Noise-Boosted Backpropagation Learning of Feedforward Threshold Neural Networks for Function Approximation

Lingling Duan, Fabing Duan, François Chapeau-Blondeau, and Derek Abbott, *Fellow, IEEE*

***Abstract*— Aiming to ensure the feasibility of the backpropagation training of feedforward threshold neural networks, each hidden unit layer is designed to be composed of a sufficiently large number of hard-limiting activation functions that are excited by mutually independent external noise components and the weighted inputs simultaneously. The application of noise to nondifferentiable activation functions enables a proper definition of the gradients, and the injected noise is treated as a network parameter that can be adaptively updated by a stochastic gradient descent learning rule. This noise-boosted backpropagation learning process is found to converge to a nonzero optimized level of noise, indicating that the injected noise is beneficial both for the learning and for the ensuing retrieval phase. For minimizing the total error energy of the function approximation in the designed threshold neural network, the proposed noise-boosted backpropagation learning method is proven to be better than directly injecting noise into network inputs or weight coefficients. The Lipschitz continuous property of the noise-smoothed activation function in the hidden unit layer is demonstrated to guarantee the local convergence of the learning process. Beyond the Gaussian injected noise, the optimal noise type is also numerically solved for training the designed threshold neural network. Test experiments for approximating nonlinear functions and real-world datasets verify the feasibility of this noise-boosted backpropagation algorithm in the threshold neural network. These results not only extend the analysis of the beneficial effects of noise similar to stochastic resonance and exploited here to the universal approximation capabilities of threshold neural networks, but also allow backpropagation training of neural networks with a much wider family of nondifferentiable activation functions.**

***Index Terms*— Function approximation, noise injection, noise-boosted backpropagation, optimal noise, stochastic resonance, threshold neural network.**

Manuscript received July 15, 2021; revised September 20, 2021; accepted October 11, 2021. Date of publication October 21, 2021; date of current version November 3, 2021. This work was supported in part by the Natural Science Foundation of Shandong Province, in part by the Taishan Scholar Project of Shandong Province of China under Grant TS20190930, in part by the Australian Research Council under Grant DP200103795, and in part by the Shandong Key Laboratory of Industrial Control Technology. The Associate Editor coordinating the review process was Arunava Naha. *(Corresponding author: Fabing Duan.)*

Lingling Duan is with the Institute of Complexity Science, Qingdao University, Qingdao 266071, China, and also with the Department of Mathematics, Jining University, Jining 273155, China.

Fabing Duan is with the Institute of Complexity Science, Qingdao University, Qingdao 266071, China (e-mail: fabing.duan@gmail.com).

François Chapeau-Blondeau is with the Laboratoire Angevin de Recherche en Ingénierie des Systèmes (LARIS), Université d'Angers, 49000 Angers, France (e-mail: f.chapeau@univ-angers.fr).

Derek Abbott is with the Centre for Biomedical Engineering (CBME), School of Electrical and Electronic Engineering, The University of Adelaide, Adelaide, SA 5005, Australia (e-mail: derek.abbott@adelaide.edu.au).

Digital Object Identifier 10.1109/TIM.2021.3121502

I. Introduction

NOISE injection [1]–[3], as a regularization method for avoiding overfitting, has been extensively investigated for improving the generalization performance of a neural network by artificially adding noise to input data, weights, the desired signal or gradients during the backpropagation training [4]–[8]. By adding noise, only to the activation function in its hard-saturated regimes, it is found [9], [10] that the training of neural networks becomes possible and easier to optimize for a wider family of activation functions, yielding the state-of-the-art competitive results on different datasets and tasks. Interestingly, the noise injection also results in lower training loss for very deep networks [10]–[18], and the dropout or dropconnect technique can be also viewed as injecting Bernoulli noise into the nodes and hidden layers of a deep neural network during training [14], [19]. The injection of noise in neural networks becomes a focused problem in the research of exploiting the benefit of noise for machine learning.

For information-network technologies, constraints on cost, power consumption, or bandwidth limitations render very attractive low-complexity sensors or devices such as low-resolution quantizers [7]–[9], [20]–[22], [25]–[27]. Using large numbers of such low-complexity devices is useful to increase the robustness of sensor networks or the lifetime of monitoring systems [7], [20]–[24]. Accordingly, activation functions with hard-limiting input–output characteristics stand as an appealing choice for implementing neural networks in digital hardware [7], [9], [20]–[22]. However, the conventional backpropagation algorithm [28] cannot be used in such networks, because the gradients of the objective function with respect to the network parameters are often undefined due to the nondiffentiability of the activation function [7], [9], [26]. Thus, noise injection has become a useful alternative strategy in updating the weights of such threshold neural networks [7], [9], [26].

It is interesting to notice that the benefits of injecting noise whilst training threshold neural networks can be viewed as a type of stochastic resonance effect [29]–[31], because there is also a nonzero amount of noise for improving the performance of nonlinear systems (in this case, threshold neural networks). Actually, the suprathreshold stochastic resonance model of a parallel array of McCulloch–Pitts neurons can be viewed as a feedforward neural network with only one hidden layer [23], [32]–[36]. A series of studies by Kosko *et al.* [13], [31],

[37]–[40] demonstrated that the backpropagation algorithm can be viewed as a special case of generalized Expectation–Maximization, and the injection of carefully chosen noise can speed convergence of backpropagation training with sufficient guaranteed conditions. Recently, Ikemoto *et al.* [26] established a stochastic resonance based feedforward threshold neural network, as shown in Fig. 1, whose hidden units consist of a number of threshold neurons. For a sufficiently large number of threshold neurons, these hidden units asymptotically converge to a smooth input–output characteristic obtained as the statistical mean of the threshold neuron with respect to the noise probability function density (PDF) [26], [41], [42]. This architectural feature indicated in Fig. 1 ensures the feasibility of the proposed backpropagation training of threshold neural networks in the framework of stochastic resonance.

However, in these noise-aided neural networks with hard-limiting activation functions, the appropriate amount of injected noise is manually configured, and not "intelligently" or adaptively learned [9], [10], [14], [26], [32]–[35]. More recently, we investigated the threshold neural networks for data classification and handwritten digit recognition by adaptively optimizing the injected Gaussian noise level during the learning process [41].

In this direction, this article will now focus on the universal approximation capabilities of threshold neural networks assisted by the optimized amount of injected noise for nonlinear functions and real-world multivariate datasets. We first theoretically prove that, in the sense of a convex loss function of total error energy, the method of injecting artificial noise into the hidden layer is never worse than adding noise into inputs or weight coefficients [1]–[3], [18] and it might even outperform it. Then, we regard the injected noise as a learnable network parameter as well as the connected weights, and propose the online noise-boosted backpropagation learning algorithm for adaptively adjusting the network parameters by the stochastic gradient descent learning rule. In the training process for supervised learning, the gradient descent learning rule of network parameters is continued until the last training data are accounted for. This constitutes one epoch of training neural networks. Then, the updated weights and the adjusted noise level in the previous epoch are treated as the initial network parameters for the next epoch, and the adjustments to weights and noise are made on an epoch-by-epoch basis.

After a certain number of learning epochs, the trained weights and the converged noise level are recorded and employed to establish the trained threshold neural network, whose hidden units are with a finite (but large) number of hard-limiting activation functions in practice. The weak convergence of the learning process is proven based on the Lipschitz continuous property of the noise-smoothed activation function in the hidden layer. Moreover, beside the Gaussian injected noise, the optimal noise type is also numerically solved by the kernel function method for training the threshold neural network. Experimental tests in approximating nonlinear functions and multivariate regression of real-world benchmark datasets are conducted with the trained threshold neural network. The obtained results show the applicability of the proposed noise-boosted backpropagation learning algorithm and the enhanced threshold neural network performance it entails.

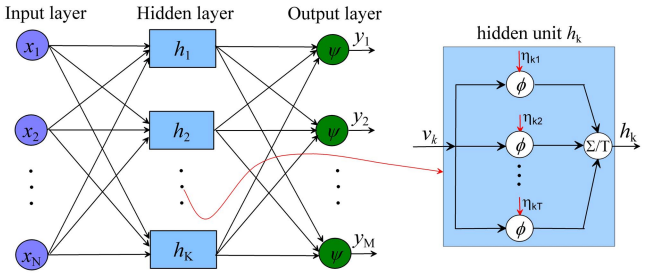

Fig. 1. Block diagram representation of the constructed feedforward threshold neural network with the noise injection in the hidden layer.

