ABSTRACT
In healthcare industry, Neural Network has attained a milestone in solving many real-life classification problems varies from very simple to complex and from linear to non-linear. To improve the training process by reducing the training time, Adaptive Skipping Training algorithm named as Half of Threshold (HOT) has been proposed. To perform the fast classification and also to improve the computational efficiency such as accuracy, error rate, etc., the highlighted characteristics of proposed HOT algorithm has been integrated with Strassen’s matrix multiplication algorithm and derived a novel, hybrid and computationally efficient algorithm for training and validating the neural network named as Strassen’s Half of Threshold (ShOT) Training Algorithm. The experimental outcome based on the simulation demonstrated that the proposed SHOT algorithm outperforms both BPN and HOT algorithm in terms of training time which is reduced with the range of 7% to 54% and its efficiency which is improved with the range of 3% to 15% on various dataset such as Hepatitis, SPECT, Heart, Liver Disorders, Breast Cancer Wisconsin (Diagnostic), Drug Consumption, Cardiotocography, Splice-junction Gene Sequences and Thyroid Disease dataset that are extracted from Machine Learning Dataset Repository of UCI. It can be integrated with any type of supervised training algorithm.

INDEX TERMS
Training speed, fast learning, fast training, classification problem, adaptive skipping training.

I. INTRODUCTION
Every second, the amount of healthcare data that is being generated by the healthcare industry is growing exponentially (approximately 30% of the world’s data volume) [1]. This growth rate will reach 36% annually by 2025. At the same time, it is mined to extract the valuable information. Many of today’s creative applications with big scale datasets challenge the natural intelligence of the human brain, which is the most intelligent system on the planet, due to exponential growth in many scientific and medical sectors [2]. Learning new patterns in large scale datasets manually, in a fast and intelligent manner, is beyond the capacity or patience of any human being. To address this issue, researchers developed the concept of Neural Network (NN). Since 1943, Neural Network has attained a milestone in solving many real-life classification problems varies from very simple to complex and from linear to non-linear [3]. When it is viewed from the technical / implementation aspect, training the neural network on very large datasets with the traditional back-propagation algorithm is still facing many challenges. One of the biggest challenges that are faced by neural network is the training rate. The elements that leverage the Neural Network’s training rate are Network structure [4], Training dataset size,
computational efficiency, and problem to classify [5]. The elements listed above relate to each other.

Based on the problem that has been considered for classification, the datasets will be generated and utilized. Very large numbers of training datasets must be fed into the neural network for training to increase the efficiency of the training algorithm as well as to generalize the network. The network structure can, however, expand automatically for a larger training dataset, leading to increased training time as well as reduced efficiency. In practical terms, a larger training dataset usually requires a very long training time with more epochs that leverage the training speed. The Half of Threshold (HoT) Adaptive Skipping Training Algorithm is applied to increase the training speed by minimizing Neural Network training time by randomly presenting the samples in training datasets to boost the training performance. Also, as the size of the network structure increases, it contributes to increase in the weight matrix size.

Among the operation that takes place during the training of neural networks using back-propagation algorithm, Matrix multiplication is the most highly computational process. For making matrix multiplication faster, Strassen’s algorithm [6] is prescribed to multiply the matrices, which is shown in the Theorem 1. By combing the highlighted characteristics of Adaptive Skipping Training Algorithm and Strassen’s algorithm, the overall training time consumed by the neural network will be reduced much with leads to increase in efficiency. By integrating the highlighted characteristics of Half of Threshold (HOT) Adaptive Skipping Training algorithm with Strassen’s algorithm, a novel, hybrid, and computationally efficient algorithm called Strassen’s Half of Threshold (SHT) Adaptive Skipping Training Algorithm, for training the neural network, has been proposed. Because of this proposed algorithm, the cumulative training time consumed by the neural network will be significantly increased resulting in better training performance.

II. RELATED STUDY

Many researchers have contributed many works towards improving the performance of training algorithm by increasing the training speed, improving the accuracy / decreasing the error rate, etc., in different enhancement: estimation of initial weight optimally, second order algorithm for faster learning and maintaining generalization and adaptive learning rate and momentum which has been surveyed in this session. Proper initialization of NN initial weights in the training algorithm’s beginning point minimizes the number of iterations in the training process, resulting in faster training. Initial weights have been demonstrated to affect the BPN technique [7]. In most cases, modest random numbers are chosen as the NN’s initial weights. Nguyen and Widrow [4] assign a fraction of the intended response range to each hidden node, and Drago and Ridella [8] utilize a technique called statistically controlled activation weight initialization (SCAWI) that calculates the maximum value that the weights should adopt at first to avoid neurons becoming saturated throughout the adaptation process. Some studies recommended for utilizing a probability distribution of the mean squared error [9] and the DPT (Delta Pre-Training) approach [10] uses different sets of small initial weights for initializing and training the NN for several times and also if the weight space is well-conditioned, then DPT is a decent concept. If the best of this group fails to meet the requirements, the process is restarted. Many people support this method, although it is essentially a trial-and-error approach with no mathematical foundation. Premature Saturation, for example, can be caused by initial weight values that are excessively large. As a result, the ASCE task committee advises that random values between $-0.30$ and $+0.30$ should be assigned for weights and thresholds as a starting point. [11]

For Single Hidden Layer Feedforward Neural Networks (SLFNs), a technique called Extreme Learning Machine (ELM), which is new and fast in learning, was published in 2004 [12], [13] that selects the weights randomly for input and derives the output weights for output analytically. Sensitivity analysis was employed in the development of the novel initialization strategy for neural networks [14], [15]. The outputs of the first layer are first assigned random values. Once the original values have been modified using sensitivity formulae, the weights are then determined using linear equations. The main benefits of this method can obtain a good solution in only one epoch and with minimal time for computation. Starting with erroneous weight values, on the other hand, can trap the network in local minima or limit learning progress. To speed up the learning process, the initial weights were carefully chosen.

