Received September 9, 2019, accepted September 25, 2019, date of publication October 17, 2019, date of current version November 5, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2948059

Fault and Noise Tolerance in the Incremental Extreme Learning Machine

HO CHUN LEUNG, CHI SING LEUNG, (Senior Member, IEEE), AND ERIC WING MING WONG, (Senior Member, IEEE)
Department of Electronic Engineering, City University of Hong Kong, Hong Kong

Corresponding author: Chi Sing Leung (eeleungc@cityu.edu.hk)

This work was supported by the City University of Hong Kong under Grant 7005063 and Grant 9610431.

ABSTRACT The extreme learning machine (ELM) is an efficient way to build single-hidden-layer feedforward networks (SLFNs). However, its fault tolerant ability is very weak. When node noise or node failure exist in a network trained by the ELM concept, the performance of the network is greatly degraded if a countermeasure is not taken. However, this kind of countermeasure for the ELM or incremental learning is seldom reported. This paper considers the situation that a trained SLFN suffers from the coexistence of node fault and node noise. We develop two fault tolerant **incremental** ELM algorithms for the regression problem, namely node fault tolerant incremental ELM (NFTI-ELM) and node fault tolerant convex incremental ELM (NFTCI-ELM). The NFTI-ELM determines the output weight of the newly inserted node only. We prove that in terms of the training set mean squared error (MSE) of faulty SLFNs, the NFTI-ELM converges. Our numerical results show that the NFTI-ELM is superior to the conventional ELM and incremental ELM algorithms under faulty situations. To further improve the performance, we propose the NFTCI-ELM algorithm. It not only determines the output weight of the newly inserted node, but also updates all previously trained output weights. In terms of training set MSE of faulty SLFNs, the NFTCI-ELM converges, and it is superior to the NFTI-ELM.

INDEX TERMS Single hidden layer network, incremental learning, extreme learning machine, multiplicative noise, open fault.

I. INTRODUCTION

A single-hidden-layer feedforward network (SLFN) [1], [2] is able to act as a universal approximator. In the traditional training approach [3], [4], we need to determine all the connection weights, including the input biases, the input weights, and the output weights of hidden nodes. The traditional approach may trigger some well-known problems. For example, when there are many hidden nodes, the computational complexity is very high.

Instead of training all the weights, Huang *et al.* [5], [6] proposed the extreme learning machine (ELM) concept, in which the parameters of hidden nodes were generated randomly. For the SLFN case, only the output weights were required to be trained. In [5], Huang *et al.* formally proved that a SLFN with randomly generated hidden nodes could act as a universal approximator too. Recently, a number of studies on the ELM abilities were reported [7]–[9]. Also, many applications make use of the ELM concept. For instance, Pan *et al.* [10] proposed an ELM model for simulating a visual neuron system and for extracting the leukocyte from images. Minhas *et al.* [11] developed a human action recognition framework to assign the action label for the video based on the ELM concept. Wang *et al.* [12] combined the ELM mapping with a multi-view framework to extract some features with good representation for training, which could be feasible to a multi-view discriminant analysis [13] too. The ELM concept [14] could work with an autoencoder for clustering and subspace clustering [15]. In [16]–[18], other ELM applications were described in details. However, ELM algorithms used in these applications require to fix the size of hidden layer during training. Hence, it is less flexible than an incremental algorithm, which adds hidden nodes incrementally into a neural network during training.

In [5], [19], Huang *et al.* proposed **two incremental ELM algorithms** for the SLFN model, namely incremental ELM

The associate editor coordinating the review of this manuscript and approving it for publication was Xi Peng.

(I-ELM) and convex incremental ELM (CI-ELM). The concept of the incremental learning is that we add hidden nodes incrementally into an existing network until the predefined condition reaches. They also mentioned that some weights between the input nodes and hidden nodes in the ELM could be disconnected in some situations [20]. If the disconnection occurred in an uncontrolled manner, it could be described as a faulty situation. However, these two incremental algorithms were designed for fault-free situations only. We believed that fault and noise could greatly degrade the performance for the I-ELM and CI-ELM, if special procedures were not considered [21], [22].

In the realization of a neural network, fault and noise are unavoidable due to some practical issues. For instance, in the hardware implementation of neural network [23], various kinds of fault may happen [24], [25], such as fault in weights and fault in an activation function. If fault happens in the activation functions, we can model it as node fault [26]–[28]. Moreover, noise may happen in either a digital or analog implementation. For the digital implementation, when the floating format is used, the round-off error of a number is proportional to its magnitude. Hence, we can use the multiplicative noise model [29], [30] to describe the error. For the analog implementation, noise always exists in the amplifier output [31]. In addition, transient noise and fault may happen when the implementation is at nanoscale [32].

In the last decades, many fault tolerant approaches for neural networks were developed. For instance, the authors in [28], [33]–[36] proposed a failure/chaos injection approach, which was suitable for online mode training. However, this approach is unable to capture the failure behavior when the training iterations are not sufficient. Another approach is to formulate the training process as a constrained optimization [37]–[39], in which the constraints are defined by the fault tolerant level. Apparently, this approach does not guarantee to have a feasible solution if the constraints are too strict. **The above approaches are not suitable for the incremental learning mode.**

The weight decay concept could improve the fault tolerant ability [40], [41]. However, the objective function of the weight decay concept is not identical to the training set error of faulty networks. Hence, even for fault tolerant radial basis function (RBF) networks [22], [42], an optimal fault tolerant solution cannot be obtained, regardless of how the weight decay parameter is tuned. **To the best of our knowledge, only few results in the fault tolerant incremental learning were reported [43].**

This paper focuses on regression problem and the incremental learning mode, in which we incrementally insert hidden nodes in an SLFN. We propose two ELM based fault tolerant incremental learning algorithms. They are called node fault tolerant I-ELM (NFTI-ELM) and node fault tolerant CI-ELM (NFTCI-ELM), respectively. They are able to handle the coexistence of node fault and multiplicative node noise. We first define a fault tolerant objective function for SLFNs. By considering the change of the fault tolerant objective values, we derive the way to determine the output weight of the newly inserted hidden node. For the NFTI-ELM, the previously trained output weights remain unchanged. Simulation results confirm that the fault tolerant performance of the NFTI-ELM are better than those of the CI-ELM and I-ELM. To boost up the fault tolerant ability, the NFTCI-ELM is developed based on the CI-ELM concept. Unlike the NFTI-ELM, the NFTCI-ELM updates all previously trained output weights after determining the output weight of the newly inserted node. We prove that in terms of the faulty training set MSE, the two proposed algorithms converge. Simulation results show that the fault tolerant performance of the NFTCI-ELM is the best among all the compared algorithms. The compared algorithms include the I-ELM, the CI-ELM and the batch mode ELM. The batch mode ELM gives us a baseline since it is the best under fault-free situations. We use the well-known statistical significance test to verify the improvement for our proposed algorithms is statistically significant. Also, even the incorrect fault level is used for training, the performance of the NFTI-ELM and NFTCI-ELM are still better than that of the I-ELM and CI-ELM.

Our major contributions are summarized as follows.

- Two fault tolerant incremental learning algorithms are proposed, namely the NFTI-ELM and NFTCI-ELM, for SLFNs.
- The convergences of the two algorithms are proved in terms of the training set MSE.

The rest of this paper is organized as follows. Section II provides the background on the basic ELM model. The effect of node failure and node noise on the SLFN model is presented in Section III. Section IV describes the details of the two proposed algorithms. Section V presents the simulation results. Section VI concludes the paper.

II. ELM AND INCREMENTAL LEARNING

This paper considers to use SLFNs for nonlinear regression. The training set is denoted as $\mathcal{D}_t = \left\{(\boldsymbol{x}_l, y_l) : l = 1, \ldots, N\right\}$, where $\boldsymbol{x}_l \in \mathbb{R}^d$ is the l-th training input vector, d is the dimensions of data, and $y_l \in \mathbb{R}$ is its associated training output. Similarly, the test set is denoted as $\mathcal{D}_f = \left\{(\boldsymbol{x}'_{l'}, y'_{l'}) : l' = 1, \ldots, N'\right\}$, where $\boldsymbol{x}'_{l'} \in \mathbb{R}^d$ is the input vector of the $l'-$th test sample, and $y'_{l'} \in \mathbb{R}$ is its associated output.

The output of an SLFN [1], [2], [44] is expressed as

$$f_n(\boldsymbol{x}) = \sum_{i=1}^{n} \beta_i h_i(\boldsymbol{x}), \tag{1}$$

where n is the number of hidden nodes, $h_i(\boldsymbol{x})$ is the output of the i-th hidden node, and β_i is the output weight of the i-th hidden node. In this paper, we use a sigmoid function as the activation function. The output of the i-th hidden node is given by

$$h_i(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-(\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{x} + b_i)\right)}, \tag{2}$$

TABLE 1. Computational complexities.

	Total complexity
I-ELM & NFTI-ELM	$O(n \times d \times N)$
CI-ELM & NFTCI-ELM	$O(n \times d \times N) + O(n^2)$
Batch mode ELM	$O(n \times d \times N) + O(n^2 \times N) + O(n^3)$

where $\boldsymbol{w}_i$ and b_i are denoted as the input weight vector and input bias term of the i-th hidden node, respectively. It should be noticed that other activation functions [5], [19] are also applicable to our proposed algorithms.

In the ELM model, the values of $\boldsymbol{w}_i$ and b_i are selected randomly. Under fault-free situations, the training set error is given by

$$\mathcal{E} = \sum_{l=1}^{N}\left(y_l - \sum_{i=1}^{n}\beta_i h_i(\boldsymbol{x}_l)\right)^2 = \left\|\boldsymbol{y} - \sum_{i=1}^{n}\beta_i \boldsymbol{h}_i\right\|_2^2, \tag{3}$$

where $\boldsymbol{h}_i = \left[h_i(\boldsymbol{x}_1), ..., h_i(\boldsymbol{x}_N)\right]^{\mathrm{T}}$ is the collection of outputs of the $i-$th hidden node for all training samples, and $\boldsymbol{y} = \left[y_1, ..., y_N\right]^{\mathrm{T}}$ is the collection of its training outputs. In [45], based on (3), a batch mode ELM algorithm was proposed. By minimizing $\mathcal{E}$, the optimal output weights of hidden nodes $\boldsymbol{\beta}^*$ can be obtained by

$$\boldsymbol{\beta}^* = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{y}, \tag{4}$$

where $\boldsymbol{\beta}^* = \left[\beta_1, ..., \beta_n\right]^{\mathrm{T}}$ is the collection of the output weights of hidden nodes, and $\boldsymbol{H} = \left[\boldsymbol{h}_1, ..., \boldsymbol{h}_n\right]$ is the output matrix of the hidden layer. In Table 1, the total complexity for the batch mode ELM is $O(n \times d \times N) + O(n^2 \times N) + O(n^3)$.

