IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

Multi-Disease Classification Model using Strassen's Half of Threshold (SHoT) Training Algorithm in Healthcare Sector

Manjula Devi Ramasamy[1], Keerthika Periasamy[2], Lalitha Krishnasamy[3], Rajesh Kumar Dhanaraj[4], Seifedine Kadry[5], Yunyoung Nam[6]

[1]Department of Computer Science and Engineering, Kongu Engineering College, Tamil Nadu, India. rmanjuladevi.gem@gmail.com
[2]Department of Computer Science and Engineering, Kongu Engineering College, Tamil Nadu, India. keerthikame@gmail.com
[3]Department of Computer Science and Engineering, Kongu Engineering College, Tamil Nadu, India. vrklalitha24@gmail.com
[4]School of Computing Science and Engineering, Galgotias University, Greater Noida, India. sangeraje@gmail.com
[5]Department of Applied data Science, Noroff University College, Kristiansand, Norway. Skadry@gmail.com
[6]Department of Computer Science and Engineering, Soonchunhyang University, South Korea

Corresponding author: Yunyoung Nam (e-mail: ynam@sch.ac.kr).

This research was supported by Korea Institute for Advancement of Technology(KIAT) grant funded by the Korea Government(MOTIE) (P0012724, The Competency Development Program for Industry Specialist) and the Soonchunhyang University Research Fund.

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 and its efficiency on various dataset such as 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 (SHoT) 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.

2. 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[8] assign a fraction of the intended response range to each hidden node, and Drago and Ridella[9] 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 [10] and the DPT (Delta Pre-Training) approach [11] 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. [12].

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[13][14] 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 [15][16]. 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. The momentum coefficient makes the weight modification to be projected up in the slope of the error surface rather than down in the slope as recommended when the latest negative gradient is crossing the prior update [17]. 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. [18]. 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 [19]. 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 to the goal error [20]. Slowing down parameter updates by allowing the learning rate to fluctuate adaptively is one way to avoid this. This will allow the network to make better responses after each weight update. The essential concept behind adaptive learning rate is that if performance falls short of the error objective at each epoch, the learning rate is increased by a constant value. Another constant parameter reduces the learning rate as performance improves. Several dynamic approaches for adaptively assigning the learning rate have been defined, based on the factor inclined to examine. Learning techniques based on the Lyapunov stability theory have been suggested for NNs[21]. The structure of Lyapunov Function-based learning algorithm (LF I) and its modified variant (LF II) are same as that of BPN method, with the exception that the suggested algorithms substitute the fixed learning rate with an adaptable learning rate. The gradient in error is closed to zero when the training reaches the smooth area. The adaptive learning rate will then be high, and weight adjustment will be delayed, resulting in slow convergence to the goal error.

Following that, the algorithm for changing the weight during the training phase has been provided, which derives the second order differential equation from the cost functions. The quasi-Newton methods or Levenberg–Marquardt (LM) algorithms are the most often used second order training algorithms [22][23] and Conjugate Gradient (CG) methods [24]. To perform the fast classification and to improve the accuracy, a new training algorithm is proposed named as GA-BEL that combines Genetic Algorithm (GA) and brain-inspired emotional learning (BEL) algorithm [25]. Based on the optimization technique Particle Swarm Optimization (PSO), proposed a model for learning named as PSO-FLN is proposed by M.H.Ali et.al for Fast Learning Network(FLN) [26] that has been experimented with the intrusion detection system dataset KDD99 which outperforms well in all aspect. Using the Extreme Learning Machine (ELM) as a baseline, an algorithm to perform fast learning is applied on the RFNN (Regular Fuzzy Neural Network) is proposed [27] and a new fast learning method (FLM) has been presented for feedforward neural networks [28]. Next, based on the concept of Adaptive Skipping, a new and fast training approach for ANN (Artificial Neural Network) is instituted by presenting the input samples for training randomly [3][5] and based on the fuzzy system [29]. To train SLFN (single hidden layer feedforward neural network) and to optimize the weight of SLFN, an algorithm is proposed by that hybridize the self-organizing map (SOM) algorithm with ELM algorithm [30]. CGP-based Artificial Neural Network (CGPANN), based on the Cartesian genetic programming (CGP) technique, is a fast-learning neuroevolutionary algorithm applicable for both feedforward and recurrent networks [31]. Even though the following methods produce good outcomes, they are computationally intensive. Convergence slowly has been identified as a serious issue for all learning methods of BPN. An innovative, hybrid, and computationally efficient neural network training algorithm named as Strassen's Half of Threshold (SHoT) Adaptive Skipping Training Algorithm has been presented to improve the training speed of BPN.

