Research Journal of Applied Sciences, Engineering and Technology 12(8): 790-813, 2016
DOI:10.19026/rjaset.12.2780
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: November 11, 2013 Accepted: November 29, 2013 Published: April 15, 2016

Research Article

CAST: A constant Adaptive Skipping Training Algorithm for Improving the Learning Rate of Multilayer Feedforward Neural Networks

R. Manjula Devi and S. Kuppuswami
Faculty of Computer Science and Engineering, Kongu Engineering College, Perundurai, Erode

Abstract: Multilayer Feedforward Neural Network (MFNN) has been administered widely for solving a wide range of supervised pattern recognition tasks. The major problem in the MFNN training phase is its long training time especially when it is trained on very huge training datasets. In this accordance, an enhanced training algorithm called Constant Adaptive Skipping Training (CAST) Algorithm is proposed in this research paper which intensifies on reducing the training time of the MFNN through stochastic manifestation of training datasets. The stochastic manifestation is accomplished by partitioning the training dataset into two completely separate classes, classified and misclassified class, based on the comparison result of the calculated error measure with the threshold value. Only the input samples in the misclassified class are exhibited to the MFNN for training in the next epoch, whereas the correctly classified class is skipped constantly which dynamically reducing the number of training input samples exhibited at every single epoch. Thus decreasing the size of the training dataset constantly can reduce the total training time, thereby speeding up the training process. This CAST algorithm can be merged with any training algorithms used for supervised task, can be used to train the dataset with any number of patterns and also it is very simple to implement. The evaluation of the proposed CAST algorithm is demonstrated effectively using the benchmark datasets - Iris, Waveform, Heart Disease and Breast Cancer for different learning rate. Simulation study proved that CAST training algorithm results in faster training than LAST and standard BPN algorithm.

Keywords: Adaptive skipping, learning rate, MFNN, neural network, training algorithm, training speed

INTRODUCTION

Multilayer Feedforward Neural Network (MFNN) with a single hidden layer has been explored as the best neural network architecture for nonlinear classification problem due to its capability to approximate any nonlinear function mapping (Mehra and Wah, 1992; Hornik *et al*., 1989; Huang *et al*., 2000). The Back Propagation (BPN) is the most popular supervised training algorithm that has been used to train MFNN extensively for the past two decades (Razavi and Tolson, 2011). It is fragmented into two phases: Training Phase (also called as Learning Phase) and Testing Phase (also called as Evaluation Phase). Among these two phases, the training phase plays an important role in establishing nonlinear models. In order to obtain better performance, it still requires many epochs for training the simple problem using MFNN. So the BPN is unfortunately very slow. And also BPN training performance is literally associated with the type and size of network architecture, the number of epochs and patterns to be trained, training speed, and the dimensionality of the training datasets.

In order to enhance the training performance, the training speed is the factor that is considered to be very important. The training speed is highly depends on the dimensionality of training dataset. In general, training MFNN with a larger training datasets will generalize the network well. But, lengthy training time is needed for larger training dataset (Behera *et al*., 2006) which influence the training speed.

This research proposes a new training algorithm to improve the training speed by reducing the training time of MFNN through the stochastic manifestation of training datasets. The correctly classified class input samples in the training datasets will be skipped constantly from the training for the consecutive n epochs. Thereby, the CAST algorithm dynamically diminishing the number of training input pattern samples constantly exhibited at every single epoch. Thus diminishing the size of the training datasets constantly can reduce the total training time, thereby speeding up the training process. Hence, the overall training time for actual training of the MFNN is often reduced by several hundred times than in the standard training algorithm. This method is carried out by merging into any algorithm used for training the supervised task.

The content of this research paper is materialized as follows. The brief review of the previous works done

Corresponding Author: R. Manjula Devi, Faculty of Computer Science and Engineering, Kongu Engineering College, Perundurai, Erode

relevant to the research problem is given and then the formulation of the given research problem s shown. The proposed CAST algorithm is presented. Followed by Performance evaluation of CAST using the benchmark datasets for the classification problems is simulated. Finally, the experimental results are summarized and analyzed along with the conclusions of the research paper.

RELATED WORKS

In order to speed up the MFNN training process, many researchers have investigated the above detriments and devoted many of their research works through various formation ranges from different amendments of existing algorithms to evolution of new algorithms. The formation of improving the training speed and maintain the generalization includes initialization of optimal initial weight (Nguyen and Widrow, 1990; Varnava and Meade Jr., 2011), adaptation of learning rate (Plagianakos *et al*., 1998), adaptation of the momentum term (Shao and Zheng, 2009), adaptation of the momentum term in parallel with learning rate adaptation (Behera *et al*., 2006), and using second order algorithm (Ampazis and Perantonis, 2002; Wilamowski and Yu, 2010; Yu and Wilamowski, 2012).

During the training process, the number of iterations will be scaled down through the proper initialization of the weight which in turn will increase the training speed. Some of the techniques applied for initializing the weight have been discussed here. Nguyen and Widrow (1990) initialize the layer's intermediate weight within the specified range for faster learning. Varnava and Meade Jr. (2011) used the polynomial mathematical models for obtaining the network synaptic initial value. The learning rate is one of the training parameters that fine-tune the size of the network's respective old weights during learning. Assigning the constant value of the learning rate will degrade the speed of the training which results in slow convergence. But, adaptation of learning rate using the Barzilai and Borwein is proposed by Plagianakos *et al*. (1998) in order to improve the convergence speed. Based on the factor inclined to investigate, several dynamic methods for assigning the learning rate adaptively have been codified. Behera *et al*. (2006) developed two new algorithms designated as LFI and LF II from Lyapunov theory of stability where the learning rate is assigned to the adaptive values instead of fixed value. Next, the algorithm that derives the second order differential equation from the cost functions for updating the weight during the training process has been listed. The most popular second order training algorithm are quasi-Newton methods or Levenberg–Marquardt (LM) (Wilamowski and Yu, 2010; Yu and Wilamowski, 2012) and conjugate gradient (CG) methods (Ampazis and Perantonis, 2002). Eventhough, the above second order approaches achieve good results, but they are computationally very expensive. Ampazis and Perantonis (2002) extracted the importance of the Levenberg–Marquardt and Conjugate Gradient methods and derived the two different approaches Levenberg–Marquardt with adaptive momentum (LMAM) and optimized Levenberg–Marquardt with adaptive momentum (OLMAM) second order algorithm. Wilamowski and Yu (2010) applied vector multiplication for determining the gradient vector and Hessian matrix instead of matrix multiplication (Yu and Wilamowski, 2012) which significantly reduces the cost of memory cost for training and thereby improves the training speed.

However, the disadvantages found in the traditional method are not surmounted by the above discussed techniques. All of the above mentioned efforts are focused directly or indirectly on tuning the network's training parameters. And besides, the formation discussed above consumes totally all the input samples till the training terminates. If a large amount of training data with high dimension is rendered for classification, then a problem is introduced by the above discussed technique which will slow down classification. So, the intention of this research is to impart a simple and new algorithm CAST for training the ANN in a fast manner by presenting the training input samples randomly based on the classification.

Problem formulations: BPN algorithm is an iterative gradient training algorithm designed to estimate the coefficients of weight matrices that minimizes the total Root Mean Squared Error (RMSE). The RMSE is defined between the desired output and the actual output summed over all the training pattern input to the network.

$$RMSE = \frac{1}{P}\sum_{p=1}^{P} E^{p} \tag{1}$$

E^p is calculated using the following formula

$$E^{p} = \frac{1}{2}\sum_{k=1}^{m}(t_k^p - y_k^p)^2 \tag{2}$$

Where P is the total number of training sample patterns, m is the number of nodes in the output layer, t_k^p is the target output of the kth node for the pth sample pattern, and y_k^p is the actual output of the kth node estimated by the network for the pth sample pattern.

According to the Equation (2), there is a real fact that the correctly classified input samples does not involve in the updating of weight since the error value generated by that sample pattern is zero. Here the intention of this research is to partition the training input samples into two distinct classes, classified and misclassified class, based on the comparison result of the calculated error measure with the maximum threshold value. By doing so, the training input samples whose actual output is same as target output will belong to the classified class; the remaining training input samples will belong to the misclassified class. Only the input samples in the misclassified class are presented to the next epoch (Epoch is one complete cycle of populating the MFNN with the entire training samples once) for training, whereas the correctly classified class will not be presented again for the subsequent n epochs. The adaptive skipping training algorithm is used to estimate the skipping factor value. In the LAST algorithm (Devi *et al.*, 2013), the value of skipping factor is increased linearly that is the input samples are skipped linearly. In the proposed CAST algorithm, the correctly classified class input samples will be skipped constantly from the training for the consecutive n epochs. Thereby, the CAST algorithm dynamically diminishing the number of training input pattern samples constantly exhibited at every single epoch. Thus diminishing the size of the training datasets constantly can reduce the total training time, thereby speeding up the training process. The dominance of this CAST algorithm is that its implementation is extremely simple and easy, and can lead to significant advances in the training speed.

PROPOSED CAST METHOD

Overview of CAST Architecture: The CAST algorithm that is contained in the prototypical MFNN architecture is outlined in Fig. 1.

Assume that the network contains n input nodes in the input layer, p hidden nodes in the hidden layer and m output nodes in the output layer. Since the above network is highly interconnected, the nodes in each layer are connected with all the nodes in the next layer. Let P represent the number of input patterns in the training dataset. The input matrix, X, of size p × n is presented to the network. The number of nodes in the input layer is equivalent to the number of columns in the input matrix, X. Each row in X is considered to be a real-valued vector $x_i \epsilon \Re^{n+1}$ where $1 \leq i \leq n$. The summed real-valued vector generated from the hidden layer is represented $z_i \epsilon \Re^{p+1}$ where $1 \leq i \leq p$. The estimated output real-valued vector generated from the network is denoted as $y_i \epsilon \Re^m$ where $1 \leq i \leq m$ and the corresponding target vector is represented as $t_i \epsilon\ \Re^m$ where $1 \leq i \leq m$. Let it implies the it^{th} iteration number.

Let $f_N(x)$ and $f_L(x)$ be the non-linear logistic activation function and linear activation function used for computation in the hidden and output layer respectively. Let v_{ij} be the $n \times p$ weight matrix contains input-to-hidden weight coefficient for the link from the input node i to the hidden node j and v_{oj} be the bias weight to the hidden node j. Let w_{jk} be the $p \times m$ weight matrix contains hidden-to-output weight coefficient for the link from the hidden node j to the output node k and w_{ok} be the bias weight to the output node k.