Previously, the momentum-coefficient was usually treated as a constant between 0 and 1. However, the results of the experiments revealed that the fixed coefficient value for the momentum appears to speed up learning only when the recent error function’s downward gradient and the latest weight change are in the same direction. When the newest negative gradient crosses the prior update, the momentum coefficient causes the weight modification to be projected up rather than down in the slope of the error surface, as is suggested [16]. To make learning more successful, it is critical to change the momentum coefficient value adaptively rather than keeping it constant throughout the training period. Even though the error function is not considered quadratic, Zhang et al claim that the BPN approach’s output is converged with constant learning rate and adaptive momentum. [17]. Both strong and weak convergence results are confirmed as well as it can escape local minima and so accelerating network training. The error gradient is closed to zero as the training enters the smooth area, causing the network to converge slowly.

The learning rate is constant and uniform across all weights in a layer for the BPN algorithm [18]. The values assigned for parameter will fluctuate around the minima of the performance surface as gradient descent approaches minima. The network’s parameter is changed in a fixed manner while the learning rate is constant, resulting in sluggish convergence.
Adaptive Skipping Training Algorithm has been presented to improve the training speed of BPN.

III. PROPOSED SHOT ALGORITHM

To conduct the research study effectively, a three-layer feedforward neural network with multiple layers has been suggested. The proposed neural network structure’s layout is built with \( N \) neurons as input, \( P \) neurons as hidden and \( O \) neurons as output. Since the prescribed neural network is fully interconnected, neurons present in the preceding layer are linked with each neuron in the next layer. The input layer has the same number of neurons as the training dataset’s properties.

A. BASIC NOTATION

Each sample is partitioned into a feature vector, \( X \), and a target class, \( Y \), given a training dataset with \( M \) labelled training samples. Let \( X \in \mathbb{R}^{M \times N} \) be the 2D matrix of size \( M \times N \) which is populated with the data samples from the training dataset that contains \( M \) input samples and \( N \) number of attributes.

\[
X = \begin{bmatrix}
    x_{1}^{0} & x_{1}^{1} & \cdots & x_{1}^{N} \\
    x_{2}^{0} & x_{2}^{1} & \cdots & x_{2}^{N} \\
    \vdots & \vdots & \ddots & \vdots \\
    x_{M}^{0} & x_{M}^{1} & \cdots & x_{M}^{N}
\end{bmatrix}, \quad 1 \leq j \leq N, \ 1 \leq i \leq M
\]

Since the training dataset consumed in this research is a supervised dataset, the corresponding target class label, represented as \( T \in \mathbb{R}^{M} \), for the above \( M \) input samples is shown below.

\[
T = \begin{bmatrix}
    t_{1} \\
    t_{2} \\
    \vdots \\
    t_{M}
\end{bmatrix}, \quad t_{i} \in \mathbb{R}^{M}, \ 1 \leq i \leq M
\]

Let \( V \in \mathbb{R}^{N \times P} \) be the 2D matrix of size \( N \times P \) that holds the input-to-hidden synaptic weight coefficient that is assigned for each connection link established between \( N \) input neurons to \( P \) hidden neurons.

\[
V = \begin{bmatrix}
    v_{1}^{1} & v_{1}^{2} & \cdots & v_{1}^{P} \\
    v_{2}^{1} & v_{2}^{2} & \cdots & v_{2}^{P} \\
    \vdots & \vdots & \ddots & \vdots \\
    v_{P}^{1} & v_{P}^{2} & \cdots & v_{P}^{P}
\end{bmatrix}, \quad 1 \leq j \leq P, \ 1 \leq i \leq N
\]

Let \( \vec{v}_0 \) represents bias vector of size \( 1 \times P \) that is fed into the nodes in the hidden layer.

\[
\vec{v}_0 = [v_0^1 \ v_0^2 \ \cdots \ v_0^P], \quad \text{where} \ v_0^i \in \mathbb{R}^{P}, \ 1 \leq i \leq P
\]

Let \( W \in \mathbb{R}^{P \times O} \) be the 2D matrix of size \( P \times O \) that holds the hidden-to-output synaptic weight coefficient that is assigned for each connection link established between
P hidden neurons to O output neurons.

\[
W = \begin{bmatrix}
    w_1^1 & w_2^1 & \cdots & w_O^1 \\
    w_1^2 & w_2^2 & \cdots & w_O^2 \\
    \vdots & \vdots & \ddots & \vdots \\
    w_1^P & w_2^P & \cdots & w_O^P 
\end{bmatrix} = w_i^j \in \mathbb{R}^{P \times O}, \quad 1 \leq j \leq O, \quad 1 \leq i \leq P
\]

Let \( \vec{w}_o \) represents bias vector of size \( 1 \times O \) that is fed into the nodes in the output layer.

\[
\vec{w}_o^0 = \begin{bmatrix}
    w_1^0 & w_2^0 & \cdots & w_O^0 
\end{bmatrix}, \quad \text{where } w_i^0 \in \mathbb{R}^P, \quad 1 \leq i \leq O
\]

Let \( \varphi_h(x) \) and \( \varphi_o(x) \) represent the nonlinear sigmoid and linear activation function adopted to compute the net output in the hidden and output layer, respectively. The symbol used for representing iteration number is \( t \). Let \( sf_i \) and \( sv_i \) be the skipping factor and skipping value of the \( i^{th} \) samples in the training dataset. Let \( d_{max} \) be the error threshold value. Let \( ic \) be the number of iteration / epoch count.