3. 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.

3.1 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 $X \in R^{M \times N}$ 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.

$$X = [x_1^1\ x_1^2\ x_2^1\ x_2^2\ \cdots\ x_1^N \cdots x_2^N \vdots\vdots x_M^1\ x_M^2\ \ddots \vdots \cdots x_M^N\ ] = x_i^j \in R^{M \times N}, 1 \le j \le N, 1 \le i \le M,$$

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

$$T = [t_1\ t_2\ \vdots\ t_M\ ] = t_i \in R^M, 1 \le i \le M$$

Let $V \in R^{N \times P}$ be the 2D matrix of size N × 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.

$$V = [v_1^1\ v_2^1\ v_1^2\ v_1^2\ \cdots\ v_P^1 \cdots v_P^2 \vdots\vdots v_1^N\ v_2^N\ \ddots \vdots \cdots v_P^N\ ] = v_j^i \in R^{M \times N}, 1 \le j \le P, 1 \le i \le N$$

Let $\overrightarrow{v_o}$ represents bias vector of size 1× P that is fed into the nodes in the hidden layer.

$$\overrightarrow{v^0} = [v_1^0\ v_2^0\ \cdots\ v_P^0\ ], where\ v_i^0 \in R^P, 1 \le i \le P$$

Let $W \in R^{P \times O}$ be the 2D matrix of size P × 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.

$$W = [w_1^1\ w_2^1\ w_1^2\ w_1^2\ \cdots\ w_O^1 \cdots w_O^2 \vdots\vdots w_1^P\ w_2^P\ \ddots \vdots \cdots w_O^P\ ] = w_j^i \in R^{P \times O}, 1 \le j \le O, 1 \le i \le P$$

Let $\overrightarrow{w_o}$ represents bias vector of size 1× O that is fed into the nodes in the output layer.

$$\overrightarrow{w^0} = [w_1^0\ w_2^0\ \cdots\ w_O^0\ ], where\ w_j^0 \in R^P, 1 \le i \le O$$

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.