In [5], [19], two incremental training algorithms were developed, i.e., the I-ELM and the CI-ELM. For the I-ELM, a randomly generated hidden node is inserted into the network at each training iteration. It is noticed that only the output weight of the newly inserted node is tuned, while the previously trained weights remain unchanged. The CI-ELM uses a similar training scheme, except that it uses a simple rule stated in (26) to update the previously trained weights. Therefore, the computational complexities of these two algorithms are significantly low. In Table 1, the total complexity of adding n hidden nodes for the I-ELM is $O(n \times d \times N)$. For the CI-ELM, the total complexity of adding n hidden nodes is $O(n \times d \times N) + O(n^2)$. The experimental results showed that both I-ELM and CI-ELM have a nice performance [5], [19] under fault-free situations. However, under faulty situations, the I-ELM and CI-ELM result in poor performance, as (3) does not encounter the fault tolerant ability.

III. FAULTY SLFNS

A. NODE NOISE AND NODE FAULT

When we use finite precision technology [29], [30] to implement hidden nodes, the hidden node outputs may deviate from their original values. The deviations are usually proportional to the magnitude of their original output values [29], [30]. In addition, in analog circuits, the deviations from the original output values are usually specified in terms of percentage error. The deviations can be modelled as multiplicative noise [29], [46]. In the multiplicative noise model, the hidden node outputs are described as

$$\tilde{h}_i(\boldsymbol{x}) = (1+\delta_i)h_i(\boldsymbol{x}), \quad \forall i = 1, \cdots, n, \tag{5}$$

where δ_i's are the normalized noise factors. This paper assumes that the normalized noise factors are identically independently distributed random variables. Their mean are equal to zero, and their variance are equal to σ^2.

In some situations, physical fault may happen. For example, when a communication link between a hidden node and an output node is broken, the output signal of the hidden node cannot be transmitted to the output node. In this case, we use the open node fault model to describe the outputs [21], [28], [47]. The hidden node outputs are given by

$$\tilde{h}_i(\boldsymbol{x}) = \alpha_i h_i(\boldsymbol{x}), \quad \forall i = 1, \cdots, n, \tag{6}$$

where α_i's are fault factors. They express whether the outputs of hidden nodes are tied to zero or not. When $\alpha_i = 0$, the output of the $i-$th node is tied to zero. Otherwise, the $i-$th hidden node operates correctly. This paper assumes that the fault factors α_i's are the identically independently distributed binary random variables. The probability mass function of α_i is given by

$$\mathrm{Prob}\{\alpha_i = 0\} = p, \text{ and } \mathrm{Prob}\{\alpha_i = 1\} = 1 - p. \tag{7}$$

When multiplicative node noise and open node fault coexist, the hidden node outputs are given by

$$\tilde{h}_i(\boldsymbol{x}) = (1+\delta_i)\alpha_i h_i(\boldsymbol{x}), \quad \forall i = 1, \cdots, n. \tag{8}$$

In (8), once a hidden node is opened, its output is tied to zero, regardless of the multiplicative noise level. When a hidden node is not opened, its output is then influenced by the multiplicative noise only.

From the statistical properties of α_i and δ_i, we obtain the following statistical properties of the hidden node outputs:

$$\left\langle \tilde{h}_i(\boldsymbol{x}) \right\rangle = (1-p)h_i(\boldsymbol{x}) \tag{9a}$$

$$\left\langle \tilde{h}_i(\boldsymbol{x})\tilde{h}_j(\boldsymbol{x}) \right\rangle = (1-p)^2 h_i(\boldsymbol{x})h_j(\boldsymbol{x}), \quad \text{for } i \neq j \tag{9b}$$

$$\left\langle \tilde{h}_i^2(\boldsymbol{x}) \right\rangle = (1-p)(1+\sigma^2)h_i^2(\boldsymbol{x}) \tag{9c}$$

where $\langle\cdot\rangle$ is the expectation operator over α_i and δ_i.

B. TRAINING SET ERROR AND TRAINING OBJECTIVE

For a faulty network, the training set error for a particular fault pattern is given by

$$\tilde{\mathcal{E}} = \sum_{l=1}^{N}\left(y_l - \sum_{i=1}^{n}\beta_i \tilde{h}_i(\boldsymbol{x}_l)\right)^2. \tag{10}$$

From (9)–(10), the average training set MSE over all possible fault patterns can be expressed as:

$$\bar{\mathcal{E}} = \sum_{l=1}^{N}\left(y_l^2 - 2(1-p)y_l\sum_{i=1}^{n}\beta_i h_i(\boldsymbol{x}_l)\right.$$

$$+(1-p)^2\sum_{i=1}^{n}\sum_{i\neq j}^{n}\beta_i h_i(\boldsymbol{x}_l)\beta_j h_j(\boldsymbol{x}_l)$$
$$+(1-p)(1+\sigma^2)\sum_{i=1}^{n}\beta_i^2 h_i^2(\boldsymbol{x}_l)\Bigg). \quad (11)$$

Defining

$$\boldsymbol{y}=\left[y_1,\cdots,y_N\right]^{\mathrm{T}} \text{ and } \boldsymbol{h}_i=\left[h_i(\boldsymbol{x}_1),\cdots,h_i(\boldsymbol{x}_N)\right]^{\mathrm{T}}, \quad (12)$$

we can rewrite (11) as

$$\begin{aligned}\bar{\mathcal{E}} &= p\,\|\boldsymbol{y}\|_2^2+(1-p)\left\|\boldsymbol{y}-\sum_{i=1}^{n}\beta_i\boldsymbol{h}_i\right\|_2^2\\ &\quad+(1-p)(p+\sigma^2)\sum_{i=1}^{n}\beta_i^2\,\|\boldsymbol{h}_i\|_2^2\\ &\quad-(1-p)p\left\|\sum_{i=1}^{n}\beta_i\boldsymbol{h}_i\right\|_2^2. \quad (13)\end{aligned}$$

The expression stated in (13) gives us a direct way to compute the training set MSE of faulty SLFNs, which is lower bounded by zero. The advantage of using (13) is that we do not need to generate a large number of faulty networks. Since the term $p\,\|\boldsymbol{y}\|_2^2$ in (13) is not a function of β_i's and the term $(1-p)$ is a constant, the training objective can be simplified to

$$\mathcal{L}=\left\|\boldsymbol{y}-\sum_{i=1}^{n}\beta_i\boldsymbol{h}_i\right\|_2^2+(p+\sigma^2)\sum_{i=1}^{n}\beta_i^2\,\|\boldsymbol{h}_i\|_2^2 -p\left\|\sum_{i=1}^{n}\beta_i\boldsymbol{h}_i\right\|_2^2. \quad (14)$$

As the training objective in (14) is derived from the training set MSE in (13), the training objective is lower bounded by $-p\,\|\boldsymbol{y}\|_2^2/(1-p)$. Based on (14), we can develop the fault tolerant incremental algorithms for SLFNs.

IV. FAULT TOLERANT ELM ALGORITHMS

The concept of the incremental learning is that we add hidden nodes one-by-one to the network. For ease of the description, we define some notations for the incremental learning, given by

$$\boldsymbol{f}_n=\sum_{i=1}^{n}\beta_i\boldsymbol{h}_i, \quad (15)$$
$$\boldsymbol{e}_n=\boldsymbol{y}-\boldsymbol{f}_n, \quad (16)$$
$$v_n=\sum_{i=1}^{n}\beta_i^2\,\|\boldsymbol{h}_i\|_2^2. \quad (17)$$

The vector $\boldsymbol{f}_n$ represents the collection of the network outputs for all training samples when n hidden nodes are used. The vector $\boldsymbol{e}_n$ represents the collection of the errors for all training samples when n hidden nodes are used. The value v_n represents the regularizer term. With (15), (16) and (17), for an SLFN with n hidden nodes, the objective function is given by

$$\mathcal{L}_n=\|\boldsymbol{e}_n\|_2^2+(p+\sigma^2)v_n-p\,\|\boldsymbol{f}_n\|_2^2. \quad (18)$$

In (18), the term "$(p+\sigma^2)v_n-p\,\|\boldsymbol{f}_n\|_2^2$" is the effect of the node fault and node noise.

A. NODE FAULT TOLERANT I-ELM

1) ALGORITHM

In the NFTI-ELM, when a node is newly inserted to the network at the n-th iteration, we determine the output weight of the newly inserted node and keep all previously trained weights $\{\beta_1,\cdots,\beta_{n-1}\}$ unchanged.

At the n-th iteration, the objective function, stated in (18), can be expressed as

$$\begin{aligned}\mathcal{L}_n &= \|\boldsymbol{e}_n\|_2^2+(p+\sigma^2)v_n-p\,\|\boldsymbol{f}_n\|_2^2 \quad (19)\\ &= \|\boldsymbol{e}_{n-1}\|_2^2-2\beta_n\boldsymbol{e}_{n-1}^{\mathrm{T}}\boldsymbol{h}_n+\beta_n^2\,\|\boldsymbol{h}_n\|_2^2\\ &\quad+(p+\sigma^2)v_{n-1}+(p+\sigma^2)\beta_n^2\,\|\boldsymbol{h}_n\|_2^2\\ &\quad-p\,\|\boldsymbol{f}_{n-1}\|_2^2-2p\beta_n\boldsymbol{f}_{n-1}^{\mathrm{T}}\boldsymbol{h}_n-p\beta_n^2\,\|\boldsymbol{h}_n\|_2^2. \quad (20)\end{aligned}$$

The change of the objective values between two consecutive iterations is given by

$$\begin{aligned}\triangle_n &= \mathcal{L}_n-\mathcal{L}_{n-1} \quad (21)\\ &= -2\beta_n(\boldsymbol{e}_{n-1}+p\boldsymbol{f}_{n-1})^{\mathrm{T}}\boldsymbol{h}_n\\ &\quad+(1+\sigma^2)\beta_n^2\,\|\boldsymbol{h}_n\|_2^2. \quad (22)\end{aligned}$$

It should be noticed that $\triangle_n$ is a quadratic function of β_n with a minimum value equal to a negative value. To maximize the reduction of the objective value, we consider the partial derivative of $\triangle_n$, given by

$$\frac{\partial\triangle_n}{\partial\beta_n}=-2(\boldsymbol{e}_{n-1}+p\boldsymbol{f}_{n-1})^{\mathrm{T}}\boldsymbol{h}_n+2\beta_n(1+\sigma^2)\,\|\boldsymbol{h}_n\|_2^2. \quad (23)$$

By setting $\partial\triangle_n/\partial\beta_n=0$, we obtain

$$\beta_n=\beta^*=\frac{(\boldsymbol{e}_{n-1}+p\boldsymbol{f}_{n-1})^{\mathrm{T}}\boldsymbol{h}_n}{(1+\sigma^2)\,\|\boldsymbol{h}_n\|_2^2}. \quad (24)$$

2) CONVERGENCE AND COMPLEXITY

By substituting the optimal β_n into (22), the change of the objective value $\triangle_n$ becomes

$$\triangle_n|_{\beta_n=\beta^*}=-\frac{\left((\boldsymbol{e}_{n-1}+p\boldsymbol{f}_{n-1})^{\mathrm{T}}\boldsymbol{h}_n\right)^2}{(1+\sigma^2)\,\|\boldsymbol{h}_n\|_2^2}. \quad (25)$$

Apparently, $\triangle_n$ in (25) is negative, which means $\mathcal{L}_n \leq \mathcal{L}_{n-1}$. In other words, when a hidden node is inserted to the SLFN, the sequence of objective values $\{\mathcal{L}_1,\mathcal{L}_2,\ldots,\mathcal{L}_n\}$ is decreasing. As mentioned, the training set MSE (13) is lower bounded by zero, and the objective value is derived from the training set MSE. Hence, the objective value is lower bounded too. As the training objective $\mathcal{L}$ is decreasing and lower bounded, we notice that our proposed NFTI-ELM, in terms of $\mathcal{L}$, converges.