**B. WORKING PRINCIPLE OF SHOT**

Step 1 (Initialization Phase): Initialize the following parameters of the constructed neural network.

- Weight value \( \epsilon [-1, 1] \)
- Biases value to small random values, typically between \(-1 \) and \( 1 \)
- Learning rate, \( \eta \)
- Skipping value, \( sf_i \), to zero
- Skipping factor, \( sf_i \), to zero
- Error threshold, \( d_{max} \)

**Step 2 (Terminating Condition):** Check whether the terminating criterion is attained or not. If it is not attained, repeat the step 3-8. Otherwise, go to step 9.

**Step 3:** Repeat the step 4-8 for \( M \) (number of training samples in the training dataset, \( X \)) times, \( 1 \leq k \leq M \)

**Step 4 (Present the Training Sample):** Training dataset samples are distributed to the input layer in the network which will just propagate it without any computation.

**Step 5 (Forward Propagation):** The following steps are calculated till the output layer starting from the hidden layer through the propagation process:

**Step 5.1:** For the Hidden layer, compute the activation values as:

\[
h_{\text{net}}_i^t = v_i^0 + \sum_{j=1}^{N} x_j^i \cdot v_j^i, \quad 1 \leq i \leq P
\]

Estimate the actual output.

\[
z_i^t = \varphi_h(h_{\text{net}}_i^t), \quad 1 \leq i \leq P
\]

\[
z_i^t = \frac{1}{1 + e^{-h_{\text{net}}_i^t}}, \quad 1 \leq i \leq P \quad (1)
\]
Step 5.2: For the Output layer, compute the activation values as:

Apply Theorem 1 to estimate the net output value using Strassen’s Fast Multiplication of Matrices Algorithm

\[ o_{\text{net}} = w^{0}_i + \sum_{j=1}^{P} \cdot j \cdot w^{j}_i, \quad 1 \leq i \leq O \]

Estimate the actual output

\[ y_i = \varphi_o (o_{\text{net}}), \quad 1 \leq i \leq O \]

\[ y_i = \frac{1}{1 + e^{-o_{\text{net}}}}, \quad 1 \leq i \leq O \]  

(2)

Theorem 1 (Strassen’s Theorem): Two \( N \times N \) matrices can be multiplied using only \( N^{\log_2 7} \approx N^{2.8074...} \) scalar multiplications.

Step 6: Error Signal Calculation

Using the squared error function, the error signal for each output neuron is calculated and performed summation over the error signal to get the total error:

\[ E = \sum_{k} \frac{1}{2} (y_k - o_k)^2, \quad 1 \leq k \leq M \]  

(3)

Step 6.1: Finding the Error derivative for hidden to output weight

Adjust the network’s weights by calculating the partial error derivative with respect to the weight to minimize the error, E, globally.

\[ \Delta W = -\eta \frac{\partial E}{\partial W} \]

\[ \Delta W^j = -\eta \frac{\partial E}{\partial W^j} \quad 1 \leq k \leq O \]

Expand the above error function using chain rule

\[ \frac{\partial E}{\partial W^j} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial W^j} \]

\[ \frac{\partial E}{\partial W^j} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial W^j} \]

The derivation of the error with respect to the activation function is derived here

\[ \frac{\partial \text{net}_k}{\partial W^j} = \frac{\partial z^j w^j_k}{\partial W^j} = z^j \]  

(4)

The derivation of the activation function with respect to the net input is shown here

\[ \frac{\partial y_k}{\partial \text{net}_k} = \frac{\partial (1 + e^{-\text{net}_k})}{\partial \text{net}_k} \]

\[ \frac{\partial y_k}{\partial \text{net}_k} = \frac{\partial (1 + e^{-\text{net}_k})}{\partial \text{net}_k} \]

Rewriting the above equation,

\[ e^{-\text{net}_k} \approx \frac{1}{(1 + e^{-\text{net}_k})^2} \]

The derivation of the net input with respect to the synaptic weight is shown here

\[ \frac{\partial E}{\partial W^j} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial W^j} \]

Substituting the value of each derivative,

\[ \Delta W^j = -\eta \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial W^j} \]

\[ \Delta W^j = -\eta \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial W^j} \]

(6)

Step 6.2: Finding the Error derivative for input to hidden weight

\[ \Delta V = -\eta \frac{\partial E}{\partial V} \]

\[ \Delta V^j = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial \text{net}_k} \frac{\partial \text{net}_k}{\partial W^j} \]

The derivation of the activation function with respect to the net input is shown here

\[ \frac{\partial y_k}{\partial \text{net}_j} = \frac{\partial (1 + e^{-\text{net}_j})}{\partial \text{net}_j} \]

\[ \frac{\partial y_k}{\partial \text{net}_j} = \frac{\partial (1 + e^{-\text{net}_j})}{\partial \text{net}_j} \]

\[ \frac{\partial y_k}{\partial \text{net}_j} = \frac{\partial (1 + e^{-\text{net}_j})}{\partial \text{net}_j} \]

\[ \frac{\partial y_k}{\partial \text{net}_j} = \frac{\partial (1 + e^{-\text{net}_j})}{\partial \text{net}_j} \]

\[ \frac{\partial y_k}{\partial \text{net}_j} = \frac{\partial (1 + e^{-\text{net}_j})}{\partial \text{net}_j} \]