II. Threshold Neural Network and Noise-Boosted Backpropagation Learning Algorithm

Consider a three-layer feedforward neural network ($N \times K \times M$) with the $N \times 1$ input vector $\boldsymbol{x}$ and the $M \times 1$ target output vector $\boldsymbol{s}$, as shown in Fig. 1. The hidden layer and the output layer have K and M neurons, respectively. The $K \times N$ weight matrix $\boldsymbol{W}$ connects the hidden neurons to the input vector $\boldsymbol{x}$, and the $M \times K$ weight matrix $\boldsymbol{U}$ connects the output layer to the hidden one. The $M \times 1$ output vector $\boldsymbol{y}$ is given by

$$\boldsymbol{y} = \psi(\boldsymbol{U}\boldsymbol{h}) \tag{1}$$

where $\boldsymbol{h}$ is the $K \times 1$ output vector of the hidden layer and $\psi(\cdot)$ is the activation function of the neurons in the output layer. As shown in Fig. 1, the kth hidden unit

$$h_k = \frac{1}{T}\sum_{t=1}^{T}\phi(v_k + \eta_{kt}) \tag{2}$$

which consists of T activation functions $\phi(\cdot)$ activated by the same local field $v_k = [\boldsymbol{W}]^{(k)}\boldsymbol{x}$ but T mutually independent noise variables η_{kt} with a common PDF $f_\eta(\eta)$ for $t = 1, 2, \ldots, T$. Here, $[\boldsymbol{W}]^{(k)}$ denotes the kth row of the weight matrix $\boldsymbol{W}$. It is noted that, for a sufficiently large number T, the hidden unit h_k converges to

$$h_k^{\infty} = \lim_{T\to\infty}\frac{1}{T}\sum_{t=1}^{T}\phi(v_k + \eta_{kt}) = \mathrm{E}_\eta[\phi(v_k + \eta)] \tag{3}$$

where the expectation operator $\mathrm{E}_\eta(\cdot) = \int \cdot f_\eta(\eta)d\eta$ [26], [32], [34], [35], [42].

Let $\{\boldsymbol{x}(\ell), \boldsymbol{s}(\ell)\}_{\ell=1}^{L}$ denote L examples of the training set to train the network in a supervised learning manner. For each input vector $\boldsymbol{x}(\ell)$, the error between the output $y_m(\ell)$ of neuron m in the output layer and the mth element $s_m(\ell)$ of the desired response vector $\boldsymbol{s}(\ell)$ is $e_m(\ell) = s_m(\ell) - y_m(\ell)$. Summing all errors contributed by all neurons in the output layer, the instantaneous error energy of the whole network is defined as

$$\mathcal{E}(\ell) = \frac{1}{2}\sum_{m=1}^{M}e_m^2(\ell) = \frac{1}{2}\|\boldsymbol{s}(\ell) - \boldsymbol{y}(\ell)\|^2 \tag{4}$$

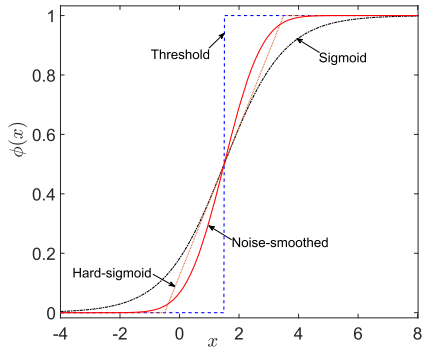

Fig. 2. Plots of the threshold activation function in (7), the noise-smoothed activation function h_k^{∞} in (8) with $\sigma = 1$, the hard-sigmoid activation function in (15) and the sigmoid function in (16). Here, the threshold parameter $\theta_k = 1.5$.

where the factor $1/2$ is introduced for deriving the gradients compactly in the following. With L examples of the training set, the total error energy or the empirical risk is computed as

$$\mathcal{E}_{\text{tot}} = \sum_{\ell=1}^{L} \mathcal{E}(\ell) = \frac{1}{2} \sum_{\ell=1}^{L} \|\boldsymbol{s}(\ell) - \boldsymbol{y}(\ell)\|^2. \tag{5}$$

Lemma 1: When the activation function $\psi(x)$ in the output layer is an affine function, the total error energy $\mathcal{E}_{\text{tot}}$ of the threshold neural network with noise-boosted hidden layer in Fig. 1 is no more than that obtained by injecting noise into input data or weights of the network.

Proof of Lemma 1 is given in Appendix A. Lemma 1 indicates that, compared with the approach of injecting noise into input data or weight coefficients [1]–[3] for training the threshold neural network, the benefit of the added noise components indicated in Fig. 1 consists in potentially achieving a smaller total error energy. Another motivation for the architectural structure of Fig. 1 will be demonstrated by the feasibility of the backpropagation training of the noise-boosted threshold neural networks as follows.

Without the noise injection, the common backpropagation algorithm [19], [26], [31] is not applicable for training a feedforward threshold neural network, because the threshold function is nondifferentiable at the discontinuity points and with zero gradients for the piecewise constants [16], [43]. However, due to the injection of noise, it is seen that the hidden unit h_k in (3) is effectively equivalent to a noise-smoothed differentiable activation function h_k^{∞} that is a function of the input data $\boldsymbol{x}$, the weight vector $[\boldsymbol{W}]^{(k)}$ and the noise PDF f_η (including the noise level σ). A special example of the noise-smoothed threshold activation function h_k^{∞} is shown in Fig. 2 and its gradients become accessible. Therefore, we can adopt the backpropagation learning rule to minimize the instantaneous error energy $\mathcal{E}(\ell)$ through applying the gradient descent method, and treat the noise-level σ as a learnable network parameter. Defining the gradients $\partial\mathcal{E}(\ell)/\partial\boldsymbol{W}$, $\partial\mathcal{E}(\ell)/\partial\boldsymbol{U}$ and $\partial\mathcal{E}(\ell)/\partial\sigma$ of the instantaneous error energy $\mathcal{E}(\ell)$ with respect to weight matrices $\boldsymbol{W}$, $\boldsymbol{U}$ and the noise-level σ, the update rule for the ℓth training example can be expressed as

$$\Theta(\ell) = \Theta(\ell-1) - \alpha \frac{\partial \mathcal{E}(\ell)}{\partial \Theta}\Big|_{\Theta=\Theta(\ell-1)} \tag{6}$$

where the network parameter $\Theta \in \{\boldsymbol{U}, \boldsymbol{W}, \sigma\}$, $\Theta(0)$ denotes the initial values and the learning rate $\alpha > 0$.

The update procedure of (6) is carried on example-by-example, from the first training example $\{\boldsymbol{x}(0), \boldsymbol{s}(0)\}$ to the last one $\{\boldsymbol{x}(L), \boldsymbol{s}(L)\}$, which constitutes the epoch p of training over the whole training set. Then, the adjustments to the network parameters are continued on an epoch-by-epoch. For clarity, the proposed noise-boosted backpropagation learning for the designed threshold neural network in Fig. 1 is presented in the Algorithm 1.