Algorithm 1 summarizes the proposed NFTI-ELM. It can be seen that the computational complexity for Steps (7)-(10) at each iteration is $O(d \times N)$. Hence, the total complexity for adding n nodes is equal to $O(n \times d \times N)$, as shown in Table 1. It should be noticed that the computational complexity of the I-ELM at each iteration is equal to $O(d \times N)$ too.

Algorithm 1 NFTI-ELM

```
1:  Set $n$ equal to zero, $n = 0$.
2:  Set the initial residue error to $\boldsymbol{y}$, $\boldsymbol{e}_0 = \boldsymbol{y}$.
3:  Set the initial network output to a zero vector, $\boldsymbol{f}_0 = \boldsymbol{0}$.
4:  while $n \le n_{\max}$ do
5:      Increment $n$ by 1.
6:      Insert a new node to the SLFN. Its input bias $b_n$ and
        input weights $\boldsymbol{a}_n$ are randomly generated.
7:      Compute the corresponding output vector $\boldsymbol{h}_n$ of this
        hidden node.
8:      Compute the output weight of the newly inserted
        node:
        $$\beta_n = \frac{(\boldsymbol{e}_{n-1} + p\boldsymbol{f}_{n-1})^{\mathrm{T}} \boldsymbol{h}_n}{(1+\sigma^2) \|\boldsymbol{h}_n\|_2^2}.$$
9:      $\boldsymbol{f}_n = \boldsymbol{f}_{n-1} + \beta_n \boldsymbol{h}_n$.
10:     $\boldsymbol{e}_n = \boldsymbol{y} - \boldsymbol{f}_n$.
11: end while
```

B. NODE FAULT TOLERANT CI-ELM

1) ALGORITHM

Under the fault-free situation [19], when we are allowed to update the previously trained weights, the performance can be enhanced. However, as shown in Fig. 1(a)-(i), the original CI-ELM cannot handle the faulty situation. Hence, it is necessary to develop a fault tolerant version for the CI-ELM.

At the n-th iteration, after we determine the output weight β_n of the newly inserted node, we update all previously trained weights by

$$\beta_i^{new} = (1-\beta_n)\beta_i, \tag{26}$$

for $i = 1$ to $n-1$. With this new update scheme in β_i's, the recursive definitions for $\boldsymbol{f}_n$, $\boldsymbol{e}_n$ and v_n become

$$\boldsymbol{f}_n = (1-\beta_n)\boldsymbol{f}_{n-1} + \beta_n \boldsymbol{h}_n \tag{27}$$

$$\boldsymbol{e}_n = \boldsymbol{y} - \boldsymbol{f}_n, \tag{28}$$

$$v_n = (1-\beta_n)^2 v_{n-1} + \beta_n^2 \|\boldsymbol{h}_n\|_2^2, \tag{29}$$

where $\boldsymbol{f}_0 = \boldsymbol{0}$, $\boldsymbol{e}_0 = \boldsymbol{y}$ and $v_0 = 0$.

From (26)–(29), the objective value at the n-th iteration can be expressed as

$$\begin{aligned}\mathcal{L}_n = {} & \|\boldsymbol{e}_{n-1}\|_2^2 - 2\beta_n \boldsymbol{e}_{n-1}^{\mathrm{T}}(\boldsymbol{h}_n - \boldsymbol{f}_{n-1}) \\ & + \beta_n^2 \left\|\boldsymbol{h}_n - \boldsymbol{f}_{n-1}\right\|_2^2 + (p+\sigma^2)v_{n-1} \\ & - 2\beta_n(p+\sigma^2)v_{n-1} + \beta_n^2(p+\sigma^2)v_{n-1} \\ & + \beta_n^2(p+\sigma^2)\|\boldsymbol{h}_n\|_2^2 \\ & - p\left\|\boldsymbol{f}_{n-1}\right\|_2^2 - 2\beta_n p \boldsymbol{f}_{n-1}^{\mathrm{T}}(\boldsymbol{h}_n - \boldsymbol{f}_{n-1}) \\ & - \beta_n^2 p \left\|\boldsymbol{h}_n - \boldsymbol{f}_{n-1}\right\|_2^2 .\end{aligned} \tag{30}$$

Algorithm 2 NFTCI-ELM

```
1:  Set $n$ equal to zero, $n = 0$.
2:  Set the initial residue error to $\boldsymbol{y}$, $\boldsymbol{e}_0 = \boldsymbol{y}$.
3:  Set the initial network output to a zero vector, $\boldsymbol{f}_0 = \boldsymbol{0}$.
4:  Set $v_0 = 0$, $\boldsymbol{r}_0 = \boldsymbol{0}$.
5:  while $n \le n_{\max}$ do
6:      Increment $n$ by 1.
7:      Insert a new hidden node to the SLFN. Its input bias
        $b_n$ and input weights $\boldsymbol{a}_n$ are randomly generated.
8:      Compute the corresponding output vector $\boldsymbol{h}_n$ of this
        hidden node.
9:      Compute $\boldsymbol{r}_n = \boldsymbol{h}_n - \boldsymbol{f}_{n-1}$.
10:     Compute the new weight:
        $$\beta_n = \frac{\left((\boldsymbol{e}_{n-1} + p\boldsymbol{f}_{n-1})^{\mathrm{T}}(\boldsymbol{r}_n) + (p+\sigma^2)v_{n-1}\right)}{(1-p)\|\boldsymbol{r}_n\|_2^2 + (p+\sigma^2)(v_{n-1} + \|\boldsymbol{h}_n\|_2^2)}.$$
11:     $\boldsymbol{f}_n = (1-\beta_n)\boldsymbol{f}_{n-1} + \beta_n \boldsymbol{h}_n$.
12:     $\boldsymbol{e}_n = \boldsymbol{y} - \boldsymbol{f}_n$.
13:     $v_n = (1-\beta_n)^2 v_{n-1} + \beta_n^2 \|\boldsymbol{g}_n\|_2^2$.
14:     $\beta_i = (1-\beta_n)\beta_i$,  for all $i = 1, \cdots, n-1$.
15: end while
```

The change of the objective value between two consecutive iterations is then given by

$$\triangle_n = \mathcal{L}_n - \mathcal{L}_{n-1} \tag{31}$$

$$\begin{aligned} &= -2\beta_n \left((\boldsymbol{e}_{n-1} + p\boldsymbol{f}_{n-1})^{\mathrm{T}} \boldsymbol{r}_n + (p+\sigma^2)v_{n-1}\right) \\ &\quad + \beta_n^2 \left((1-p)\|\boldsymbol{r}_n\|_2^2 + (p+\sigma^2)(v_{n-1} + \|\boldsymbol{h}_n\|_2^2)\right), \end{aligned} \tag{32}$$

where

$$\boldsymbol{r}_n = \boldsymbol{h}_n - \boldsymbol{f}_{n-1}. \tag{33}$$

Again, for the NFTCI-ELM, $\triangle_n$ is a quadratic function of β_n with a minimum value equal to a negative value. To maximize the reduction of the objective value, we should set

$$\beta_n = \beta^* = \frac{(\boldsymbol{e}_{n-1} + p\boldsymbol{f}_{n-1})^{\mathrm{T}}(\boldsymbol{r}_n) + (p+\sigma^2)v_{n-1}}{(1-p)\|\boldsymbol{r}_n\|_2^2 + (p+\sigma^2)(v_{n-1} + \|\boldsymbol{h}_n\|_2^2)}. \tag{34}$$

2) CONVERGENCE AND COMPLEXITY

By substituting the optimal β_n into (32), the change of the objective value $\triangle_n$ becomes

$$\triangle_n|_{\beta_n=\beta^*} = -\frac{\left((\boldsymbol{e}_{n-1} + p\boldsymbol{f}_{n-1})^{\mathrm{T}}(\boldsymbol{r}_n) + (p+\sigma^2)v_{n-1}\right)^2}{(1-p)\|\boldsymbol{r}_n\|_2^2 + (p+\sigma^2)(v_{n-1} + \|\boldsymbol{h}_n\|_2^2)}. \tag{35}$$

Again, $\triangle_n$ in (35) is negative, which means $\mathcal{L}_n \le \mathcal{L}_{n-1}$. The sequence of objective values $\{\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n\}$ is decreasing when a hidden node is inserted to the SLFN. As the objective value is derived from the training set MSE, which