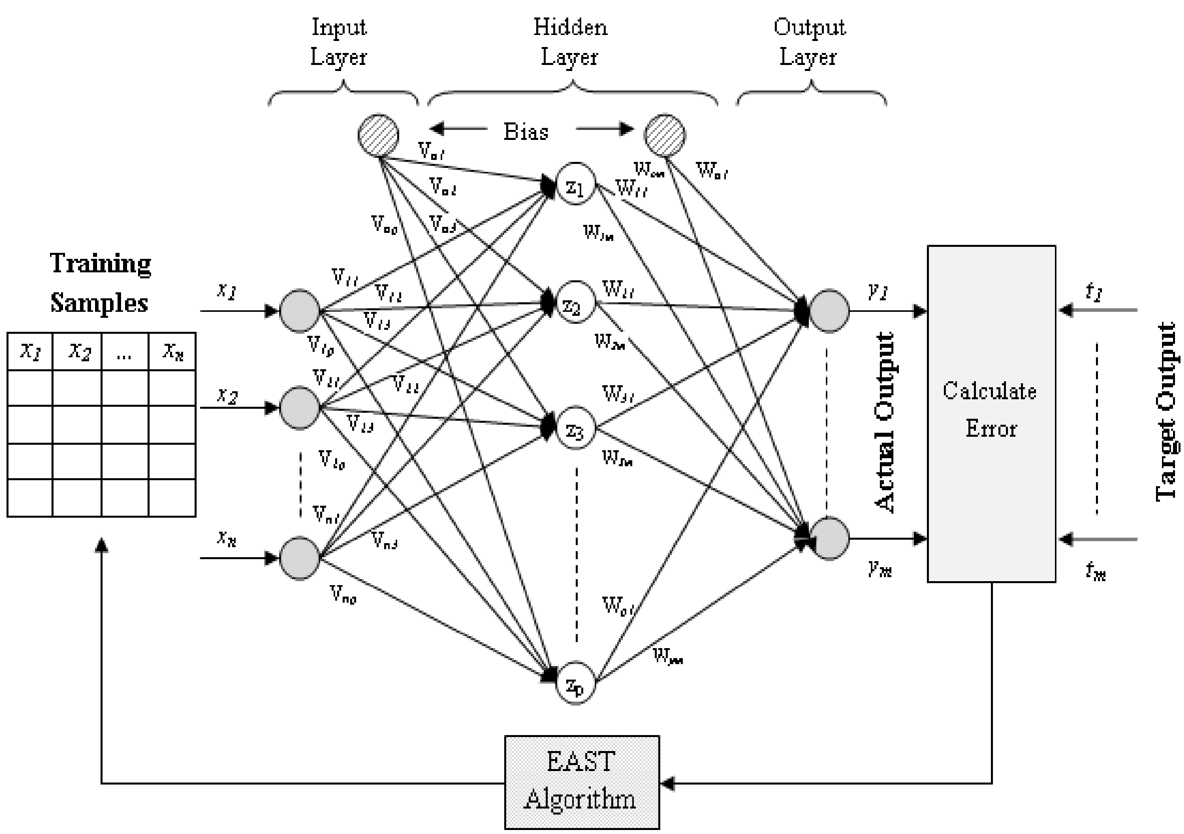

Fig. 1: Architecture of MFNN with CAST algorithm

Proposed CAST Algorithm: The working principle of the CAST algorithm that is incorporated in the BPN algorithm is summarized below:

Step 1: **Weight initialization:** Initialize weights to small random values;
Step 2: **Furnish the input sample:** Disseminate to the input layer an input sample vector x_k having desired output vector y_k;
Step 3: **Forward phase:** Starting from the first hidden layer and propagating towards the output layer:

- **Calculate the activation values** for the Hidden layer as:
  - Estimate the net output value:

$$z_{inj}(it) = v_{oj}(it) + \sum_{i=1}^{n} x_i(it).vij(it) \quad (3)$$

  - Estimate the actual output:

$$z_j(it) = \frac{1}{1+e^{-z_{inj}}} \quad (4)$$

- **Calculate the activation values** for the Output layer as:
  - Estimate the net output value:

$$y_{ink}(it) = w_{ok}(it) + \sum_{j=1}^{p} z_j(it).w_{jk}(it) \quad (5)$$

  - Estimate the actual output:

$$y_k(it) = \frac{1}{1+e^{-y_{ink}}} \quad (6)$$

Step 4: **Output errors:** Calculate the error terms at the output layer as:

$$\delta_k(it) = [t_k - y_k(it)].f'(y_k(it)) \quad (7)$$

Differentiate the activation function in Equation 6:

$$f'(y_k(it)) = \frac{\partial(y_k(it))}{\partial x} = y_k(it) \times (1 - y_k(it)) \quad (8)$$

Substitute the resultant value of Equation (8) in (7):

$$\delta_k(it) = y_k(it).[1 - y_k(it)].[t_k - y_k(it)] \quad (9)$$

Step 1: **Backward phase:** Propagate error backward to the input layer through the hidden layer using the error term.

$$\ddot{a}_j(it) = \left[\sum_{k=1}^{m} \delta_j(it).w_{jk}(it)\right].f'\left(z_j(it)\right) \quad (10)$$

Differentiate the activation function in Equation 4:

$$f'\left(z_j(it)\right) = \frac{\partial\left(z_j(it)\right)}{\partial x} = z_j(it) \times \left(1 - z_j(it)\right) \quad (11)$$

Substitute the resultant value of Equation (11) in (10):

$$\delta_j(it) = \left[\sum_{k=1}^{m} \delta_j(it).w_{jk}(it)\right] z_j(it).\left[1 - z_j(it)\right] \quad (12)$$

Step 2: **Weight amendment:** Update weights using the Delta-Learning Rule.

Weight amendment: For Output Unit.

$$W_{jk}(it+1) = W_{jk}(it) + \alpha(it).\delta_k(it).z_j(it) \quad (13)$$

Weight amendment: For Hidden Unit.

$$V_{ij}(it+1) = V_{ij}(it) + \alpha(it)\,\delta_j(it)\,x_i(it) \quad (14)$$

Step 3: **CAST Algorithm:** Incorporating the CAST algorithm.

- **Compare** the error value, $|t_k - y_k|$ with threshold value, d_{max}.

$$|t_k - y_k(it)| < d_{max} \quad (15)$$

If equation 15 generates 0, then the x_i is correct.

- **Compute**: The probability value for all input samples.

$$prob(x^i) = \begin{cases} 0, \text{if } x_i \text{ is correct and epoch number} < n \\ 1, otherwise \end{cases} \quad (16)$$

- **Calculate** the skipping factor, sf_i, for all input samples
  - Initialize the value of sf_i to zero (for first epoch)
  - Increment the value of sf_i constantly for correctly classified samples alone.
- **Skip** the training samples with prob (=0) for the next sf_i epoch

Step 4: **Repeat** steps 1-7 until the halting criterion is satisfied, which may be chosen as the Root Mean Square Error (RMSE), elapsed epochs and desired accuracy.

Working flow of CAST: The block diagram of the proposed strategy is illustrated in the Fig. 2.

Empirical result and analysis: This section holds about the description of the dataset used for the research, the experimental design and results.

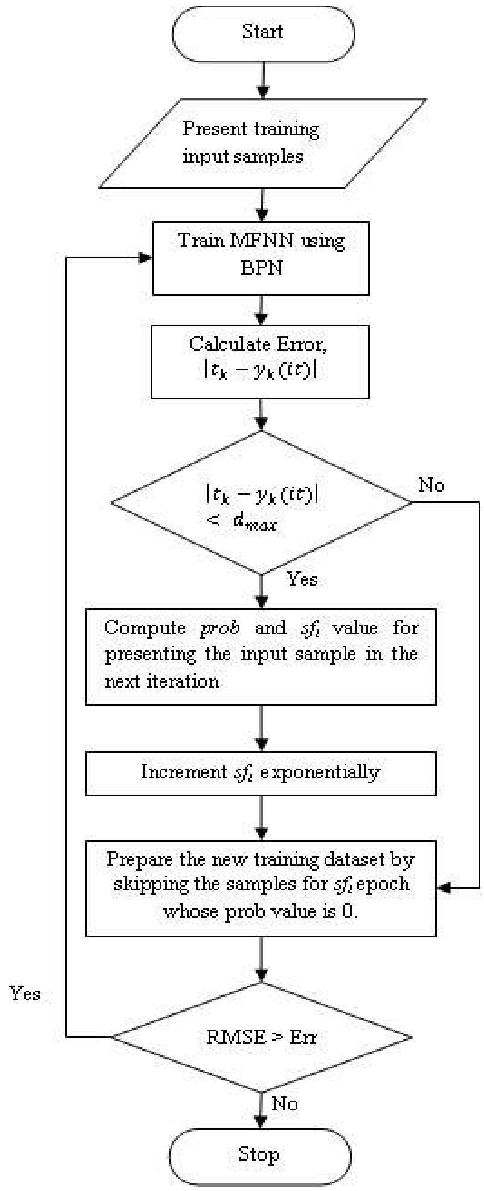

Fig. 2: Flow diagram of CAST training algorithm

Dataset properties: In this section, the performance of the proposed CAST algorithm is evaluated on the benchmark two-class classification and multi-class classification problems. The benchmark datasets used for two-class classification problem are Iris and Waveform Data Set, and multiclass classification problem are Heart and Breast Cancer Data Set. The fore-mentioned datasets are fetched from the UCI (University of California at Irvine) Machine Learning Repository (Asuncion and Newman, 2007). The extracted results are compared with the existing BPN and LAST algorithms for both two- and multiclass classification problems.

The specification of the benchmark datasets utilized for training in the research is summarized in the Table 1.

Table 1: Specification of benchmark data sets

Datasets	No. of attributes	No. of classes	No. of instances
Iris	4	3	150
Waveform	21	3	5000
Heart	13	2	270
Breast cancer	31	2	569

Table 2: Selected training architectures and parameters

Datasets	Learning rate	MLP architecure	Momentum
Iris	1e - 4 1e – 3	4 × 5 × 1	0.8
Waveform	1e – 4 1e – 3	21×10×1	0.7
Heart	1e – 4 1e – 3	13×5×1	0.9
Breast cancer	1e – 4	31×15×1	0.9

Experimental design: A 3-layer feedforward neural network is adopted for the simulations of all the training algorithms with the selected training architecture and training parameters mentioned in the Table 2. The simulations of all the training algorithms are repeated for two different learning rates such as 1e-4 (0.0001) and 1e-3(0.001).

The simulations of all the above training algorithms are done using MATLAB R2010b on a machine with the configuration of Intel® Core I5-3210M processor, 4 GB of RAM and CPU speed of 2.50GHz.

According to the idea of Nguyen-Widrow algorithm (Nguyen and Widrow, 1990), the MFNN weight coefficients are initialized with the random values within the specified range -0.5 to +0.5. The Fivefold cross validation method is applied to train and test the above training algorithms. Each dataset is split into five disjoint subsets. Among these subsets, a single subset is retained for testing, and the remaining four subsets are used for training. The validation process is repeated five times with each of the five subset used exactly once for testing.

- Experimental Result
- Multiclass Problems
- Iris Data Set

The IRIS dataset is furnished with 150 iris flower samples collected equally from three different varieties of iris flowers. The varieties are listed as Iris Setosa, Iris Versicolour and Iris Virginica. These varieties are identified based on the four characteristics of iris flower such as width and length of Iris sepal, and width and length of Iris petal. Among these varieties, Iris Setosa is easier to be separated from the other two varieties, while the other two varieties, Iris Virgincia and Iris Versicolour, are partially obscured and harder to be distinguished.

The total number of IRIS input samples consumed by BPN, LAST and CAST training algorithms at every single epoch is graphically represented in the Fig. 3 and 4 with the learning rate of 1e-4 and 1e-3 respectively.

Figure 5 and 6 illustrates the epoch wise training time comparison between BPN, LAST and CAST training algorithm for the learning rates 1e-4 and 1e-3 respectively.

Waveform data set: The Waveform database generator data set consists of measurements of 5000 wave's samples. The 5000 wave's samples are equally scattered (about 33%) among the three classes of waves (Asuncion and Newman, 2007). These samples are collected from the generation of 2 of 3 "base" waves. It contains 21 attributes of numeric values which are involved in the categorization of each class of waves.

The total number of Waveform input samples consumed by BPN, LAST and CAST training algorithms at every single epoch is graphically represented in the Fig. 7 and 8 with the learning rate of 1e-4 and 1e-3 respectively.

Figure 9 and 10 illustrates the epoch wise training time comparison between BPN, LAST and CAST training algorithm for the learning rates 1e-4 and 1e-3 respectively.

Two-Class problem:

Heart data set: The Statlog Heart disease database consists of 270 patient's samples. The presence or absence of each patient's heart disease is predicted using 13 attributes. Among these 270 patient's samples, 150 samples are the samples of heart disease which is 'absent' and 120 samples of heart disease which is 'present'.

The total number of Heart input samples consumed by BPN, LAST and CAST training algorithms at every single epoch is graphically represented in the Fig. 11 and 12 with the learning rate of 1e-4 and 1e-3 respectively.

Figure 13 and 14 illustrates the epoch wise training time comparison between BPN, LAST and CAST training algorithm for the learning rates 1e-4 and 1e-3 respectively.

Breast cancer data set: The Wisconsin Breast Cancer Diagnosis Dataset contains 569 patient's breasts samples among which 357 diagnosed as benign and 212 diagnosed as malignant class. Each patient's characteristics are recorded using 32 numerical features.