Algorithm 1 Noise-Boosted Backpropagation Learning

Input: $\{\boldsymbol{x}(\ell), \boldsymbol{s}(\ell)\}_{\ell=1}^{L}$, $\boldsymbol{W}(0)$, $\boldsymbol{U}(0)$, $\sigma(0)$, P, α.
Output: $\Theta(L) \in \{[\boldsymbol{U}]_{mk}, [\boldsymbol{W}]_{kn}, \sigma\}$.
for *training epoch* $p = 1 \to P$ **do**
  $\mathcal{E}_{\text{tot}} \leftarrow 0$;
  for *training example* $\ell = 1 \to L$ **do**
    Feedforward procedure:
    $v_k(\ell) \leftarrow [\boldsymbol{W}]^{(k)}(\ell-1)\boldsymbol{x}(\ell)$;
    $h_k^{\infty}(\ell) \leftarrow \mathrm{E}_\eta[\phi(v_k(\ell)+\eta)]$;
    $y_m(\ell) \leftarrow \psi\{[\boldsymbol{U}]^{(m)}(\ell-1)\boldsymbol{h}(\ell)\}$;
    $e_m(\ell) \leftarrow s_m(\ell) - y_m(\ell)$;
    $\mathcal{E}(\ell) \leftarrow \frac{1}{2}\sum_{m=1}^{M} e_m^2(\ell)$;
    $\mathcal{E}_{\text{tot}} \leftarrow \mathcal{E}_{\text{tot}} + \mathcal{E}(\ell)$;
    Backpropagation procedure:
    $\Theta(\ell) \leftarrow \Theta(\ell-1) - \alpha \frac{\partial \mathcal{E}(\ell)}{\partial \Theta}\Big|_{\Theta=\Theta(\ell-1)}$;
  end
  $\Theta(0) \leftarrow \Theta(L)$.
end

III. Results of Threshold Neural Networks With Noise Injection

A. Threshold Neuron

We first consider the hidden unit h_k consisting of the threshold activation function [32], [34], or the McCulloch–Pitts neuron [44], as

$$\phi(u) = \begin{cases} 1, & u \geq \theta_k \\ 0, & u < \theta_k \end{cases} \tag{7}$$

where θ_k is the threshold parameter and also assumed to be learnable. The mutually independent noise components η_{kt} injected into the hidden units h_k are assumed to be with the common Gaussian PDF $f_\eta(x) = \exp(-x^2/2\sigma^2)/(2\pi\sigma^2)^{1/2}$ and the same noise level σ. The activation function in the output layer is taken as the linear transformation $\psi(x) = x$.

With the noise-smoothed activation function of (7) and the injected Gaussian noise, we can express the hidden unit h_k^{∞} as

$$h_k^{\infty} = \mathrm{E}_\eta[\phi(v_k+\eta)] = \int_{\left(\frac{\theta_k - v_k}{\sigma}\right)}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx \tag{8}$$

and deduce the gradients $\partial\mathcal{E}(\ell)/\partial\Theta$ in Appendix B for the network parameter $\Theta \in \{\boldsymbol{U}, \boldsymbol{W}, \sigma\}$.

Lemma 2: For the designed threshold neural network with the hidden units h_k^{∞} of (8), the noise-boosted backpropagation Algorithm 1 is weakly convergent.

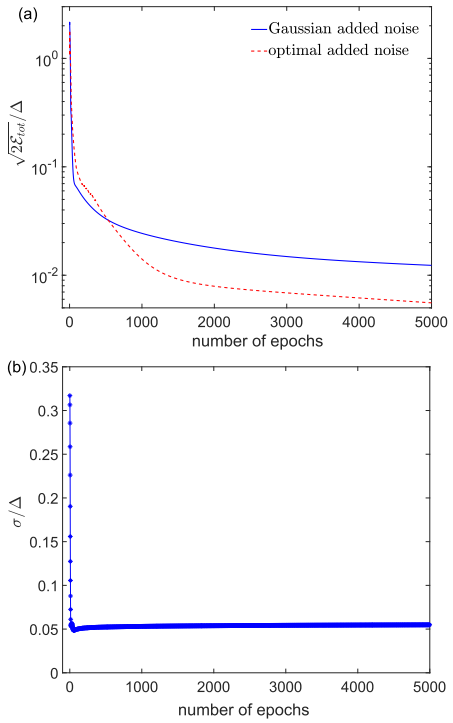

Fig. 3. Learning curves of (a) rms error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ (blue solid line) and (b) injected Gaussian noise-level σ/Δ versus the number of epochs by the noise-boosted backpropagation Algorithm 1 for the unidimensional function in (9). Here, the maximum difference of the function in (9) is $\Delta = 3.005$ in the interval $[-2, 2]$. The red dashed line represents the learning curve of $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ with the optimal injected noise obtained by the PDF $f_\eta^*(\eta)$ of (12) in Section III-C.

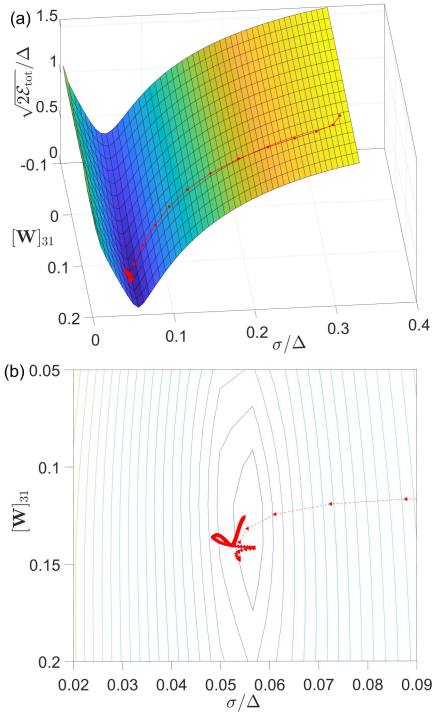

Fig. 4. (a) Surface and (b) corresponding contour of the rms error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ as a function of the element $[\boldsymbol{W}]_{31}$ of the weight matrix $\boldsymbol{W}$ and the noise-level σ/Δ with respect to the unidimensional function in (9). The learning curve of $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ (red triangle trajectory) versus $[\boldsymbol{W}]_{31}$ and σ/Δ is also plotted. The other parameters are the same as shown in Fig. 3.

It is then proven in Appendix C that the gradients are Lipschitz continuous in the definition domain of $-\infty < t < \infty$ and $\sigma \geq 0$, and the noise-boosted backpropagation Algorithm 1 is weakly convergent. Thus, Lemma 2 holds. Therefore, we can find a local optimum solution of the noise level in the training procedure of threshold neural networks by the Algorithm 1, and when the convergence is obtained at a noise level $\sigma > 0$, the beneficial role of injected Gaussian noise will be manifested.

First, we investigate the approximation capability of the constructed feedforward network for some benchmark nonlinear functions. The training set $\{\boldsymbol{x}(\ell), \boldsymbol{s}(\ell)\}_{\ell=1}^{L}$ is generated from an unidimensional function

$$f(x) = \sin(2x) + 2\exp(-10x^2). \tag{9}$$

Here, the length of data is $L = 41$, input examples $x(\ell)$ are generated equally spaced in the interval $[-2, 2]$ and the corresponding function values $s(\ell) = f[x(\ell)]$ are also recorded for $\ell = 1, 2, \ldots, L$. Then, a feedforward neural network $(N \times K \times M)$ with $N = 1$ input neuron, $K = 50$ hidden units h_k, and $M = 1$ output neuron is trained to fit the training set $\{\boldsymbol{x}(\ell), \boldsymbol{s}(\ell)\}_{\ell=1}^{L}$ sampled from the target function of (9). Here, the learning rate takes $\alpha = 0.01$, the initial noise-level $\sigma(0) = 1$, and the initial weight vectors $\boldsymbol{W}(0)$ and $\boldsymbol{U}(0)$ are uniformly distributed in the interval $[-1, 1]$.

Using the proposed noise-boosted backpropagation Algorithm 1, the learning curves of the total error energy $\mathcal{E}_{\rm tot}$ and the noise-level σ can be obtained as a function of the number of epochs. Here, for reference, the maximum difference Δ of the target function in the interval $[a, b]$ is defined as

$$\Delta = \max f(x) - \min f(x) \quad \forall x \in [a, b]. \tag{10}$$

For the unidimensional target function in (9) in the interval $[-2, 2]$, the maximum difference $\Delta = 3.005$. Then, the root-mean-square (rms) error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ (blue solid line) and the noise-level σ/Δ are plotted as a function of the number of epochs in Fig. 3(a) and (b), respectively. It is shown in Fig. 3(a) that, upon increasing the epoch number, the rms error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ first greatly decreases and then reaches convergence effectively. For instance, $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta \leq 10^{-1}$ after about 45 epochs of training. In addition, it is also observed in Fig. 3(b) that, after 200 epochs of training, the noise-level σ/Δ $(*)$ also converges to 0.054, which clearly indicates a (local) optimized nonzero Gaussian noise being necessary to achieve the best performance $\mathcal{E}_{\rm tot}$ of the trained threshold neural network. However, the noise-level σ/Δ does not stay at the local optimized point 0.054 and experiences a slight increase as the training epoch number increases, as shown in Fig. 3(b). The reason causing this result can be explained in Fig. 4: the total error energy $\mathcal{E}_{\rm tot}$ is not a strictly convex function of the noise level and the weight coefficients (for simplicity, only illustrating the element $[\boldsymbol{W}]_{31}$ of the weight matrix $\boldsymbol{W}$), and the learning curve of the rms error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ moves slowly in the flat bottom of the valley.