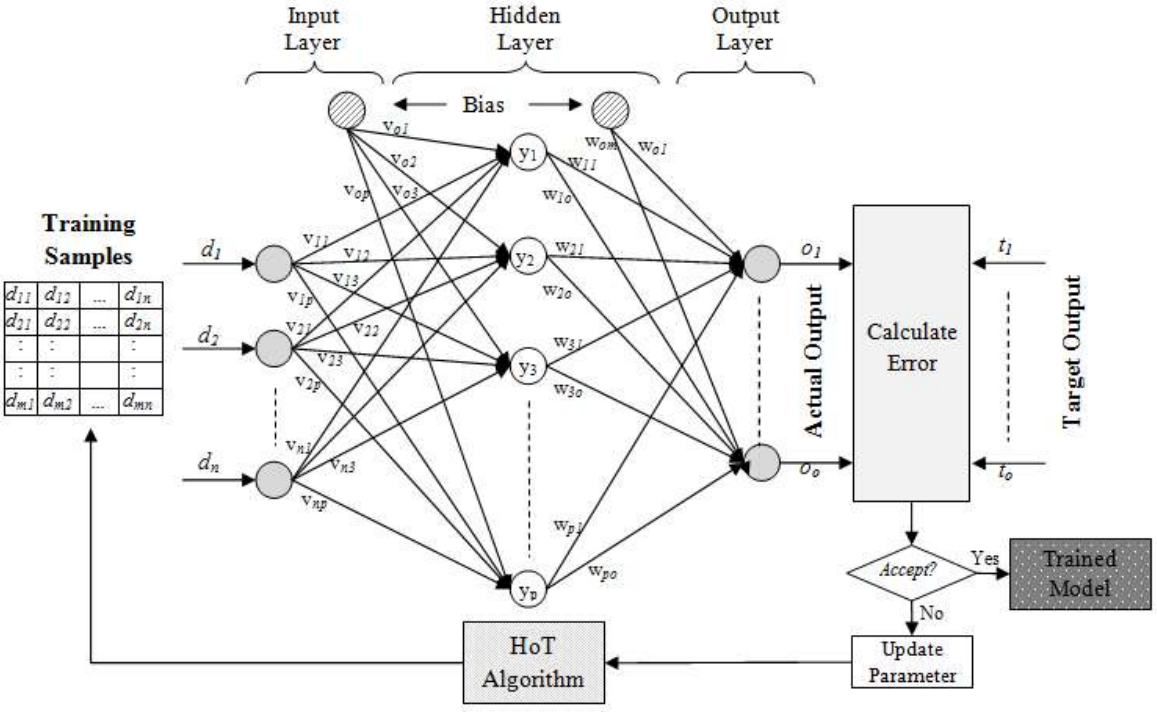

Figure.1: Proposed SHOT Framework

3.2 Working Principle of SHOT

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

- Weight value ϵ $[-1\ 1\ ]$
- Biases value to small random values, typically between -1 and 1
- Learning rate,η
- 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, $1 \le k \le M$

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:

Apply Theorem 1 to estimate the net output value using Strassen's Fast Multiplication of Matrices Algorithm which is specified in Algorithm 1.

$$h\_net_i^t = v_i^{0\,t} + \sum_{j=1}^{N} x_k^j . v_i^{j\,t}, \qquad 1 \le i \le P$$

Estimate the actual output.

$$z^i = \varphi_h(h\_net_i^t), \qquad 1 \le i \le P$$
$$= \frac{1}{1 + e^{-h\_net_i^t}}, \qquad 1 \le i \le P$$

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

$$o\_net_i = w_i^0 + \sum_{j=1}^{P} z^j . w_i^{j\,t}, \ 1 \le i \le O$$

Estimate the actual output

$$y_i^t = \varphi_o(o\_net_i), \qquad 1 \le i \le O$$
$$= \frac{1}{1 + e^{-o\_net_i}}, \qquad 1 \le i \le O$$

Theorem 1[Strassen's Theorem]: Two N × N matrices can be multiplied using only $N^7 \approx N^{2.8074\ldots}$ scalar multiplications.

Step 6: Error Signal Calculation

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

$$E = \sum_k \frac{1}{2}(t_k - y_k)^2, \quad 1 \le k \le M$$

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.

$$\Delta W\ \alpha - \frac{\partial E}{\partial W}$$
$$\Delta w_k^j\ \alpha - \frac{\partial E}{\partial w_k^j}\ 1 \le k \le O$$
$$\Delta w_k^j = -\eta \frac{\partial E}{\partial w_k^j}\ 1 \le k \le O$$

Expand the above error function using chain rule

$$\frac{\partial E}{\partial w_k^j} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial w_k^j}$$
$$\frac{\partial E}{\partial w_k^j} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial o\_net_k} \frac{\partial o\_net_k}{\partial w_k^j}$$

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

$$\frac{\partial o\_net_k}{\partial w_k^j} = \frac{\partial z^j w_k^j}{\partial w_k^j} = z^j$$

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

$$\frac{\partial y_k}{\partial o\_net_k} = \frac{\partial \left(\frac{1}{1 + e^{-o\_net_k}}\right)}{\partial o\_net_k}$$
$$\frac{\partial y_k}{\partial o\_net_k} = \frac{\partial (1 + e^{-o\_net_k})^{-1}}{\partial o\_net_k}$$
$$= -1(1 + e^{-o\_net_k})^{-2} \cdot e^{-o\_net_k} \cdot -1$$
$$\frac{\partial y_k}{\partial o\_net_k} = \frac{e^{-o\_net_k}}{(1 + e^{-o\_net_k})^2}$$

Rewriting the above equation,

$$\frac{e^{-net_k}}{(1 + e^{-net_k})^2} = \frac{1}{(1 + e^{-net_k})} \frac{e^{-net_k}}{(1 + e^{-net_k})}$$
$$= \frac{1}{(1 + e^{-net_k})} \left(1 - \frac{1}{(1 + e^{-net_k})}\right)$$

$$= y_k . (1 - y_k)$$

$$\frac{\partial y_k}{\partial o\_net_k} = y_k . (1 - y_k)$$

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

$$\frac{\partial E}{\partial y_k} = \frac{\partial \frac{1}{2}(t_k - y_k)^2}{\partial y_k}$$
$$= \frac{1}{2} \cdot 2(t_k - y_k) \cdot \frac{\partial (t_k - y_k)}{\partial y_k}$$
$$= (t_k - y_k) \cdot (0 - 1)$$
$$= -(t_k - y_k)$$