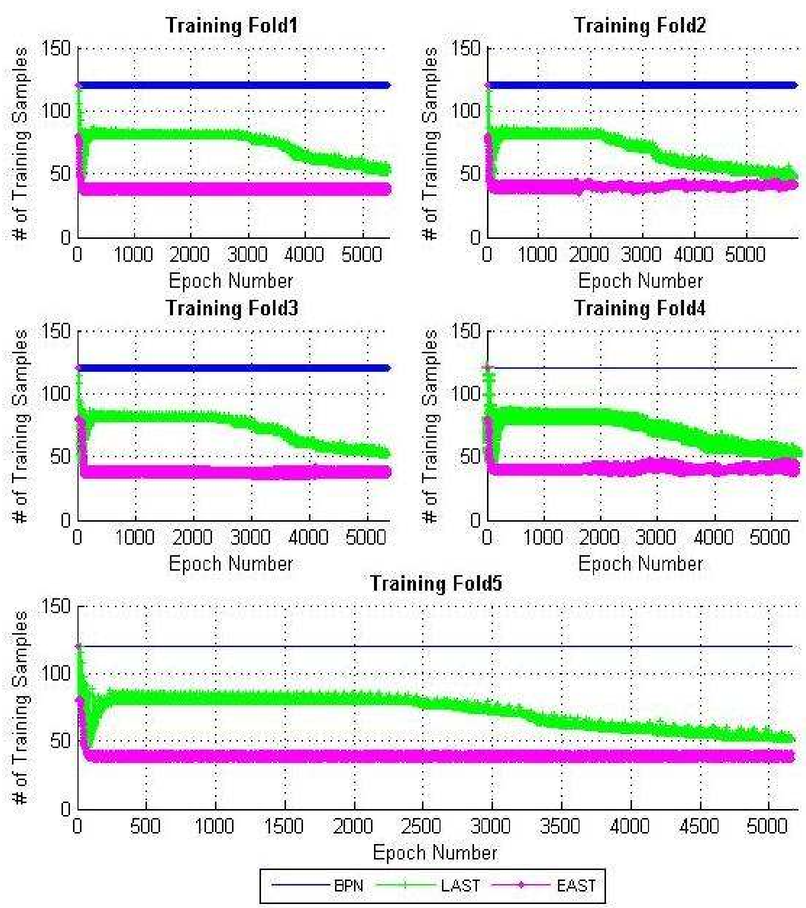

Fig. 3: IRIS epoch wise input samples with 1e-4 learning rate

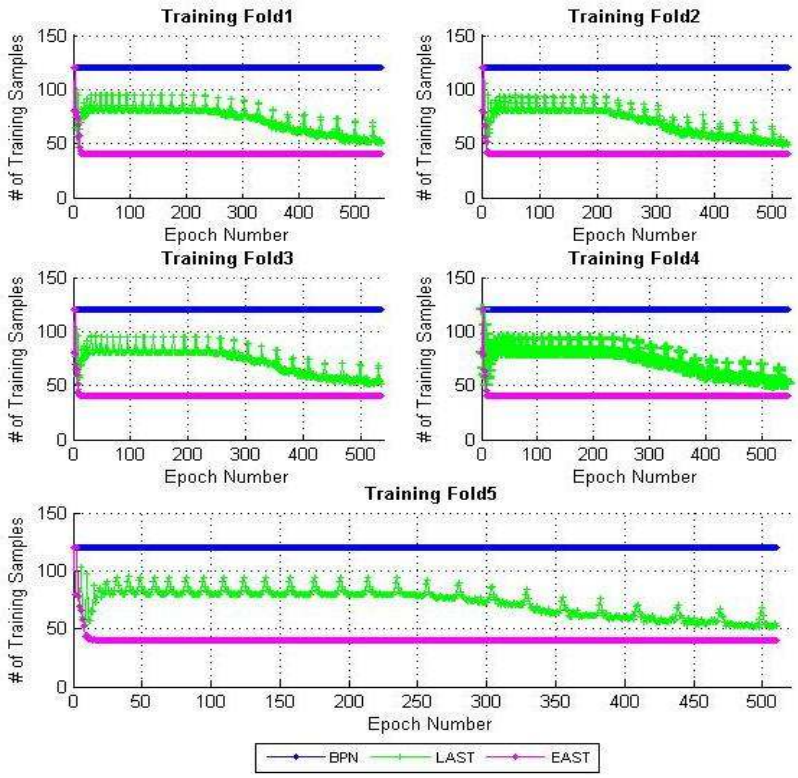

Fig. 4: IRIS epoch wise input samples with 1e-3 learning rate

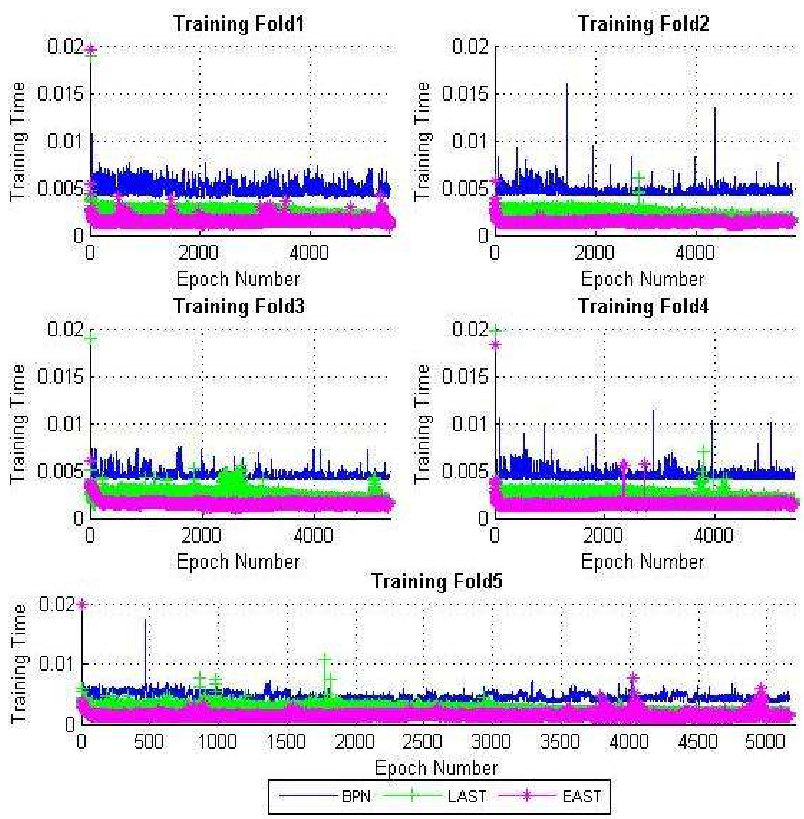

Fig. 5: IRIS epoch wise training time with 1e-4 learning rate

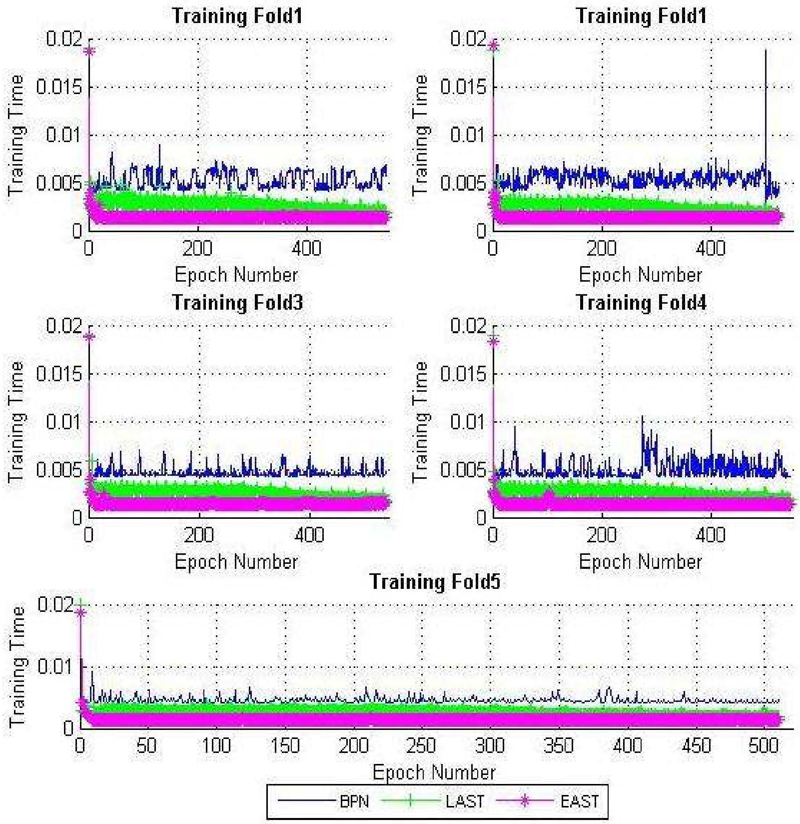

Fig. 6: IRIS epoch wise training time with 1e-3 learning rate

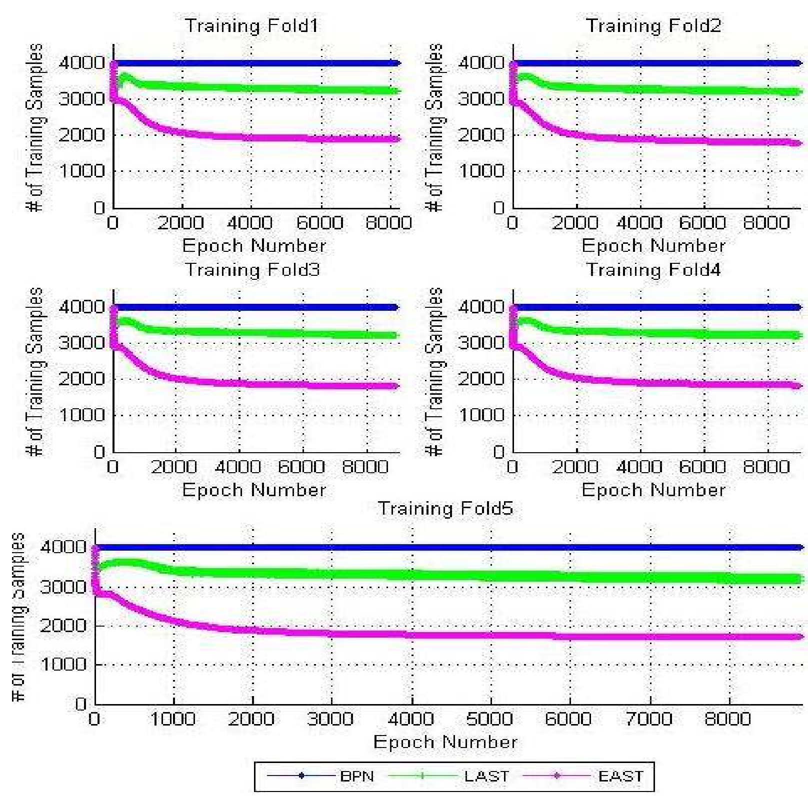

Fig. 7: Waveform epoch wise input samples with 1e-4 learning rate

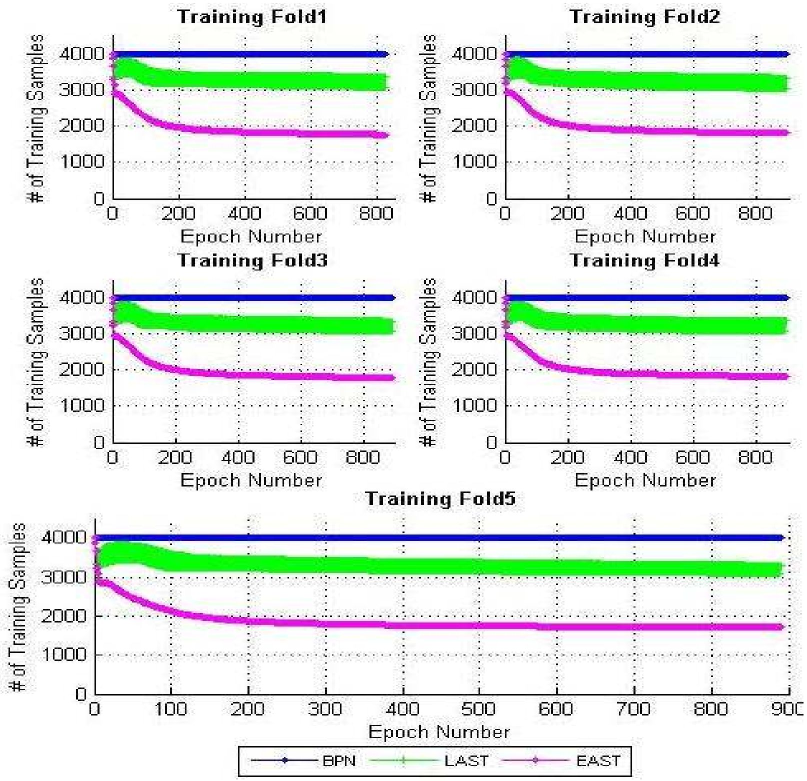

Fig. 8: Waveform epoch wise input samples with 1e-3 learning rate

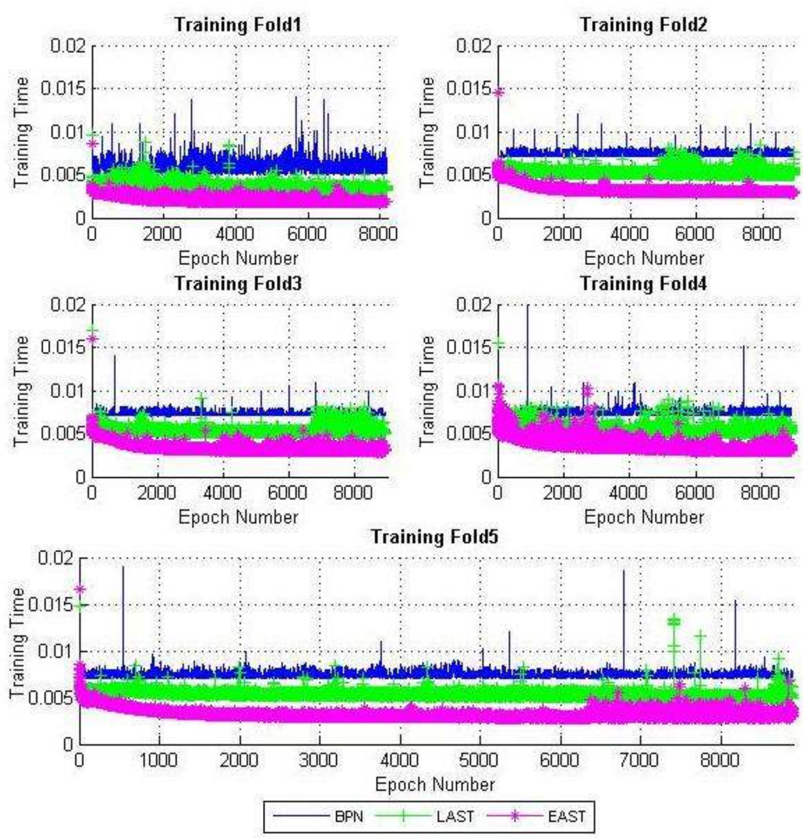

Fig. 9: Waveform epoch wise training time with 1e-4 learning rate

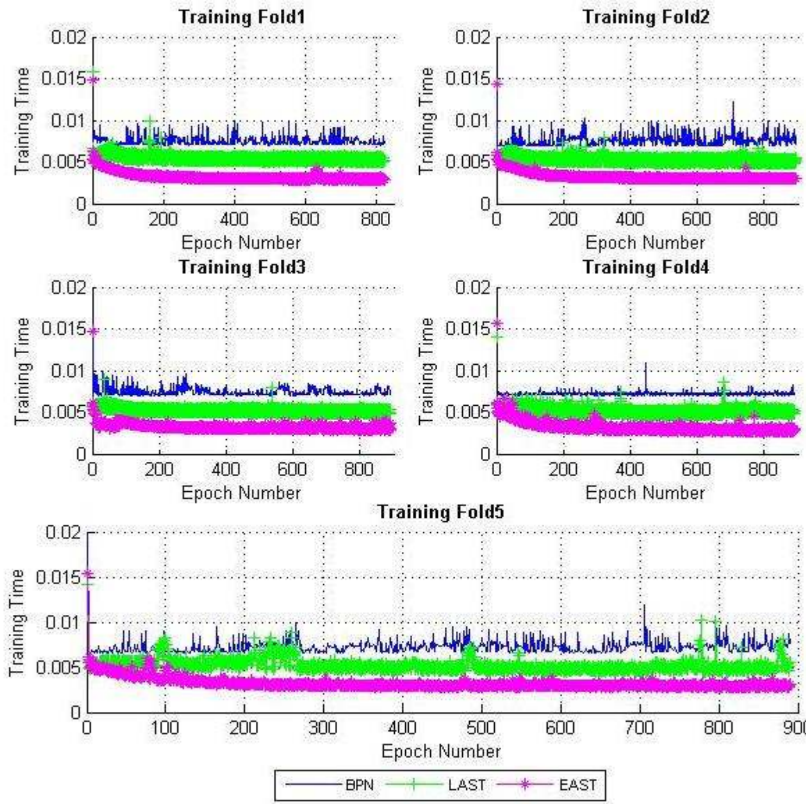

Fig. 10: Waveform epoch wise training time with 1e-3 learning rate

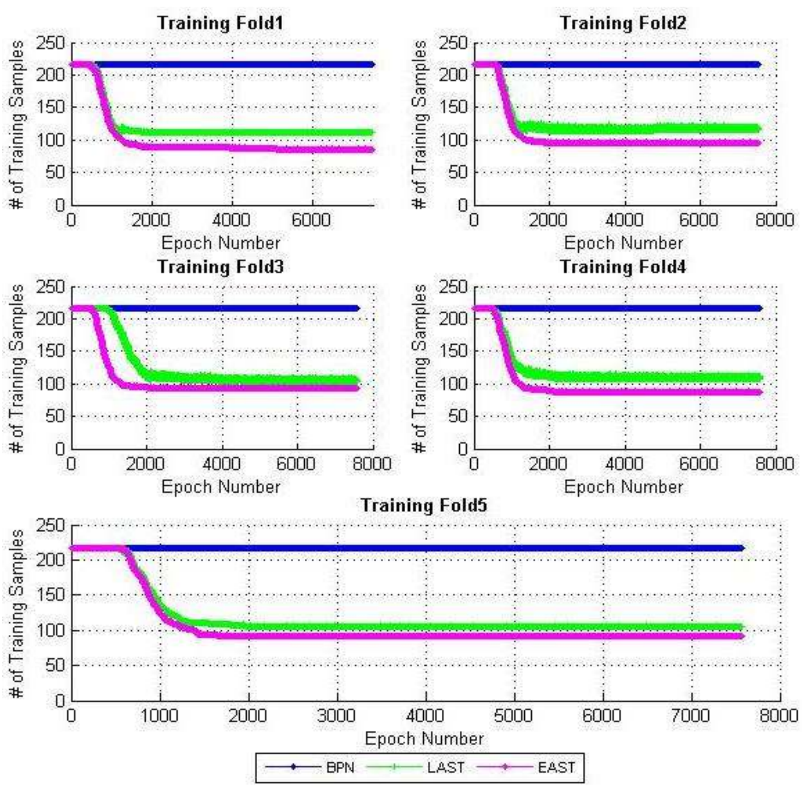

Fig. 11: Heart epoch wise input samples with 1e-4 learning rate

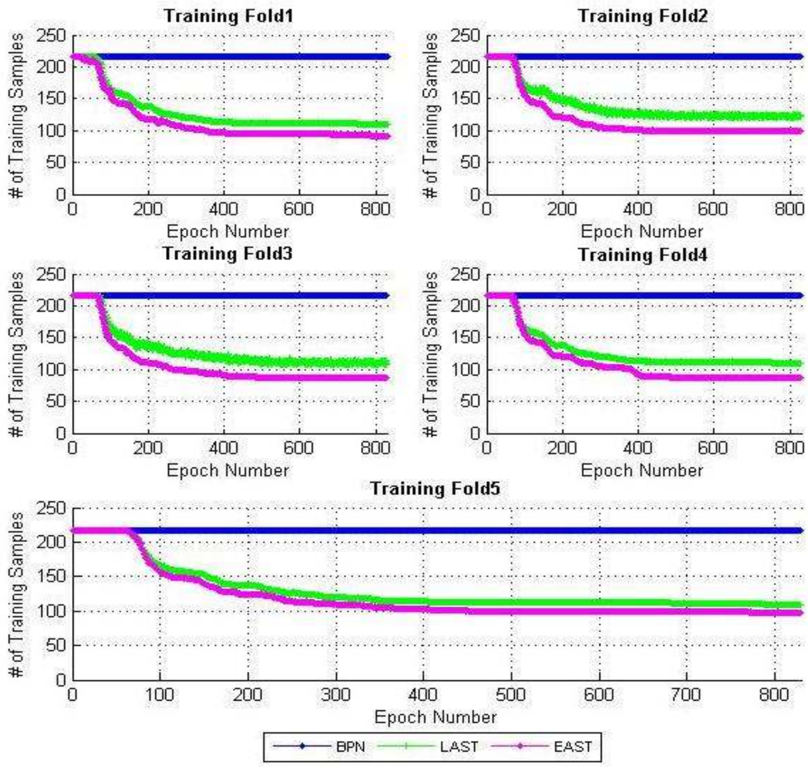

Fig. 12: Heart epoch wise input samples with 1e-3 learning rate

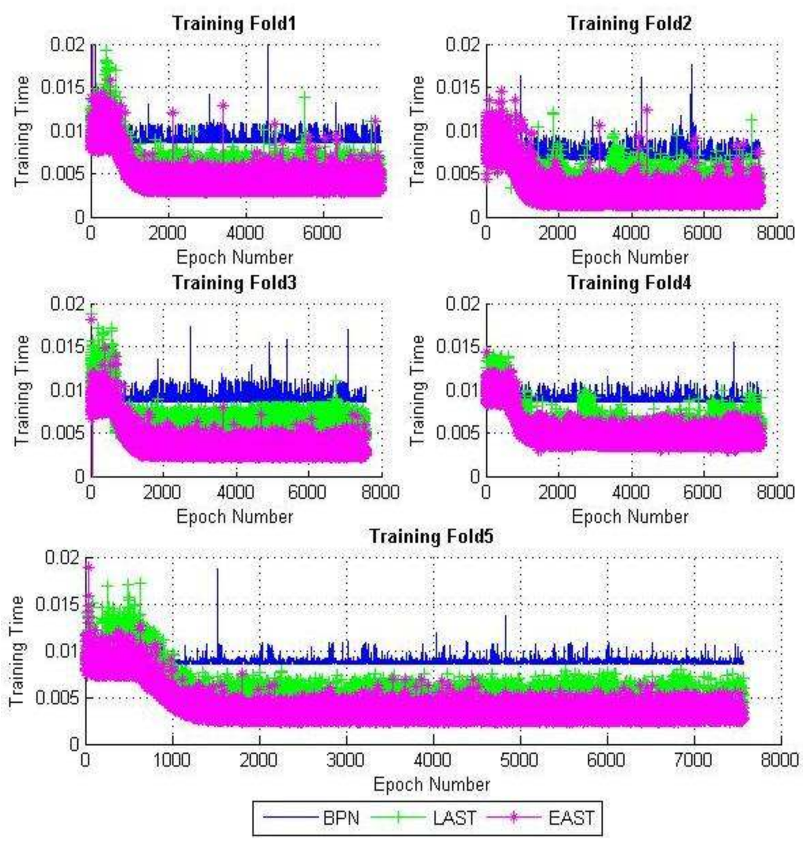