\[ \frac{\partial y_k}{\partial \text{net}_j} = \frac{\partial (1 + e^{-\text{net}_j})}{\partial \text{net}_j} \]

\[ \frac{\partial y_k}{\partial \text{net}_j} = \frac{\partial (1 + e^{-\text{net}_j})}{\partial \text{net}_j} \]

\[ \frac{\partial y_k}{\partial \text{net}_j} = \frac{\partial (1 + e^{-\text{net}_j})}{\partial \text{net}_j} \]

\[ \frac{\partial y_k}{\partial \text{net}_j} = \frac{\partial (1 + e^{-\text{net}_j})}{\partial \text{net}_j} \]

\[ \frac{\partial y_k}{\partial \text{net}_j} = \frac{\partial (1 + e^{-\text{net}_j})}{\partial \text{net}_j} \]
Rewriting the above equation,
\[
\frac{e^{-h_{\text{net}_j}}}{1 + e^{-h_{\text{net}_j}}}^2 = \frac{1}{(1 + e^{-h_{\text{net}_j}})(1 + e^{-h_{\text{net}_j}})}
= \frac{1}{(1 + e^{-h_{\text{net}_j}})} \left( 1 - \frac{1}{(1 + e^{-h_{\text{net}_j}})} \right)
= \zeta_i \left( 1 - \zeta_i \right)
\]
\[
\frac{\partial z_i}{\partial h_{\text{net}_j}} = \zeta_i \left( 1 - \zeta_i \right)
\tag{7}
\]

The derivation of the net input with respect to the synaptic weight is shown here
\[
\frac{\partial z_i}{\partial v_i} = \frac{\partial x_k}{\partial v_i} = x_k
\]
\[
\Delta v_i = -\eta \frac{\partial E}{\partial v_i}
\tag{8}
\]

Substituting the value of each derivative,
\[
\Delta v_i = -\eta \frac{\partial E}{\partial v_i}
= -\eta \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial d_{\text{net}_k}} \frac{\partial d_{\text{net}_k}}{\partial h_{\text{net}_j}} \frac{\partial h_{\text{net}_j}}{\partial v_i}
= \left[ \eta \delta_k \zeta_i \right] \zeta_i \left( 1 - \zeta_i \right) x_k
\text{ Using equ.} (7), (8)
\]

\[
\Delta v_i = \eta \delta_k x_k
\]

\textbf{Step 7: Backward Propagation}

\textbf{Step 7.1:} Update weights hidden to output weight using the Delta-Learning Rule
\[
w^i_k = w^i_k + \eta \delta_k z_i
\tag{9}
\]

\textbf{Step 7.2:} Update weights input to hidden weight using the Delta-Learning Rule
\[
v^i_j = v^i_j + \eta \delta_i x^j_k
\tag{10}
\]

\textbf{Step 8: AST Algorithm}

\textbf{Step 8.1:} Calculate the difference between the neural network output’s target ($t_k$) and actual($y_k$) value, $t_k - y_k$.

\textbf{Step 8.2:} Compare the difference (step 8.1) with the error half of the threshold value. Whenever the samples are classified correctly, the following condition returns zero.
\[
|t_k - y_k| < d_{\text{max}}/2
\]

\textbf{Step 8.3:} Determine the computing probability of all the input samples based on $t_k - y_k$.

\textbf{Step 8.4:} If the $\text{prob}(x_i)$ is 0, then the corresponding input samples have been classified correctly and it will be skipped from training for next $sf_i$ epochs.

\textbf{Step 8.5:} If the $\text{prob}(x_i)$ is 0, then skipping value $sv_i$ is increased by $sf_i$. When the skipping value $sv_i$ becomes zero, then the $i^{th}$ input samples will be presented again for training.

\textbf{Step 8.6:} Based on the skipping value $sv_i$, the modified training dataset is constructed which will be presented in the next epoch.

\textbf{Step 9:} Stop the training process

A machine learning algorithm must be tested on test data once it has learned the fundamental patterns in the training data. It is termed an efficient machine learning classifier model if it performs well on the test data and generalizes the produced dataset, which is measured using the classifier’s performance matrices.

\section{IV. SIMULATION-BASED EXPERIMENTAL RESULT AND ITS ANALYSIS}

To conduct the research successfully, the proposed supervised machine learning SHOT algorithm is simulated with the following machine configurations: Intel®Core I5 generation-3210M processor, CPU speed with 2.50GHz and MATLAB Software R2010b version.

\subsection{A. DATASET DESCRIPTION}

To assess the performance of the existing and proposed SHOT algorithms, both the algorithms were tested on datasets acquired from UCI’s Machine Learning Dataset Repository for binary and multi-class classification problems [14]. The data collection for the Hepatitis dataset is loaded with 155 samples of data collected containing 19 attributes and 2 classes of binary classification. The SPECT Heart dataset has 267 samples in its data collection with 22 attributes and 2 classes of binary classification. The data collection for the Liver Disorders dataset is loaded with 345 samples of data collected containing 7 attributes and 2 classes of binary classification. The data collection for the Breast Cancer Wisconsin (Diagnostic) dataset is loaded with 569 samples of data collected containing 32 attributes and 2 classes of binary classification. The data collection for the Drug Consumption dataset is loaded with 1885 samples of data collected containing 32 attributes and 7 classes of multi-class classification. The data collection for the Cardiotocography dataset is loaded with 2126 samples of data collected containing 23 attributes and 3 classes of multi-class classification. The data collection for the Splice-Junction Gene Sequences dataset has 3190 samples in its data collection with 19 attributes and 3 classes of multi-class classification. The data collection for the Thyroid Disease dataset is loaded with 7200 samples of data collected containing 19 attributes and 3 classes of multi-class classification. The training dataset properties are shown in Table 1.