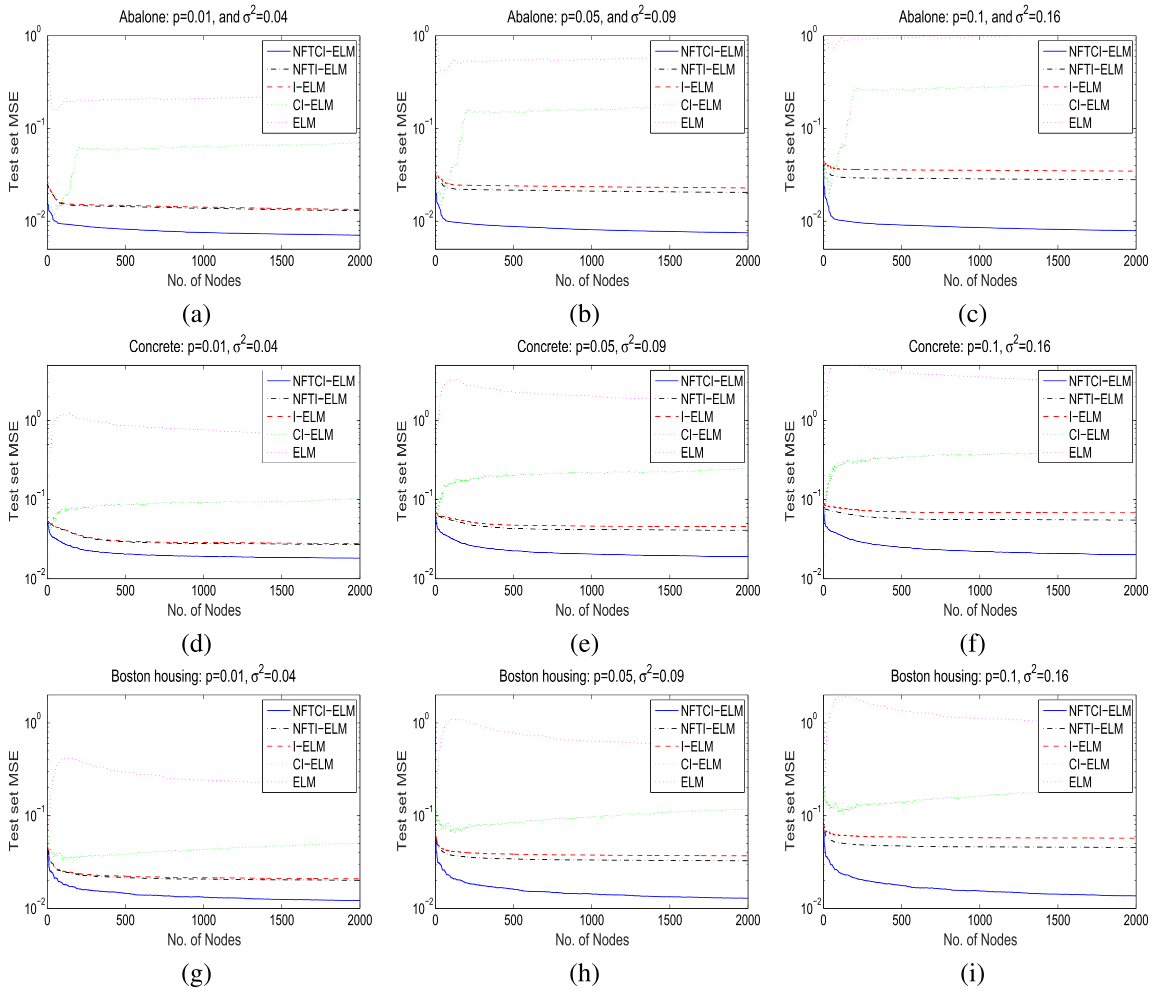

FIGURE 1. **The performance of algorithms versus the number of additive nodes. The multiplicative noise and probability of open fault intensity levels are σ^2 and p. The datasets are Abalone, Concrete, and boston housing.**

is lower bounded by zero, the objective value is also lower bounded. Since the training objective $\mathcal{L}$ is decreasing and lower bounded, we notice that the proposed NFTCI-ELM, in terms of $\mathcal{L}$, converges.

The procedures of the NFTCI-ELM are summarized in Algorithm 2. At the n-th iteration, the computational complexity is $O(d \times N) + O(n)$. Hence, the total complexity of adding n hidden nodes is given by $O(n \times d \times N) + O(n^2)$, as shown in Table 1. Comparing to the NFTI-ELM, the update of β_i's in the NFTCI-ELM increases the computational complexity.

V. SIMULATION RESULTS

The performance of the proposed NFTI-ELM and NFTCI-ELM are verified by comparing it against other ELM algorithms, i.e., the I-ELM, the CI-ELM and the batch mode ELM, under faulty situations. For the batch mode ELM, the output weights of hidden nodes $\boldsymbol{\beta}$ can be obtained by the least square solution of SLFN, i.e., $\boldsymbol{\beta} = (\boldsymbol{H}^T\boldsymbol{H})^{-1}\boldsymbol{H}^T\boldsymbol{y}$. It gives us a baseline for comparison since given a set of hidden nodes it provides the best solution under fault-free situations.

TABLE 2. **Properties of the ten data sets.**

Dataset	Number of features	Training set size	Test set size
Abalone	9	2000	2177
Concrete	9	500	530
Boston Housing	14	250	256
Wine Quality	12	2000	2898
ASN	6	750	753
Auto MPG	8	195	203
Mortgage	16	500	549
Weather Ankara	10	800	809
Parkinsons Telemonitoring	21	2937	2937
Computer Activity	22	4096	4096

For the simulation, ten well-known benchmark datasets in the field of regression problem are used.

A. DATASETS AND SETTINGS

Ten commonly used datasets for regression problem are selected from the UCI[1] and KEEL[2] dataset

[1]https://archive.ics.uci.edu/ml/datasets.html
[2]https://sci2s.ugr.es/keel/category.php?cat=reg

TABLE 3. Concurrent fault situations: Average Test set MSE of the ELM, I-ELM, CI-ELM, NFTI-ELM and NFTCI-ELM over 400 trials.

Dataset	Fault level	I-ELM		NFTI-ELM		CI-ELM		NFTCI-ELM		Batch mode ELM	
		AVG MSE	STD MSE	AVG MSE	STD MSE	AVG MSE	STD MSE	AVG MSE	STD MSE	AVG MSE	STD MSE
Abalone	p=0.01, σ^2=0.04	0.01374	0.00044	0.01336	0.00043	0.06497	0.02057	0.00795	0.00031	0.24578	0.03792
	p=0.05, σ^2=0.09	0.02291	0.00044	0.02063	0.00041	0.16337	0.05533	0.00852	0.00033	0.65088	0.10204
	p=0.1, σ^2=0.16	0.03449	0.00047	0.02806	0.00041	0.28288	0.09737	0.00887	0.00033	1.14138	0.17960
Concrete	p=0.01, σ^2=0.04	0.02973	0.00112	0.02910	0.00113	0.13071	0.07152	0.01997	0.00068	0.90262	0.08209
	p=0.05, σ^2=0.09	0.04708	0.00122	0.04295	0.00128	0.32294	0.19242	0.02201	0.00073	2.41047	0.22092
	p=0.1, σ^2=0.16	0.06894	0.00148	0.05716	0.00155	0.55638	0.33866	0.02444	0.00081	4.23528	0.38885
Boston Housing	p=0.01, σ^2=0.04	0.02378	0.00232	0.02320	0.00232	0.05846	0.02082	0.01515	0.00183	0.29707	0.05097
	p=0.05, σ^2=0.09	0.03961	0.00270	0.03588	0.00261	0.13605	0.05622	0.01656	0.00188	0.78633	0.13948
	p=0.1, σ^2=0.16	0.05955	0.00334	0.04886	0.00303	0.23073	0.09910	0.01805	0.00194	1.37898	0.24656
Wine Quality	p=0.01, σ^2=0.04	0.02919	0.00076	0.02834	0.00075	0.03905	0.01145	0.01666	0.00042	1.03617	0.12810
	p=0.05, σ^2=0.09	0.04938	0.00074	0.04438	0.00074	0.07852	0.03074	0.01731	0.00044	2.75945	0.34433
	p=0.1, σ^2=0.16	0.07487	0.00074	0.06075	0.00077	0.12736	0.05407	0.01797	0.00046	4.84500	0.60587
ASN	p=0.01, σ^2=0.04	0.03519	0.00111	0.03407	0.00115	0.23471	0.11031	0.01851	0.00057	1.05027	0.09731
	p=0.05, σ^2=0.09	0.06450	0.00133	0.05745	0.00144	0.60447	0.29629	0.02032	0.00059	2.80474	0.26181
	p=0.1, σ^2=0.16	0.10152	0.00176	0.08134	0.00183	1.05331	0.52124	0.02206	0.00070	4.92852	0.46080
AutoMPG	p=0.01, σ^2=0.04	0.01874	0.00103	0.01809	0.00101	0.05431	0.02445	0.00931	0.00080	0.11996	0.01599
	p=0.05, σ^2=0.09	0.03454	0.00162	0.03073	0.00151	0.13287	0.06583	0.01013	0.00083	0.31360	0.04337
	p=0.1, σ^2=0.16	0.05449	0.00256	0.04361	0.00222	0.22872	0.11588	0.01089	0.00086	0.54865	0.07648
Mortgage	p=0.01, σ^2=0.04	0.00814	0.00039	0.00758	0.00037	0.01464	0.00764	0.00113	0.00007	0.00772	0.00119
	p=0.05, σ^2=0.09	0.02104	0.00091	0.01777	0.00080	0.03900	0.02056	0.00173	0.00009	0.02100	0.00320
	p=0.1, σ^2=0.16	0.03733	0.00154	0.02818	0.00124	0.06917	0.03619	0.00257	0.00012	0.03775	0.00563
Weather Ankara	p=0.01, σ^2=0.04	0.01932	0.00047	0.01798	0.00046	0.05588	0.02400	0.00211	0.00010	0.03531	0.00356
	p=0.05, σ^2=0.09	0.05032	0.00077	0.04249	0.00073	0.14965	0.06452	0.00337	0.00016	0.09519	0.00959
	p=0.1, σ^2=0.16	0.08947	0.00123	0.06750	0.00106	0.26472	0.11355	0.00503	0.00022	0.16927	0.01689
Parkinsons Telemonitoring	p=0.01, σ^2=0.04	0.05006	0.00078	0.04928	0.00078	0.06146	0.01212	0.03898	0.00061	11.73207	0.71329
	p=0.05, σ^2=0.09	0.06917	0.00079	0.06445	0.00079	0.10168	0.03262	0.03978	0.00061	31.49087	1.91687
	p=0.1, σ^2=0.16	0.09330	0.00089	0.07996	0.00088	0.15135	0.05742	0.04070	0.00061	55.39183	3.37270
Computer Activity	p=0.01, σ^2=0.04	0.05352	0.00242	0.05082	0.00241	0.22960	0.14329	0.01896	0.00083	2.06417	0.20566
	p=0.05, σ^2=0.09	0.11696	0.00233	0.10114	0.00239	0.59454	0.38546	0.02158	0.00103	5.54616	0.55257
	p=0.1, σ^2=0.16	0.19708	0.00237	0.15250	0.00246	1.03937	0.67837	0.02405	0.00121	9.76138	0.97220

repositories [48]–[57]. Table 2 summarizes the properties of the ten datasets. Abalone [49] aims at predicting the age of abalones by using 8 features as input. Concrete [50] aims at predicting the concrete compressive strength by using 8 features as input. Boston Housing [51] aims at predicting the housing value in Boston by using 13 features as input. Wine Quality [52] aims at predicting the quality of wine with a score between 0 and 10 by using 11 features as input. Airfoil Self-Noise (ASN) [53] aims at predicting the sound pressure level in an anechoic wind tunnel by using 5 features as input.