Fig. 13: Heart epoch wise training time with 1e-4 learning rate

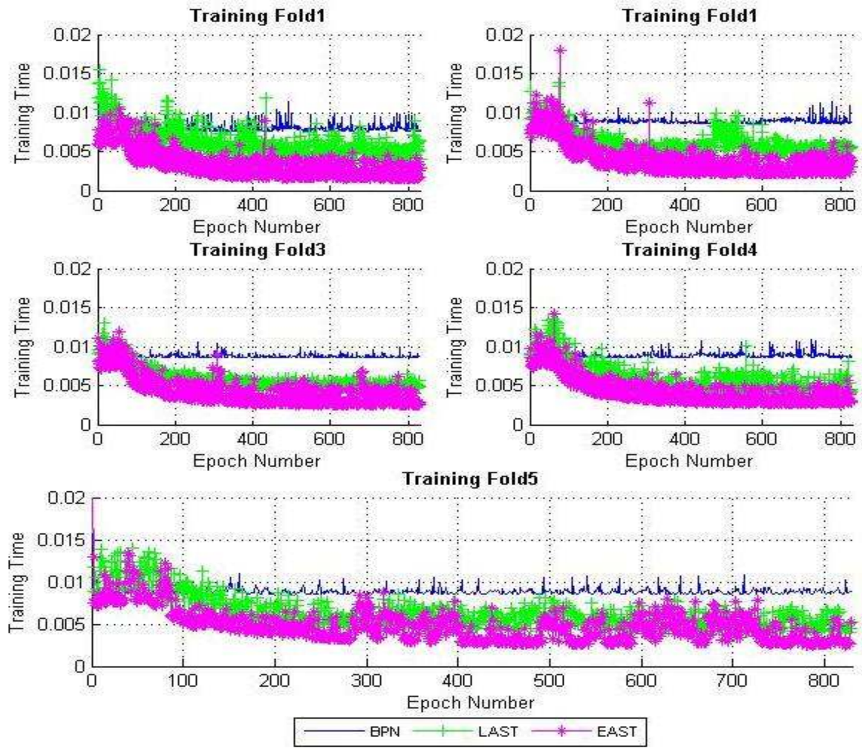

Fig. 14: Heart epoch wise training time with 1e-3 learning rate

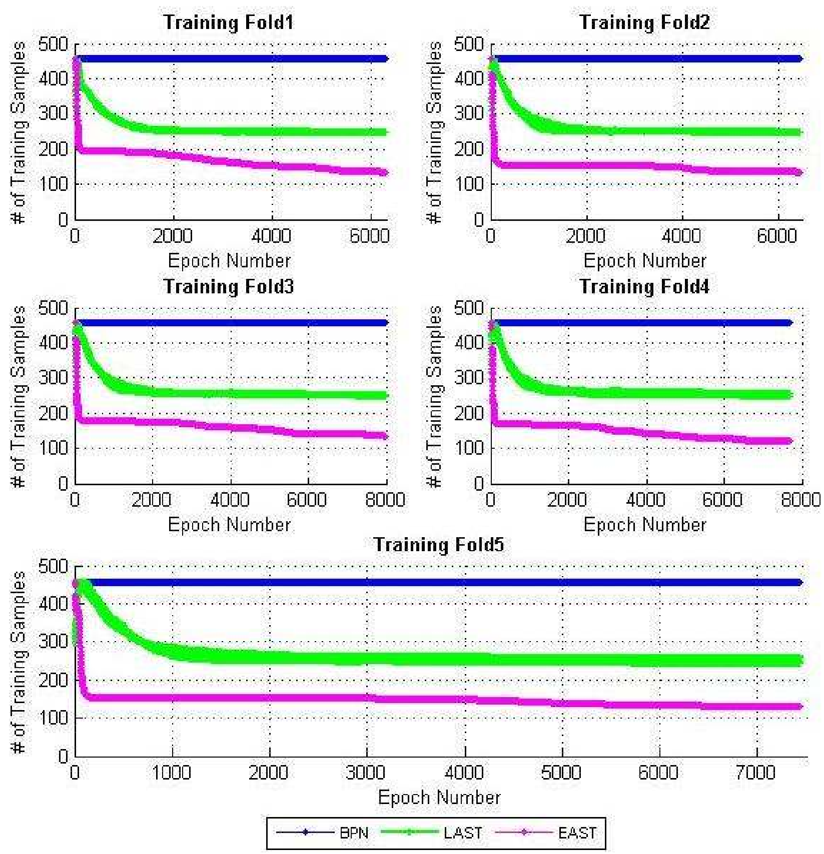

Fig. 15: Breast cancer epoch wise input samples with 1e-4 learning rate

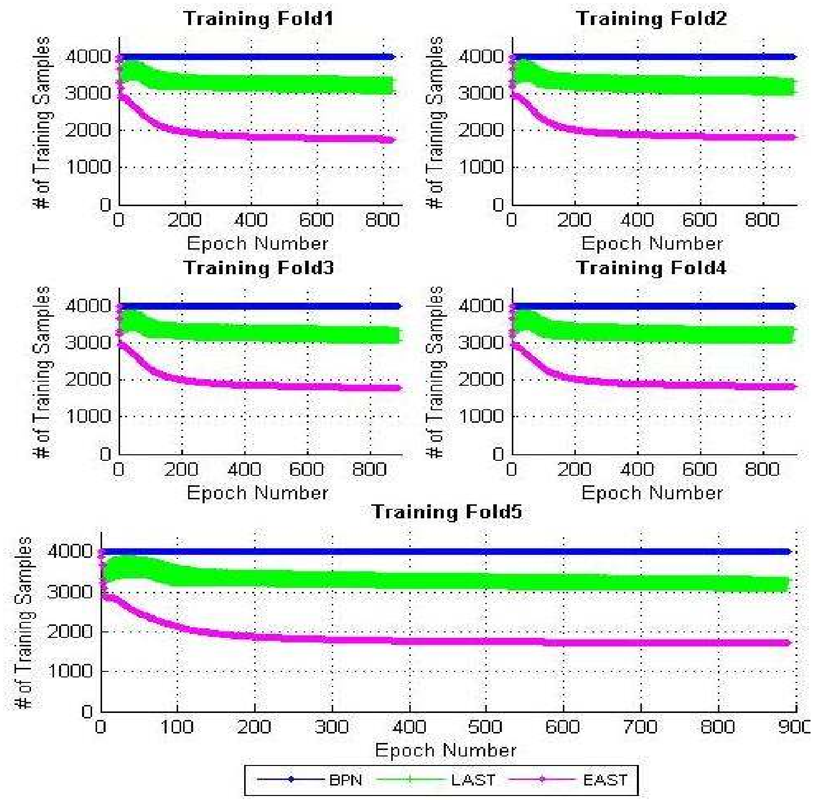

Fig. 16: Breast cancer epoch wise input samples with 1e-3 learning rate

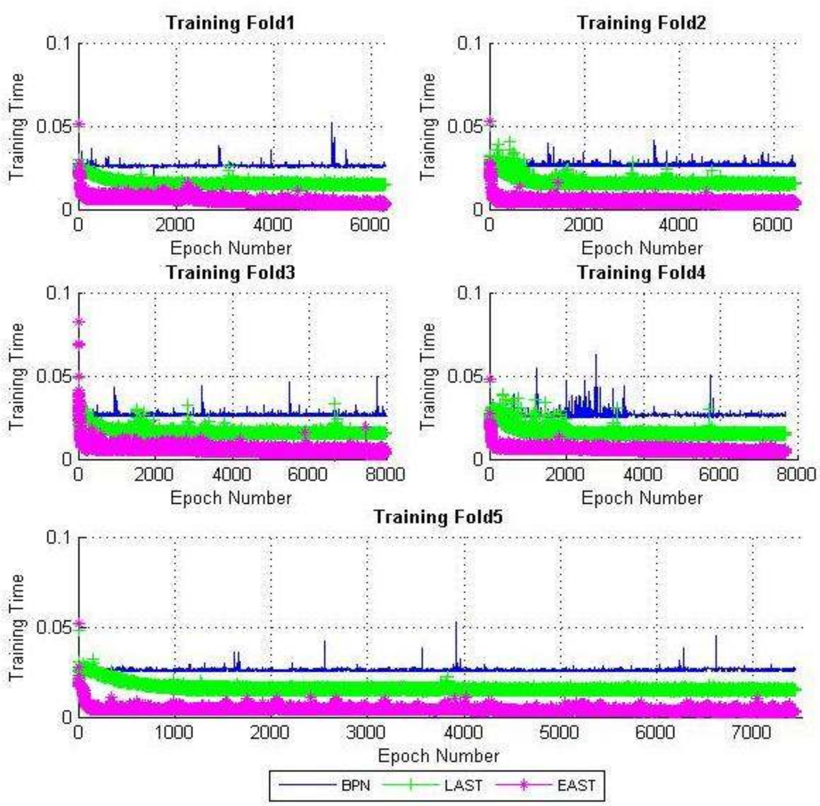

Fig. 17: Breast cancer epoch wise training time with 1e-4 learning rate

The total number of Heart input samples consumed by BPN, LAST and CAST training algorithms at every single epoch is graphically represented in the Fig. 15 and 16 with the learning rate of 1e-4 and 1e-3 respectively.

Figure 17 and 18 illustrates the epoch wise training time comparison between BPN, LAST and CAST training algorithm for the learning rates 1e-4 and 1e-3 respectively.

Result analysis and comparison: Table 3 to 10 shows the experimental results of BPN, LAST and CAST algorithm observed at each step across five repeats of fivefold cross validation using two different learning rates such as 1e-4 and 1e-3.