\begin{table}
\centering
\caption{Dataset properties.}
\begin{tabular}{|c|c|c|c|c|}
\hline
\textbf{Dataset} & \textbf{Number of Instances} & \textbf{Number of Attributes} & \textbf{Number of Classes} & \textbf{Classification Type} \\
\hline
Hepatitis & 155 & 19 & 2 & Binary \\
SPECT Heart & 267 & 22 & 2 & Binary \\
Liver Disorders & 342 & 7 & 2 & Binary \\
Breast Cancer Wisconsin (Diagnostic) & 569 & 32 & 2 & Binary \\
Drug Consumption & 1885 & 32 & 7 & Multiclass \\
Cardiotocography & 2126 & 23 & 3 & Multiclass \\
Splice-junction Gene Sequences & 3190 & 19 & 3 & Multiclass \\
Thyroid Disease & 7200 & 19 & 3 & Multiclass \\
\hline
\end{tabular}
\end{table}
TABLE 2. Comparison results of various learning algorithm trained using Hepatitis dataset with $\theta = 1e^{-4}$.

| Fold Number | Epoch | BPN | HoT | SHoT |
|-------------|-------|-----|-----|------|
|             |       | Training Time (in Sec) | Accuracy (%) | Training Time (in Sec) | Accuracy (%) | Training Time (in Sec) | Accuracy (%) |
| 1           | 2789  | 83.6642 | 66.67 | 68.7064 | 86.67 | 64.6961 | 86.67 |
| 2           | 2486  | 86.8873 | 66.67 | 54.1146 | 93.33 | 44.4945 | 100  |
| 3           | 2361  | 78.1140 | 73.33 | 60.7247 | 73.33 | 33.7581 | 100  |
| 4           | 2664  | 61.4551 | 80  | 49.8330 | 86.67 | 56.8143 | 93.33 |
| 5           | 2627  | 82.0119 | 80  | 49.5782 | 73.33 | 30.4874 | 80   |
| 6           | 3097  | 63.6796 | 100 | 57.3314 | 100  | 64.8616 | 86.67 |
| 7           | 2005  | 79.3600 | 66.67 | 67.6834 | 86.67 | 42.2932 | 93.33 |
| 8           | 2428  | 64.2809 | 80  | 50.2056 | 86.67 | 27.9781 | 93.33 |
| 9           | 2495  | 79.6651 | 100 | 67.4787 | 80  | 61.0754 | 93.33 |
| 10          | 2810  | 69.0190 | 73.33 | 44.5243 | 86.67 | 47.0059 | 100  |
| Avg         |       | 74.8137 | 78.6670 | 57.0180 | 85.3340 | 47.3465 | 92.6660 |

B. EXPERIMENTAL SETUP AND RESULT

To perform the experiment, the supervised machine learning algorithm is simulated with the use of 3-layer multilayer feedforward neural network. For enhancing the training performance by attaining more accurate prediction, the ten-fold cross validation technique is adopted for training the network model in which all the training samples are given for training.

The performance of the proposed SHoT method is evaluated using four real benchmark classification datasets snatched from the UCI Machine Learning Repository: Hepatitis, SPect Heart, Liver Disorders, Breast Cancer Wisconsin (Diagnostic), Drug Consumption, Cardiotocography, Splice-Junction Gene Sequences, and Thyroid Disease. Training time and Accuracy have been used as performance indicators to assess the performance of various supervised machine learning algorithms. Training time refers to the amount of time spent by the classifier throughout the training process. The percentage of correctly categorized samples is used to define accuracy.

\[
\text{Accuracy} = \frac{\text{Number of samples that are classified correctly}}{\text{Total number of samples}}
\]

1) HEPATITIS DATASET

The results of various learning algorithms for training Hepatitis dataset are summarized in Table 2 for each fold of tenfold cross validation and are compared to the proposed SHoT approaches. In comparison to existing methods, the accuracy produced by the suggested SHOT methods has improved. Furthermore, when compared to Artificial Neural Networks utilizing BPN algorithm, the average time for the complete training process consumed by HoT and SHoT methods is reduced by 24% and 37%, respectively, and by SHoT technique is reduced by 17% when compared to HoT algorithm.

2) SPECT HEART DATASET

The results of various learning algorithms for training SPect Heart dataset are summarized in Table 3 for each fold of tenfold cross validation and are compared to the proposed SHOT approaches. In comparison to existing methods, the accuracy produced by the suggested SHOT methods has improved. Furthermore, when compared to Artificial Neural Networks utilizing BPN algorithm, the average time for the complete training process consumed by HoT and SHoT methods is reduced by 10% and 46%, respectively, and by SHoT technique is reduced by 39% when compared to HoT algorithm.

3) LIVER DISORDERS DATASET

The results of various learning algorithms for training Liver Disorders dataset are summarized in Table 4 for each fold of tenfold cross validation and are compared to the proposed SHOT approaches. In comparison to existing methods, the accuracy produced by the suggested SHOT methods has improved. Furthermore, when compared to Artificial Neural Networks utilizing BPN algorithm, the average time for the complete training process consumed by HoT and SHoT methods is reduced by 9% and 28%, respectively, and by SHoT technique is reduced by 20% when compared to HoT algorithm.
4) BREAST CANCER WISCONSIN (DIAGNOSTIC) DATASET
The results of various learning algorithms for training Breast Cancer Wisconsin (Diagnostic) dataset are summarized in Table 5 for each fold of tenfold cross validation and are compared to the proposed SHOt approaches. In comparison to existing methods, the accuracy produced by the suggested SHOt methods has improved. Furthermore, when compared to Artificial Neural Networks utilizing BPN algorithm, the average time for the complete training process consumed by HoT and SHOt methods is reduced by 38% and 54%, respectively, and by SHOt technique is reduced by 25% when compared to HoT algorithm.