Next, we validate whether the converged noise level $\sigma/\Delta = 0.054$ in the above training experiment is consistent with the optimized one that corresponds to the minimum total

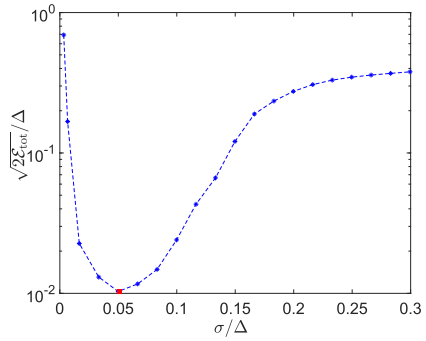

Fig. 5. RMS error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ of the total error energy $\mathcal{E}_{\rm tot}$ as a function of noise-level σ/Δ for the target function in (9). Here, each point of $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ is obtained by fixing noise-level σ/Δ but training other network parameters with the backpropagation learning rule. The other parameters are the same as shown in Fig. 2.

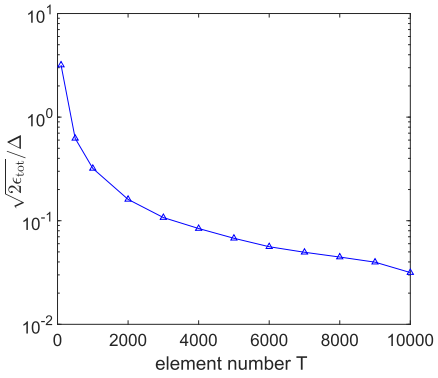

Fig. 6. RMS error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ for 10^3 testing points of the target unidimensional function of (9) versus the number T of threshold elements. Here, 10^2 trials are realized for each point of experimental results, and other parameters are the same as shown in Fig. 3.

error energy obtained by the conventional stochastic resonance method. By fixing the noise-level σ, the weight matrices $\boldsymbol{W}$, $\boldsymbol{U}$ and the threshold θ_k of the feedforward neural network are trained for 5000 epochs by the backpropagation learning rule. Then, the resonance curve of the rms error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ of the total error energy $\mathcal{E}_{\rm tot}$ is illustrated as a function of the noise level σ/Δ in Fig. 5. It is seen in Fig. 5 that the optimized noise-level σ/Δ (■) corresponding to the lowest $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ agrees well with the converged value of 0.054 shown in Fig. 3(b). This fact also demonstrates the validity and practicability of the proposed noise-boosted backpropagation learning algorithm to adaptively optimize the noise level as part of the learning process of function approximations.

After 5000 times of training epochs, the trained network parameters $\boldsymbol{W}$, $\boldsymbol{U}$, θ_k and the converged noise-level σ define the designed threshold neural network. However, it is noted that the hidden unit h_k^{∞} in (3) is a limit expression that would be obtained with threshold neurons activated by an infinite number of mutually independent noise components, which is impossible to implement in practice. Therefore, in the test experiments, the hidden unit h_k is composed of a finite number T of threshold activation functions in (7), which are activated by mutually independent noise components with the same converged noise-level σ. For 10^3 test input data $x(\ell)$ equally spaced in the interval $[-2, 2]$, we simulate the trained threshold neural network for 10^2 times. For each trail, the $K \times T$ mutually independent noise components are randomly

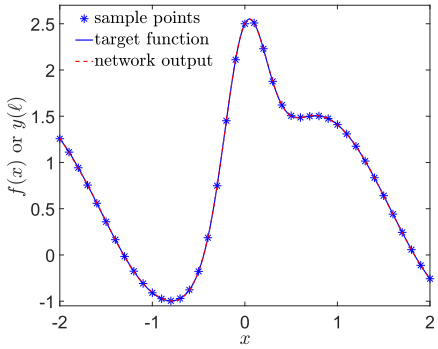

Fig. 7. Approximation (red dashed line) of the target function of (9) obtained by the trained feedforward neural network for 10^3 testing points. The $L = 41$ training data (∗) and the target function (blue solid line) of (9) are also plotted. Other parameters are the same as shown in Fig. 3.

generated and injected into the hidden units h_k in (2) for $k = 1, 2, \ldots, K$. Then, we average the outputs of the network as the approximated function for testing the target function of (9). For different numbers T of threshold elements, the rms error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$, as shown in Fig. 6, are experimentally obtained for 10^2 trials. It is seen in Fig. 6 that, for a sufficiently large number $T = 10^4$ of the threshold elements, the rms error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ has a relatively small statistical mean value 0.0316. Then, using the $T = 10^4$ threshold elements in each hidden units in the testing phase, the outputs (red dashed line) of the proposed threshold network are presented in Fig. 7. For comparison, the target function of (9) (blue line covered by the network outputs) and the $L = 41$ training data (∗) are also plotted in Fig. 7. It is seen in Fig. 7 that the trained threshold neural network assisted by the addition of noise performs well on the test for approximating the target unidimensional function of (9).

We also test a benchmark 2-D function [26]

$$f(x_1, x_2) = \max\{e^{-10x_1^2}, e^{-50x_2^2}, 1.25e^{-5(x_1^2+x_2^2)}\}. \quad (11)$$

The designed threshold neural network is with the layer size of $N \times K \times M = 2 \times 100 \times 1$. The $L = 11 \times 11$ training set contains the data $\boldsymbol{x}(\ell) = [x_1(\ell), x_2(\ell)]^{\top}$ that are equally spaced in the range $[-1, 1] \times [-1, 1]$ and the corresponding function values $s(\ell) = f[x_1(\ell), x_2(\ell)]$ of (11). Here, for the target function in (11), the maximum difference $\Delta = 1.2499$ in the range $[-1, 1] \times [-1, 1]$. Using the proposed backpropagation learning Algorithm 1, the rms error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ is obtained as $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta = 0.1864$ and the converged noise-level $\sigma/\Delta = 0.0258$ after training 2×10^4 epochs, as shown in Fig. 8(a) (blue solid line) and (b), respectively. The approximation (patched surface) of the trained neural network and the training data (⋆) are illustrated in Fig. 9(a), and the relative error $|y(x_1, x_2) - f(x_1, x_2)|/\Delta$ between the neural network outputs $y(x_1, x_2)$ and the training function $f(x_1, x_2)$ is plotted in Fig. 9(b). The maximum relative error $\max|y(x_1, x_2) - f(x_1, x_2)|/\Delta = 0.0450$. For $L = 21 \times 21$ testing data equally spaced in the range $[-1, 1] \times [-1, 1]$, the rms error $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta$ is given by $(2\mathcal{E}_{\rm tot})^{1/2}/\Delta = 2.0625$ and the maximum relative error $\max|y(x_1, x_2) - f(x_1, x_2)|/\Delta = 0.5067$.

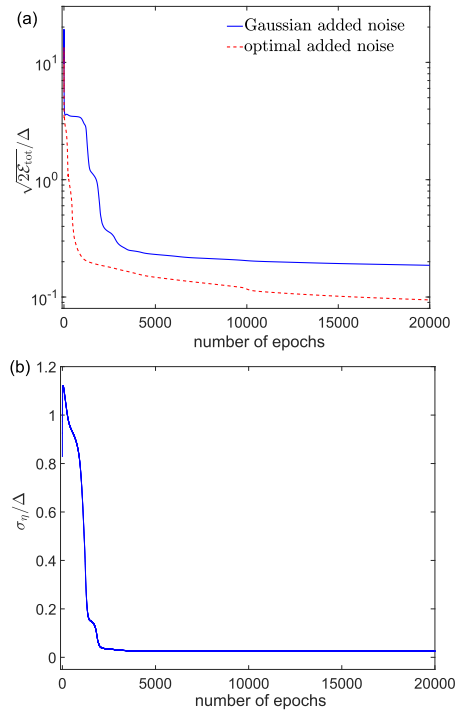

Fig. 8. Learning curves of (a) rms error $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta$ by injecting the Gaussian noise (blue solid line) and the optimal noise (red dashed line) with the PDF f_η^* of (12) in Section III-C. (b) Gaussian noise level σ/Δ in (12) versus the number of epochs. Here, the noise-boosted backpropagation Algorithm 1 is applied to the threshold neural network for approximating the 2-D function in (11). The maximum difference of the function in (11) is $\Delta = 1.2499$ in the interval $[-1, 1] \times [-1, 1]$.

