Journal of Ambient Intelligence and Humanized Computing (2023) 14:15643–15657
https://doi.org/10.1007/s12652-019-01488-8

ORIGINAL RESEARCH

Two noise tolerant incremental learning algorithms for single layer feed-forward neural networks

Muideen Adegoke[1] · Hiu Tung Wong[1] · Andrew Chi Sing Leung[1] · John Sum[2]

Received: 30 January 2019 / Accepted: 8 September 2019 / Published online: 19 September 2019

Abstract
This paper focuses on noise resistant incremental learning algorithms for single layer feed-forward neural networks (SLFNNs). In a physical implementation of a well trained neural network, faults or noise are unavoidable. As biological neural networks have ability to tolerate noise, we would like to have a trained neural network that has certain ability to tolerate noise too. This paper first develops a noise tolerant objective function that can handle multiplicative weight noise. We assume that multiplicative weight noise exist in the weights between the input layer and the hidden layer, and in the weights between the hidden layer and the output layer. Based on the developed objective function, we propose two noise tolerant incremental extreme learning machine algorithms, namely weight deviation incremental extreme learning machine (WDT-IELM) and weight deviation convex incremental extreme learning machine (WDTC-IELM). Compared to the original extreme learning machine algorithms, the two proposed algorithms have much better ability to tolerate the multiplicative weight noise. Several simulations are carried out to demonstrate the superiority of the two proposed algorithms.

Keywords Weight noise · Extreme learning machine · Neural network · Fault tolerance

1 Introduction

Extreme learning machine (ELM) algorithms (Huang et al. 2006; Huang and Chen 2007) provide a low computation solution for constructing a single layer feed-forward neural network (SLFNN). In the ELM concept, the input connection weights between the input and hidden layers are generated randomly. Hence we only need to train the output connection weights between the hidden and output layers. Although the ELM concept uses the random node concept, a SLFNN trained by the ELM concept still has the universal approximation ability (Barron 1993; Hornik et al. 1989; Hornik 1991). In the last several years, many applications of using ELM were reported. For example, a modified ELM model for imbalance data was reported (Li et al. 2018). Also, some works of using the ELM concept to handle biological data were reported (Bi et al. 2018; Wang et al. 2017).

There are two kinds of ELM algorithms. One is batch mode, in which we first generate a number of hidden nodes and then we estimate all the output weights at a time. Another one is incremental mode, in which we add the hidden nodes one-by-one into the network until the predefined stopping condition reaches. The incremental ELM (IELM) and the convex IELM (CIELM) (Huang et al. 2006; Huang and Chen 2007) are two representative incremental ELM algorithms with simple update rules. Although many ELM algorithms have been developed, few of them have the ability to tolerate network fault and noise.

In hardware implementation of neural networks, noise or faults are prone to occur. For instance when a neural network is implemented on the field-programmable gate array (FPGA) technology, we may use a low precision floating point format to represent connection weights. The roundoff error of using the floating point format can be modelled as multiplicative noise (Liu and Kaneko 1969). In an analog implementation, thermal noise and drifts always exist in

✉ Andrew Chi Sing Leung
eeleungc@cityu.edu.hk

Muideen Adegoke
maadegoke2-c@my.cityu.edu.hk

Hiu Tung Wong
htwong39-c@my.cityu.edu.hk

John Sum
pfsum@dragon.nchu.edu.tw

[1] City University of Hong Kong, Hong Kong, China

[2] National Chung Hsing University, Taichung, Taiwan

operational amplifiers. Besides, the precision of analog devices, such as resistors, are in terms of percentage error. In addition, implementing a trained network using a nano-scale device, during the operation transient noise or failure may be introduced (Pajarinen et al. 2011; Mahdiani et al. 2012). Traditional learning algorithms have poor fault or noise tolerant performance. Since biological neural networks have certain ability to tolerate noise, we would like to train neural network that has certain noise tolerant too.

To handle noise and fault, it is essential to understand how they affect the behaviour or performance of a trained network. Noise or fault tolerant learning algorithms aim at training a network to attain acceptable performance even under noise and fault situations. A survey of various kinds of imperfect conditions in the traditional network model, such as radial basis function (RBF) networks, was reported in Martolia et al. (2015). Besides, failure tolerant ability of RBF networks was extensively studied (Leung et al. 2010; Feng et al. 2017; Murakami and Honda 2007). However, few results about failure tolerant ability of ELM networks were studied.

This paper investigates the noise tolerant performance of the SLFNN model trained by the incremental ELM concept. We consider that multiplicative weight noise exist between the input and hidden layers, and between the hidden and output layers. Firstly, a noise tolerant training objective function for SLFNNs is formulated. Afterwards, two incremental ELM algorithms, namely weight deviation incremental extreme learning machine (WDT-IELM) and weight deviation convex incremental extreme learning machine (WDTC-IELM), are derived.

In the WDT-IELM algorithm, the hidden nodes are added into the existing network incrementally in the one-by-one manner. After adding a new hidden node, all the previous trained output weights are not modified .

In the WDTC-IELM algorithm, we use a strategy similar to WDT-IELM to create a SLFNN, but we use a simple updating rule to modify the previous trained output weights.

We show that for the two proposed ELM algorithms, the training objective values are non-increasing at each training iteration. We use several simulations, operated on several commonly used datasets, to validate the superiority of the two proposed algorithms. Compared to the original incremental ELM algorithms, the two proposed algorithms have much better ability to tolerate the multiplicative weight noise. In addition, we perform paired-t tests to show that the improvement of using the proposed algorithms is statistical significant.

The rest of the paper is organized as follows. The backgrounds on ELM are given in Sect. 2. In Sect. 3, weight noise models are presented and the noise tolerant objective function is derived. Sect. 4 presents the two proposed algorithms. In addition, in this section, we show that during training, the objective values are non-increasing. The simulation results are presented in Sect. 5. The paper is ended with conclusion in Sect. 6.

2 Mathematical background on extreme learning machine

The standard ELM was developed to train SLFNNs (Huang et al. 2006; Huang and Chen 2007; Huang et al. 2006). In a SLFNN, there are three layers, namely, input, hidden and output layers. In the ELM concept, the input weights in between the input layer and the hidden layer are randomly generated. They do not need to be learned or tuned. Only the output weights in between the hidden layer and the output layer nodes are required to be trained. Hence, during learning, the computational cost is not prohibitive and is much lower than that of other traditional learning algorithms such as gradient descent (Guély and Siarry 1993).

This paper considers to use the ELM concept to solve the nonlinear regression problem. Let $\mathbb{D}_{train} = \{(\boldsymbol{x_k}, t_k) : \boldsymbol{x}_k \in \mathbb{R}^D, t_k \in R, \quad k = 1, \ldots, N\}$ be the training set, where D is the number of input features, N is the number of training samples, and $\boldsymbol{x_k}$ and t_k are the inputs and target output of the k-th sample, respectively. Similarly, the test set is denoted as $\mathbb{D}_{test} = \{\boldsymbol{x}'_{k'}, t'_{k'}) : \boldsymbol{x}'_{k'} \in \mathbb{R}^{D'}, t'_{k'} \in R, \quad k' = 1, \ldots, N'\}$, where N' is the number of samples in the test set.

The output of a SLFNN with m hidden nodes is equal to

$$f_m(\boldsymbol{x}) = \sum_{j=1}^{m} \beta_j h_j(\boldsymbol{x}), \tag{1}$$

where β_j is the jth output weight, and $h_j(\boldsymbol{x})$ is the output of the jth hidden node. There are several possible activation functions, for instance, in the case of the sigmoid activation function, $h_j(\cdot)$ is given by

$$h_j(\boldsymbol{x}) = \frac{1}{1 + \exp -(\boldsymbol{a}_j^{\mathrm{T}} \boldsymbol{x} + b_j)}, \tag{2}$$

where $\boldsymbol{a}_j$ and b_j are the input weights and input bias, respectively, of the jth hidden node. Grouping $\boldsymbol{a}_j$ and b_j together, (3) can be rewritten as

$$h_j(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-\boldsymbol{w}_j^{\mathrm{T}} \boldsymbol{o}\right)}, \tag{3}$$

where $\boldsymbol{w}_j = [\boldsymbol{a}_j^{\mathrm{T}}, b_j]^{\mathrm{T}}$, and $\boldsymbol{o} = [\boldsymbol{x}_k^{\mathrm{T}}, 1]^{\mathrm{T}}$. In the ELM concept, the input weight vectors, $\boldsymbol{w}_j$'s, are generated randomly.