5) DRUG CONSUMPTION DATASET
The results of various learning algorithms for training Drug Consumption dataset are summarized in Table 6 for each fold of tenfold cross validation and are compared to the proposed SHOt approaches. In comparison to existing methods, the accuracy produced by the suggested SHOt methods has improved. Furthermore, when compared to Artificial Neural

| Fold Number | Epoch | BPN Training Time (in Sec) | Accuracy (%) | HoT Training Time (in Sec) | Accuracy (%) | SHOt Training Time (in Sec) | Accuracy (%) |
|-------------|-------|-----------------------------|--------------|-----------------------------|--------------|-------------------------------|--------------|
| 1           | 1711  | 37.0966                     | 80           | 47.3319                     | 85.71        | 53.1991                       | 91.43        |
| 2           | 1576  | 40.3681                     | 85.71        | 45.9930                     | 88.57        | 40.5809                       | 91.43        |
| 3           | 1031  | 44.5172                     | 80           | 39.8379                     | 80           | 53.7893                       | 94.29        |
| 4           | 1032  | 65.2276                     | 88.57        | 54.7378                     | 85.71        | 50.9335                       | 80           |
| 5           | 1833  | 65.8828                     | 82.86        | 51.1167                     | 91.43        | 29.6916                       | 94.29        |
| 6           | 1003  | 41.9939                     | 85.71        | 34.3110                     | 80           | 25.8202                       | 94.29        |
| 7           | 1787  | 64.4417                     | 88.57        | 52.4936                     | 88.57        | 29.6721                       | 88.57        |
| 8           | 2364  | 47.0853                     | 77.14        | 56.7969                     | 85.71        | 29.2445                       | 97.14        |
| 9           | 1646  | 45.6105                     | 88.57        | 47.4441                     | 91.43        | 29.8929                       | 85.71        |
| 10          | 1990  | 65.7836                     | 82.86        | 41.4965                     | 85.71        | 32.2293                       | 94.29        |
| **Avg:**    |       | 51.8007                     | 83.9990      | 47.1559                     | 86.2840      | 37.5053                       | 91.1440      |
Networks utilizing BPN algorithm, the average time for the complete training process consumed by HoT and SHoT methods is reduced by 7% and 23%, respectively, and by SHoT technique is reduced by 17% when compared to HoT algorithm.

6) CARDIOTOCOGRAPHY DATASET
The results of various learning algorithms for training Cardiotocography dataset are summarized in Table 7 for each fold of tenfold cross validation and are compared to the proposed SHOT approaches. In comparison to existing methods, the accuracy produced by the suggested SHOT methods has improved. Furthermore, when compared to Artificial Neural Networks utilizing BPN algorithm, the average time for the complete training process consumed by HoT and SHoT methods is reduced by 7% and 20%, respectively, and by SHoT technique is reduced by 13% when compared to HoT algorithm.

7) SPLICE-JUNCTION GENE SEQUENCES DATASET
The results of various learning algorithms for training Splice-junction Gene Sequences dataset are summarized
in Table 8 for each fold of tenfold cross validation and are compared to the proposed SHOT approaches. In comparison to existing methods, the accuracy produced by the suggested SHOT methods has improved. Furthermore, when compared to Artificial Neural Networks utilizing BPN algorithm, the average time for the complete training process consumed by HoT and SHoT methods is reduced by 9% and 26%, respectively, and by SHoT technique is reduced by 18% when compared to HoT algorithm.

### 8) THYROID DISEASE DATASET

The results of various learning algorithms for training Splice-junction Gene Sequences dataset are summarized in Table 9 for each fold of tenfold cross validation and are compared to the proposed SHOT approaches. In comparison to existing methods, the accuracy produced by the suggested SHOT methods has improved. Furthermore, when compared to Artificial Neural Networks utilizing BPN algorithm, the average time for the complete training process consumed
by HoT and SHoT methods is reduced by 39% and 54%, respectively, and by SHoT technique is reduced by 24% when compared to HoT algorithm.

C. RESULT COMPARISON AND DISCUSSION

1) TRAINING TIME
The training time consumed totally by various training algorithm such as BPN, HoT and SHoT at the end of each training fold and its average training time of all the training fold is compared and represented in Figure 2.

The average value of overall training time consumed by HoT algorithm is reduced by 24%, 10%, 9%, 38%, 7%, 7%, 9% and 39% as that of BPN algorithm for training the dataset such as Hepatitis, SpeCT, Heart, Liver Disorders, Breast Cancer Wisconsin (Diagnostic), Drug Consumption, Cardiotocography, Splice-junction Gene Sequences and Thyroid Disease dataset, respectively. The total time consumed by SHoT algorithm for training is scaled down to 37%, 46%, 28%, 54%, 23%, 20%, 26%, and 54% as that of BPN algorithm and 17%, 39%, 20%, 25%, 17%, 13%, 18% and 24% as that of HoT algorithm for training the dataset such as Hepatitis, SpeCT, Heart, Liver Disorders, Breast Cancer Wisconsin (Diagnostic), Drug Consumption, Cardiotocography, Splice-junction Gene Sequences and Thyroid Disease dataset respectively.