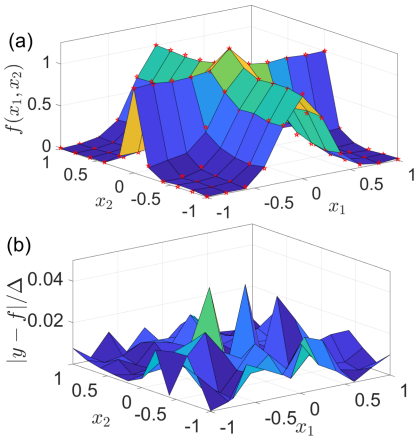

Fig. 9. (a) Outputs $y(x_1, x_2)$ of the trained neural network as the approximation (patched surface) to the training data ($\star$) of the 2-D function $f(x_1, x_2)$ in (11) in the range $[-1, 1] \times [-1, 1]$. (b) Corresponding relative error $|y(x_1, x_2) - f(x_1, x_2)|/\Delta$ between the neural network output and the testing data.

B. Validation of the Threshold Neural Network on the Real-World Dataset

Furthermore, we validate the proposed backpropagation learning Algorithm 1 on five real-world datasets [45]–[48] in the designed feedforward threshold neural network. The $N \times K \times 1$ ($K = 5, 20$, and 30) three-layer network is trained by the real-world N-dimensional datasets of Auto MPG ($N = 7$) [45], Housing ($N = 13$) [46], Airfoil noise ($N = 5$) [45], Wine quality ($N = 11$) [47], and QSAR fish toxicity ($N = 6$) [48] in a computer equipped with CPU of Intel Core i7-7820HK at 2.90 GHz and 32G RAM DDR4 at 2400 MHz. Here, the datasets contain 198, 253, 300, 980, and 908 examples, respectively. Using the two-eight rule, 80% of data are used for training, while 20% of data are employed to test the trained threshold neural network.

TABLE I

COMPARISON OF TESTING RESULTS OF $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta$ OF THE NOISE-BOOSTED THRESHOLD NEURAL NETWORK AND THE ONE WITH THE SIGMOID ACTIVATION FUNCTION

Number K / Data set	$K = 5$		$K = 20$		$K = 30$	
	threshold	sigmoid	threshold	sigmoid	threshold	sigmoid
Auto MPG [45]	0.0827	0.0913	0.0838	0.0956	0.0876	0.1001
Housing [46]	0.0857	0.0890	0.0633	0.0922	0.0734	0.0959
Airfoil noise [45]	0.1224	0.1490	0.1207	0.1492	0.1101	0.1488
Wine quality [47]	0.1753	0.1763	0.1742	0.1759	0.1724	0.1763
QSAR fish toxicity[48]	0.0994	0.1040	0.0990	0.1046	0.1034	0.1047

TABLE II

COMPARISON OF THE CONVERGENCE TIME OF TRAINING THE NOISE-BOOSTED THRESHOLD NEURAL NETWORK AND THE NETWORK WITH THE SIGMOID ACTIVATION FUNCTION FOR ATTAINING THE SAME VALUE OF THE RMS ERROR $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta$ (UNIT: SECOND)

Number K / Data set	$K = 5$		$K = 20$		$K = 30$	
	threshold	sigmoid	threshold	sigmoid	threshold	sigmoid
Auto MPG [45]	0.2663	1.0163	0.1247	0.5027	0.2225	0.7081
Housing [46]	0.7490	1.3604	0.1985	1.7214	0.5664	2.3088
Airfoil noise[45]	0.8607	1.6895	1.2088	1.4262	0.4767	1.1729
Wine quality [47]	5.2061	2.5871	0.7949	2.1044	1.1327	2.5631
QSAR fish toxicity[48]	0.6622	1.1938	0.6094	0.7719	0.2394	0.8029

Table I reports the test results of the rms error $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta$ of the noise-boosted neural network with the threshold activation function. For comparison, the classical neural network with the smooth sigmoid activation function of (16) is also tested, and the corresponding rms error $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta$ is presented in Table I. It is interesting to note in Table I that, for testing five real-world datasets, the proposed threshold neural network can predict the test data with a smaller rms error $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta$ than the neural network with the smooth sigmoid activation function does. This demonstrates the superiority of the proposed backpropagation learning method in the threshold neural network for solving the practical multivariate regression problem. Of course, the apparent cost of employing the noise-boosted threshold neural network is the demand of more memory storage for storing the mutually independent KT noise samples and requiring more addition and XOR operations in the testing phase. For instance, when the number T in $K = 30$ hidden units of the threshold neural network takes 10^4, the memory storage of the noise samples occupies about 2-Mb memory in the computer RAM.

In Table II, we also compare convergence times of training both the noise-boosted threshold neural network and the network with the sigmoid activation function to attain the same level of the rms error $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta$ of a given dataset. Although the noise-boosted threshold neural network has extra parameters of noise-level σ and threshold θ to be learned, it is shown in Table II that the convergence time of training

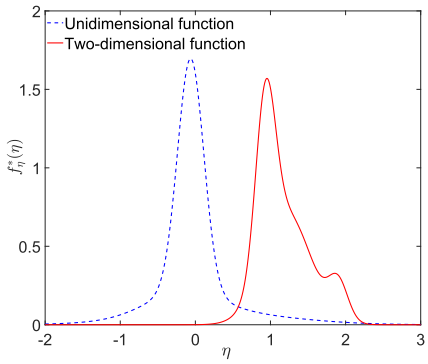

Fig. 10. Approximated optimal noise PDF $f_\eta^*(\eta)$ for the threshold neural networks with respect to the unidimensional function in (9) (blue dashed line) and the 2-D function in (11) (red solid line). The kernel function number $J = 4$. Other parameters are the same as shown in Figs. 3(a) and 8(a).

the noise-boosted threshold neural network is lower than that of the network with the sigmoid activation function in most cases. The reason is that the designed neural network with its size $N \times K \times 1$ ($K \leq 30$) is actually not deep, and the extra network parameters have low memory requirement in the training phase. The introduction of the learnable parameters of noise level σ and threshold θ leads to a more powerful learning capacity of the noise-boosted threshold neural network. Therefore, for reaching the same value of the rms error $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta = 0.009, 0.012, 0.017, 0.017$ and 0.01 for five real-world datasets respectively, Table II demonstrates another superiority of the noise-boosted threshold neural network in the convergence time of training phase.

C. Optimal Noise Type for Threshold Neural Networks

In the above-mentioned experiments, only the Gaussian noise injected into the feedforward threshold neural network is taken into account. It is natural to consider the injection of other noise types into the network and find the optimal injected noise type to achieve the minimum total error energy $\mathcal{E}_{\text{tot}}$ with respect to the noise PDF $f_\eta(\eta)$. However, it is usually analytically intractable to obtain the optimal noise PDF $f_\eta^{\text{opt}}(\eta)$ [35], [42]. Usually, a kernel method with the form as [35], [42]

$$f_\eta^*(\eta) = \sum_{j=1}^{J} \lambda_j g\left(\eta, \mu_j, \sigma_j\right) \tag{12}$$

is employed to approximate the optimal noise PDF $f_\eta^{\text{opt}}(\eta)$, where the normalization coefficients $\lambda_j \geq 0$ satisfy the constraint $\sum_{j=1}^{J} \lambda_j = 1$, and the Gaussian kernel function $g\left(\eta, \mu_j, \sigma_j\right) = \exp[-(\eta-\mu_j)^2/2\sigma_j^2]/(2\pi\sigma_j^2)^{1/2}$ is with mean μ_j and standard deviation $\sigma_j \geq 0$ for $j = 1, 2, \ldots, J$. With this approximate PDF of (12), the outputs of the hidden units in (3) can be reexpressed as

$$h_k^{\infty} = \text{E}_\eta[\phi(\upsilon_k + \eta)] = \sum_{j=1}^{J} \lambda_j \text{E}_g[\phi(\upsilon_k + \eta)] \tag{13}$$

where the expectation operator $\text{E}_g(\cdot) = \int \cdot\ g\left(\eta, \mu_j, \sigma_j\right) d\eta$. Therefore, for a given kernel function number $J > 1$, the minimization problem of the total error energy $\mathcal{E}_{\text{tot}}$ with respect to the optimal noise can be simplified as a finite-dimensional constrained optimization