Substituting the value of each derivative,

$$\Delta w_k^j = -\eta \frac{\partial E}{\partial w_k^j} = -\eta \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial o\_net_k} \frac{\partial o\_net_k}{\partial w_k^j}$$
$$= \eta (t_k - y_k).y_k(1 - y_k).z^j$$

$$= \eta \delta_k z^j \quad where\ \delta_k = (t_k - y_k).y_k.(1 - y_k)$$

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

$$\Delta V\ \alpha - \frac{\partial E}{\partial V}$$

$$\Delta v_i^j\ \alpha - \frac{\partial E}{\partial v_i^j}$$

$$\Delta v_i^j = -\eta \frac{\partial E}{\partial v_i^j}$$

$$\frac{\partial E}{\partial v_i^j} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial o\_net_k} \frac{\partial o\_net_k}{\partial z^j} \frac{\partial z^j}{\partial h\_net_j} \frac{\partial h\_net_j}{\partial v_i^j}$$

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

$$\frac{\partial z^j}{\partial h\_net_j} = \frac{\partial z^j}{\partial h\_net_j}$$
$$\frac{\partial z^j}{\partial h\_net_j} = \frac{\partial \left(\frac{1}{1 + e^{-h\_net_j}}\right)}{\partial o\_net_k}$$
$$= \frac{\partial \left(1 + e^{-h\_net_j}\right)^{-1}}{\partial h\_net_j}$$
$$= -1\left(1 + e^{-h\_net_j}\right)^{-2} \cdot e^{-h\_net_j} \cdot -1$$
$$\frac{\partial z^j}{\partial h\_net_j} = \frac{e^{-h\_net_j}}{\left(1 + e^{-h\_net_j}\right)^2}$$

Rewriting the above equation,

$$\frac{e^{-h\_net_j}}{\left(1 + e^{-h\_net_j}\right)^2} = \frac{1}{\left(1 + e^{-h\_net_j}\right)} \frac{e^{-h\_net_j}}{\left(1 + e^{-h\_net_j}\right)}$$

$$= \frac{1}{\left(1 + e^{-h\_net_j}\right)} \left(1 - \frac{1}{\left(1 + e^{-h\_net_j}\right)}\right)$$

$$= z^j.(1 - z^j)$$

$$\frac{\partial z^j}{\partial h\_net_j} = z^j.(1 - z^j)$$

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

$$\frac{\partial h\_net_j}{\partial v_i^j} = \frac{\partial x_k^j v_i^j}{\partial v_i^j} = x_k^j$$
$$\Delta v_i^j = -\eta \frac{\partial E}{\partial v_i^j}$$

Substituting the value of each derivative,

$$\Delta v_i^j = -\eta \frac{\partial E}{\partial v_i^j}$$
$$= -\eta \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial o_{net_k}} \frac{\partial o_{net_k}}{\partial z^j} \frac{\partial z^j}{\partial h_{net_j}} \frac{\partial h_{net_j}}{\partial v_i^j}$$
$$= [\eta \delta_k z^j]\, z^j(1 - z^j) x_k^j$$
$$= \eta \delta_j x_k^j$$

Step 7: Backward Propagation

Step 7.1: Update weights hidden to output weight using the Delta-Learning Rule

$$w_k^j = w_k^j + \eta \delta_k z^j$$

Step 7.2: Update weights input to hidden weight using the Delta-Learning Rule

$$v_i^j = v_i^j + \eta \delta_j x_k^j$$

Step 8: AST Algorithm

Step 8.1: Calculate the difference between the neural network output's target(t_k) and actual(y_k) value, $t_k - y_k$.

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.

$$|t_k - y_k| < d_{max}/2$$

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

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

Step 8.5: If the $prob(x_i)$ 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.

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

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.

4. 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.

4.1 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 [15]. 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 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.

Table 1. Dataset Properties

Dataset	Number of Instances	Number of Attributes	Number of Classes	Classification Type
Hepatitis	155	19	2	Binary
SPeCT Heart	267	22	2	Binary
Liver Disorders	345	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	61	3	Multiclass
Thyroid Disease	7200	21	3	Multiclass

4.2 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 adapted 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.

$$Accuracy = \frac{Number\ of\ samples\ that\ are\ classified\ correctly}{Total\ number\ of\ samples}$$

4.2.1 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.

Table 2. Comparison Results of Various Learning Algorithm Trained using Hepatitis Dataset with ŋ=1e-4

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

4.2.2 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.