2) ACCURACY
The comparison result of the accuracy consumed by various training algorithm such as BPN, LAST and SHOT is illustrated in the Figure 3.
From Figure 3 and 4, for the Hepatitis dataset, the accuracy obtained by SHoT training algorithm is 15% greater than that acquired by BPN training algorithm and 8% higher than that produced by HoT training algorithm, and the accuracy obtained by HoT training algorithm is 8% higher than that obtained by BPN. For the SPeCT Heart dataset, the accuracy obtained by SHoT training algorithm is 12% greater than that acquired by BPN training algorithm and 5% higher than that produced by HoT training algorithm, and the accuracy obtained by HoT training algorithm is 7% higher than that obtained by BPN. For the Liver Disorders dataset, the accuracy obtained by SHoT training algorithm is 8% greater than that acquired by BPN training algorithm and 5% higher than that produced by HoT training algorithm, and the accuracy obtained by HoT training algorithm is 3% higher than that obtained by BPN. For the Breast Cancer Wisconsin (Diagnostic) dataset, the accuracy obtained by SHoT training algorithm is 12% greater than that acquired by BPN training algorithm and 5% higher than that produced by HoT training algorithm, and the accuracy obtained by HoT training algorithm is 7% higher than that obtained by BPN. For the Drug Consumption dataset, the accuracy obtained by SHoT training algorithm is 8% greater than that acquired by BPN training algorithm and 3% higher than that produced by HoT training algorithm, and the accuracy obtained by HoT training algorithm is 5% higher than that obtained by BPN. For the Cardiotocography dataset, the accuracy obtained by SHoT training algorithm is 10% greater than that acquired by BPN training algorithm and 4% higher than that produced by HoT training algorithm, and the accuracy obtained by HoT training algorithm is 6% higher than that obtained by BPN. For the Splice-junction Gene Sequences dataset, the accuracy obtained by SHoT training algorithm is 8% greater than that acquired by BPN training algorithm and 5% higher than that produced by HoT training algorithm, and the accuracy obtained by HoT training algorithm is 3% higher than that obtained by BPN. For the Thyroid Disease dataset,
the accuracy obtained by SHoT training algorithm is 7% greater than that acquired by BPN training algorithm and 5% higher than that produced by HoT training algorithm, and the accuracy obtained by HoT training algorithm is 2% higher than that obtained by BPN.

V. CONCLUSION

The experimental outcome based on the simulation demonstrated that the proposed SHOT algorithm outperforms both HoT and BPN algorithm in terms of training time and its efficiency. Regarding training time, the proposed SHOT algorithm decreases the total training time it takes to train the network, which in turn increases the training speed. In comparison to its current supervised algorithm, such as HoT and BPN, the accuracy obtained by the proposed SHOT methods has been improved. Finally, the proposed SHOT approach increases the training performance for any kind of real-world supervised classification task by both training speed and by accuracy compared to the current algorithm. Also, the proposed SHOT algorithm also provides quicker convergence and results in lower values of RMSE compared to the HoT algorithm and standard BP algorithm. The current research can be extended in different ways to originate new learning algorithms such as incorporating Adaptive Skipping Training algorithm variants, applying the optimization technique, injecting the Fuzzy logic, and so on. For any NN application, the proposed training algorithm can be applied.

REFERENCES

[1] D. Faggella, Where Healthcare’s Big Data Actually Comes From, Accessed: Nov. 2, 2019. [Online]. Available: https://www.techmergence.com/where-healthcares-big-data-actually-comes-from

[2] S. Rajasekaran and G. A. V. Pai, Neural Networks, Fuzzy Systems and Evolutionary Algorithms: Synthesis and Applications. New Delhi, India: PHI Learning Pvt. Ltd., 2017.

[3] R. M. Devi, S. Kuppuswami, and R. C. Suganthan, “Fast linear adaptive skipping training algorithm for training artificial neural network,” Math. Problems Eng., vol. 2013, pp. 1–9, Jun. 2013, doi: 10.1155/2013/346949.

[4] D. Nguyen and B. Widrow, “Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights,” in Proc. Int. Joint Conf. Neural Netw. (IJCNN), vol. 4, 1990, pp. 21–26, doi: 10.1109/IJCNN.1990.137819.

[5] R. M. Devi and S. Kuppuswami, “EAST: An exponential adaptive skipping training algorithm for multilayer feedforward neural networks,” WSEAS Trans. Comput., vol. 13, pp. 138–151, Jan. 2014.

[6] A. Levi, Introduction to the Design & Analysis of Algorithms. Boston, MA, USA: Pearson, 2012.

[7] J. F. Kolen and J. B. Pollack, “Back propagation is sensitive to initial conditions,” Complex Syst., vol. 4, pp. 860–867, Jun. 1990.

[8] G. P. Drago and S. Ridella, “Statistically controlled activation weight initialization (SCAWI),” IEEE Trans. Neural Netw., vol. 3, no. 4, pp. 627–631, Jul. 1992, doi: 10.1109/72.143378.

[9] W. F. Schmidt, S. Raudys, M. A. Kraaijveld, M. Skurikhina, and R. P. W. Duin, “Initializations, back-propagation and generalization of feed-forward classifiers,” in Proc. IEEE Int. Conf. Neural Netw., Mar./Apr. 1993, pp. 598–604, doi: 10.1109/ICNN.1993.298625.

[10] G. Li, H. Alnuweiri, Y. Wu, and H. Li, “Acceleration of back propagation through initial weight pre-training with delta rule,” in Proc. IEEE Int. Conf. Neural Netw., Mar./Apr. 1993, pp. 580–585, doi: 10.1109/ICNN.1993.298622.