Given all training samples, the training set error is equal to

$$\begin{aligned}\varepsilon_m &= \sum_{k=1}^{N}(t_k - f_m(\boldsymbol{x}_k))^2 \\ &= \sum_{k=1}^{N}(t_k - \sum_{j=1}^{m}\beta_j h_j(\boldsymbol{x}_k))^2.\end{aligned} \tag{4}$$

Let $\boldsymbol{t}$ be the collection of the target outputs, given by

$$\boldsymbol{t} = [t_1, \ldots, t_N]^{\mathrm{T}},$$

and let $\boldsymbol{h}_j$ be the collection of the j hidden node outputs of all input samples, given by

$$\boldsymbol{h}_j = [h_j(\boldsymbol{x}_1), \ldots, h_j(\boldsymbol{x}_N)]^{\mathrm{T}}.$$

Equation (4) can be written in a compact form, given by

$$\varepsilon_m = \|\boldsymbol{t} - \sum_{j=1}^{m}\beta_j \boldsymbol{h}_j\|_2^2 = \|\boldsymbol{t} - \boldsymbol{H}_m\boldsymbol{\beta}_m\|_2^2, \tag{5}$$

where $\boldsymbol{\beta}_m = [\beta_1, \ldots, \beta_m]^{\mathrm{T}}$, and

$$\boldsymbol{H}_m = [\boldsymbol{h}_1, \ldots, \boldsymbol{h}_m] = \begin{pmatrix} h_1(\boldsymbol{x}_1) & \cdots & h_m(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\boldsymbol{x}_N) & \cdots & h_m(\boldsymbol{x}_N) \end{pmatrix}. \tag{6}$$

As aforementioned, the input weights $\boldsymbol{w}_j$'s are randomly generated. During training, we do not need to adjust them. The ELM concept uses the least square approach to obtain the output weights. In the batch mode ELM, given m hidden nodes, the optimal output weight vector that minimizes the training set error is given by

$$\boldsymbol{\beta}_m^* = \left(\boldsymbol{H}_m^{\mathrm{T}}\boldsymbol{H}_m\right)^{-1}\boldsymbol{H}_m^{\mathrm{T}}\boldsymbol{t}. \tag{7}$$

Instead of using the batch mode to find out the weights, the ELM concept has the incremental mode, in which we incrementally add hidden nodes into the existing network until the stopping condition reaches. When we insert a new hidden node, we need to determine its output weight only. The IELM and CIELM are two incremental ELM algorithms. In the IELM algorithm, after inserting a new hidden node, all the previous trained output weights are unchanged. The difference between the two algorithms is that the CIELM algorithm uses a simple updating rule to modify all the existing output weights. Although the ELM concept can simplify the creation process of a SLFNN, few ELM algorithms have ability to tolerate the noise situation. In the rest of the paper, we will first define a noise tolerant objective function, and then develop the noise tolerant versions of IELM and CIELM.

3 Weight noise model and objective function

In this section, we consider that a SLFNN is affected by multiplicative weight noise in the input weight vectors $\boldsymbol{w}_j$'s and the output weights β_j's. We will first describe the noise model and then develop a noise tolerant objective function.

3.1 Weight noise model

When we implement a trained network, weight noise is prone to occur. Weight noise can be regarded as the deviation from the nominal value of the weight of a well trained neural network. For instance, after training a neural network, we may implement the trained network on hardware such as FPGA. To do this, we may use a low precision floating point format to represent connection weights. The roundoff error of using the floating point format can cause an implemented weight to deviate from its nominal value. Also, in the analog implementation, noise are unavoidable. One of commonly used noise is multiplicative weight noise (Burr 1991; Liu and Kaneko 1969). In this model, the difference between the implemented weight value and its nominate value is proportional to its nominate value.

Let w_{jl} be the original value of the lth element in the jth input weight vector. Under multiplicative noise model, the deviation of an input weight w_{jl} from its nominal value is given by

$$\delta_{jl} = \upsilon_{jl} w_{jl}, \quad \forall\, j, l, \tag{8}$$

where υ_{jl}'s are independent and identically distributed (iid) random variables (RVs). Their mean is equal to zero and variance is equal to σ_w^2. In other words, the implemented value of an input weight is given by

$$\tilde{w}_{jl} = w_{kl} + \delta_{kl} = (1 + \upsilon_{jl})w_{jl}. \tag{9}$$

In (9), the magnitude of the noise component δ_{jl} is proportional to that of the nominate value w_{jl}.

Given the deviations δ_{jl}, for all $l = 1, \ldots, D+1$, of the input weights for the jth hidden node, the hidden node output is

$$\tilde{h}_j(\boldsymbol{x}) = \frac{1}{1 + \exp\left(-\tilde{\boldsymbol{w}}_j^{\mathrm{T}}\boldsymbol{o}\right)}. \tag{10}$$

Note that $\boldsymbol{o} = [\boldsymbol{x}^{\mathrm{T}}, 1]^{\mathrm{T}}$ and $\tilde{\boldsymbol{w}} = [\tilde{w}_{j1}, \ldots, \tilde{w}_{j(D+1)}]^{\mathrm{T}}$, where D is the number of input features of the neural network. We can use the first order Taylor series to expand (10), given by

$$\begin{aligned}\tilde{h}_j(\boldsymbol{x}) =& h_j(\boldsymbol{x}) + \sum_{l=1}^{D+1} \delta_{jl} \frac{\partial h_j(\boldsymbol{x})}{\partial w_{jl}} \\ =& h_j(\boldsymbol{x}) + \sum_{l=1}^{D+1} \upsilon_{jl} w_{jl} \frac{\partial h_j(\boldsymbol{x})}{\partial w_{jl}} \\ =& h_j(\boldsymbol{x}) + \sum_{l=1}^{D+1} \upsilon_{jl} w_{jl} \Delta H_{jl}(\boldsymbol{x}),\end{aligned} \tag{11}$$

where

$$\Delta H_{jl}(\boldsymbol{x}) = \frac{\partial h_j(\boldsymbol{x})}{\partial w_{jl}}. \tag{12}$$

When the sigmoid function is used as the activation function,

$$\Delta H_{jl}(\boldsymbol{x}) = o_l h_j(\boldsymbol{x})(1 - h_j(\boldsymbol{x})). \tag{13}$$

Similarly, under the multiplicative noise, the value of an output weight becomes

$$\tilde{\beta}_j = (1 + \zeta_j)\beta_j \tag{14}$$

where ζ_j's are iid RVs. Their mean is equal to zero and variance is equal to σ_β^2.

Hence for a given input pattern $\boldsymbol{x}$, the weighted output of a noisy hidden node is given by

$$\tilde{\beta}_j \tilde{h}_j(\boldsymbol{x}) = (1+\zeta_j)\beta_j \left(h_j(\boldsymbol{x}) + \sum_{l=1}^{D+1} \upsilon_{jl} w_{jl} \Delta H_{jl}(\boldsymbol{x}) \right). \tag{15}$$

Since ζ_j's and υ_{jl}'s are iid RVs and have zero mean, the expected values of the weighted outputs are given by

$$\left\langle \tilde{\beta}_j \tilde{h}_j(\boldsymbol{x}) \right\rangle = (1 + 0)\beta_j(h_j(\boldsymbol{x}) + 0) = \beta_j h_j(\boldsymbol{x}), \tag{16}$$

where $\langle \cdot \rangle$ is the expectation operator. Also, ζ_j's and υ_{jl}'s are with variances equal to σ_w^2 and σ_β^2, respectively. Hence the expected squares of the weighted outputs are given by

$$\left\langle \tilde{\beta}_j^2 \tilde{h}_j^2(\boldsymbol{x}) \right\rangle = \left(1+\sigma_\beta^2\right)\beta_j^2 \left(h_j^2(\boldsymbol{x}) + \sigma_w^2 \sum_{l=1}^{D+1} w_{jl}^2 \Delta H_{jl}^2(\boldsymbol{x}) \right). \tag{17}$$

Furthermore, for $j \neq j'$, we have

$$\left\langle \tilde{\beta}_j \tilde{h}_j(\boldsymbol{x}) \tilde{\beta}_{j'} \tilde{h}_{j'}(\boldsymbol{x}) \right\rangle = \beta_j h_j(\boldsymbol{x}) \beta_{j'} h_{j'}(\boldsymbol{x}). \tag{18}$$

3.2 Noise tolerant objective function

Traditional ELM algorithms, for regression, minimize the square error between the network output and target output. In the noise situation, we propose to minimize the expected error over all noise patterns. For a particular noise pattern, the training error set of a noisy network is given by

$$\tilde{\varepsilon}_m = \sum_{k=1}^{N} (t_k - \sum_{j=1}^{m} \tilde{\beta}_j \tilde{h}_j(\boldsymbol{x}_k))^2. \tag{19}$$