Table 3. Comparison Results of Various Learning Algorithm Trained using SPeCT Heart Dataset with ŋ=1e-4

Fold Number	BPN		HoT		SHoT	
	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)
1	89.0843	88.89	78.6745	96.3	45.8612	96.3
2	96.3294	85.19	74.7164	92.59	49.1764	100
3	88.9225	81.48	59.1024	92.59	42.6014	81.48
4	74.1832	85.19	74.4453	96.3	48.3799	96.3
5	69.0828	70.37	77.6458	74.07	42.2692	88.89

6	91.6944	88.89	58.7441	74.07	48.7352	96.3
7	85.7959	88.89	105.6180	96.3	47.8319	92.59
8	74.8809	70.37	65.3217	77.78	46.1017	81.48
9	89.1431	66.67	74.1911	92.59	42.5950	96.3
10	81.9943	81.48	87.2486	77.78	43.9491	85.19
Avg:	84.1111	80.7420	75.5708	87.0370	45.7501	91.4830

4.2.3 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.

Table 4. Comparison Results of Various Learning Algorithm Trained using Liver Disorders Dataset with ŋ=1e-4

Fold Number	BPN		HoT		SHoT	
	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)
1	37.0966	80	47.3319	85.71	53.1991	91.43
2	40.3681	85.71	45.9930	88.57	40.5809	91.43
3	44.5172	80	39.8379	80	53.7893	94.29
4	65.2276	88.57	54.7378	85.71	50.9335	80
5	65.8828	82.86	51.1167	91.43	29.6916	94.29
6	41.9939	85.71	34.3110	80	25.8202	94.29
7	64.4417	88.57	52.4936	88.57	29.6721	88.57
8	47.0853	77.14	56.7969	85.71	29.2445	97.14
9	45.6105	88.57	47.4441	91.43	29.8929	85.71
10	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

4.2.4 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.

Table 5. Comparison Results of Various Learning Algorithm Trained using Breast Cancer Wisconsin (Diagnostic) Dataset with ŋ=1e-4

Fold Number	BPN		HoT		SHoT	
	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)
1	101.3862	78.95	64.6625	94.74	43.7087	92.98
2	96.3294	78.95	63.3403	92.98	48.0980	98.25
3	88.9225	85.96	59.3364	82.46	43.7007	91.23
4	109.3300	75.44	60.7605	87.72	47.2308	96.49
5	103.9083	77.19	56.3224	91.23	43.2749	96.49
6	91.6944	89.47	56.3265	89.47	42.3389	73.68
7	85.7959	87.72	56.7478	82.46	44.3271	96.49
8	105.4725	77.19	63.7920	80.7	45.4686	98.25
9	106.1723	78.95	57.6802	96.49	45.9082	82.46
10	81.9943	85.96	59.0849	78.95	45.2612	98.25
Avg:	97.1006	81.5780	59.8053	87.7200	44.9317	92.4570

4.2.5 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 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.

Table 6. Comparison Results of Various Learning Algorithm Trained using Drug Consumption Dataset with ŋ=1e-4

Fold Number	BPN		HoT		SHoT	
	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)
1	93.4230	86.77	74.5535	89.95	62.0459	98.41
2	91.9963	90.48	99.8428	93.65	87.6630	92.59
3	104.6784	86.24	70.9701	92.59	80.6729	99.47
4	90.2273	92.06	97.3860	95.24	65.6849	92.59
5	86.8110	84.65	59.5250	87.3	58.6658	98.41
6	75.7399	86.77	69.3283	97.76	72.1315	98.41
7	75.7751	94.18	72.0838	91.01	48.8140	90.48
8	95.9685	89.42	104.3691	94.71	68.1416	96.83
9	95.6195	89.17	102.3419	88.89	86.3361	94.18
10	99.0994	84.12	92.5225	96.3	68.2187	97.88
Avg:	90.9338	88.3860	84.2923	92.7400	69.8374	95.9250

4.2.6 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.