Auto MPG [54] aims at predicting the fuel consumption for cars in miles per gallon by using 7 features as input. Mortgage [55] aims at predicting the 30 Year-Conventional Mortgage Rate in the USA by using 15 features as input. Weather Ankara [57] aims at predicting the mean temperature in Ankara by using 9 features as input. Parkinsons Telemonitoring [56] aims at predicting the clinician's Parkinson's disease symptom score on the UPDRS scale by using 20 features as input. Computer Activity aims at predicting the portion of time that CPUs run in user mode by using 21 features as input.

We adopt the validation method mentioned in [5] with these 10 datasets. For each dataset, the samples are randomly split for the training and test sets for 20 trials; hence, we have 20 partitions. For each partition, we run the simulation with 20 sets of random hidden nodes. Therefore, the total number of trails is 400. The training inputs and outputs are normalized to the range of $[-1, 1]$ and $[0, 1]$, respectively.

B. TEST SET MSE VERSUS NUMBER OF HIDDEN NODES

We demonstrate how the test set MSE changes with respect to the various numbers of hidden nodes using three datasets. The datasets are Abalone [49], Concrete [50], and Boston Housing [51]. We consider three different fault levels. They are $\{p = 0.01\, \sigma^2 = 0.04\}$, $\{p = 0.05\, \sigma^2 = 0.09\}$ and $\{p = 0.1\, \sigma^2 = 0.16\}$.

Fig. 1 shows the test set MSE versus the number of nodes under faulty situations. It is noticed that the test set MSE values of the CI-ELM and batch mode ELM are much higher than those of the other three algorithms. In other words, the fault tolerant ability of the CI-ELM and batch mode ELM is very weak. For the I-ELM, NFTI-ELM and NFTCI-ELM, when the number of hidden nodes is more than 500, the decreasing rate of the test set MSE becomes slow and reaches a plateau.

As shown in Fig. 1(c),(f),(i), the improvements for the NFTI-ELM and NFTCI-ELM are more significant under high fault levels. For instance, in the Abalone dataset, **when the fault level $\{p = 0.01, \sigma^2 = 0.04\}$ and 500 hidden nodes are considered**, the test set MSE values of the CI-ELM and the batch mode ELM are very large. When we use the I-ELM, the MSE value is equal to 0.01480. For the NFTI-ELM, the MSE value is equal to 0.01437.

TABLE 4. Paired t-test between I-ELM and NFTI-ELM. The number of trails is 400.

Dataset	Fault level	AVG I-ELM MSE	AVG NFTI-ELM MSE	AVG improvement	Standard error	**t-value**	**Confidence interval of AVG improvement**
Abalone	p=0.01,σ^2=0.04	0.01374	0.01336	0.00038	1.10E-06	**346.10792**	**[0.00038, 0.00038]**
	p=0.05,σ^2=0.09	0.02291	0.02063	0.00227	4.30E-06	**528.23804**	**[0.00226, 0.00228]**
	p=0.1,σ^2=0.16	0.03449	0.02806	0.00643	7.89E-06	**814.91264**	**[0.00641, 0.00645]**
Concrete	p=0.01,σ^2=0.04	0.02973	0.02910	0.00064	3.12E-06	**203.68579**	**[0.00063, 0.00064]**
	p=0.05,σ^2=0.09	0.04708	0.04295	0.00413	1.41E-05	**293.50131**	**[0.00411, 0.00416]**
	p=0.1,σ^2=0.16	0.06894	0.05716	0.01179	2.61E-05	**452.26897**	**[0.01174, 0.01184]**
Boston Housing	p=0.01,σ^2=0.04	0.02378	0.02320	0.00058	4.50E-06	**129.13001**	**[0.00057, 0.00059]**
	p=0.05,σ^2=0.09	0.03961	0.03588	0.00373	1.87E-05	**199.09259**	**[0.00369, 0.00377]**
	p=0.1,σ^2=0.16	0.05955	0.04886	0.01069	3.87E-05	**276.19487**	**[0.01061, 0.01076]**
Wine Quality	p=0.01,σ^2=0.04	0.02919	0.02834	0.00084	1.83E-06	**462.58796**	**[0.00084, 0.00085]**
	p=0.05,σ^2=0.09	0.04938	0.04438	0.00499	6.51E-06	**767.33841**	**[0.00498, 0.00501]**
	p=0.1,σ^2=0.16	0.07487	0.06075	0.01412	1.15E-05	**1224.98223**	**[0.0141, 0.01414]**
ASN	p=0.01,σ^2=0.04	0.03519	0.03407	0.00112	4.88E-06	**229.48804**	**[0.00111, 0.00113]**
	p=0.05,σ^2=0.09	0.06450	0.05745	0.00706	1.93E-05	**365.14729**	**[0.00702, 0.00709]**
	p=0.1,σ^2=0.16	0.10152	0.08134	0.02018	3.70E-05	**546.03198**	**[0.02011, 0.02025]**
AutoMPG	p=0.01,σ^2=0.04	0.01874	0.01809	0.00064	3.22E-06	**200.40528**	**[0.00064, 0.00065]**
	p=0.05,σ^2=0.09	0.03454	0.03073	0.00382	1.39E-05	**274.05318**	**[0.00379, 0.00384]**
	p=0.1,σ^2=0.16	0.05449	0.04361	0.01088	2.96E-05	**367.85085**	**[0.01082, 0.01094]**
Mortgage	p=0.01,σ^2=0.04	0.00814	0.00758	0.00056	1.55E-06	**362.33493**	**[0.00056, 0.00056]**
	p=0.05,σ^2=0.09	0.02104	0.01777	0.00327	7.26E-06	**450.17963**	**[0.00325, 0.00328]**
	p=0.1,σ^2=0.16	0.03733	0.02818	0.00915	1.76E-05	**518.56903**	**[0.00912, 0.00919]**
Weather Ankara	p=0.01,σ^2=0.04	0.01932	0.01798	0.00135	2.24E-06	**600.45396**	**[0.00134, 0.00135]**
	p=0.05,σ^2=0.09	0.05032	0.04249	0.00784	7.59E-06	**1032.23526**	**[0.00782, 0.00785]**
	p=0.1,σ^2=0.16	0.08947	0.06750	0.02197	1.53E-05	**1432.24333**	**[0.02194, 0.022]**
Parkinsons Telemonitoring	p=0.01,σ^2=0.04	0.05006	0.04928	0.00079	2.38E-06	**330.97561**	**[0.00078, 0.00079]**
	p=0.05,σ^2=0.09	0.06917	0.06445	0.00473	1.01E-05	**469.13099**	**[0.00471, 0.00475]**
	p=0.1,σ^2=0.16	0.09330	0.07996	0.01334	1.83E-05	**727.19767**	**[0.0133, 0.01338]**
Computer Activity	p=0.01,σ^2=0.04	0.05352	0.05082	0.00269	6.22E-06	**432.51435**	**[0.00268, 0.0027]**
	p=0.05,σ^2=0.09	0.11696	0.10114	0.01582	1.78E-05	**887.10986**	**[0.01578, 0.01585]**
	p=0.1,σ^2=0.16	0.19708	0.15250	0.04458	3.27E-05	**1361.71540**	**[0.04452, 0.04465]**

TABLE 5. Paired t-test between I-ELM and NFTCI-ELM. The number of trails is 400.

Dataset	Fault level	AVG I-ELM MSE	AVG NFTCI-ELM MSE	AVG improvement	Standard error	**t-value**	**Confidence interval of AVG improvement**
Abalone	p=0.01,σ^2=0.04	0.01374	0.00795	0.00578	1.45E-05	**398.30489**	**[0.00575, 0.00581]**
	p=0.05,σ^2=0.09	0.02291	0.00852	0.01439	1.43E-05	**1005.80872**	**[0.01436, 0.01442]**
	p=0.1,σ^2=0.16	0.03449	0.00887	0.02562	1.70E-05	**1506.37294**	**[0.02558, 0.02565]**
Concrete	p=0.01,σ^2=0.04	0.02973	0.01997	0.00976	4.31E-05	**226.49519**	**[0.00968, 0.00985]**
	p=0.05,σ^2=0.09	0.04708	0.02201	0.02507	4.66E-05	**538.17266**	**[0.02498, 0.02516]**
	p=0.1,σ^2=0.16	0.06894	0.02444	0.04451	6.36E-05	**700.32799**	**[0.04438, 0.04463]**
Boston Housing	p=0.01,σ^2=0.04	0.02378	0.01515	0.00863	5.79E-05	**148.85404**	**[0.00851, 0.00874]**
	p=0.05,σ^2=0.09	0.03961	0.01656	0.02304	7.48E-05	**308.14928**	**[0.02289, 0.02319]**
	p=0.1,σ^2=0.16	0.05955	0.01805	0.04150	1.12E-04	**369.98820**	**[0.04128, 0.04173]**
Wine Quality	p=0.01,σ^2=0.04	0.02919	0.01666	0.01252	2.96E-05	**423.19929**	**[0.01246, 0.01258]**
	p=0.05,σ^2=0.09	0.04938	0.01731	0.03207	2.82E-05	**1136.65102**	**[0.03201, 0.03212]**
	p=0.1,σ^2=0.16	0.07487	0.01797	0.05690	2.93E-05	**1939.06175**	**[0.05684, 0.05696]**
ASN	p=0.01,σ^2=0.04	0.03519	0.01851	0.01668	5.02E-05	**332.37844**	**[0.01658, 0.01677]**
	p=0.05,σ^2=0.09	0.06450	0.02032	0.04418	5.80E-05	**761.21720**	**[0.04407, 0.0443]**
	p=0.1,σ^2=0.16	0.10152	0.02206	0.07946	7.87E-05	**1010.28241**	**[0.07931, 0.07961]**
AutoMPG	p=0.01,σ^2=0.04	0.01874	0.00931	0.00943	3.79E-05	**249.00953**	**[0.00936, 0.00951]**
	p=0.05,σ^2=0.09	0.03454	0.01013	0.02442	7.93E-05	**307.83917**	**[0.02426, 0.02457]**
	p=0.1,σ^2=0.16	0.05449	0.01089	0.04360	1.32E-04	**330.53355**	**[0.04334, 0.04385]**
Mortgage	p=0.01,σ^2=0.04	0.00814	0.00113	0.00702	1.95E-05	**359.17946**	**[0.00698, 0.00705]**
	p=0.05,σ^2=0.09	0.02104	0.00173	0.01932	4.63E-05	**417.15121**	**[0.01922, 0.01941]**
	p=0.1,σ^2=0.16	0.03733	0.00257	0.03476	7.84E-05	**443.16467**	**[0.03461, 0.03492]**
Weather Ankara	p=0.01,σ^2=0.04	0.01932	0.00211	0.01722	2.30E-05	**749.16397**	**[0.01717, 0.01726]**
	p=0.05,σ^2=0.09	0.05032	0.00337	0.04695	3.85E-05	**1218.18460**	**[0.04688, 0.04703]**
	p=0.1,σ^2=0.16	0.08947	0.00503	0.08444	6.15E-05	**1371.92234**	**[0.08432, 0.08456]**
Parkinsons Telemonitoring	p=0.01,σ^2=0.04	0.05006	0.03898	0.01108	2.86E-05	**387.36573**	**[0.01102, 0.01114]**
	p=0.05,σ^2=0.09	0.06917	0.03978	0.02939	3.13E-05	**939.94386**	**[0.02933, 0.02945]**
	p=0.1,σ^2=0.16	0.09330	0.04070	0.05260	3.90E-05	**1347.32374**	**[0.05252, 0.05268]**
Computer Activity	p=0.01,σ^2=0.04	0.05352	0.01896	0.03456	1.06E-04	**326.91231**	**[0.03435, 0.03477]**
	p=0.05,σ^2=0.09	0.11696	0.02158	0.09537	1.00E-04	**949.28717**	**[0.09518, 0.09557]**
	p=0.1,σ^2=0.16	0.19708	0.02405	0.17303	1.03E-04	**1684.10952**	**[0.17283, 0.17323]**