From (16)–(18), the expected error over all noise patterns is given by

$$\begin{aligned} J_m =& \langle \tilde{\varepsilon}_m \rangle \\ =& \left\langle \sum_{k=1}^{N} (t_k - \sum_{j=1}^{m} \tilde{\beta}_j \tilde{h}_j(\boldsymbol{x}_k))^2 \right\rangle \\ =& \sum_{k=1}^{N} (t_k - \sum_{j=1}^{m} \beta_j h_j(\boldsymbol{x}_k))^2 \\ &+ \sigma_\beta^2 \sum_{k=1}^{N} \sum_{j=1}^{m} \beta_j^2 h_j^2(\boldsymbol{x}_k) \\ &+ (1 + \sigma_\beta^2) \sum_{k=1}^{N} \sum_{j=1}^{m} \beta_j^2 \sum_{l=1}^{D+1} \sigma_w^2 w_{jl}^2 \Delta_{jl}^2(\boldsymbol{x}_k) \end{aligned} \tag{20}$$

$$= \|\boldsymbol{e}_m\|_2^2 + \sigma_\beta^2 \kappa_m + (1 + \sigma_\beta^2)\sigma_w^2 \tau_m, \tag{21}$$

where

$$\boldsymbol{e}_m = \boldsymbol{t} - \boldsymbol{f}_m \tag{22}$$

$$\boldsymbol{f}_m = \sum_{j=1}^{m} \beta_j \boldsymbol{h}_j \tag{23}$$

$$\boldsymbol{h}_j = [\boldsymbol{h}_j(\boldsymbol{x}_1), \ldots, \boldsymbol{h}_j(\boldsymbol{x}_N)]^{\mathrm{T}} \tag{24}$$

$$\kappa_m = \sum_{j=1}^{m} \beta_j^2 \|\boldsymbol{h}_j\|_2^2 \tag{25}$$

$$\tau_m = \sum_{k=1}^{N} \sum_{j=1}^{m} \beta_j^2 \sum_{l=1}^{D+1} \sigma_w^2 w_{jl}^2 \Delta H_{jl}^2(\boldsymbol{x}_k). \tag{26}$$

In (21), the expected error of a SLFFN contains three terms. The first term $\|\boldsymbol{e}_m\|_2^2$ is the error of a noiseless SLFFN. The second term is the degradation from the noise in the output weights β_j's. The third term is the the degradation from the noise in the input weights w_{jl}'s. In the ELM concept, the input weights w_{jl} are randomly generated, and their values are then fixed. Only the output weights are required to be trained.

4 Noise tolerant incremental algorithms

This section presents the two incremental ELM algorithms, namely WDT-IELM and WDTC-IELM, for training SLFNNs. The WDTC-IELM algorithm use the same strategy to add hidden nodes into the network, but we use a simple updating rule to modify the existing output weights.

4.1 WDT-IELM

The WDT-IELM algorithm is a noise tolerant version of the original IELM. The WDT-IELM algorithm incrementally adds hidden nodes one-by-one into the network. Suppose that a SLFFN already has $m-1$ hidden nodes. The incremental strategy is that we determine the value of the output weight β_m of the newly inserted node and do not modify the existing output weights $\{\beta_1, \ldots, \beta_{m-1}\}$. According to that strategy, we have the following recursive relationships for $\boldsymbol{f}_m$, κ_m, and τ_m, given by

$$\boldsymbol{f}_m = \boldsymbol{f}_{m-1} + \beta_m \boldsymbol{h}_m \tag{27}$$

$$\kappa_m = \kappa_{m-1} + \beta_m^2 \|\boldsymbol{h}_m\|_2^2 \tag{28}$$

$$\tau_m = \tau_{m-1} + \beta_m^2 \sum_{k=1}^{N} \sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k). \tag{29}$$

With (27), the recursive equation for the error vector is given by

$$\boldsymbol{e}_m = \boldsymbol{e}_{m-1} - \beta_m \boldsymbol{h}_m. \tag{30}$$

Based on (21), and (27)–(30), J_m can be expressed as

$$\begin{aligned} J_m =& J_{m-1} - 2\beta_m \boldsymbol{e}_{m-1}^{\mathrm{T}} \boldsymbol{h}_m + (1+\sigma_\beta^2)\beta_m^2 \|\boldsymbol{h}_m\|_2^2 \\ &+ (1+\sigma_\beta^2)\sigma_w^2 \beta_m^2 \sum_{k=1}^{N} \sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k). \end{aligned} \tag{31}$$

Let

$$\begin{aligned} R_m =& J_m - J_{m-1} \\ =& -2\beta_m \boldsymbol{e}_{m-1}^{\mathrm{T}} \boldsymbol{h}_m + (1+\sigma_\beta^2)\beta_m^2 \|\boldsymbol{h}_m\|_2^2 \\ &+ (1+\sigma_\beta^2)\sigma_w^2 \beta_m^2 \sum_{k=1}^{N} \sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k). \end{aligned} \tag{32}$$

To maximize the reduction in the expected error over all noise patterns, we should consider

$$\frac{\partial R_m}{\partial \beta_m} = 0. \tag{33}$$

Then the optimal value of β_m is given by

$$\beta_m^* = \frac{\boldsymbol{e}_{m-1}^{\mathrm{T}} \boldsymbol{h}_m}{(1+\sigma_\beta^2)(\|\boldsymbol{h}_m\|_2^2 + \sigma_w^2 \nu_m)}, \tag{34}$$

where

$$\nu_m = \sum_{k=1}^{N} \sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k). \tag{35}$$

With this optimal value, R_m is given by

$$R_m = -\frac{(\boldsymbol{e}_{m-1}^{\mathrm{T}} \boldsymbol{h}_m)^2}{\left((1+\sigma_\beta^2)(\|\boldsymbol{h}_m\|_2^2 + \sigma_w^2 \nu_m)\right)^2}. \tag{36}$$

Apparently, the expected training error of the noisy network always reduces. The summary of the WDT-IELM algorithm is given in Algorithm 1. It should be noticed during training, we do not need to keep τ_m and κ_m.

Algorithm 1 The updating procedure of WDT-IELM

1: Initialization: $\boldsymbol{t} = [t_1, \cdots, t_N]^{\mathrm{T}}$, $m = 0$, $\boldsymbol{e}_0 = \boldsymbol{0}$, and $\boldsymbol{f}_0 = \boldsymbol{0}$.
2: **while** $m \le m_{\max}$ **do**
3: $m = m + 1$.
4: Generate a hidden node whose input weight vector $\boldsymbol{w}_m$ randomly generated.
5: For this new hidden node, computes its output values $h_m(\boldsymbol{x}_k)$'s for all training sample.
6: Form a vector $\boldsymbol{h}_m = [h_m(\boldsymbol{x}_1), \cdots, h_m(\boldsymbol{x}_N)]^{\mathrm{T}}$.
7: Compute $\nu_m = \sum_{k=1}^{N} \sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k)$.
8: Compute the output weight of the newly inserted node: $\beta_m = \frac{\boldsymbol{e}_{m-1}^{\mathrm{T}} \boldsymbol{h}_n}{(1+\sigma_\beta^2)(\|\boldsymbol{h}_m\|_2^2 + \sigma_w^2 \nu_m)}$.
9: $\boldsymbol{f}_m = \boldsymbol{f}_{m-1} + \beta_m \boldsymbol{h}_m$.
10: $\boldsymbol{e}_m = \boldsymbol{t} - \boldsymbol{f}_n$.
11: **end while**

4.2 WDTC-IELM

In Huang and Chen (2007), the CIELM algorithm was presented. It aims at improving the approximation ability of IELM. The performance demonstration in Huang and Chen (2007) showed that the CIELM algorithm outperforms the original IELM algorithm. However, it was designed for the noiseless situation. Hence, it is equally important to develop a noise tolerant version of CIELM.

In the proposed WDTC-IELM algorithm, we also incrementally insert hidden nodes into the network in the one-by-one manner. After determine the current output weight β_m, we update all the previous trained weights, given by

$$\boldsymbol{\beta}_j^{new} = (1 - \boldsymbol{\beta}_m)\boldsymbol{\beta}_j^{old} \qquad j = 1, 2, \ldots, m-1. \tag{37}$$

With (37), the recursive equations for $\boldsymbol{f}_m$, $\boldsymbol{e}_m$, κ_m, and τ_m becomes

$$\boldsymbol{f}_m = (1-\beta_m)\boldsymbol{f}_{m-1} + \beta_m \boldsymbol{h}_m \tag{38}$$

$$\boldsymbol{e}_m = \boldsymbol{t} - \boldsymbol{f}_m = \boldsymbol{e}_{m-1} + \beta_m(\boldsymbol{f}_{m-1} - \boldsymbol{h}_m) \tag{39}$$

$$\kappa_m = (1-\beta_m)^2\kappa_{m-1} + \beta_m^2\|\boldsymbol{h}_m\|_2^2 \tag{40}$$

$$\tau_m = (1-\beta_m)^2\tau_{m-1} + \beta_m^2 \sum_{k=1}^{N}\sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k). \tag{41}$$