From these Table 3 to 10, the CAST algorithm yields improved computational training speed in terms of the total number of trained input samples as well as total training time over BPN and less than LAST. But, when the skipping factor goes higher, the accuracy of the system is affected highly.

Training samples comparison: The comparison results of the total number of input samples consumed for training by BPN, LAST and CAST with the

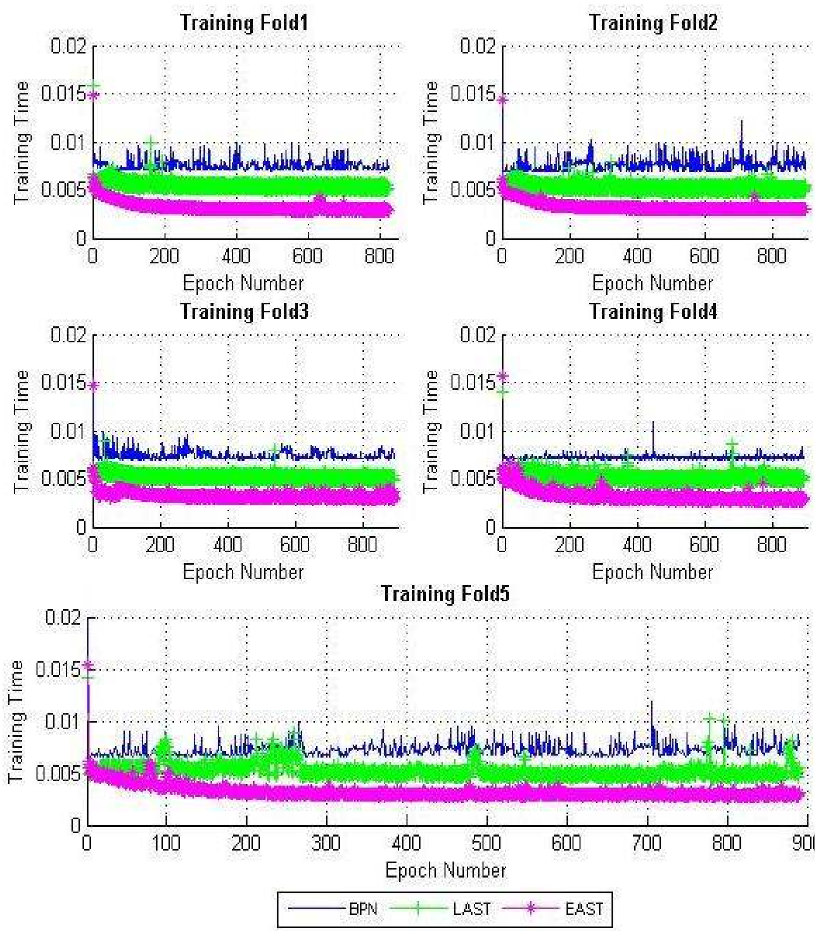

Fig. 18: Breast cancer epoch wise training time with 1e-3 learning rate

Table 3: Comparison results trained by the Iris dataset with 1e-4 learning rate

		BPN			LAST			CAST		
Testing fold	Number of epochs	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)
1	5442	653040	26.7909	83.33	395718	13.1303	80	208755	8.2995	73.33
2	5902	708240	27.2332	83.33	396670	13.5337	83.33	240293	8.5218	76.67
3	5332	639840	23.6228	80	379759	12.9799	83.33	206029	8.2960	80
4	5439	652680	24.1885	83.33	383028	13.2143	80	223245	8.2565	80
5	5161	619320	23.2492	83.33	365940	12.7051	76.67	203116	7.8261	76.67
Average:		654624	25.0169	82.664	384223	13.1127	80.666	216288	8.23998	77.334