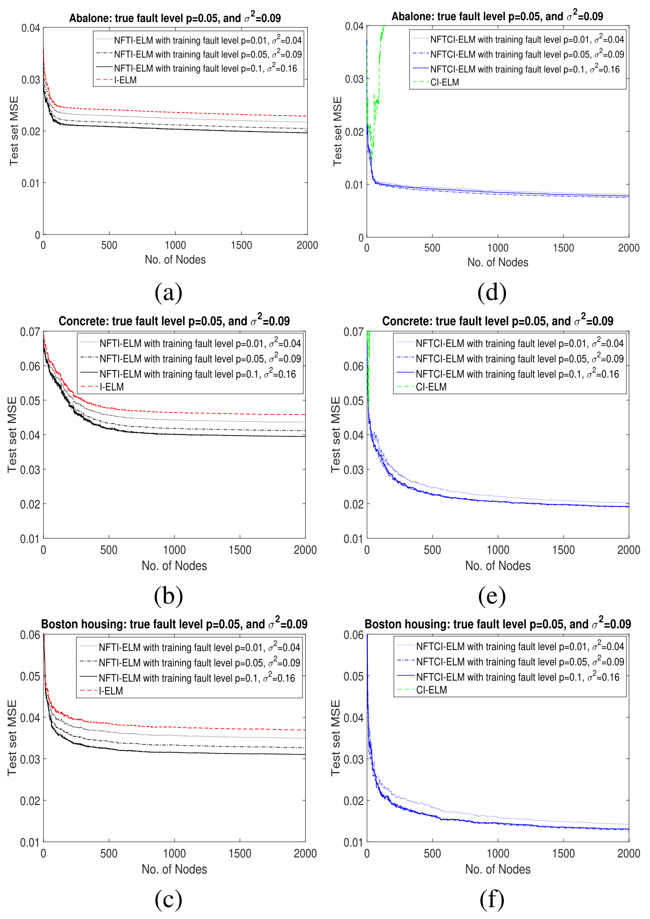

FIGURE 2. The performance of NFTI-ELM and NFTCI-ELM under true fault level $\{p = 0.05, \sigma^2 = 0.09\}$ trained in different fault levels, i.e., $\{p = 0.01, \sigma^2 = 0.04\}$, $\{p = 0.05, \sigma^2 = 0.09\}$ and $\{p = 0.1, \sigma^2 = 0.16\}$.

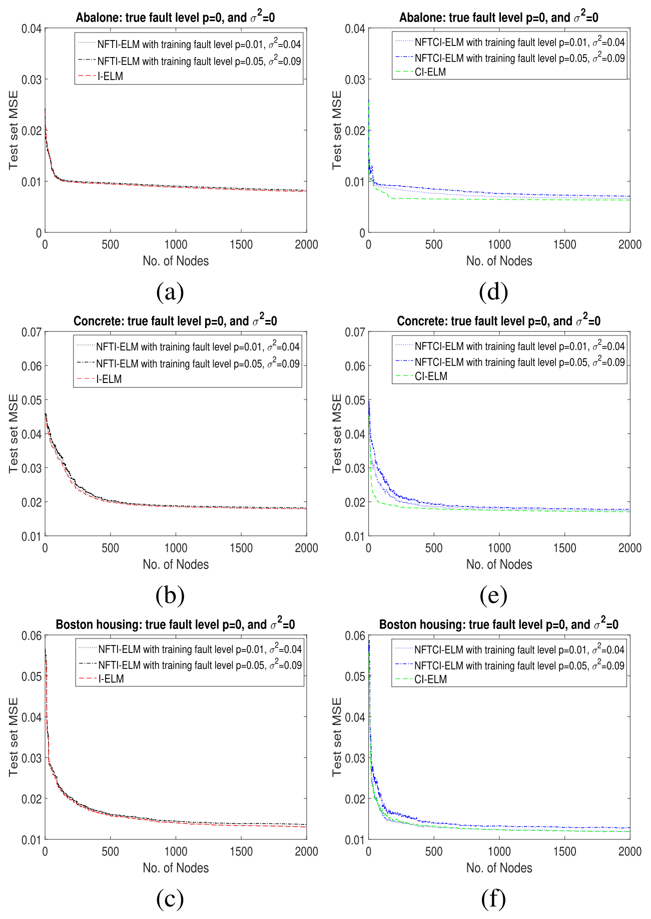

FIGURE 3. The performance of NFTI-ELM and NFTCI-ELM under true fault level $\{p = 0, \sigma^2 = 0\}$ (fault-free situation) trained in different fault levels, i.e., $\{p = 0.01, \sigma^2 = 0.04\}$ and $\{p = 0.05, \sigma^2 = 0.09\}$.

Compared to the I-ELM, the improvement for the NFTI-ELM is 0.00043. For the CI-ELM, the MSE value is equal to 0.00822. **Compared to the I-ELM, the improvement for the NFTCI-ELM is 0.00658.**

When the fault level raises to $\{p = 0.1, \sigma^2 = 0.16\}$, the MSE value of the I-ELM is equal to 0.03577. For the NFTI-ELM, the MSE value is reduced to 0.02908. **Compared to the I-ELM, the improvement for the NFTI-ELM is 0.00669.** For the NFTCI-ELM, the MSE value is further reduced to 0.00911. **Compared to the I-ELM, the improvement for the NFTCI-ELM is 0.02666.**

C. COMPARISON WITH THE I-ELM, CI-ELM AND BATCH MODE ELM

We compare the performance of our NFTI-ELM and NTCI-ELM with the I-ELM, CI-ELM and batch mode ELM in terms of test set MSE using 500 hidden nodes. **We consider three different fault levels. They are** $\{p = 0.01\, \sigma^2 = 0.04\}$, $\{p = 0.05, \sigma^2 = 0.09\}$ **and** $\{p = 0.1, \sigma^2 = 0.16\}$. **As mentioned, we partition the dataset into the training set and test set 20 times. For each partition, we generate 20 sets of random hidden nodes. Hence, we run the simulation 400 times for each dataset and each fault level.**

Table 3 shows the average test set MSE values of the algorithms for 400 trails under the three fault levels. The results indicate that the performance of our NFTI-ELM and NFTCI-ELM are much better than those of the other algorithms. For instance, in Abalone dataset with the fault level $\{p = 0.05, \sigma = 0.09\}$, the MSE values of the batch mode ELM, I-ELM and CI-ELM are 0.65088, 0.02291 and 0.16337, respectively. On the other hand, the MSE values of our NFTI-ELM and NFTCI-ELM are 0.02063 and 0.00852, respectively. We notice that the NFTCI-ELM has the best performance, which has the lowest MSE values among the other algorithms.

Moreover, compared to other algorithms, the NFTCI-ELM are relatively insensitive to the fault level. For instance, in Abalone dataset, when the fault level is equal to $\{p = 0.01, \sigma = 0.04\}$, the MSE value of the NFTI-ELM is 0.01336 from Table 3. When the fault level is equal to $\{p = 0.1, \sigma = 0.16\}$, the MSE value of the NFTI-ELM increases to 0.02806. For the NTCI-ELM, when the fault level is equal to $\{p = 0.01, \sigma = 0.04\}$, the MSE value is 0.00795. Even we greatly increase the fault level to $\{p = 0.1, \sigma = 0.16\}$, the MSE value slightly increases to 0.00887 only. This phenomenon also happens in other datasets.

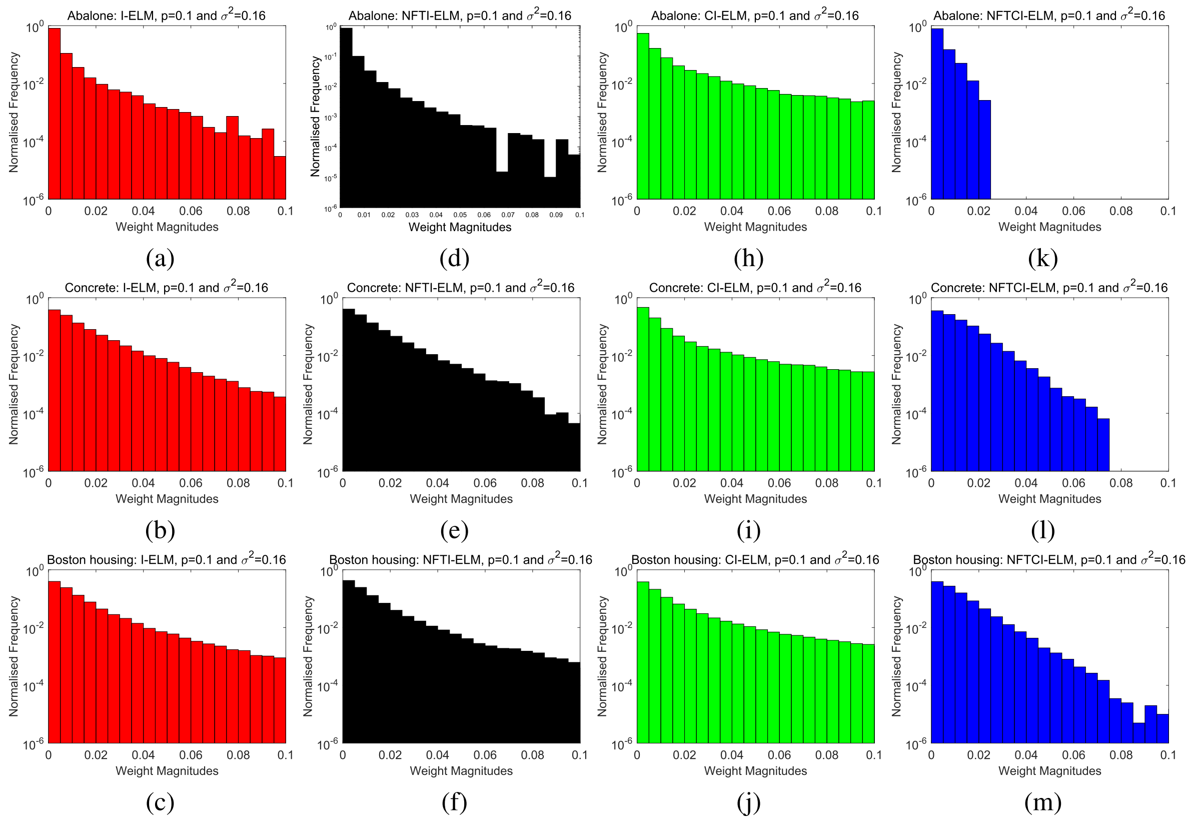

FIGURE 4. **The histogram of β's, which is normalized frequency versus weight magnitude. The fault rate are $p = 0.1$ and $\sigma^2 = 0.16$. The datasets are Abalone, Concrete, boston housing, Wine quality and ASN.**