From (38) and (41), the training objective J_m can be expressed as

$$\begin{aligned} J_m =& \|\boldsymbol{e}_{m-1}\|_2^2 + 2\beta_m \boldsymbol{e}_{m-1}^{\mathrm{T}}(\boldsymbol{f}_{m-1} - \boldsymbol{h}_m) \\ &+ \beta_m^2\|\boldsymbol{f}_{m-1} - \boldsymbol{h}_m\|_2^2 \\ &+ \sigma_\beta^2\left((1-\beta_m)^2\kappa_{m-1} + \beta_m^2\|\boldsymbol{h}_m^2\|_2^2\right) \\ &+ (1+\sigma_\beta^2)\sigma_w^2\Big((1-\beta_m)^2\tau_m \\ &+ \beta_m^2 \sum_{k=1}^{N}\sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k)\Big) \end{aligned} \tag{42}$$

$$\begin{aligned} =& J_{m-1} \\ &+ \beta_m^2\Big[\|\boldsymbol{f}_{m-1} - \boldsymbol{h}_m\|_m^2 + \sigma_\beta^2(\kappa_{m-1} + \|\boldsymbol{h}_m\|_2^2) \\ &+ (1+\sigma_\beta^2)\sigma_w^2\Big(\tau_{m-1} + \sum_{k=1}^{N}\sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k)\Big)\Big] \\ &+ 2\beta_m\Big[\boldsymbol{e}_{m-1}^T(\boldsymbol{f}_{m-1} - \boldsymbol{h}_m) - \sigma_\beta^2\kappa_{m-1} \\ &- (1+\sigma_\beta^2)\sigma_w^2\tau_{m-1}\Big]. \end{aligned} \tag{43}$$

Then we can obtain the difference between two consecutive iterations, given by

$$R_m = J_m - J_{m-1} \tag{44}$$

$$\begin{aligned} =& \beta_m^2\Big[\|\boldsymbol{f}_{m-1} - \boldsymbol{h}_m\|_m^2 + \sigma_\beta^2(\kappa_{m-1} + \|\boldsymbol{h}_m\|_2^2) \\ &+ (1+\sigma_\beta^2)\sigma_w^2\Big(\tau_{m-1} + \sum_{k=1}^{N}\sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k)\Big)\Big] \\ &+ 2\beta_m\Big[\boldsymbol{e}_{m-1}^T(\boldsymbol{f}_{m-1} - \boldsymbol{h}_m) - \sigma_\beta^2\kappa_{m-1} \\ &- (1+\sigma_\beta^2)\sigma_w^2\tau_{m-1}\Big] \end{aligned} \tag{45}$$

Similar to the WDT-IELM algorithm, in the WDTC-IELM the optimal value of β_m is given by

$$\beta_m^* = \frac{\boldsymbol{e}_{m-1}^T(\boldsymbol{f}_{m-1} - \boldsymbol{h}_m) - \sigma_\beta^2\kappa_{m-1} - (1+\sigma_\beta^2)\sigma_w^2\tau_{m-1}}{\Omega} \tag{46}$$

where

$$\begin{aligned} \Omega =& \|\boldsymbol{f}_{m-1} - \boldsymbol{h}_m\|_m^2 + \sigma_\beta^2(\kappa_{m-1} + \|\boldsymbol{h}_m\|_2^2) \\ &(1+\sigma_\beta^2)\sigma_w^2\Big(\tau_{m-1} + \sum_{k=1}^{N}\sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k)\Big). \end{aligned} \tag{47}$$

With this optimal value, R_m is given by

$$R_m = -\frac{\Pi^2}{\Omega}. \tag{48}$$

where

$$\Pi = \boldsymbol{e}_{m-1}^T(\boldsymbol{f}_{m-1} - \boldsymbol{h}_m) - \sigma_\beta^2\kappa_{m-1} - (1+\sigma_\beta^2)\sigma_w^2\tau_{m-1}). \tag{49}$$

In (48), the denominator Ω is positive. *Hence, the expected training error of the noisy network always reduces.* The summary of the WDTC-IELM algorithm is given in Algorithm 2. It can be seen that in the WDTC-IELM, we need to keep two addition variables τ_m and κ_m.

Algorithm 2 The updating procedure of WDTC-IELM

1: Initialization: $\boldsymbol{t} = [t_1, \cdots, t_N]^{\mathrm{T}}$, $m = 0$, $\boldsymbol{e}_0 = \boldsymbol{0}$, $\boldsymbol{f}_0 = \boldsymbol{0}$, $\kappa_m = 0$ and $\tau_m = 0$.
2: **while** $m \le m_{\max}$ **do**
3: $m = m + 1$.
4: Generate a hidden node whose input weight vector $\boldsymbol{w}_m$ randomly generated.
5: For this new hidden node, computes its output values $h_m(\boldsymbol{x}_k)$'s for all training sample.
6: Form a vector $\boldsymbol{h}_m = [h_m(\boldsymbol{x}_1), \cdots, h_m(\boldsymbol{x}_N)]^{\mathrm{T}}$.
7: Compute Ω, given by
$\Omega = \|\boldsymbol{f}_{m-1} - \boldsymbol{h}_m\|_m^2 + \sigma_\beta^2(\kappa_{m-1} + \|\boldsymbol{h}_m\|_2^2)$
$+(1+\sigma_\beta^2)\sigma_w^2\Big(\tau_{m-1} + \sum_{k=1}^{N}\sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k)\Big)$.
8: Compute the output weight of the newly inserted node:
$\beta_m = \frac{\boldsymbol{e}_{m-1}^T(\boldsymbol{f}_{m-1} - \boldsymbol{h}_m) - \sigma_\beta^2\kappa_{m-1} - (1+\sigma_\beta^2)\sigma_w^2\tau_{m-1}}{\Omega}$.
9: $\boldsymbol{f}_m = (1-\beta_m)\boldsymbol{f}_{m-1} + \beta_m\boldsymbol{h}_m$.
10: $\boldsymbol{e}_m = \boldsymbol{t} - \boldsymbol{f}_m = \boldsymbol{e}_{m-1} + \beta_m(\boldsymbol{f}_{m-1} - \boldsymbol{h}_m)$.
11: $\kappa_m = \kappa_{m-1} + \beta_m^2\|\boldsymbol{h}_m\|_2^2$.
12: $\tau_m = \tau_{m-1} + \beta_m^2 \sum_{k=1}^{N}\sum_{l=1}^{D+1} w_{ml}^2 \Delta H_{ml}^2(\boldsymbol{x}_k)$.
13: **end while**

Table 1 Details of the seven UCI datasets

Data-set	Number of samples	Number of features
Abalone	4177	8
Concrete	1030	9
Airfoil self noise (ASN)	1503	5
Bodyfat	252	7
Chemical sensor	498	8
Building energy	4208	14
Housing	506	13

5 Numerical study

In this section, we evaluate the performance of the proposed algorithms with some real life datasets obtained from the UCI machine learning repository (Lichman 2013). Datasets that we use to validate the performance of the algorithms are Abalone, Housing price, Concrete Compressive strength, Airfoil Self Noise (ASN), BodyFat, Chemical Sensor, and Building Energy. Table 1 presents the properties of the datasets. The datasets are pre-processed. The target outputs of these datasets are normalized to the range of [0, 1], while the input features of the data sets are normalized to the range of [−1, 1]. In addition, we randomly generate the input weights of the hidden nodes from range [−1, 1].

For fair comparison, we use the tenfold evaluation method. The samples of a dataset are randomly partitioned into ten subsets. The summary of the partitioning is given in Table 2. In our simulation, a subset is used as the test set, and the remaining nine subsets are used as training data. The noise levels that we test are ($\{\sigma_\beta^2 = \sigma_w^2 = 0.04$, $\sigma_\beta^2 = \sigma_w^2 = 0.09, \sigma_\beta^2 = \sigma_w^2 = 0.09$ and $\sigma_\beta^2 = \sigma_w^2 = 0.25\}$). According to the analysis in Liu and Kaneko (1969), the noise level $\sigma_\beta^2 = \sigma_w^2 = 0.04$ responds around 2–3 mantissa bits in the digital implementation. For other noise levels, we can consider that the standard deviation of noise is around 30–50 % of the nominate value in the analog implementation.

5.1 Number of hidden nodes

We use three datasets to demonstrate how the test set errors change with respect to the various numbers of hidden nodes. The three datasets are Abalone, Concrete, and Boston Housing. Three noise levels $\{\sigma_\beta^2 = \sigma_w^2 = 0.04$, $\sigma_\beta^2 = \sigma_w^2 = 0.09, \sigma_\beta^2 = \sigma_w^2 = 0.25\}$ are considered. Figure 1 shows the test set MSE versus the number of nodes for a typical run. It can be seen that the test set errors of the CIELM are much higher than those of the other three algorithms. That means, the noise tolerant ability of the original CIELM is very poor. For the IELM, WDT-IELM, and WDTC-IELM algorithms, when the number of hidden nodes is around 400–500, the decreasing rate of the test set error is very slow. Thus, we treat 500 hidden nodes as a reference point to conduct a deeper analysis in the rest parts of the paper.

For the figure, the WDT-IELM algorithm is better than the IELM algorithm. When the noise level is high, the improvement on the test set error becomes more significant. In addition, when we use the WDTC-IELM algorithm, we can further improve the test set MSE. For instance, for the Abalone data set with noise level equal to $\sigma_\beta^2 = \sigma_w^2 = 0.04$, the test set MSE of the original I-ELM algorithm is 0.01074.