Table 7. Comparison Results of Various Learning Algorithm Trained using Cardiotocography Dataset with ŋ=1e-4

Fold Number	BPN		HoT		SHoT	
	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)
1	103.0846	87.13	86.8696	89.67	66.8196	98.12
2	102.4240	88.3	67.6612	93.9	80.9484	90.61
3	79.5093	81.87	67.9806	93.43	96.9357	96.24
4	96.7652	81.87	113.0080	90.61	75.3049	99.06
5	107.7290	82.46	90.9146	84.98	59.5661	93.9
6	112.9056	85.96	85.0187	86.38	114.5687	92.02
7	88.8774	86.55	70.4095	91.55	79.2127	95.31
8	95.2970	89.47	113.6688	95.31	77.7937	91.55
9	108.2056	86.55	109.7794	81.87	66.6012	90.14
10	76.4921	84.21	95.8990	97.65	62.9317	98.12
Avg:	97.1290	85.4370	90.1209	90.5350	78.0683	94.5070

4.2.7 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.

Table 8. Comparison Results of Various Learning Algorithm Trained using Splice-junction Gene Sequences Dataset with ŋ=1e-4

Fold Number	BPN		HoT		SHoT	
	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)
1	87.5880	88.4	88.1602	90.91	76.1594	96.55
2	103.3754	84.95	91.4030	91.85	77.1280	95.61
3	107.0662	86.83	83.5459	88.09	68.4007	99.37
4	113.8473	85.58	95.0122	92.48	68.4088	94.36
5	76.2946	87.77	77.9273	98.75	74.2995	98.12
6	85.9242	91.22	74.1780	92.48	66.8254	97.18

7	78.8116	94.04	93.3203	88.71	80.4364	98.12
8	102.8460	92.79	86.7354	88.09	69.1237	92.48
9	97.7427	89.66	86.9934	93.1	68.7194	91.54
10	109.7966	86.83	97.9197	88.4	66.3883	96.87
Avg:	96.3293	88.81	87.5196	91.29	71.5889	96.02

4.2.8 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.

Table 9. Comparison Results of Various Learning Algorithm Trained using Thyroid Disease Dataset with ŋ=1e-4

Fold Number	BPN		HoT		SHoT	
	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)	Training Time (in Sec)	Accuracy (%)
1	101.3862	93.19	64.6625	87.64	42.7087	96.81
2	96.3294	86.53	63.3403	89.44	48.0980	95.56
3	88.9225	86.11	59.3364	94.03	43.7007	91.39
4	109.3300	87.08	60.7605	90	50.2308	95.56
5	103.9083	92.36	56.3224	90.83	43.2749	92.5
6	91.6944	90.97	56.3265	94.72	42.3389	97.78
7	96.7959	85.42	56.7478	92.92	44.3271	97.36
8	105.4725	88.75	63.7920	89.31	45.4686	94.31
9	106.1723	90.56	57.6802	88.47	51.0082	96.11
10	84.9943	87.92	59.0849	92.08	41.2612	98.75
Avg:	98.5006	88.8890	59.8053	90.9440	45.2417	95.6130

4.3 Result Comparison and Discussion

4.3.1 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 3.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3103746, IEEE Access

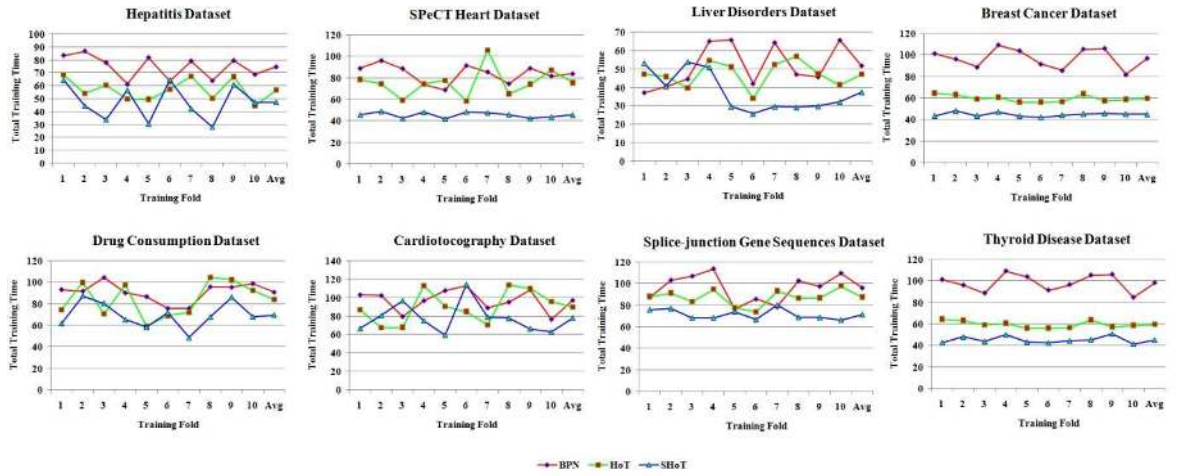

Figure 3: Training Time Comparison

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.