$$\min_{\{\lambda_j, \mu_j, \sigma_k\}} \mathcal{E}_{\text{tot}}$$
$$\text{s.t. } \lambda_j \geq 0, \quad \sum_{j=1}^{J} \lambda_j = 1, \ \sigma_j \geq 0 \tag{14}$$

with respect to parameters λ_j, μ_j and σ_j for $j = 1, 2, \ldots, J$. Under such circumstances, the training procedure of (6) updates the weight matrices $\boldsymbol{W}$ and $\boldsymbol{U}$ by the gradients $\partial\mathcal{E}(\ell)/\partial\boldsymbol{W}$ and $\partial\mathcal{E}(\ell)/\partial\boldsymbol{U}$, but searches the approximate optimal PDF $f_\eta^*(\eta)$ of (12) by the sequential quadratic programming (SQP) method [35], [42], [49], which can resort to the existing commercial software package of constrained nonlinear optimization [53]. At each iteration of a quadratic programming subproblem, an approximation is made of the Hessian of the Lagrangian function using a quasi-Newton (BFGS) updating method [49]. Then, in the proposed backpropagation Algorithm 1, the Gaussian noise needs to be replaced by the updating procedure of the PDF $f_\eta^*(\eta)$, while the learning rules of the weight matrices still keep updating by their corresponding gradients. For clarity, the optimal noise-boosted backpropagation learning is described in Algorithm 2. Moreover, it is seen that the hidden unit in (13) is a linear combination of the expectations $\text{E}_g[\phi(\upsilon_k+\eta)]$, therefore the Lemma 2 holds and the optimal noise-boosted backpropagation Algorithm 2 with online searching the optimal noise type is still weakly convergent.

Algorithm 2 Optimal Noise-Boosted Backpropagation Learning

Input: $\{\boldsymbol{x}(\ell), \boldsymbol{s}(\ell)\}_{\ell=1}^{L}$, $\boldsymbol{W}(0)$, $\boldsymbol{U}(0)$, $\lambda_j(0)$, $\mu_j(0)$, $\sigma_j(0)$, P, α.
Output: $\Theta(L) \in \{[\boldsymbol{U}]_{mk}, [\boldsymbol{W}]_{kn}\}$, $f_\eta^*(\eta)$.
for *training epoch* $p = 1 \to P$ **do**
  $\mathcal{E}_{\text{tot}} \leftarrow 0$;
  for *training example* $\ell = 1 \to L$ **do**
    Feedforward procedure:
    $\upsilon_k(\ell) \leftarrow [\boldsymbol{W}]^{(k)}(\ell-1)\boldsymbol{x}(\ell)$;
    $h_k^{\infty}(\ell) \leftarrow \sum_{j=1}^{J} \lambda_j \text{E}_g[\phi(\upsilon_k+\eta)]$;
    $y_m(\ell) \leftarrow \psi\{[\boldsymbol{U}]^{(m)}(\ell-1)\boldsymbol{h}(\ell)\}$;
    $e_m(\ell) \leftarrow s_m(\ell) - y_m(\ell)$;
    $\mathcal{E}(\ell) \leftarrow \frac{1}{2}\sum_{m=1}^{M} e_m^2(\ell)$;
    $\mathcal{E}_{\text{tot}} \leftarrow \mathcal{E}_{\text{tot}} + \mathcal{E}(\ell)$;
    Backpropagation procedure:
    $\Theta(\ell) \leftarrow \Theta(\ell-1) - \alpha \frac{\partial\mathcal{E}(\ell)}{\partial\Theta}\Big|_{\Theta=\Theta(\ell-1)}$;
    $\{\lambda_j(\ell), \mu_j(\ell), \sigma_j(\ell)\} \leftarrow$ SQP method;
  end
  $\Theta(0) \leftarrow \Theta(L)$.
end

Using this optimal noise-boosted backpropagation learning rule in Algorithm 2, the learning curves of the rms error $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta$ for the unidimensional function in (9) and the 2-D function in (11) are shown in Figs. 3(a) and 8(a) (red dashed lines), respectively. It is seen in Figs. 3(a) and 8(a) that the trained threshold neural network with the optimized noise indicated by the PDF $f_\eta^*(\eta)$ can outperform that with the injected Gaussian noise. The finally obtained noise PDFs $f_\eta^*(\eta)$ after the training procedure are also shown in Fig. 10. According to the optimized noise PDF $f_\eta^*(\eta)$, we generate

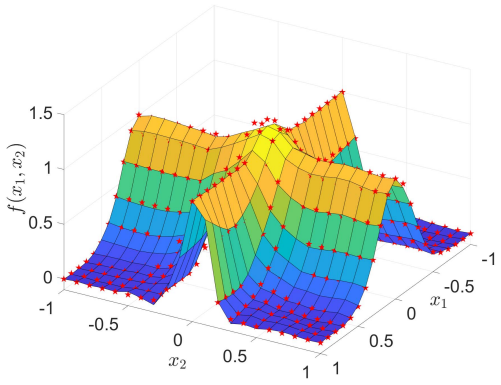

Fig. 11. Outputs of the trained threshold neural network as the approximation (patched surface) to the 21 × 21 testing data (⋆) of the 2-D function $f(x_1, x_2)$ in (11) in the range $[-1, 1] \times [-1, 1]$. The number of threshold activation functions in each hidden unit is $T = 10^4$ and other parameters are the same as shown in Fig. 9.

$T = 10^4$ samples of the noise using the slice-sampling method [50] and inject them into each hidden unit of the threshold networks for testing data. For instance, the approximation (patched surface) of the trained neural network and the 21 × 21 testing data (⋆) of the 2-D function in (11) are illustrated in Fig. 11 in the range of $(x_1, x_2) \in [-1, 1] \times [-1, 1]$. The rms error $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta = 0.6254$ and the maximum relative error $\max|y(x_1, x_2) - f(x_1, x_2)|/\Delta = 0.2048$. Compared with the results of $(2\mathcal{E}_{\text{tot}})^{1/2}/\Delta = 2.0625$ and the maximum relative error $\max|y(x_1, x_2) - f(x_1, x_2)|/\Delta = 0.5067$ by injecting Gaussian noise, the trained feedforward neural network with the optimized noise performs well on the test data. There are also some distinct differences between the network outputs and the testing points of the 2-D function in (11), and the reason lies in the finite number J of the kernel function in (12). It is known that, as the number of kernel functions J increases, the approximate optimal noise PDF $f_\eta^*(\eta)$ can converge to the optimal one $f_\eta^{\text{opt}}(\eta)$ if it exists [42], [49]. However, the acceleration of the training procedure is an open problem for the application of the noise-boosted backpropagation learning algorithm to the threshold neural network, and further studies of the optimal-injected noise type are being carried out for function approximation and pattern classification.

D. Hard-Sigmoid and Sigmoid Thresholds

Next, it is interesting to investigate whether the proposed algorithm is applicable to other activation functions or not. Here, we consider a kind of piecewise linear functions named as the hard-sigmoid [9]

$$\phi(u) = \begin{cases} 1, & u > \bar{\theta}_k \\ 0.25(u - \theta_k) + 0.5, & \underline{\theta}_k \le u \le \bar{\theta}_k \\ 0, & u < \underline{\theta}_k \end{cases} \tag{15}$$

and a frequently-used sigmoid activation function defined as

$$\phi(u) = [1 + e^{-(x-\theta_k)}]^{-1} \tag{16}$$

where the threshold parameters are θ_k, $\bar{\theta}_k = \theta_k + 2$ and $\underline{\theta}_k = \theta_k - 2$ [1]–[3], [9]. For a special threshold value $\theta_k = 1.5$, the input–output characteristics of the hard-sigmoid function in (15) and the sigmoid function in (16) are illustrated in Fig. 2.

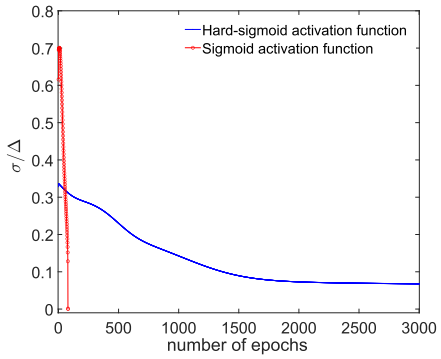

Fig. 12. Learning curves of the noise-level σ/Δ versus the number of epochs for the noise-boosted backpropagation algorithm with respect to the unidimensional function in (9). Here, hidden units h_k of the feedforward neural networks (1 × 50 × 1) are composed of the hard-sigmoid function in (15) and the sigmoid function in (16), respectively. The other parameters are the same as in Fig. 3.