Table 2 The partitioning of the datasets in the tenfold

Data sets	Sample sizes of subsets									
	Subset 1	Subset 2	Subset 3	Subset 4	Subset 5	Subset 6	Subset 7	Subset 8	Subset 9	Subset 10
Abalone	418	418	418	418	418	418	418	418	418	415
Concrete	103	103	103	103	103	103	103	103	103	103
Airfoil self noise (ASN)	150	150	150	150	150	150	150	150	150	153
Bodyfat	25	25	25	25	25	25	25	25	25	27
Chemical Sensor	50	50	50	50	50	50	50	50	50	48
Building energy	421	421	421	421	421	421	421	421	421	419
Housing	51	51	51	51	51	51	51	51	51	47

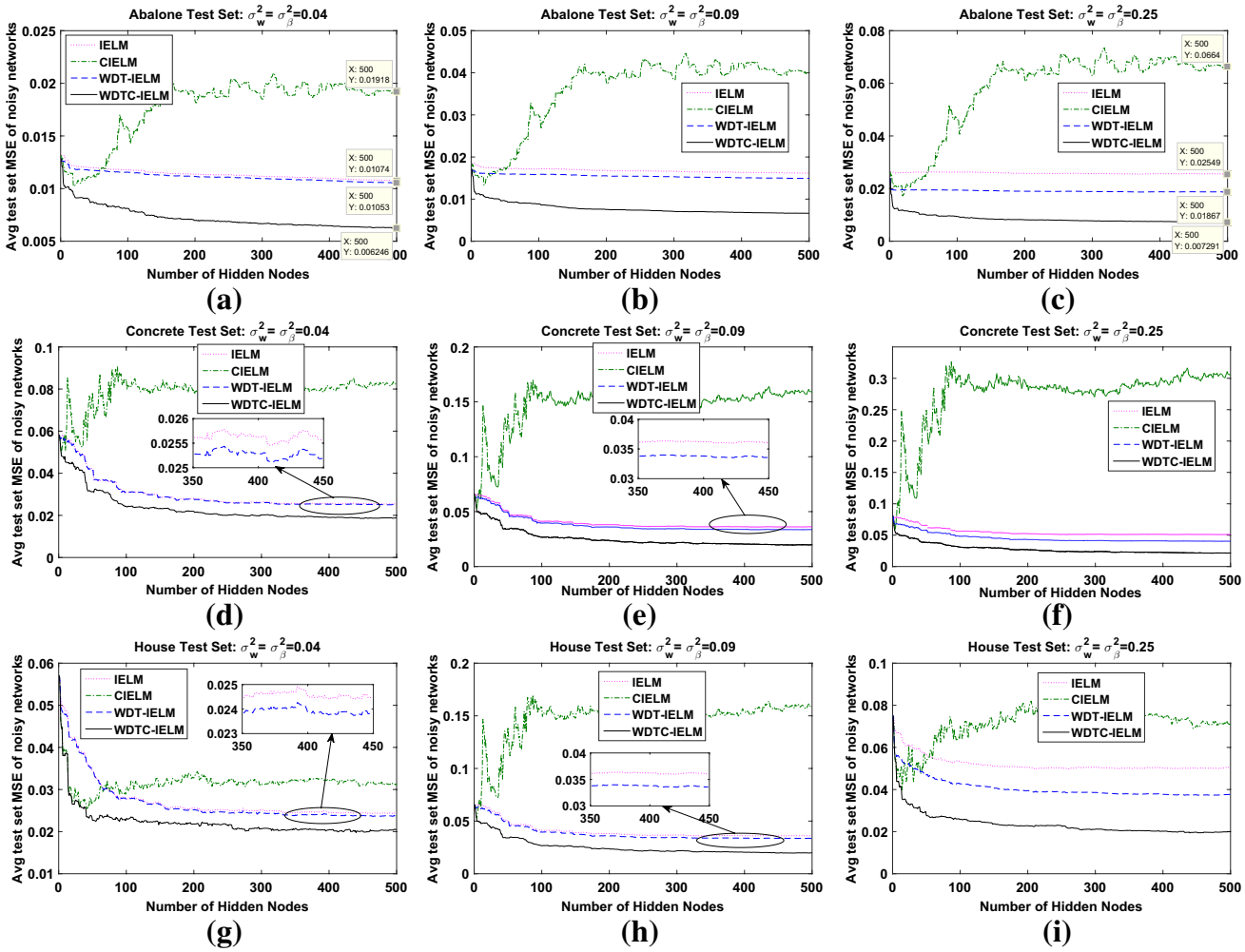

Fig. 1 Test set MSE versus number of hidden nodes. Three noise levels are consider. They are $\sigma_\beta^2 = \sigma_w^2 = 0.04$, $\sigma_\beta^2 = \sigma_w^2 = 0.09$, $\sigma_\beta^2 = \sigma_w^2 = 0.25$. **a–c** Are for Abalone dataset. **d–f** Are for Concrete Compressive strength dataset. **g–i** Are for Housing Price dataset

When we use the WDT-IELM algorithm, we can reduce the test set MSE to 0.01053. Using the WDTC-IELM algorithm, we can further reduce the test set MSE to 0.006246. The MSE difference between IELM and WDTC-IELM is 0.005172.

5.2 Performance comparison

To further investigate the performance of those algorithms, we use the tenfold evaluation strategy. The setting of the tenfold is shown in Table 2. The average test set performance over the tenfold in the seven datasets is summarized in Table 3. Besides, for an easy and quick view of the result in Table 3, we also provide a chart view of the performance in Fig. 2. The table contains $7 \times 4 \times 4 = 112$ entities. Each entity is the average MSE of the tenfolds (ten runs).

From the table, the test set MSE values from WDT-IELM are smaller than those of IELM. This is obvious at large noise level as shown in Fig. 2. Besides, the test set MSE values from WDTC-IELM are much smaller than those of the other three algorithms. For instance, for the BodyFat dataset, when the noise level is 0.04, the test set MSE value of the original IELM is 0.020329. When we use WDT-IELM, we reduce the test set MSE value to 0.019707. Furthermore, when we use the WDTC-IELM algorithm, the test set MSE value can be reduced to 0.010461. The improvement of using the WDTC-IELM algorithm is more significant for high noise levels. When the noise level is 0.25, the test set MSE value of the original IELM is 0.063316. When we use WDT-IELM, we further reduce the test set MSE value to 0.046406. Furthermore, when we use the WDTC-IELM algorithm, the test set MSE value can be reduced to 0.012829.

Table 3 The performance of the four algorithms over the tenfold. The number of hidden nodes in the networks is 500

Data set	Noise level	IELM AVG MSE (STD)	WDT-IELM AVG MSE (STD)	CIELM AVG MSE (STD)	WDTC-IELM AVG MSE (STD)
Abalone	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.011783 (0.000938)	0.011560 (0.000937)	0.020905 (0.001155)	0.007181 (0.000754)
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.017275 (0.000977)	0.015978 (0.000973)	0.042613 (0.001670)	0.007562 (0.000796)
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.026747 (0.001161)	0.021641 (0.001096)	0.070163 (0.002897)	0.008205 (0.000831)
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.031176 (0.001242)	0.023635 (0.001129)	0.089067 (0.003462)	0.008544 (0.000871)
Concrete	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.025125 (0.001773)	0.024660 (0.001771)	0.081572 (0.004145)	0.018160 (0.001696)
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.036084 (0.001891)	0.033387 (0.001869)	0.159236 (0.007044)	0.018807 (0.001746)
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.051595 (0.002221)	0.043229 (0.002060)	0.304412 (0.014399)	0.020196 (0.001805)
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.061589 (0.002194)	0.045867 (0.002029)	0.400578(0.017670)	0.021036 (0.001866)
Airfoil self noise (ASN)	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.030828 (0.002594)	0.029885 (0.002568)	0.068495 (0.004323)	0.017614 (0.002150)
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.049538 (0.002893)	0.044534 (0.002797)	0.116458(0.005878)	0.018152 (0.002146)
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.078103 (0.003992)	0.062197 (0.003523)	0.199054(0.010623)	0.019508 (0.002172)
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.088394 (0.004238)	0.062641 (0.003548)	0.349192 (0.012665)	0.019850 (0.002204)
Bodyfat	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.020329 (0.004176)	0.019707 (0.004235)	0.022267 (0.003534)	0.010461 (0.003271)
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.028394 (0.004417)	0.026290 (0.004513)	0.031199 (0.003772)	0.010922 (0.003435)
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.043682 (0.004384)	0.034604 (0.004472)	0.051883 (0.004338)	0.011654 (0.003632)
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.063316 (0.004514)	0.046406 (0.004556)	0.099397 (0.008975)	0.012829 (0.003924)
Chemical sensor	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.015480 (0.000928)	0.014666 (0.000879)	0.030556 (0.002067)	0.003908 (0.000634)
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.027633 (0.001375)	0.024117 (0.001150)	0.051147 (0.003091)	0.004240 (0.000657)
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.047376 (0.002041)	0.035701 (0.001430)	0.093500(0.005328)	0.004497 (0.000716)
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.061687 (0.002665)	0.041432 (0.001609)	0.194894 (0.010652)	0.005578 (0.000764)
Building energy	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.033315 (0.002004)	0.032760 (0.001970)	0.033096 (0.002196)	0.024433 (0.001759)
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.043506 (0.002079)	0.040445 (0.001985)	0.043062 (0.002483)	0.024665 (0.001663)
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.065593 (0.002526)	0.055578 (0.002225)	0.056860 (0.002981)	0.025735 (0.001618)
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.071086 (0.002598)	0.053335 (0.002148)	0.080842 (0.003679)	0.026818 (0.001555)
Housing	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.020332 (0.006023)	0.019815 (0.006108)	0.022122 (0.005755)	0.012956 (0.006287)
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.029526 (0.006505)	0.026570 (0.006406)	0.032400 (0.006259)	0.013440 (0.006491)
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.045813 (0.006505)	0.037162 (0.007546)	0.060647 (0.006416)	0.014920 (0.006651)
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.062733 (0.007802)	0.045019 (0.007958)	0.082054 (0.007856)	0.015648 (0.006513)

Furthermore, we discover that the WDTC-IELM are relatively insensitive to the noise level. Consider the Abalone dataset.