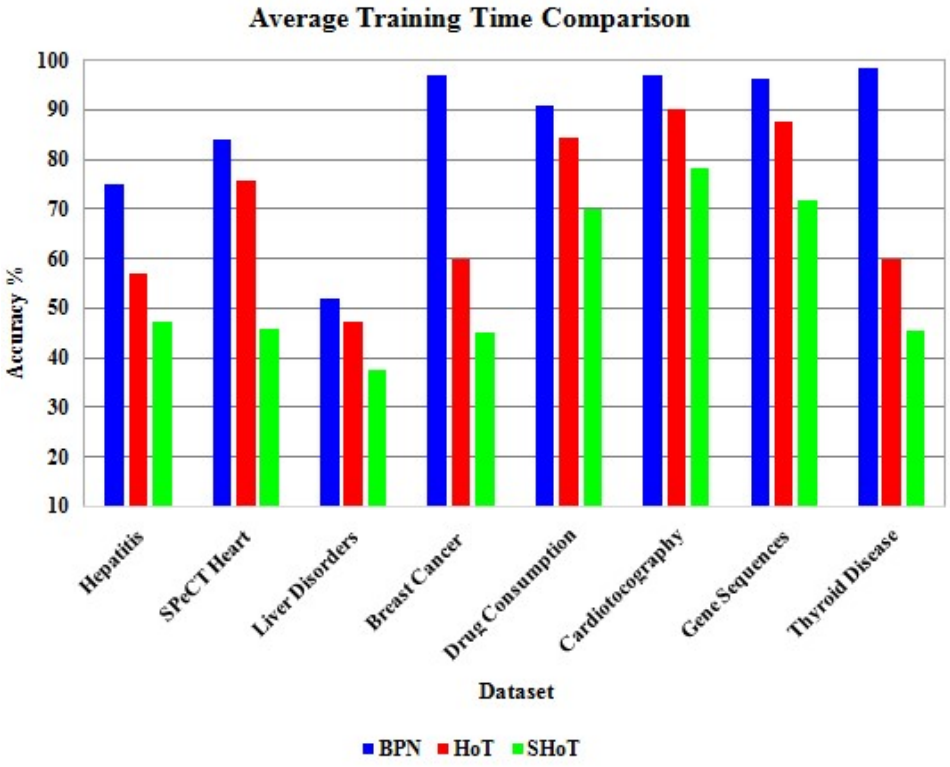

Figure 4: Average Training Time Comparison

4.3.2 Accuracy

The comparison result of the accuracy consumed by various training algorithm such as BPN, LAST and SHOT is illustrated in the Figure 4.

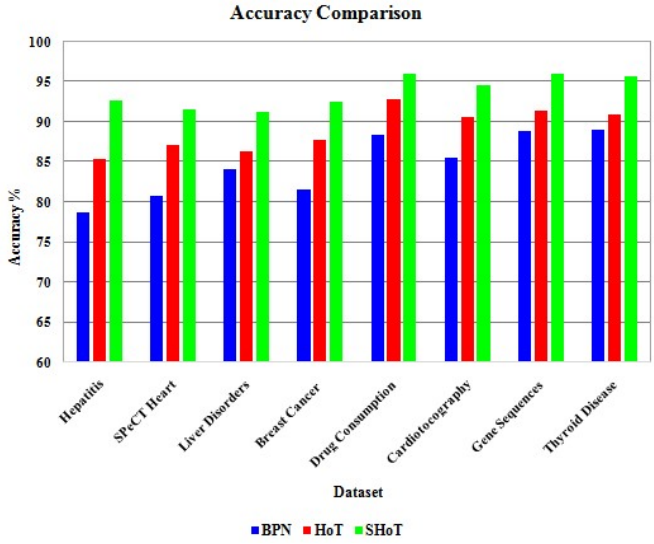

Figure 5: Accuracy Comparison

From Figure 4 and 5, 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.

5. Conclusions

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,” Tech Emerg, vol. 11, 2018.
[2] S. Rajasekaran and G. A. V. Pai, Neural Networks, Fuzzy Systems and Evolutionary Algorithms: Synthesis and Applications. PHI Learning Pvt. Ltd., 2017.
[3] R. Manjula Devi, S. Kuppuswami, and R. C. Suganthe, “Fast linear adaptive skipping training algorithm for training artificial neural network,” Math. Probl. Eng., vol. 2013, 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,” IJCNN. Int. Jt. Conf. Neural Networks, pp. 21–26, 1990, 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, 2014.

