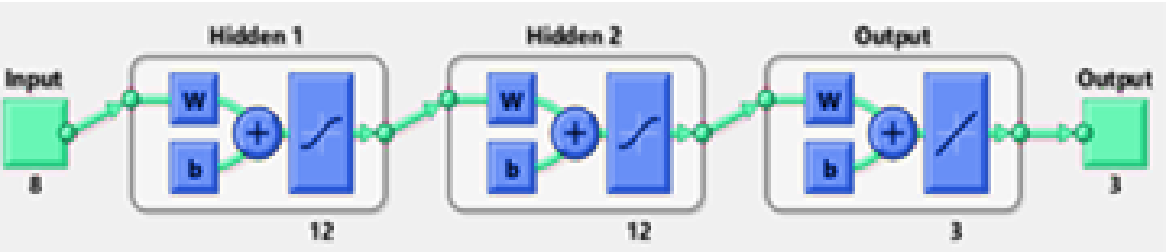

Figure 4.9 The best MLPNN structure for printing applications

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

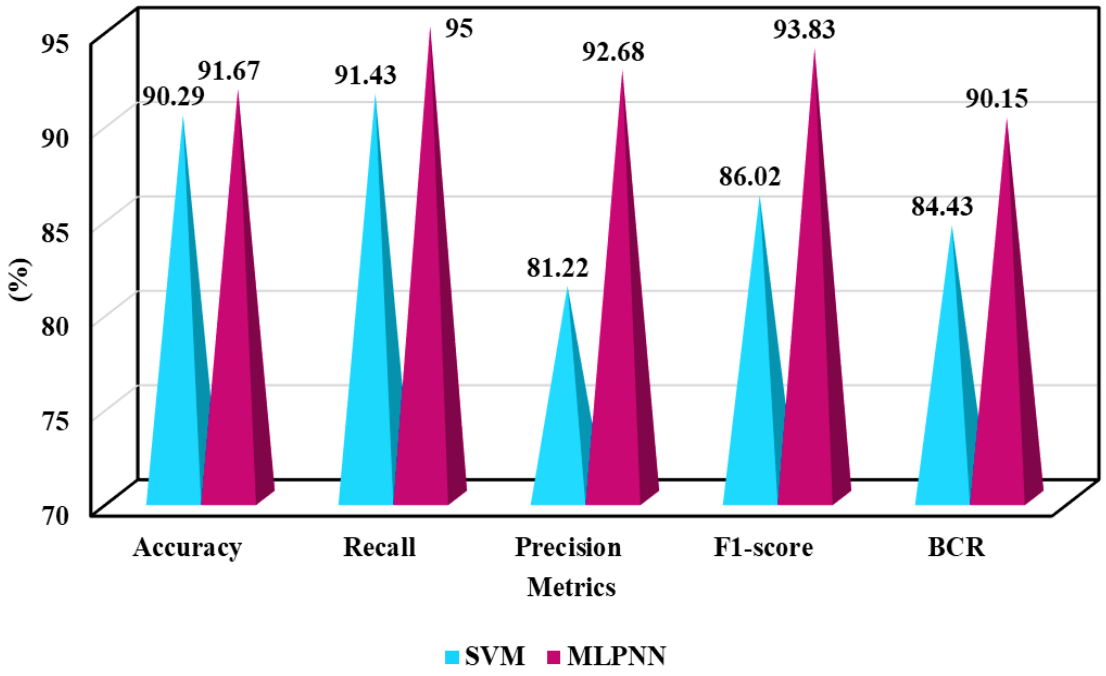

Figure 4.10 Performance comparison between SVM and MLPNN

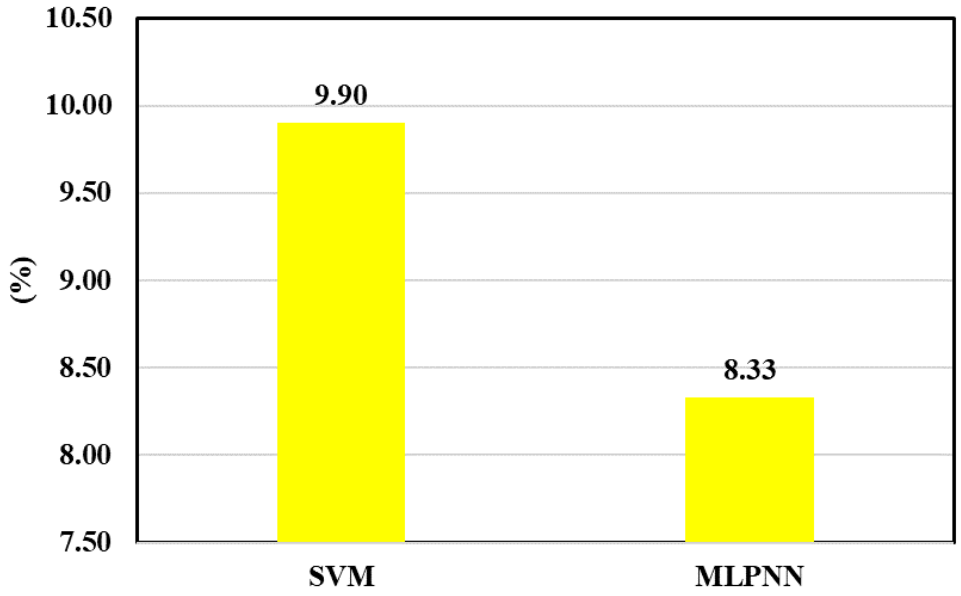

Figure 4.11 Performance comparison between SVM and MLPNN in terms of MCR

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.