In Appendix B, the partial derivatives of the instantaneous error energy $\mathcal{E}(\ell)$ with respect to network parameter $\Theta \in \{\boldsymbol{W}, \boldsymbol{U}, \theta_k, \sigma\}$ are derived. Using the Algorithm 1, we train the feedforward neural networks ($N \times K \times M = 1 \times 50 \times 1$) with hidden units composed of the hard-sigmoid function in (15) and the sigmoid function in (16), respectively. For approximating the unidimensional function in (9), the learning curves of the noise level σ/Δ are plotted in Fig. 12. It is interesting to note that the converged noise-level $\sigma/\Delta = 0.0677$ is not zero, and the injection of noise improves training the feedforward neural networks composed of the hard-sigmoid function in (15). However, it is seen in Fig. 12 that the injection of noise is unnecessary for the neural networks composed of the sigmoid function in (16). The reason is that the sigmoid function in (16) is continuous and has no constant parts, and then the gradient of the total error energy with respect to the network parameters is always accessible and nonzero for the backpropagation learning. However, the feedforward network with the zero-gradient activation function, e.g. (7) and (15), could benefit from the noise injection into the hidden layer for approximating function approximation via the proposed backpropagation learning.

IV. Conclusion

In this article, an online backpropagation learning algorithm enhanced with the injection of noise is proposed for training the feedforward threshold neural network. When the activation function has hard-limiting input–output characteristics, the conventional gradient-based backpropagation learning procedure meets the difficulty of zero or undefined gradients but is inapplicable in practice. The proposed backpropagation learning algorithm complements the conventional one by injecting mutually independent noise components into a sufficiently large number of activation functions. Due to the noise injection, the hidden unit in the hidden layer of networks can be viewed as a noise-smoothed function, and the loss function with respect to the network parameter becomes continuously differentiable, with a nonvanishing gradient everywhere that can be used for efficient backpropagation training.

Fig. 13. Examples of (a) input handwritten image and (b) output image of the autoencoder with size of $(784 \times 256 \times 64 \times 10) \times (64 \times 256 \times 784)$. (c) Generated image of the decoder part of the autoencoder by feeding a deviated low-dimensional vector into the code layer.

A key feature of the proposed noise-boosted backpropagation learning algorithm is that the noise level or the noise PDF, as well as other network parameters, are learnable and updated by the gradient-based rule. An essential observation is that the learning process is generally found to converge to a nonzero optimized level of noise, indicating that the nonzero optimized noise has a beneficial effect, both in the learning but also in the retrieval phase of the neural network. With the trained weights and the converged nonzero noise level or the approximated optimal noise PDF, experimental results show the applicability of the designed noise-boosted backpropagation algorithm for approximating nonlinear functions and practical multivariate regression of real-world dataset by the trained threshold neural networks operated with the injection of mutually independent noise components in hidden layers.

This proposed noise-boosted backpropagation learning algorithm benefits from the noise in the hidden layer to allow the training of neural networks with a much wider family of activation functions that are nondifferentiable or with zero derivatives. However, some open questions remain. For instance, the numerical computation of the gradient of the total error energy with respect to model parameters is time consuming, the convergence rate of the training procedure is rather low, and the convergent noise-level sways and is not stable (see learning curves in Figs. 3 and 8). Thus, some improved learning rules, such as Adam [51] and Levenberg–Marquardt method [49], can potentially be developed to accelerate the training process of the studied neural networks in this article. Although the optimal noise type is analyzed, a crucial problem of the existence and uniqueness of the optimal injected noise for the threshold neural networks remains to be solved.

In addition, the improvement of injecting noise into the hidden layer on the generalization performance of the threshold neural network is not studied for the inputs corrupted by some preexisting noise. In the line with the argument of noise injection equivalent to a smooth Tikhonov regularization [1] of the total error energy, the noise injected into the hidden layer of the threshold neural network is also expected to possibly play a regularization role in constraining the network model to be less sensitive to the noisy inputs.

Finally, the noise-boosted threshold neural network considered in this article has a shallow and simple size of $N \times K \times M$. A natural question is whether the proposed backpropagation learning algorithm can potentially be applied to deeper neural networks, e.g. deep convolutional neural networks [13] and long-short-term memory (LSTM) networks [9], [18], or not. Moreover, the learning manner of the proposed neural network is based on the supervised learning principle. To investigate the possibility of also operating this network in an unsupervised manner, we here make a trial of the proposed backpropagation learning algorithm in an autoencoder with a deep size of $(784 \times 256 \times 64) \times 10 \times (64 \times 256 \times 784)$. Besides the first input layer with the linear activation function, other layers are all composed of the noise-smoothed function defined in (8). The fourth layer is the code layer, and the $784 \times 256 \times 64 \times 10$ network part performs the encoding or compressing function of transforming the high-dimensional 784×1 vector into a low-dimensional 10×1 code vector, and the decoder with the $10 \times 64 \times 256 \times 784$ network part does the exact opposite of the compression for recovering the low-dimensional data from the code layer [19]. The whole $(784 \times 256 \times 64) \times 10 \times (64 \times 256 \times 784)$ network is called an autoencoder [19]. From the MNIST dataset [54] without labels, each black-and-white 28×28 pixel handwritten image is mapped into a 784×1 input vector $\boldsymbol{x}$ for training the designed autoencoder via the unsupervised learning method [19]. It is interesting to note that this deep autoencoder with the noise-smoothed activation function can learn well and yields the low-dimensional code as a tool to reduce the dimensionality of data. For example, after 10^3 epochs of the unsupervised learning of 5×10^4 handwritten images, an input handwritten image illustratively shown in Fig. 13(a) is compressed by the encoder as a low-dimensional vector $[0.3366, 0.6974, 0.2757, 0.6797, 0, 0.4564, 0.2855, 0, 0.6105, 1.0]^\top$ at the code layer. By contrast, with this low-dimensional vector, the decoder produces a visualization of data in Fig.13(b). The unsupervised learning rule of minimizing the cross-entropy error between the pixel intensities of the original image and the reconstructed one [19] is also generalized via the introduction of noise into deep autoencoder networks.

To test the fast retrieval of the designed autoencoder, a vector $[0.3, 0.7, 0.3, 0.7, 0.1, 0.4, 0.3, 0.1, 0.6, 0.7]^\top$, deviating from the encoded vector with the 10° intersection angle, is feed into the decoder part of the autoencoder, and generates the reconstructed image as shown in Fig. 13(c). It is seen in Fig. 13(a) and (c) that the generated image is very similar to the original input one and clearly reestablishes the handwritten numeral of 7, which effectively demonstrates the generalization of the noise-boosted autoencoder. This example also illustrates that the noise-boosted proposed threshold neural network can be used in deeper architectures and with unsupervised learning. Therefore, further investigations of the noise-boosted backpropagation learning of feedforward threshold neural networks can be extended in various directions.