D. STATISTICAL TEST ANALYSIS

To verify the superiority of our algorithms, we run the statistical test, i.e., paired t-test. The paired t-test illustrates whether the improvements for our algorithms are statistically significant or not. As the fault tolerant performance of the batch mode ELM and CI-ELM are poor, we only run the t-test for the NFTI-ELM against I-ELM and the NFTCI-ELM against I-ELM in Table 4 and Table 5, respectively. For the paired t-test with 400 trials and 95% confidence level, the critical t-value is 1.6486.

In Table 4, all obtained t-values are far beyond the critical t-value, i.e., 1.6486. Also, all confidence intervals of the improvements exclude zero. For instance, in Abalone dataset with the fault level $\{p = 0.01, \sigma^2 = 0.04\}$, the obtained t-value is 346.10792, and the confidence interval is [0.000378, 0.000382]. Similarly, all obtained t-values are much greater than the critical t-value in Table 5. Both paired t-tests conclude that the improvement for our NFTI-ELM and NFTCI-ELM are statistically significant.

E. INCONSISTENCE IN FAULT LEVEL BETWEEN OPERATION AND TRAINING

In our proposed algorithms, we assume that the fault level $\{p, \sigma^2\}$, which is the true fault level during operation, is used for training. However, this true fault level may be unknown in some cases. In other words, the fault level used for training may different from the true fault level during operation. Hence, we investigate the situation that we use an incorrect fault level for training. Fig. 2 and Fig. 3 illustrate the impact of using an incorrect fault level on the performance of our algorithms. Given the true fault level $\{p = 0.05, \sigma^2 = 0.09\}$, the performance of the NFTI-ELM and NFTCI-ELM with different training fault levels, i.e., $\{p = 0.01, \sigma^2 = 0.04\}$, $\{p = 0.05, \sigma^2 = 0.09\}$ and $\{p = 0.1, \sigma^2 = 0.16\}$, are shown in Fig. 2.

Fig. 2(a)-(c) shows the performance of the NFTI-ELM. We notice that even the incorrect fault level for training is used, the performance of the NFTI-ELM is still better than that of the I-ELM. For instance, in Abalone dataset, the MSE values of the NFTI-ELM with three training fault levels, i.e., $\{p = 0.01, \sigma^2 = 0.04\}$, $\{p = 0.05, \sigma^2 = 0.09\}$ and $\{p = 0.1, \sigma^2 = 0.16\}$, are 0.02171, 0.02045 and 0.01963, respectively. Apparently, these MSE values are all smaller than the MSE value of the I-ELM, i.e., 0.02286.

Fig. 2(d)-(f) shows the performance of the NFTCI-ELM. Similarly, the performance of the NFTCI-ELM is much better than that of the CI-ELM when the incorrect fault level for training is used. For instance, in Abalone dataset, the MSE values of the NFTCI-ELM with three training fault levels, i.e., $\{p = 0.01, \sigma^2 = 0.04\}$, $\{p = 0.05, \sigma^2 = 0.09\}$ and $\{p = 0.1, \sigma^2 = 0.16\}$, are 0.008093, 0.007507 and 0.007793, respectively. Again, these MSE values are much smaller than the MSE value of the CI-ELM, i.e., 0.179. Also, it is noticed that the NFTCI-ELM achieves the smallest MSE value, i.e., 0.007507, when the correct training fault level $\{p = 0.05, \sigma^2 = 0.09\}$ is used. This phenomenon also happens in Concrete and Boston Housing datasets.

Moreover, given the fault-free situation, i.e., true fault level $\{p = 0, \sigma^2 = 0\}$, the performance of the NFTI-ELM and NFTCI-ELM with different training fault levels are shown in Fig. 3. Fig. 3(a)-(c) shows the performance of the NFTI-ELM. We notice that even the NFTI-ELM is used in the fault-free situation, the NFTI-ELM performs as well as the I-ELM. For instance, in Abalone dataset, the MSE values of the NFTI-ELM with different training fault levels, i.e., $\{p = 0.01, \sigma^2 = 0.04\}$ and $\{p = 0.05, \sigma^2 = 0.09\}$, are 0.007992 and 0.008222, respectively. These MSE values are similar to that of the I-ELM, i.e., 0.007977. Meanwhile, the NFTCI-ELM has similar phenomenon, as shown in Fig. 3(d)-(f). Therefore, our NFTI-ELM and NFTCI-ELM still perform excellently even though the true fault level is unknown, and the incorrect fault level is used for training.

F. DISTRIBUTION ANALYSIS OF β

The CI-ELM performs poorly under faulty situations. We believe that this behaviour is related to the distribution of the trained output weights β's. When the weights' magnitudes are large, the network output is very sensitive to the weight perturbation, i.e., fault and noise. This results in the poor performance of the CI-ELM under faulty situations.

Fig. 4(a)-(m) shows the histograms of the output weights' magnitudes, which are the normalized frequencies versus the weight magnitudes. The results from the three datasets with the fault level $\{p = 0.1, \sigma^2 = 0.16\}$ are displayed. In Fig. 4(h)-(j), the CI-ELM contains many weights in large magnitudes. Thus, the trained networks of the CI-ELM are sensitive to fault and noise. This behaviour results in poor fault tolerant performance. On the other hand, the NFTCI-ELM has few weights in large magnitudes, as shown in Fig. 4(k)-(m). Hence, the trained networks of the NFTCI-ELM are insensitive to fault and noise.

VI. CONCLUSION

In this paper, we propose two node fault tolerant incremental algorithms for SLFNs, namely NFTI-ELM and NFTCI-ELM. The two proposed algorithms aim to maximize the reduction of the training set MSE of faulty networks between two incremental steps.

For the NFTI-ELM, we train the output weight of the newly inserted node, whereas the weights in other nodes remain unchanged. The simulation results show that the fault tolerant performance of the NFTI-ELM is better than that of other ELM algorithms, including I-ELM, CI-ELM and batch mode ELM. In order to boost the performance, the NFTCI-ELM is then developed. The idea is to train the output weight of the newly inserted node, and to update the previously trained weights. The simulation results show that the performance of the NFTCI-ELM is better than that of the NFTI-ELM.

Meanwhile, the statistical test confirms our NFTI-ELM and NFTCI-ELM have a statistically significant improvement under faulty situations. Since the results show that the NFTCI-ELM is the best among other algorithms, one may argue that we do not need to consider the NFTI-ELM. However, the computational complexity of the NFTI-ELM is lower than that of the NFTCI-ELM. For someone concerning training speed, the NFTI-ELM might be another choice. In general, we believe that the NFTI-ELM and NFTCI-ELM should have certain potential to be applied to the hardware implementation of neural networks in the future.

This paper focuses on regression problems. Our fault tolerant incremental algorithms, as well as the original I-ELM and CI-ELM, aim at minimizing the reduction in MSE between two consecutively incremental steps. As classification problems focus on the recognition error, it may not be appropriate to directly extend our algorithms for classification problems. Hence, one of the future works is to develop fault tolerant incremental algorithms for classification problems.

REFERENCES

[1] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[2] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Netw.*, vol. 4, no. 2, pp. 251–257, 1991.

[3] W. Wei, G. Feng, Z. Li, and Y. Xu, "Deterministic convergence of an online gradient method for BP neural networks," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 533–540, May 2005.

[4] C. Chen and X. Yan, "Optimization of a multilayer neural network by using minimal redundancy maximal relevance-partial mutual information clustering with least square regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 6, pp. 1177–1187, Jun. 2015.

[5] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[6] J. Tang, C. Deng, and G.-B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 809–821, Apr. 2016.

[7] J. Xin, Z. Wang, L. Qu, G. Yu, and Y. Kang, "A-ELM: Adaptive distributed extreme learning machine with MapReduce," *Neurocomputing*, vol. 174, pp. 368–374, Jan. 2016.

[8] X. Liu, S. Lin, J. Fang, and Z. Xu, "Is extreme learning machine feasible? A theoretical assessment (Part I)," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 7–20, Jan. 2015.

[9] S. Lin, X. Liu, J. Fang, and Z. Xu, "Is extreme learning machine feasible? A theoretical assessment (Part II)," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 1, pp. 21–34, Jan. 2015.

[10] C. Pan, D. S. Park, Y. Yang, and H. M. Yoo, "Leukocyte image segmentation by visual attention and extreme learning machine," *Neural Comput. Appl.*, vol. 21, no. 6, pp. 1217–1227, 2012.

[11] R. Minhas, A. Baradarani, S. Seifzadeh, and Q. M. J. Wu, "Human action recognition using extreme learning machine based on visual vocabularies," *Neurocomputing*, vol. 73, nos. 10–12, pp. 1906–1917, 2010.

[12] Q. Wang, Y. Dou, X. Liu, Q. Lv, and S. Li, "Multi-view clustering with extreme learning machine," *Neurocomputing*, vol. 214, pp. 483–494, Nov. 2016.

[13] P. Hu, D. Peng, Y. Sang, and Y. Xiang, "Multi-view linear discriminant analysis network," *IEEE Trans. Image Process.*, vol. 28, no. 11, pp. 5352–5365, Nov. 2019.

[14] C. K. L. Lekamalage, T. Liu, Y. Yang, Z. Lin, and G.-B. Huang, "Extreme learning machine for clustering," in *Proc. ELM*, Cham, Switzerland, Springer, vol. 1, 2015, pp. 435–444.