Table 4: Comparison results trained by the IRIS dataset with 1e-3 learning rate

		BPN			LAST			CAST		
Testing fold	Number of epochs	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)
1	547	65640	2.8833	83.33	39896	1.4390	83.33	22339	0.7867	76.67
2	526	63120	2.4651	80	37281	1.2867	80	21369	0.7537	80
3	535	64200	2.4906	80	39165	1.3472	80	21735	0.7667	76.67
4	545	65400	2.7546	83.33	39697	1.3740	83.33	22120	0.7756	80
5	510	61200	2.3283	83.33	37425	1.2840	83.33	20735	0.7306	76.67
Average:		63912	2.58438	81.998	38693	1.34618	81.998	21660	0.76266	78.002

Table 5: Comparison results trained by the waveform dataset with 1e-4 learning rate

		BPN			LAST			CAST		
Testing fold	Number of epochs	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)
1	8187	32748000	47.6683	84.9	27229320	28.9716	85.1	16974989	17.2826	79.8
2	8973	35892000	66.7460	83.7	29669915	52.8073	84.6	17897431	30.3537	80.2
3	8929	35716000	65.7213	84.6	29656457	47.9644	84.5	17812293	30.2254	81.1
4	8903	35612000	64.8988	83.2	29571880	47.3533	83.1	17806977	29.0942	80.9
5	8887	35548000	64.3973	82.1	29476116	47.3203	82.5	17144339	28.6921	79.9
Average:		35103200	61.8863	83.7	29082110	44.8834	83.96	17527206	27.12961	80.38

Table 6: Comparison results trained by the waveform dataset with 1e-3 learning rate

		BPN			LAST			CAST		
Testing fold	Number of epochs	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)
1	823	3292000	6.1784	84.4	2729243	4.5310	85.6	1611594	2.6747	81.1
2	894	3576000	6.7595	83.8	2944663	4.7575	84.5	1785336	2.9381	80.6
3	891	3564000	6.6254	82.9	2944567	4.6765	83.9	1761213	2.8975	79.9
4	890	3560000	6.4547	83.5	2938903	4.6199	83.6	1784880	2.8904	80.5
5	890	3560000	6.4537	84.1	2937498	4.6656	84.6	1659327	2.8696	80.1
Average:		3510400	6.49434	83.74	2898975	4.6501	84.44	1720470	2.85406	80.44

Table 7: Comparison results trained by the heart dataset with 1e-4 learning rate

		BPN			LAST			CAST		
Testing fold	Number of epochs	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)
1	7485	1616760	58.0715	81.48	81.48	43.3506	83.33	713559	23.2651	75.93
2	7529	1626264	60.2075	83.33	83.33	46.7666	81.48	809372	25.3458	74.07
3	7569	1634904	67.8729	83.33	83.33	48.6806	83.33	820114	27.8431	75.93
4	7567	1634472	66.8935	81.48	81.48	47.8751	79.63	813699	26.6308	79.63
5	7567	1634472	66.5249	81.48	81.48	47.3221	81.48	811180	25.9578	77.78
Average:		1629374	63.91406	82.22	959597	46.799	81.85	793585	25.8085	76.67

Table 8: Comparison results trained by the heart dataset with 1e-3 learning rate

		BPN			LAST			CAST		
Testing fold	Number of epochs	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)
1	830	179280	7.3662	81.48	107845	4.9837	83.33	95137	3.3133	74.07
2	828	178848	7.361153	83.33	116169	5.238218	81.48	98116	3.382314	75.93
3	829	179064	7.265956	83.33	108534	4.492601	83.33	90205	3.533761	75.93
4	829	179064	7.326156	79.63	107736	4.772563	81.48	93136	3.554815	74.07
5	829	179064	7.341574	81.48	107736	5.274545	81.48	99092	3.993784	77.78
Average:		179064	7.332208	81.85	109604	4.95233	82.22	95137	3.555595	75.56

Table 9: Comparison results trained by the breast cancer dataset with 1e-4 learning rate

		BPN			LAST			CAST		
Testing fold	Number of epochs	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)
1	6279	2856945	162.5596	87.72	1659497	100.1092	87.72	1055844	34.0808	83.33
2	6460	2939300	172.0937	86.64	1718322	105.6381	86.64	966328	30.7942	79.82
3	7976	3629080	210.8542	88.6	2140909	131.4230	85.72	1286262	46.8745	84.21
4	7691	3499405	203.5600	86.84	2074540	125.0857	85.97	1138979	43.9744	80.07
5	7439	3392184	193.7257	87.61	1996086	119.5164	87.61	1097278	31.3622	84.07
Average:		3263383	188.5586	87.482	1917870.8	116.354	87.13	1108938	37.417214	82.3

Table 10: Comparison results trained by the breast cancer dataset with 1e-3 learning rate

		BPN			LAST			CAST		
Testing fold	Number of epochs	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)	Total number of input samples	Training time (in Sec)	Accuracy (%)
1	609	277095	16.5255	87.72	161260	10.3436	85.97	101916	5.4285	83.33
2	647	294385	17.2322	86.64	172059	10.5972	86.64	107089	5.8950	84.21
3	785	357175	21.3841	88.6	210885	13.4171	87.72	132372	6.4982	84.21
4	750	341250	19.7409	86.84	202580	12.1622	85.97	128676	5.8950	83.33
5	743	338808	19.7142	87.61	199366	11.9810	87.61	120608	5.7421	84.07
Average:		321742.6	18.91938	87.482	189230	11.7002	86.782	118132	5.8918	83.83

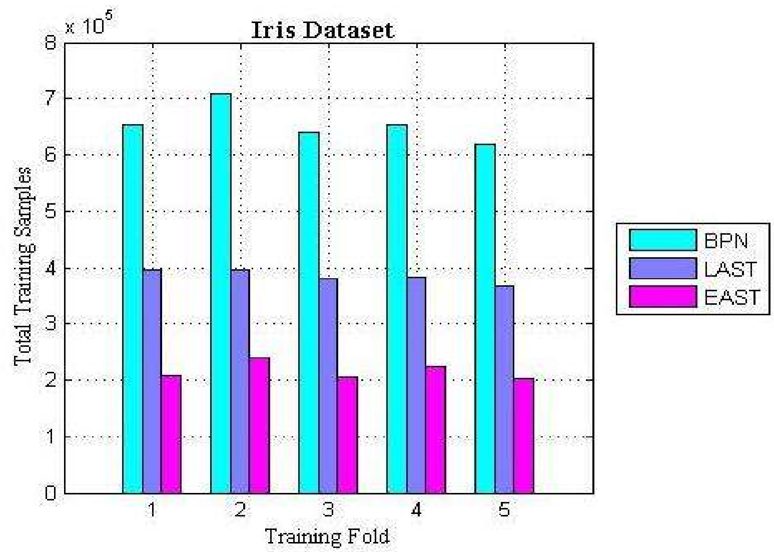

Fig. 19: Comparison result of IRIS input samples with 1e-4 learning rate

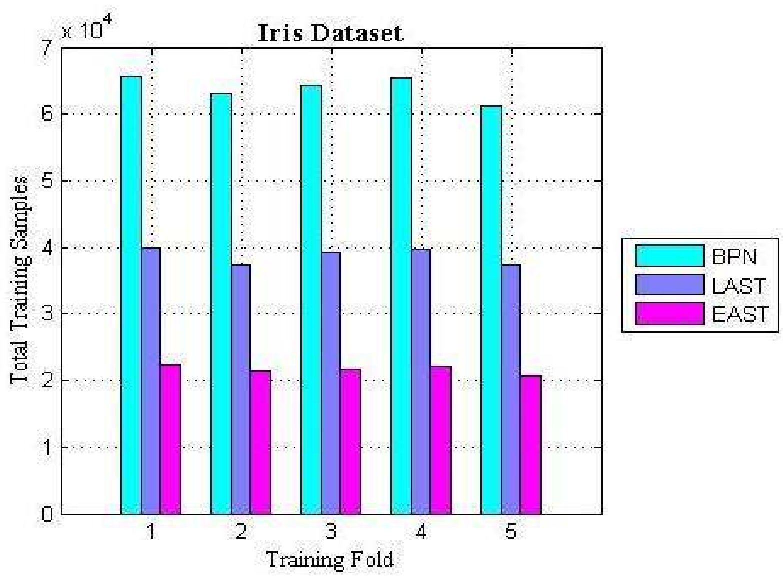

Fig. 20: Comparison result of IRIS input samples with 1e-3 learning rate

learning rate of 1e-4 and 1e-3 are shown in Fig. 19 to 26.

From the Fig. 19, it is portrayed that the total number of IRIS data samples consumed by CAST algorithm for training under the learning rate of 1e-4 is reduced by an average of nearly 67% and 44% of BPN and LAST algorithm respectively.

From the Fig. 20, it is portrayed that the total number of IRIS data samples consumed by CAST algorithm for training under the learning rate of 1e-3 is reduced by an average of nearly 66% and 44% of BPN and LAST algorithm respectively.

From the Fig. 21, it is portrayed that the total number of Waveform data samples consumed by CAST algorithm for training under the learning rate of 1e-4 is reduced by an average of nearly 50% and 40% of BPN and LAST algorithm respectively.

From the Fig. 22, it is portrayed that the total number of Waveform data samples consumed by CAST algorithm for training under the learning rate of 1e-3 is reduced by an average of nearly 51% and 41% of BPN and LAST algorithm respectively.

From the Fig. 23, it is portrayed that the total number of Heart data samples consumed by CAST algorithm for training under the learning rate of 1e-4 is reduced by an average of nearly 51% and 17% of BPN and LAST algorithm respectively.

From the Fig. 24, it is portrayed that the total number of Heart data samples consumed by CAST algorithm for training under the learning rate of 1e-3 is