- When the noise level is 0.04, the test set MSE of IELM is 0.011783. When the noise level increases to 0.25, the test set MSE increases to 0.031176.
- When the noise level is 0.04, the test set MSE of CIELM is 0.020905. When the noise level increases to 0.25, the test set MSE increases to 0.089067.
- When the noise level is 0.04, the test set MSE of WDT-IELM is 0.011560. When the noise level increases to 0.25, the test set MSE increases to 0.023635.
- When the noise level is 0.04, the test set MSE of WDT-IELM is 0.007181. When the noise level increases to 0.25, the test set MSE increases to 0.008544.

The above phenomenon also happens in the other six datasets. Since the WDTC-IELM is the best among the four algorithms, one may argue that we do not need to consider the WDT-IELM. However, the WDTC-IELM algorithm needs to update all the previous trained weights and has a more complicated training procedure, as shown Algorithms 1 and 2.

5.3 Paired T-test analysis

From Fig 2 and Table 3, in terms of average test set MSE, the two proposed algorithms are better than the two original algorithms. In this section, we would like to check if the improvements are statistical significant or not. We would like to perform significant test, i.e., paired t-test, to check if the performance of the proposed algorithms are statistical

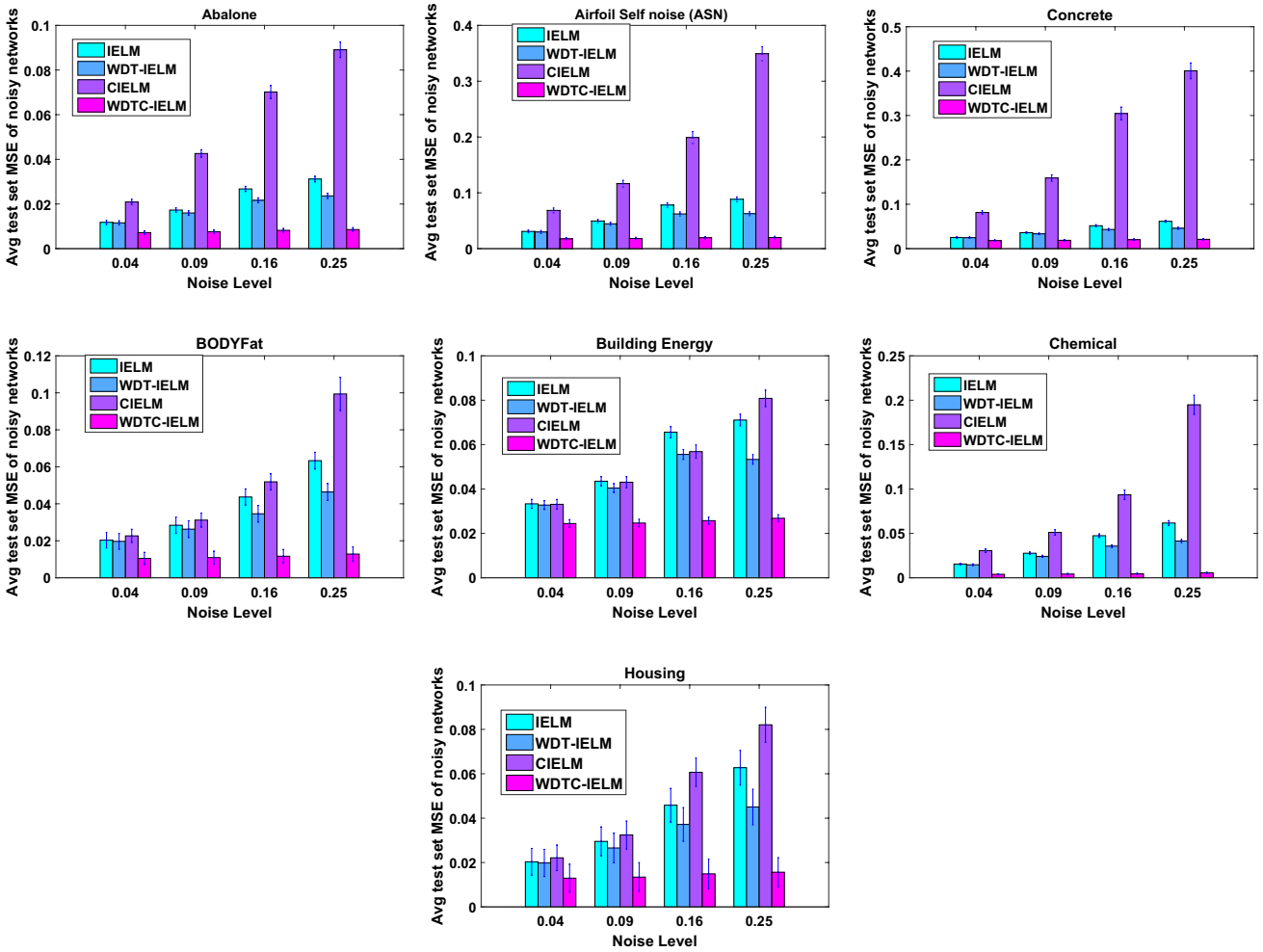

Fig. 2 Test set MSE versus Noise Level for each data set considered. Four noise levels are considered. They are $\sigma_\beta^2 = \sigma_w^2 = 0.04$, $\sigma_\beta^2 = \sigma_w^2 = 0.09$, $\sigma_\beta^2 = \sigma_w^2 = 0.16$, and $\sigma_\beta^2 = \sigma_w^2 = 0.25$

significant or not. Since the CIELM is with very poor performance, we do not perform the paired t-test on it.

Tables 4 and 5 summarize the paired t-test results, i.e., the IELM versus WDT-IELM, and IELM versus WDTC-IELM. For the one-tail test with 10 trials and 95% confidence level, the critical t-value is 1.8331.

Before we perform the paired test, we should check if the data pass the normality test or not. In this paper, we consider the Anderson–Darling goodness-of-fit hypothesis test. For the hypothesis test, the critical p value is 0.05. That is the p value of the data should be greater than 0.05. Since there are three algorithms involved, four noise levels, and seven datasets, there are 84 sets of data. Among those sets, most of cases meet the normality test. Only nine cases do not the normality test. The nine cases appear in three datasets: the ASN dataset, the Chemical Sensor dataset, and the Housing dataset. In the ASN dataset, there are three cases: WDT-IELM (noise level=0.04) with p value equal to 0.0498, WDT-IELM (noise level=0.09) with p value equal to 0.0162, and IELM (noise level=0.09) with p value equal to 0.0313. In the Housing dataset, there are two cases: IELM (noise level=0.16) with p value equal to 0.0481 and IELM (noise level = 0.25) with p value equal to 0.0381. In the Chemical Sensor dataset, there are four cases: IELM (noise level=0.16) with p value equal to 0.028, IELM (noise level=0.25) with p value equal to 0.0129, WDT-IELM (noise level=0.16) with p value equal to 0.03, and WDT-IELM (noise level=0.25) with p value equal to 0.0345.

In Table 4, all t-values are greater than the critical t-value (i.e. 1.8331). Besides, all confidence intervals of the average improvements excluded the zero. For example, in the Bodyfat dataset with the noise level equal to 0.25, the t-value is 89.1, which is greater than 1.8331. Besides, the confidence interval of average improvement is in between [0.016480, 0.017339]. With these results, there is strong evidence that the WDT-IELM is better than IELM.