[6] A. Levitin, Introduction to the design & analysis of algorithms. Boston: Pearson, 2012.
[7] J. F. Kolen, J. F. Kolen, J. B. Pollack, and J. B. Pollack, "Back Propagation is Sensitive to Initial Conditions," in Complex Systems, 1990, pp. 860–867.
[8] D. Nguyen and B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," in 1990 IJCNN International Joint Conference on Neural Networks, 1990, pp. 21–26 vol.3, doi: 10.1109/IJCNN.1990.137819.
[9] G. P. Drago and S. Ridella, "Statistically controlled activation weight initialization (SCAWI)," IEEE Trans. Neural Networks, vol. 3, no. 4, pp. 627–631, Jul. 1992, doi: 10.1109/72.143378.
[10] 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 IEEE International Conference on Neural Networks, pp. 598–604, doi: 10.1109/ICNN.1993.298625.
[11] G. Li, H. Alnuweiri, Y. Wu, and H. Li, "Acceleration of back propagation through initial weight pre-training with delta rule," in IEEE International Conference on Neural Networks, pp. 580–585, doi: 10.1109/ICNN.1993.298622.
[12] "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).
[13] Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew, "Extreme learning machine: a new learning scheme of feedforward neural networks," in 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No.04CH37541), vol. 2, pp. 985–990, doi: 10.1109/IJCNN.2004.1380068.
[14] 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.
[15] 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, 2006.
[16] B. Guijarro-Berdinas, O. Fontenla-Romero, B. Pérez-Sánchez, and A. Alonso-Betanzos, "A new initialization method for neural networks using sensitivity analysis," in International Conference on Mathematical and Statistical Modeling, Spain, 2006, vol. 2830.
[17] H. Shao and G. Zheng, "A New BP Algorithm with Adaptive Momentum for FNNs Training," in 2009 WRI Global Congress on Intelligent Systems, 2009, pp. 16–20, doi: 10.1109/GCIS.2009.136.
[18] G. P. Zhang, "Neural networks for classification: a survey," IEEE Trans. Syst. Man Cybern. Part C (Applications Rev., vol. 30, no. 4, pp. 451–462, 2000, doi: 10.1109/5326.897072.
[19] S. Haykin, Neural networks and learning machines, 3/E. Pearson Education India, 2010.
[20] A. M.Kh.S., O. Kh.B., and N. Sh.A., "Back Propagation Algorithm: The Best Algorithm Among the Multi-layer Perceptron Algorithm," 2009.
[21] L. Behera, S. Kumar, and A. Patnaik, "On Adaptive Learning Rate That Guarantees Convergence in Feedforward Networks," IEEE Trans. Neural Networks, vol. 17, no. 5, pp. 1116–1125, Sep. 2006, doi: 10.1109/TNN.2006.878121.
[22] B. M. Wilamowski and Hao Yu, "Improved Computation for Levenberg–Marquardt Training," IEEE Trans. Neural Networks, vol. 21, no. 6, pp. 930–937, Jun. 2010, doi: 10.1109/TNN.2010.2045657.
[23] H. Yu and B. M. Wilamowski, "Neural Network Training with Second Order Algorithms," 2012, pp. 463–476.
[24] N. Ampazis and S. J. Perantonis, "Two highly efficient second-order algorithms for training feedforward networks," IEEE Trans. Neural Networks, vol. 13, no. 5, pp. 1064–1074, Sep. 2002, doi: 10.1109/TNN.2002.1031939.
[25] 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.
[26] 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.
[27] 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.
[28] S. Wang, F.-L. Chung, 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.
[29] K. Nanthini and R. M. Devi, "Adaptive fuzzy C-means for human activity recognition," in International Conference on Information Communication and Embedded Systems (ICICES2014), Feb. 2014, pp. 1–5, doi: 10.1109/ICICES.2014.7033836.
[30] I. Jammoussi and M. Ben 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.
[31] M. Mahsal Khan, A. Masood Ahmad, G. Muhammad Khan, and J. F. Miller, "Fast learning neural networks using Cartesian genetic programming," Neurocomputing, vol. 121, pp. 274–289, 2013, doi: https://doi.org/10.1016/j.neucom.2013.04.005.