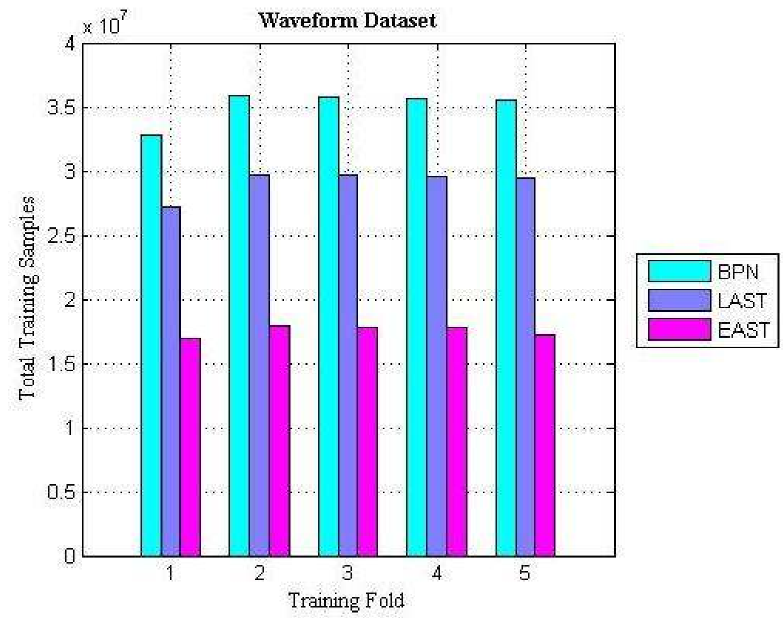

Fig. 21: Comparison result of waveform input samples with 1e-4 learning rate

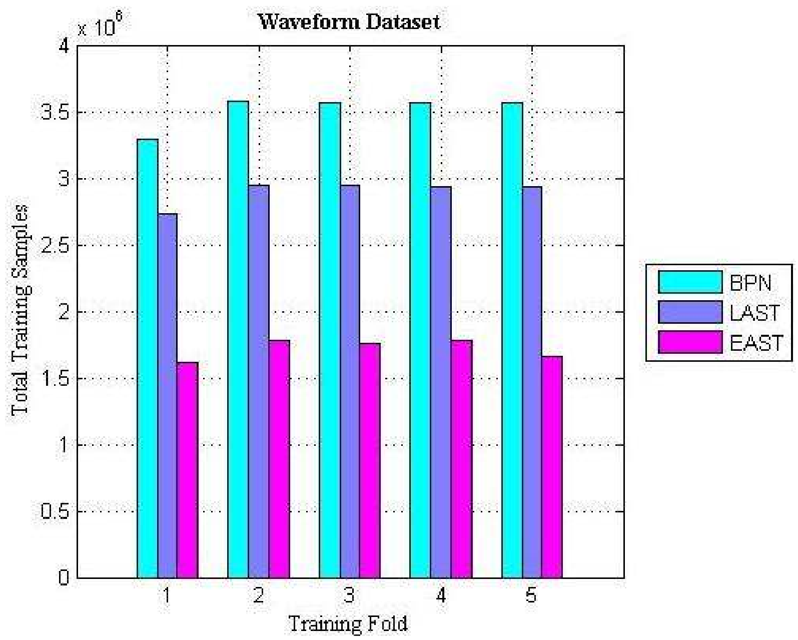

Fig. 22: Comparison result of waveform input samples with 1e-3 learning rate

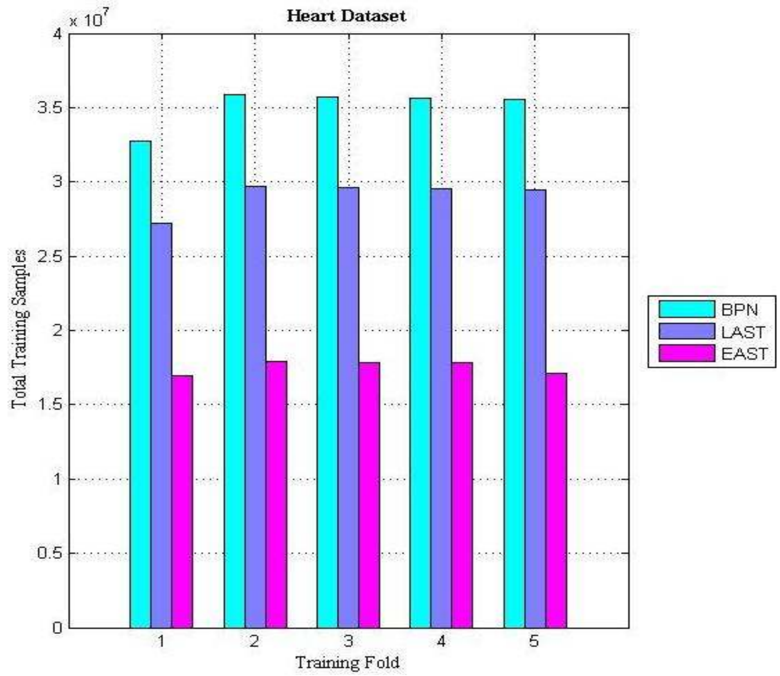

Fig. 23: Comparison result of heart input samples with 1e-4 learning rate

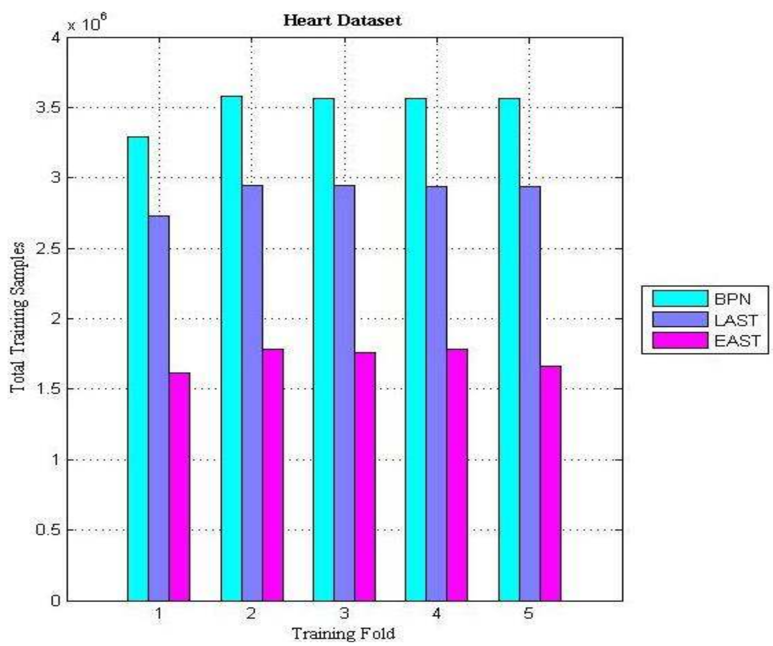

Fig. 24: Comparison result of heart input samples with 1e-3 learning rate

reduced by an average of nearly 47% and 13% of BPN and LAST algorithm respectively.

From the Fig. 25, it is portrayed that the total number of Breast Cancer data samples consumed by CAST algorithm for training under the learning rate of 1e-3 is reduced by an average of nearly 66% and 42% of BPN and LAST algorithm respectively.

From the Fig. 26, it is portrayed that the total number of Breast Cancer data samples consumed by CAST algorithm for training under the learning rate of 1e-3 is reduced by an average of nearly 63% and 38% of BPN and LAST algorithm respectively.

Training time comparison: Thus decreasing the size of the trained input samples can reduce the training time which is shown in this section, thereby increasing the speed of the training process. Figure 27 to 34 illustrates the training time comparison between BPN,

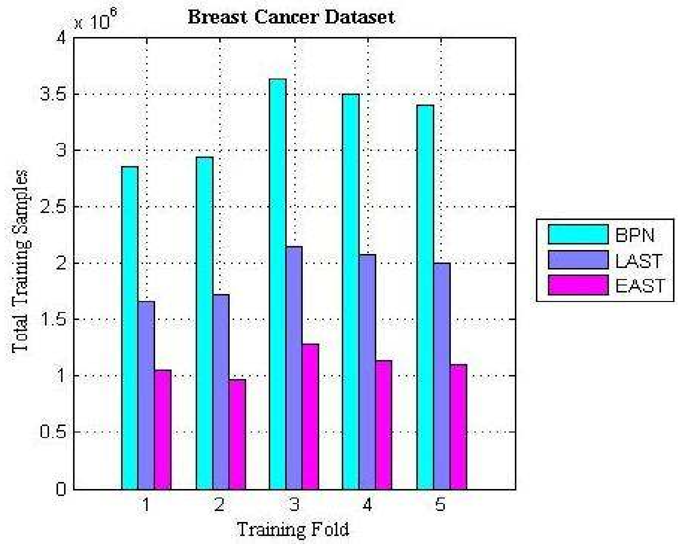

Fig. 25: Comparison result of breast cancer input samples with 1e-4 learning rate

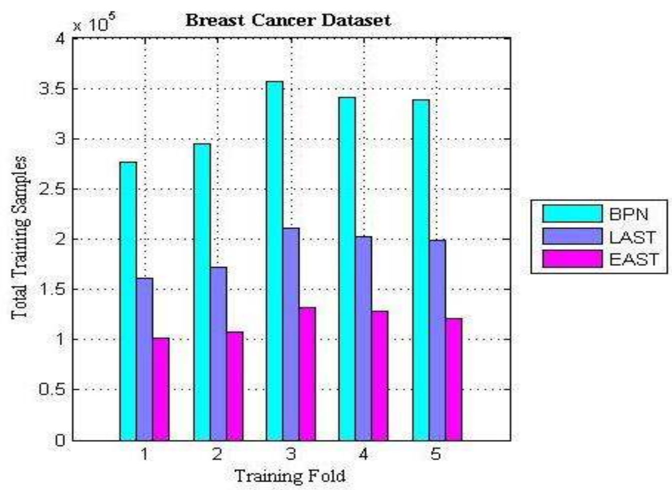

Fig. 26: Comparison result of breast cancer input samples with 1e-3 learning rate

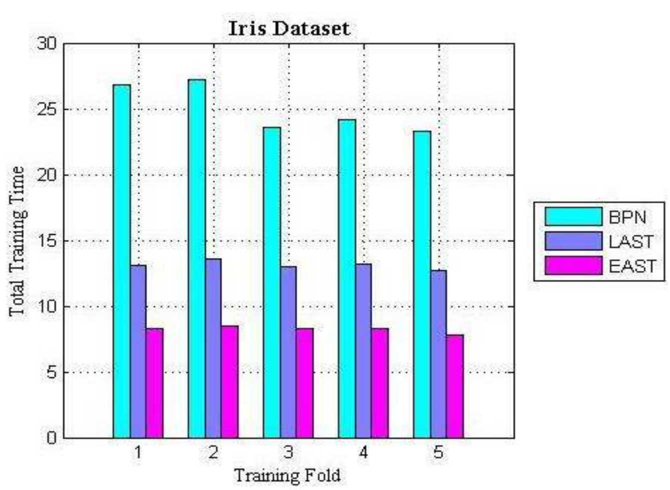

Fig. 27: Comparison result of IRIS training time with 1e-4 learning rate

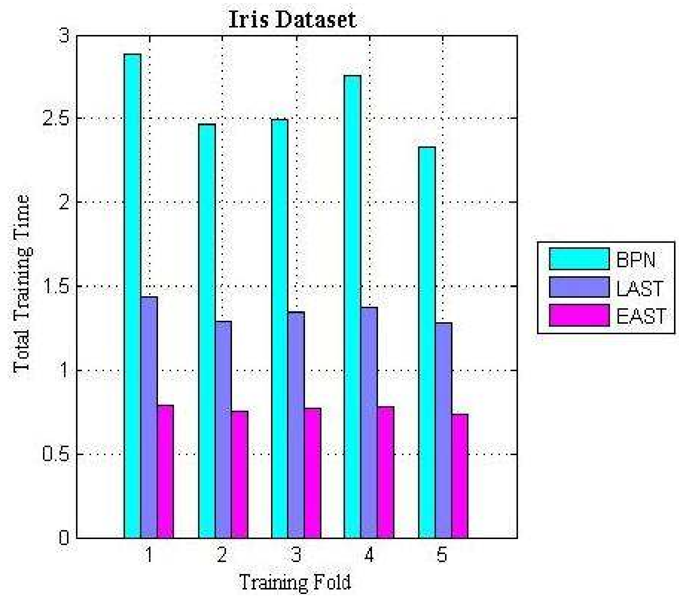

Fig. 28: Comparison result of IRIS training time with 1e-3 learning rate

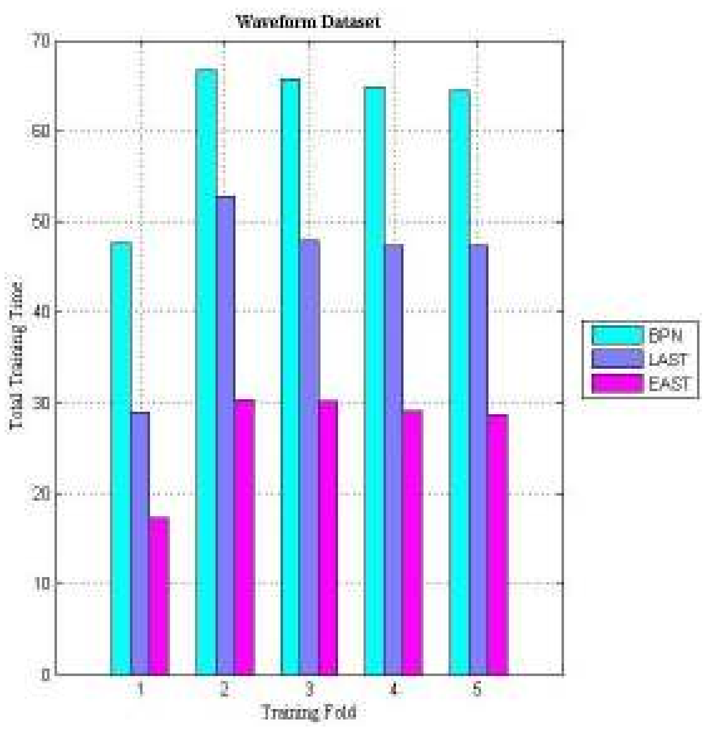

Fig. 29: Comparison result of waveform training time with 1e-4 learning rate

LAST and CAST training methods for different learning rate of 1e-4 and 1e-3.