[11] R. S. Govindaraju, “Artificial neural networks in hydrology. II: Hydrologic applications,” J. Hydrol. Eng., vol. 5, no. 2, pp. 124–137, Apr. 2000, doi: 10.1061/(ASCE)1084-0699(2000)5:2(124).

[12] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: A new learning scheme of feedforward neural networks,” in Proc. IEEE Int. Joint Conf. Neural Netw., vol. 2, Jul. 2004, pp. 985–990, doi: 10.1109/IJCNN.2004.1380068.

[13] S. Ding, H. Zhao, Y. Zhang, X. Xu, and R. Nie, “Extreme learning machine: Algorithm, theory and applications,” Artif. Intell. Rev., vol. 44, no. 1, pp. 103–115, Jun. 2015, doi: 10.1007/s10462-013-9405-z.

[14] E. Castillo, B. Guijarro-Berdinas, O. Fontenla-Romero, A. Alonso-Betanzos, and Y. Bengio, “A very fast learning method for neural networks based on sensitivity analysis,” J. Mach. Learn. Res., vol. 7, no. 7, pp. 1–24, 2006.

[15] B. Guijarro-Berdinas, O. Fontenla-Romero, B. Pérez-Sánchez, and A. Alonso-Betanzos, “A new initialization method for neural networks using sensitivity analysis,” in Proc. Int. Conf. Math. Stat. Modeling, Spain, vol. 2830, 2006, pp. 1–9.

[16] H. Shao and G. Zheng, “A new BP algorithm with adaptive momentum for FNNs training,” in Proc. WRI Global Cong. Intell. Syst., 2009, pp. 16–20, doi: 10.1109/GCIS.2009.68.

[17] G. P. Zhang, “Neural networks for classification: A survey,” IEEE Trans. Syst., Man, Cybern., C, Appl. Rev., vol. 30, no. 4, pp. 451–462, Nov. 2000, doi: 10.1109/36.907072.

[18] S. Haykin, Neural Networks and Learning Machines. London, U.K.: Pearson, 2010.

[19] M. K. S. Alsmadi, K. B. Omar, and S. A. Noah, “Back propagation algorithm: The best algorithm among the multi-layer perceptron algorithm,” IJCSNS, vol. 9, no. 4, pp. 378–383, 2009.

[20] L. Behera, S. Kumar, and A. Patnaik, “On adaptive learning rate that guarantees convergence in feedforward networks,” IEEE Trans. Neural Netw., vol. 17, no. 5, pp. 1116–1125, Sep. 2006, doi: 10.1109/TNN.2006.871812.

[21] B. M. Wilamowski and H. Yu, “Improved computation for Levenberg–Marquardt training,” IEEE Trans. Neural Netw., vol. 21, no. 6, pp. 930–937, Jun. 2010, doi: 10.1109/TNN.2010.2045657.

[22] H. Yu and B. M. Wilamowski, “Neural network training with second order algorithms,” in Human—Computer Systems Interaction: Backgrounds and Applications 2 (Advances in Intelligent and Soft Computing), vol. 99, Z. S. Hippe, J. L. Kulikowski, and T. Mroczek, Eds., Berlin, Germany: Springer, 2012, doi: 10.1007/978-3-642-23172-8_30.

[23] N. Ampazis and S. J. Perantonis, “Two highly efficient second-order algorithms for training feedforward networks,” IEEE Trans. Neural Netw., vol. 13, no. 5, pp. 1064–1074, Sep. 2002, doi: 10.1109/TNN.2002.1031939.

[24] Y. Mei, G. Tan, and Z. Liu, “An improved brain-inspired emotional learning algorithm for fast classification,” Algorithms, vol. 10, no. 2, p. 70, Jun. 2017, doi: 10.3390/a10020070.

[25] M. H. Ali, B. A. D. Al Mohammed, A. Ismail, and M. F. Zolkipli, “A new intrusion detection system based on fast learning network and particle swarm optimization,” IEEE Access, vol. 6, pp. 20255–20261, 2018, doi: 10.1109/ACCESS.2018.2820092.

[26] C. He, Y. Liu, T. Yao, F. Xu, Y. Hu, and J. Zheng, “A fast learning algorithm based on extreme learning machine for regular fuzzy neural network,” J. Intell. Fuzzy Syst., vol. 36, no. 4, pp. 3263–3269, Apr. 2019, doi: 10.3233/JIFS-18046.

[27] S. Wang, F.-L. Chang, J. Wang, and J. Wu, “A fast learning method for feedforward neural networks,” Neurocomputing, vol. 149, pp. 295–307, Feb. 2015, doi: 10.1016/j.neucom.2014.01.065.

[28] K. Nanthini and R. M. Devi, “Adaptive fuzzy C-means for human activity recognition,” in Proc. Int. Conf. Inf. Commun. Embedded Syst. (ICICES), Feb. 2014, doi: 10.1109/ICICES.2014.7033836.

[29] I. Jammoussi and M. B. Nasr, “A hybrid method based on extreme learning machine and self organizing map for pattern classification,” Comput. Intell. Neurosci., vol. 2020, pp. 1–9, Aug. 2020, doi: 10.1155/2020/2918276.

[30] M. M. Khan, A. M. Ahmad, G. M. Khan, and J. F. Miller, “Fast learning neural networks using Cartesian genetic programming,” Neurocomputing, vol. 121, pp. 274–289, Dec. 2013, doi: 10.1016/j.neucom.2013.04.005.