[15] X. Peng, J. Feng, S. Xiao, W.-Y. Yau, J. T. Zhou, and S. Yang, "Structured autoencoders for subspace clustering," *IEEE Trans. Image Process.*, vol. 27, no. 10, pp. 5076–5086, Oct. 2018.

[16] E. Cambria *et al.*, "Extreme learning machines [trends & controversies]," *IEEE Intell. Syst.*, vol. 28, no. 6, pp. 30–59, Nov./Dec. 2013.

[17] 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, 2015.

[18] X. Li, W. Mao, and W. Jiang, "Extreme learning machine based transfer learning for data classification," *Neurocomputing*, vol. 174, no. 1, pp. 203–210, 2016.

[19] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, nos. 16–18, pp. 3056–3062, 2007.

[20] G.-B. Huang, Z. Bai, L. L. C. Kasun, and C. M. Vong, "Local receptive fields based extreme learning machine," *IEEE Comput. Intell. Mag.*, vol. 10, no. 2, pp. 18–29, May 2015.

[21] R. A. Nawrocki and R. M. Voyles, "Artificial neural network performance degradation under network damage: Stuck-at faults," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, San Jose, CA, USA, Jul. 2011, pp. 442–449.

[22] C.-S. Leung, W. Y. Wan, and R. Feng, "A regularizer approach for RBF networks under the concurrent weight failure situation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1360–1372, Jun. 2017.

[23] J. B. Burr, "Digital neural network implementations," in *Neural Networks, Concepts, Applications, and Implementations*, vol. 3. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995, pp. 237–285.

[24] V. Piuri, M. Sami, and R. Stefanelli, "Fault tolerance in neural networks: Theoretical analysis and simulation results," in *Proc. 5th Annu. Eur. Comput. Conf., Adv. Comput. Technol., Rel. Syst. Appl.*, May 1991, pp. 429–436.

[25] F. M. Dias and A. Antunes, "Fault tolerance of artificial neural networks: An open discussion for a global model," *Int. J. Circuits, Syst. Signal Process.*, vol. 4, no. 1, pp. 9–16, Aug. 2010.

[26] R. Eickhoff and U. Rückert, "Tolerance of radial basis functions against stuck-at-faults," in *Artificial Neural Networks: Formal Models and Their Applications—ICANN* (Lecture Notes in Computer Science), vol. 3697, W. Duch, J. Kacprzyk, E. Oja, and S. Zadrożny, Eds. Berlin, Germany: Springer, 2005, pp. 1003–1008.

[27] R. Eickhoff and U. Rückert, "Robustness of radial basis functions," *Neurocomputing*, vol. 70, nos. 16–18, pp. 2758–2767, Oct. 2007.

[28] K. I.-J. Ho, C.-S. Leung, and J. Sum, "Convergence and objective functions of some fault/noise-injection-based online learning algorithms for RBF networks," *IEEE Trans. Neural Netw.*, vol. 21, no. 6, pp. 938–947, Jun. 2010.

[29] B. Liu and T. Kaneko, "Error analysis of digital filters realized with floating-point arithmetic," *Proc. IEEE*, vol. 57, no. 10, pp. 1735–1747, Oct. 1969.

[30] J. L. Holi and J.-N. Hwang, "Finite precision error analysis of neural network hardware implementations," *IEEE Trans. Comput.*, vol. 42, no. 3, pp. 281–290, Mar. 1993.

[31] G. Carvajal and M. Figueroa, "Model, analysis, and evaluation of the effects of analog VLSI arithmetic on linear subspace-based image recognition," *Neural Netw.*, vol. 55, pp. 72–82, Jul. 2014.

[32] H. R. Mahdiani, S. M. Fakhraie, and C. Lucas, "Relaxed fault-tolerant hardware implementation of neural networks in the presence of multiple transient errors," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1215–1228, Aug. 2012.

[33] A. P. Piotrowski, P. M. Rowinski, and J. J. Napiorkowski, "Comparison of evolutionary computation techniques for noise injected neural network training to estimate longitudinal dispersion coefficients in rivers," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 1354–1361, 2012.

[34] T. Cho, K. Katahira, K. Okanoya, and M. Okada, "Node perturbation learning without noiseless baseline," *Neural Netw.*, vol. 24, no. 3, pp. 267–272, 2011.

[35] S. U. Ahmed, M. Shahjahan, and K. Murase, "Injecting chaos in feedforward neural networks," *Neural Process. Lett.*, vol. 34, no. 1, pp. 87–100, 2011.

[36] O. Osoba and B. Kosko, "Noise-enhanced clustering and competitive learning algorithms," *Neural Netw.*, vol. 37, pp. 132–140, Jan. 2013.

[37] D. Deodhare, M. Vidyasagar, and S. S. Keethi, "Synthesis of fault-tolerant feedforward neural networks using minimax optimization," *IEEE Trans. Neural Netw.*, vol. 9, no. 5, pp. 891–900, Sep. 1998.

[38] C. Neti, M. H. Schneider, and E. D. Young, "Maximally fault-tolerant neural networks and nonlinear programming," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 2, Jun. 1990, pp. 483–496.

[39] T. Horita and I. Takanami, "An FPGA-based multiple-weight-and-neuron-fault tolerant digital multilayer perceptron," *Neurocomputing*, vol. 99, pp. 570–574, Jan. 2013.

[40] J. L. Bernier, J. Ortega, E. Ros, I. Rojas, and A. Prieto, "A quantitative study of fault tolerance, noise immunity, and generalization ability of MLPs," *Neural Comput.*, vol. 12, no. 12, pp. 2941–2964, 2000.

[41] M. Conti, S. Orcioni, and C. Turchetti, "Training neural networks to be insensitive to weight random variations," *Neural Netw.*, vol. 13, no. 1, pp. 125–132, 2000.

[42] Y. Xiao, R.-B. Feng, C.-S. Leung, and J. Sum, "Objective function and learning algorithm for the general node fault situation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 863–874, Apr. 2016.

[43] H.-C. Leung, C.-S. Leung, and E. W. M. Wong, "Fault-tolerant incremental learning for extreme learning machines," in *Proc. 23rd Int. Conf. Neural Inf. Process. (ICONIP)*, A. Hirose, S. Ozawa, K. Doya, K. Ikeda, M. Lee, and D. Liu, Eds. Kyoto, Japan: Springer, 2016, pp. 168–176.

[44] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Trans. Inf. Theory*, vol. 39, no. 3, pp. 930–945, May 1993.

[45] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," *Neural Netw.*, vol. 2, pp. 985–990, Jul. 2004.

[46] J. L. Bernier, J. Ortega, I. Rojas, E. Ros, and A. Prieto, "Obtaining fault tolerant multilayer perceptrons using an explicit regularization," *Neural Process. Lett.*, vol. 12, no. 2, pp. 107–113, Oct. 2000.

[47] A. Hashmi, H. Berry, O. Temam, and M. Lipasti, "Automatic abstraction and fault tolerance in cortical microachitectures," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 3, pp. 1–10, Jun. 2011.

[48] M. Lichman. (2013). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml

[49] M. Sugiyama and H. Ogawa, "Optimal design of regularization term and regularization parameter by subspace information criterion," *Neural Netw.*, vol. 15, no. 3, pp. 349–361, 2002.

[50] D. L. Ly and H. Lipson, "Optimal experiment design for coevolutionary active learning," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 394–404, Jun. 2014.

[51] Q. Zhang, X. Hu, and B. Zhang, "Comparison of ℓ_1-norm SVR and sparse coding algorithms for linear regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 8, pp. 1828–1833, Aug. 2015.

[52] R. Ekambaram, S. Fefilatyev, M. Shreve, K. Kramer, L. O. Hall, D. B. Goldgof, and R. Kasturi, "Active cleaning of label noise," *Pattern Recognit.*, vol. 51, pp. 463–480, Mar. 2016.

[53] S. Li, Z.-H. You, H. Guo, X. Luo, and Z.-Q. Zhao, "Inverse-free extreme learning machine with optimal information updating," *IEEE Trans. Cybern.*, vol. 46, no. 5, pp. 1229–1241, May 2016.

[54] S. Wager, T. Hastie, and B. Efron, "Confidence intervals for random forests: The jackknife and the infinitesimal jackknife," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1625–1651, May 2014.

[55] M. Graczyk, T. Lasota, Z. Telec, and B. Trawiński, "Nonparametric statistical analysis of machine learning algorithms for regression problems," in *Knowledge-Based and Intelligent Information and Engineering Systems KES* (Lecture Notes in Computer Science), vol 6276, R. Setchi, I. Jordanov, R. J. Howlett, and L. C. Jain, Eds. Berlin, Germany: Springer, 2010, pp. 111–120.

[56] M. A. Little, P. E. McSharry, E. J. Hunter, J. Spielman, and L. O. Ramig, "Suitability of dysphonia measurements for telemonitoring of Parkinson's disease," *IEEE Trans. Bio-Med. Eng.*, vol. 56, no. 4, pp. 1015–1022, Apr. 2009.

[57] I. A. Hodashinsky, K. S. Sarin, and D. D. Zykov, "Takagi-Sugeno fuzzy systems structure identification based on piecewise linear initialization," in *Proc. Int. Siberian Conf. Control Commun. (SIBCON)*, May 2015, pp. 1–4.

HO CHUN LEUNG received the B.Eng. degree in information engineering from the City University of Hong Kong, Hong Kong, in 2015, where he is currently pursuing the Ph.D. degree with the Department of Electronic Engineering. His research interests include machine learning, neural networks, computer graphic, and telecommunication.

CHI SING LEUNG (M'05–SM'15) received the Ph.D. degree in computer science from The Chinese University of Hong Kong, in 1995. He is currently a Professor with the Department of Electronic Engineering, City University of Hong Kong. He has published over 120 journal articles in the areas of digital signal processing, neural networks, and computer graphics. His research interests include neural computing and computer graphics. He was a member of Organizing Committee of ICONIP2006. He is a governing board member and the Vice President of the Asian Pacific Neural Network Assembly (APNNA). In 2005, he received the 2005 IEEE Transactions on Multimedia Prize Paper Award for his article titled The Plenoptic Illumination Function. He was the Program Chair of ICONIP2009 and ICONIP2012. He is/was a Guest Editor of several journals, including *Neural Computing and Applications*, *Neurocomputing*, and *Neural Processing Letters*.

ERIC WING MING WONG (S'87–M'90–SM'00) received the B.Sc. and M.Phil. degrees in electronic engineering from the Chinese University of Hong Kong, Hong Kong, in 1988 and 1990, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Massachusetts, Amherst, MA, USA, in 1994. He is currently an Associate Professor with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong. His research interests include analysis and design of telecommunications and computer networks, energy-efficient data center design, green cellular networks, and optical networking.