From the Fig. 27, the total training time for training IRIS dataset by CAST algorithm is reduced to an average of 67% of BPN algorithm and 37% of LAST algorithm for the learning rate of 1e-4.

From the Fig. 28, the total training time for training IRIS dataset by CAST algorithm is reduced to an average of 70% of BPN algorithm and 43% of LAST algorithm for the learning rate of 1e-3.

From the Fig. 29, the total training time for training waveform dataset by CAST algorithm is reduced to an average of 56% of BPN algorithm and 40% of LAST algorithm for the learning rate of 1e-4.

From the Fig. 30, the total training time for training waveform dataset by CAST algorithm is reduced to an

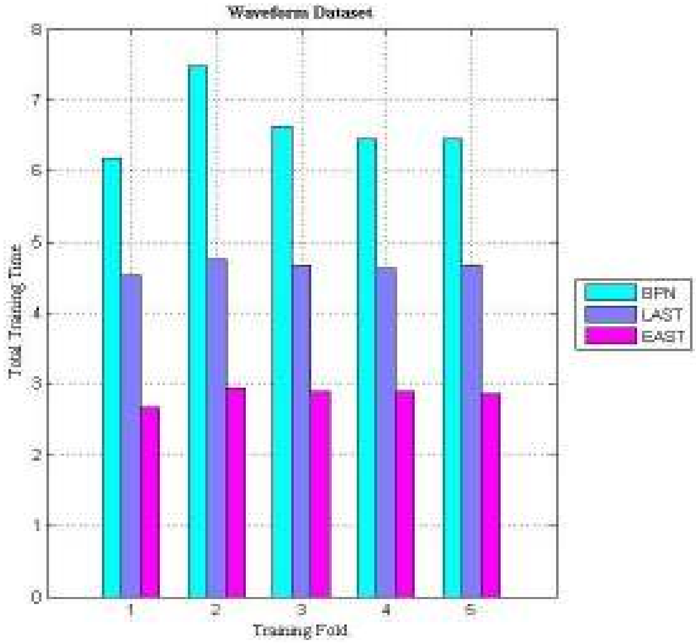

Fig. 30: Comparison result of waveform training time with 1e-3 learning rate

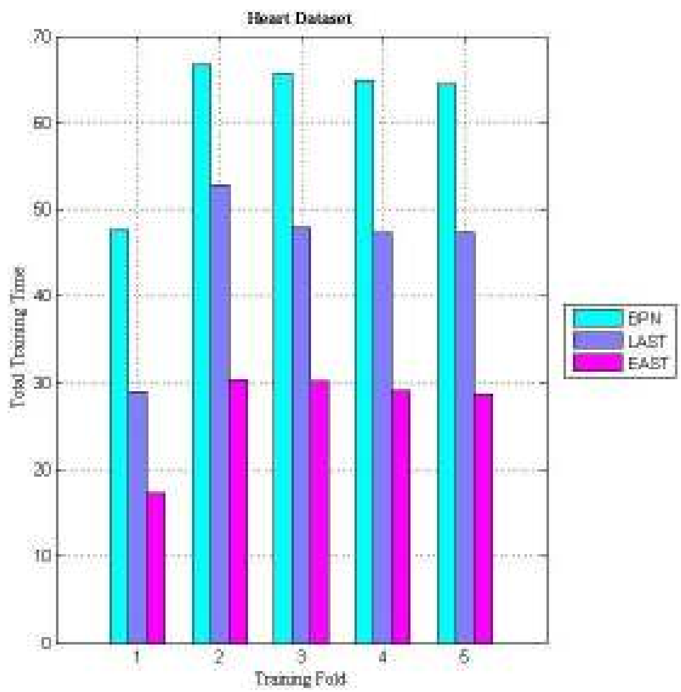

Fig. 31: Comparison result of heart training time with 1e-4 learning rate

average of 56% of BPN algorithm and 39% of LAST algorithm for the learning rate of 1e-3.

From the Fig. 31, the total training time for training Heart dataset by CAST algorithm is reduced to an average of 60% of BPN algorithm and 45% of LAST algorithm for the learning rate of 1e-4.

From the Fig. 32, the total training time for training Heart dataset by CAST algorithm is reduced to an average of 52% of BPN algorithm and 28% of LAST algorithm for the learning rate of 1e-3.

From the Fig. 33, the total training time for training Breast Cancer by CAST algorithm is reduced to an average of 80% of BPN algorithm and 68% of LAST algorithm for the learning rate of 1e-4.

From the Fig. 34, the total training time for training Breast Cancer dataset by CAST algorithm is reduced to

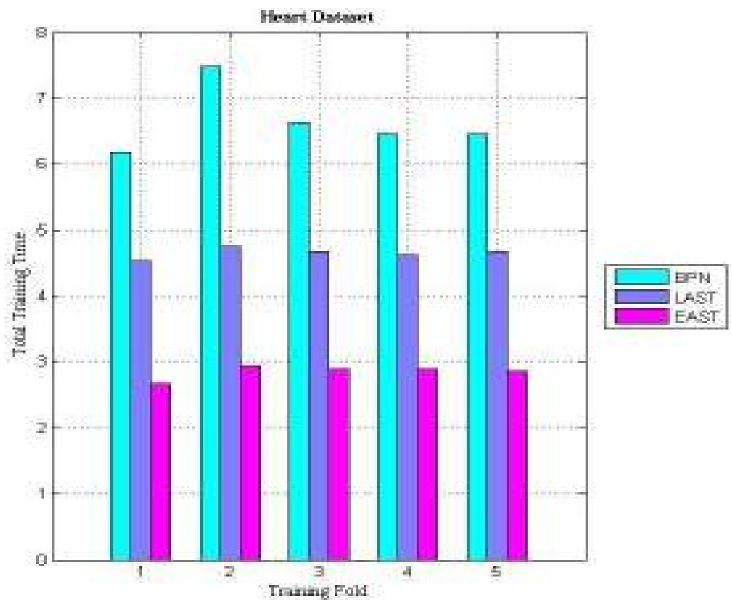

Fig. 32: Comparison result of heart training time with 1e-3 learning rate

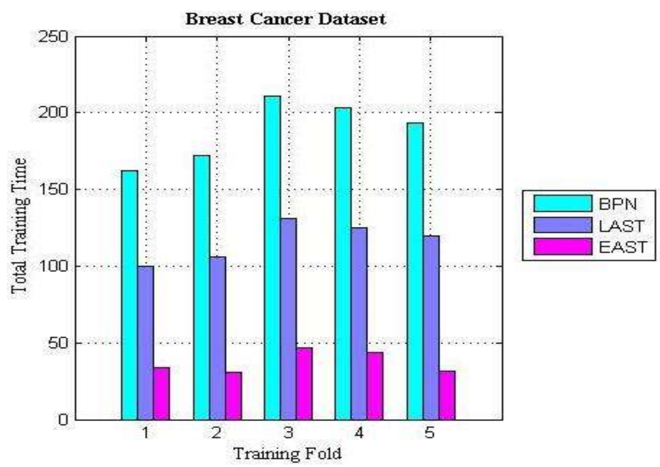

Fig. 33: Comparison result of breast cancer training time with 1e-4 learning rate

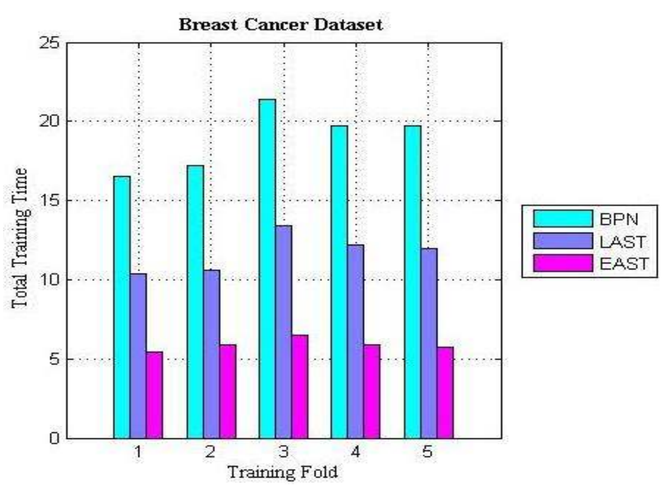

Fig. 34: Comparison result of breast cancer training time with 1e-3 learning rate

an average of 69% of BPN algorithm and 50% of LAST algorithm for learning rate of 1e-3.

Although the training performance of CAST achieves faster, it still lacks in the accuracy rate due to high skipping factor. So, further work should be concentrated on how to improve the accuracy rate of the training algorithm also.

CONCLUSION

In this brief, a simple and fast training algorithm called Constant Adaptive Skipping Training (CAST) Algorithm is presented. The simulation results showed that, compared to other training methods, the new algorithm improves the training speed by significantly reducing the total number of training input samples consumed by MFNN for training at every single epoch. Hence, the overall training time for actual training of the MFNN is often reduced by an average of 50% than in the standard training algorithm. It is concluded that the proposed CAST algorithm are much faster than the standard BPN and LAST algorithm and also the proposed CAST Algorithm can be merged in addition with any algorithm used for training any real-world supervised task classification.

REFERENCES

Ampazis, N. and S.J. Perantonis, 2002. Two highly efficient second-order algorithms for training feedforward networks. IEEE T. Neural Networ., 13(5): 1064-1074.

Asuncion, A. and D.J. Newman, 2007. UCI Machine Learning Repository. School of Information and Computer Science, University of California, Irvine, CA. Retrieved form: http://www.ics.uci.edu/~mlearn/.

Behera, L., S. Kumar and A. Patnaik, 2006. On adaptive learning rate that guarantees convergence in feedforward networks. IEEE T. Neural Networ., 17(5): 1116-1125.

Devi, R.M., S. Kuppuswami and R.C. Suganthe, 2013. Fast linear adaptive skipping training algorithm for training artificial neural network. Math. Probl. Eng., 2013(2013): 9.

Hornik, K., M. Stinchcombe and H. White, 1989. Multilayer feedforward networks are universal approximators. Neural Networks, 2(5): 359-366.

Huang, G.B., Y.Q. Chen and H.A. Babri, 2000. Classification ability of single hidden layer feedforward neural networks. IEEE T. Neural Networ., 11(3): 799-801.

Mehra, P. and B.W. Wah, 1992. Artificial Neural Networks: Concepts and Theory. 1st Edn., IEEE Computer Society Press, Los Alamitos, Calif, pp: 667.

Nguyen, D. and B. Widrow, 1990. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. Proceeding of the IJCNN International Joint Conference on Neural Networks. San Diego, CA, USA, 3: 21-26.

Plagianakos, V.P., D.G. Sotiropoulos and M.N. Vrahatis, 1998. A nonmonotone backpropagation training method for neural networks. Department of Mathematics, University of Patras, Technical Report No. TR98-04.

Razavi, S. and B.A. Tolson, 2011. A new formulation for feedforward neural networks. IEEE T. Neural Networ., 22(10): 1588-1598.

Shao, H. and G. Zheng, 2009. A new BP algorithm with adaptive momentum for FNNs training. Proceeding of the WRI Global Congress on Intelligent Systems. Xiamen, China, 4: 16-20.

Varnava, T.M. and A.J. Meade Jr., 2011. An initialization method for feedforward artificial neural networks using polynomial bases. Adv. Adaptive Data Anal., 3: 385-400.

Wilamowski, B.M. and H. Yu, 2010. Improved computation for levenberg–Marquardt training. IEEE T. Neural Networ., 21: 930-937.

Yu, H. and B.M. Wilamowski, 2012. Neural Network Training with Second Order Algorithms. In: Hippe, Z.S. *et al*. (Eds.), Human-Computer Systems Interaction: Backgrounds and Applications 2. Advances in Intelligent and Soft Computing, Springer-Verlag, Berlin, Heidelberg, 99: 463-476.