Table 4 The paired t-test result between WDT-IELM and IELM

Data set	Noise level	IELM vs. WDT-IELM					
		AVG IELM MSE	AVG WDT-IELM MSE	AVG improvement	t-value	p-value	Confidence interval of AVG improvement
Abalone	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.011783	0.011560	0.000223	52.0	1.80×10^{-12}	[0.000214, 0.000232]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.017275	0.015978	0.001296	115.3	1.41×10^{-15}	[0.001271, 0.001322]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.026747	0.021641	0.005107	124.3	7.19×10^{-16}	[0.005014, 0.005199]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.031176	0.023635	0.007542	114.2	1.54×10^{-15}	[0.007392, 0.007691]
Concrete	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.025125	0.024660	0.000465	30.8	1.98×10^{-10}	[0.000431, 0.000499]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.036084	0.033387	0.002697	69.0	1.43×10^{-13}	[0.002609, 0.002786]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.051595	0.043229	0.008366	73.1	8.57×10^{-14}	[0.008107, 0.008625]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.061589	0.045867	0.015721	83.5	2.56×10^{-14}	[0.015295, 0.016147]
Airfoil self noise (ASN)	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.030828	0.029885	0.000943	60.5	4.63×10^{-13}	[0.000908, 0.000978]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.049538	0.044534	0.005004	86.9	1.79×10^{-14}	[0.004874, 0.005135]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.078103	0.062197	0.015906	76.7	5.50×10^{-14}	[0.015436, 0.016375]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.088394	0.062641	0.025754	82.2	2.97×10^{-14}	[0.025045, 0.026463]
Bodyfat	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.020329	0.019707	0.000622	20.9	6.20×10^{-9}	[0.000555, 0.000689]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.028394	0.026290	0.002104	41.5	1.37×10^{-11}	[0.001989, 0.002219]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.043682	0.034604	0.009078	91.9	1.08×10^{-14}	[0.008854, 0.009301]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.063316	0.046406	0.016909	89.1	1.43×10^{-14}	[0.016480, 0.017339]
Chemical sensor	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.015480	0.014666	0.000814	39.8	1.99×10^{-11}	[0.000768, 0.000860]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.027633	0.024117	0.003516	42.8	1.04×10^{-11}	[0.003330, 0.003701]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.047376	0.035701	0.011676	55.2	1.06×10^{-12}	[0.011197, 0.012154]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.061687	0.041432	0.020256	54.0	1.29×10^{-12}	[0.019407, 0.021104]
Building energy	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.033315	0.032760	0.000555	34.1	8.03×10^{-11}	[0.000518, 0.000592]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.043506	0.040445	0.003061	74.6	7.09×10^{-14}	[0.002968, 0.003154]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.065593	0.055578	0.010015	87.3	1.73×10^{-14}	[0.009756, 0.010275]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.071086	0.053335	0.017751	100.3	4.93×10^{-15}	[0.017350, 0.018151]
Housing	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.020332	0.019815	0.000518	11.3	1.27×10^{-6}	[0.0000414, 0.000621]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.029526	0.026570	0.002956	28.4	4.08×10^{-10}	[0.002720, 0.003191]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.045813	0.037162	0.008651	29.4	2.96×10^{-10}	[0.007986, 0.009317]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.062733	0.045019	0.017714	36.70	4.10×10^{-11}	[0.016622, 0.018805]

Again, in Table 5, all the t-values are much greater than the critical t-value. This phenomenon revealed the improvement of WDTC-IELM is statistically significant too.

As mentioned in the above, there are a few cases that do not meet the normality test. *However, from Table 3, for those cases, the improvements of using WDT-IELM and WDTC-ILM are greater than the standard deviations of IELM, WDT-IELM and WDTC-IELM. Hence, we can conclude that the improvements of WDT-IELM and WDTC-IELM are significant.*

For example, for the ASN with the noise level equal to 0.09, for the IELM, the test set MSE of is 0.049538 and the standard deviation is 0.002893. When we use the WDT-IELM, the test set MSE of is 0.044534 and the standard deviation is 0.002797. Clearly, the improvement of using WDT-IELM is around 0.005004 and is around two times of the standard deviations. When we use the WDTC-IELM, the test set MSE of is 0.018152 and the standard deviation is 0.002. Clearly, the improvement of using WDT-IELM is around 0.031386 and is around nines times of the standard deviations.

5.4 Performance improvement ratio

We also present the performance improvement ratios of WDT-IELM and WDTC-IELM. We calculate the performance improvement ratio:

$$((P_e - P_p)/P_e) * 100, \tag{50}$$

where P_e and P_p are the performances of existing and proposed models, respectively. Table 6 summarizes the the performance improvement ratios of WDT-IELM and

Table 5 The paired t-test result between WDTC-IELM and IELM

Data set	Noise level	IELM vs. WDTC-IELM					
		AVG IELM MSE	AVG WDTC-IELM MSE	AVG Improvement	t-value	p-value	Confidence interval of AVG improvement
Bodyfat	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.022278	0.010835	0.011443	20.4	7.74×10^{-9}	[0.010172, 0.012714]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.034085	0.011378	0.022707	28.9	3.45×10^{-10}	[0.020930, 0.024483]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.052936	0.012189	0.040746	32.5	1.21×10^{-10}	[0.037912, 0.043580]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.070029	0.013115	0.056914	35.9	4.93×10^{-11}	[0.053333, 0.060495]
Abalone	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.012662	0.007209	0.005453	206.4	7.48×10^{-18}	[0.005393, 0.005513]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.019211	0.007580	0.011631	308.8	1.99×10^{-19}	[0.011546, 0.011716]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.028824	0.008017	0.020807	426.4	1.09×10^{-20}	[0.020700, 0.020917]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.035775	0.008502	0.027273	561.4	9.19×10^{-22}	[0.027163, 0.027383]
Airfoil self noise (ASN)	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.031486	0.016750	0.014736	259.8	9.44×10^{-19}	[0.014608, 0.014864]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.051227	0.017497	0.033730	323.9	1.30×10^{-19}	[0.033495, 0.033966]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.081452	0.018467	0.062985	359.3	5.10×10^{-20}	[0.062589, 0.063382]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.107319	0.019475	0.087845	404.0	1.76×10^{-20}	[0.087353, 0.088336]
Housing	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.022812	0.013174	0.009637	16.8	4.32×10^{-8}	[0.008336, 0.010940]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.034105	0.013651	0.020454	29.9	2.54×10^{-10}	[0.018908, 0.022001]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.054031	0.015027	0.039005	66.7	1.92×10^{-13}	[0.037683, 0.040327]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.073486	0.015348	0.058138	41.6	1.34×10^{-11}	[0.054976, 0.061300]
Concrete	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.027174	0.017707	0.009468	76.6	5.61×10^{-14}	[0.009188, 0.009747]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.038863	0.018418	0.020445	67.7	1.70×10^{-13}	[0.019762, 0.021128]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.055533	0.019621	0.035911	73.1	8.53×10^{-14}	[0.034799, 0.037023]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.070241	0.020741	0.049500	74.5	7.14×10^{-14}	[0.047997, 0.051003]
Chemical sensor	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.015348	0.003805	0.011543	206.3	7.51×10^{-18}	[0.011417, 0.011670]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.029669	0.004125	0.025544	225.5	3.37×10^{-18}	[0.025288, 0.025800]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.050609	0.004567	0.046042	240.3	1.91×10^{-18}	[0.045608, 0.046475]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.065332	0.005175	0.060157	259.1	9.66×10^{-19}	[0.059632, 0.060683]
Building Energy	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	0.035929	0.024715	0.011214	303.4	2.34×10^{-19}	[0.011130, 0.011297]
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	0.050040	0.025368	0.024672	281.2	4.63×10^{-19}	[0.024473, 0.024870]
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	0.071174	0.026145	0.045030	305.8	2.18×10^{-19}	[0.044697, 0.045363]
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	0.091444	0.027190	0.064254	309.4	1.96×10^{-19}	[0.063784, 0.064724]

WDTC-IELM. The 3rd and 4th columns show the the performance improvement ratios of WDT-IELM and WDTC-IELM to IELM, respectively. The 3rd and 4th columns show the the performance improvement ratios of WDT-IELM and WDTC-IELM to C-IELM. The table clearly shows that our algorithms outperform the existing ones. For example, in the ASN dataset we have the following performance improvement ratios.

- When the noise level is 0.04, the improvement ratio of WDT-IELM to IELM is 3.06%. When the noise level increases to 0.25, the improvement ratio increases to 29.14%.
- When the noise level is 0.04, the improvement ratio of WDTC-IELM to IELM is 42.86%. When the noise level increases to 0.25, the improvement ratio increases to 77.54%.
- When the noise level is 0.04, the improvement ratio of WDT-IELM to CIELM is 56.37%. When the noise level increases to 0.25, the improvement ratio increases to 82.06%.
- When the noise level is 0.04, the improvement ratio of WDTC-IELM to CIELM is 74.28%. When the noise level increases to 0.25, the improvement ratio increases to 94.32%.

Table 6 Performances Improvement ratio of the comparison algorithms: IELM, CIELM,WDT-IELM and WDTC-IELM

Data set	Noise level	Performance improvement ratio			
		WDT-IELM to IELM (%)	WDTC-IELM to IELM (%)	WDT-IELM to CIELM (%)	WDTC-IELM to CIELM (%)
Abalone	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	1.89	39.05	44.70	65.65
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	7.50	56.22	62.50	82.25
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	19.09	69.33	69.16	88.31
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	24.19	72.59	73.46	90.41
Concrete	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	1.85	27.72	69.77	77.74
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	7.48	47.88	79.03	88.19
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	16.22	60.86	85.80	93.37
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	25.53	65.85	88.55	94.75
Airfoil self noise (ASN)	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	3.06	42.86	56.37	74.28
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	10.10	63.36	61.76	84.41
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	20.37	75.02	68.75	90.20
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	29.14	77.54	82.06	94.32
Bodyfat	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	3.06	48.54	13.06	53.85
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	7.41	61.53	15.73	64.99
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	20.78	73.32	33.30	77.54
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	26.71	79.74	53.31	87.09
Chemical sensor	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	5.26	74.76	52.00	87.21
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	12.72	84.66	52.85	91.71
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	24.64	90.51	61.82	95.19
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	32.84	90.96	78.74	97.14
Building energy	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	1.67	26.66	1.02	26.18
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	7.04	43.31	6.08	42.72
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	15.27	60.77	2.26	54.74
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	24.97	62.27	34.03	66.83
Housing	$\sigma_\beta^2 = \sigma_w^2 = 0.04$	2.55	36.28	10.43	41.44
	$\sigma_\beta^2 = \sigma_w^2 = 0.09$	10.01	54.48	18.00	58.52
	$\sigma_\beta^2 = \sigma_w^2 = 0.16$	18.88	67.43	38.73	75.40
	$\sigma_\beta^2 = \sigma_w^2 = 0.25$	28.24	75.06	45.13	80.93

The above occurrence also happens in the other six datasets. Furthermore, we give visual figures as shown in Fig. 3. The figure makes a better view of Table 6. From the figure and table, the improvement ratios of our proposed algorithms (WDT-IELM and WDTC-IELM) are more significant for high noise levels.

6 Conclusion

In this paper, we have developed a noise resistant objective function for concurrent multiplicative weight noise in the input weights and output weights. Based on the developed objective function, we then propose two increment ELM algorithms that can handle weight noise. We also show that during the incremental training, the training objective is non-increasing. From the simulation results, the proposed incremental algorithms are much better than the original incremental algorithms. Besides, based on the paired t-test results and the performance improvement ratio results, the improvement of the proposed algorithms are statistical significant.

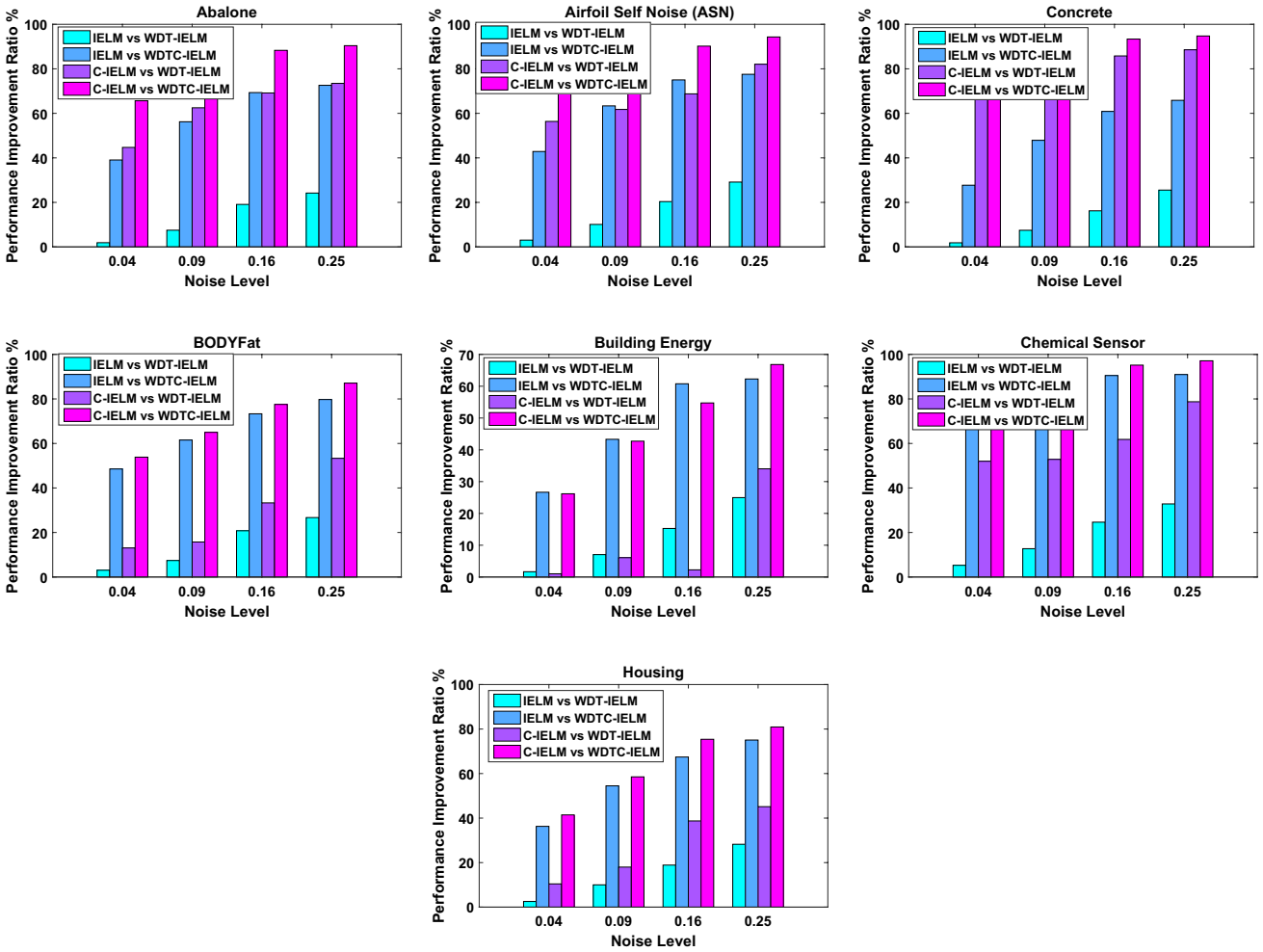

Fig. 3 Performance improvement ratios. Four noise levels are considered. They are $\sigma_\beta^2 = \sigma_w^2 = 0.04$, $\sigma_\beta^2 = \sigma_w^2 = 0.09$, $\sigma_\beta^2 = \sigma_w^2 = 0.16$, and $\sigma_\beta^2 = \sigma_w^2 = 0.25$

Acknowledgements The work is supported by a research grant from City University of Hong Kong (No. 9610431).

References

Barron AR (1993) Universal approximation bounds for superpositions of a sigmoidal function. IEEE Trans Inf Theory 39(3):930–945

Bi X, Ma H, Li J, Ma Y, Chen D (2018) A positive and unlabeled learning framework based on extreme learning machine for drug-drug interactions discovery. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-018-0960-7

Lichman M (2013) UCI machine learning repository. http://archive.ics.uci.edu/ml. Accessed 2019

Burr JB (1991) Digital neural network implementations. Neural Netw Concept Appl Implement 3:237–285

Feng RB, Han ZF, Wan WY, Leung CS (2017) Properties and learning algorithms for faulty RBF networks with coexistence of weight and node failures. Neurocomputing 224:166–176

Guély F, Siarry P (1993) Gradient descent method for optimizing various fuzzy rule bases. In: Second IEEE international conference on fuzzy systems, 1993, pp 1241–1246

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

Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural Netw 4(2):251–257

Huang GB, Zhu QY, Siew CK (2006) Extreme learning machine: theory and applications. Neurocomputing 70(1–3):489–501

Huang GB, Chen L, Siew CK (2006) Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans Neural Netw 17(4):879–892

Huang GB, Chen L (2007) Convex incremental extreme learning machine. Neurocomputing 70(16–18):3056–3062

Leung CS, Wang HJ, Sum J (2010) On the selection of weight decay parameter for faulty networks. IEEE Trans Neural Netw 21(8):1232–1244

Li Y, Zhang S, Yin Y, Xiao W, Zhang J (2018) Parallel one-class extreme learning machine for imbalance learning based on Bayesian approach. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-018-0994-x

Liu B, Kaneko T (1969) Error analysis of digital filters realized with floating-point arithmetic. Proc IEEE 57(10):1735–1747

Mahdiani HR, Fakhraie SM, Lucas C (2012) Relaxed fault-tolerant hardware implementation of neural networks in the presence of multiple transient errors. IEEE Trans Neural Netw Learn Systems 23(8):1215–1228

Martolia R, Jain A, Singla L (2015) Analysis & survey on fault tolerance in radial basis function networks. In: 2015 IEEE international conference on computing, communication & automation (ICCCA), pp 469–473

Murakami M, Honda N (2007) Fault tolerance comparison of IDS models with multilayer perceptron and radial basis function networks. In: International joint conference on neural networks 2007 (IJCNN2007), pp 1079–1084, IEEE

Pajarinen J, Peltonen J, Uusitalo MA (2011) Fault tolerant machine learning for nanoscale cognitive radio. Neurocomputing 74(5):753–764

Wang SJ, Muhammad K, Phillips P, Dong Z, Zhang YD (2017) Ductal carcinoma in situ detection in breast thermography by extreme learning machine and combination of statistical measure and fractal dimension. J Ambient Intell Human Comput. https://doi.org/10.1007/s12652-017-0639-5