APPENDIX A
PROOF OF LEMMA 1

Proof: When we inject noise $\boldsymbol{\xi}$ into the input $\boldsymbol{x}$, the kth hidden unit has the same local field $v_k = [\boldsymbol{W}]^{(k)}(\boldsymbol{x} + \boldsymbol{\xi})$ and the output $h_k = \sum_{t=1}^{T} \phi(v_k)/T = \phi(v_k)$. In this circumstance, the total error energy in (5) can be calculated as

$$\begin{aligned}\widetilde{\mathcal{E}}_{\text{tot}} &= \frac{1}{2}\mathrm{E}_{\boldsymbol{\xi}}\left[\sum_{\ell=1}^{L}\sum_{m=1}^{M}\big(s_m(\ell) - y_m(\ell)\big)^2\right] \\ &= \sum_{\ell=1}^{L}\frac{1}{2}\mathrm{E}_{\boldsymbol{\xi}}\left\{\left\|\boldsymbol{s}(\ell) - \psi\Big[\boldsymbol{U}\phi\Big(\boldsymbol{W}\big(\boldsymbol{x}(\ell) + \boldsymbol{\xi}(\ell)\big)\Big)\Big]\right\|^2\right\}\end{aligned} \tag{17}$$

where $\|\cdot\|$ denotes the Euclidean norm of a vector $\boldsymbol{x}$. For the convex function $\|\boldsymbol{x}\|^2$ and using the Jensen inequality, we find

$$\begin{aligned}\widetilde{\mathcal{E}}_{\text{tot}} &\geq \sum_{\ell=1}^{L}\frac{1}{2}\Big\|\boldsymbol{s}(\ell)-\mathrm{E}_{\boldsymbol{\xi}}\{\psi\Big[\boldsymbol{U}\phi\Big(\boldsymbol{W}\big(\boldsymbol{x}(\ell)+\boldsymbol{\xi}(\ell)\big)\Big)\Big]\}\Big\|^2\\ &= \sum_{\ell=1}^{L}\frac{1}{2}\Big\|\boldsymbol{s}(\ell)-\psi\Big\{\boldsymbol{U}\mathrm{E}_{\boldsymbol{\xi}}\Big[\phi\Big(\boldsymbol{W}\boldsymbol{x}(\ell)+\boldsymbol{W}\boldsymbol{\xi}(\ell)\Big)\Big]\Big\}\Big\|^2\\ &= \sum_{\ell=1}^{L}\frac{1}{2}\Big\|\boldsymbol{s}(\ell)-\psi\Big\{\boldsymbol{U}\mathrm{E}_{\boldsymbol{\eta}}\Big[\phi\Big(\boldsymbol{W}\boldsymbol{x}(\ell)+\boldsymbol{\eta}(\ell)\Big)\Big]\Big\}\Big\|^2\\ &= \sum_{\ell=1}^{L}\frac{1}{2}\Big\|\boldsymbol{s}(\ell)-\psi\big\{\boldsymbol{U}\boldsymbol{h}^{\infty}\big\}\Big\|^2=\mathcal{E}_{\text{tot}} \end{aligned} \tag{18}$$

where the injected noise vector $\boldsymbol{\eta}(\ell)=\boldsymbol{W}\boldsymbol{\xi}(\ell)$ and the output vector $\boldsymbol{h}^{\infty}=[h_1^{\infty},h_2^{\infty},\ldots,h_K^{\infty}]^{\top}$ of the hidden layer. Moreover, the proof is the same for injecting noise into the weight matrix $\boldsymbol{W}$ by replacing the injected noise vector with $\boldsymbol{\eta}(\ell)=\boldsymbol{W}\boldsymbol{x}(\ell)$. Thus, the Lemma 1 holds.

Appendix B
Partial Derivatives in (6)

The partial derivative of $\mathcal{E}(\ell)$ with the mth row and the k-column element $[\boldsymbol{U}]_{mk}$ of the weigh matrix $\boldsymbol{U}$ can be calculated as

$$\frac{\partial\mathcal{E}(\ell)}{\partial[\boldsymbol{U}]_{mk}}=-e_m(\ell)\frac{\partial y_m(\ell)}{\partial[\boldsymbol{U}]_{mk}} \tag{19}$$

where $[\boldsymbol{U}]_{mk}$ denotes the weight connecting the neuron m in the output layer with the hidden unit h_k^{∞}. The partial derivative of $\mathcal{E}(\ell)$ with the element $[\boldsymbol{W}]_{kn}$ of the weigh matrix $\boldsymbol{W}$ is

$$\frac{\partial\mathcal{E}(\ell)}{\partial[\boldsymbol{W}]_{kn}}=-\frac{\partial h_k^{\infty}(\ell)}{\partial[\boldsymbol{W}]_{kn}}\sum_{m=1}^{M}e_m(\ell)\frac{\partial y_m(\ell)}{\partial h_k^{\infty}(\ell)} \tag{20}$$

where $[\boldsymbol{W}]_{kn}$ denotes the weight connecting the hidden unit h_k^{∞} with the nth element of the input vector $\boldsymbol{x}$. Similarly, the partial derivative of $\mathcal{E}(\ell)$ with respect to the noise-level σ can be calculated as

$$\frac{\partial\mathcal{E}(\ell)}{\partial\sigma}=-\sum_{k=1}^{K}\sum_{m=1}^{M}e_m(\ell)\frac{\partial y_m(\ell)}{\partial h_k^{\infty}(\ell)}\frac{\partial h_k^{\infty}(\ell)}{\partial\sigma}. \tag{21}$$

When the activation function of the output layer is taken as $\psi(x)=x$, we have the gradients

$$\frac{\partial y_m(\ell)}{\partial[\boldsymbol{U}]_{mk}}=h_k^{\infty}(\ell),\quad \frac{\partial y_m(\ell)}{\partial h_k^{\infty}(\ell)}=[\boldsymbol{U}]_{mk}. \tag{22}$$

For the threshold activation function in (7), we can further compute the gradients

$$\frac{\partial h_k^{\infty}}{\partial[\boldsymbol{W}]_{kn}}=\frac{x_n}{\sqrt{2\pi}\sigma}\exp\Big[-\frac{(\theta_k-\upsilon_k)^2}{2\sigma^2}\Big] \tag{23}$$

$$\frac{\partial h_k^{\infty}}{\partial\theta_k}=-\frac{1}{\sqrt{2\pi}\sigma}\exp\Big[-\frac{(\theta_k-\upsilon_k)^2}{2\sigma^2}\Big] \tag{24}$$

$$\frac{\partial h_k^{\infty}}{\partial\sigma}=\frac{\theta_k-\upsilon_k}{\sqrt{2\pi}\sigma^2}\exp\Big[-\frac{(\theta_k-\upsilon_k)^2}{2\sigma^2}\Big]. \tag{25}$$

Then, substituting (22)–(25) into (6), the learning rule can be implemented.

For the hard-sigmoid activation function in (15), we can express the hidden unit h_k^{∞} in (3) as

$$\begin{aligned}h_k^{\infty}=&\frac{1}{4}(\upsilon_k-\underline{\theta}_k)\Big[F\Big(\frac{\overline{\theta}_k-\upsilon_k}{\sigma}\Big)-F\Big(\frac{\underline{\theta}_k-\upsilon_k}{\sigma}\Big)\Big]\\ &+\frac{\sigma}{4\sqrt{2\pi}}\Big[\exp\Big(-\frac{(\underline{\theta}_k-\upsilon_k)^2}{2\sigma^2}\Big)-\exp\Big(-\frac{(\overline{\theta}_k-\upsilon_k)^2}{2\sigma^2}\Big)\Big]\\ &-F\Big(\frac{\overline{\theta}_k-\upsilon_k}{\sigma}\Big)+1\end{aligned} \tag{26}$$

where $F(x)=\int_{-\infty}^{x}(1/((2\pi)^{1/2}))e^{-(\eta^2/2)}d\eta$ is the standardized Gaussian cumulative distribution function. Then, the partial derivatives of (26) with respect to the weight $[\boldsymbol{W}]_{kn}$, the threshold parameter θ_k and noise-level σ become

$$\frac{\partial h_k^{\infty}}{\partial[\boldsymbol{W}]_{kn}}=\frac{x_n}{4}\Big[F\Big(\frac{\overline{\theta}_k-\upsilon_k}{\sigma}\Big)-F\Big(\frac{\underline{\theta}_k-\upsilon_k}{\sigma}\Big)\Big] \tag{27}$$

$$\frac{\partial h_k^{\infty}}{\partial\theta_k}=-\frac{1}{4}\Big[F\Big(\frac{\overline{\theta}_k-\upsilon_k}{\sigma}\Big)-F\Big(\frac{\underline{\theta}_k-\upsilon_k}{\sigma}\Big)\Big] \tag{28}$$

$$\frac{\partial h_k^{\infty}}{\partial\sigma}=\frac{1}{4\sqrt{2\pi}}\Big[\exp\Big(-\frac{(\underline{\theta}_k-\upsilon_k)^2}{2\sigma^2}\Big)-\exp\Big(-\frac{(\overline{\theta}_k-\upsilon_k)^2}{2\sigma^2}\Big)\Big]. \tag{29}$$

For the sigmoid function in (16), the hidden unit h_k^{∞} in (3) is given by

$$h_k^{\infty}=\frac{1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\infty}\frac{\exp[-\eta^2/(2\sigma^2)]}{1+\exp[-(\upsilon_k-\theta_k+\eta)]}d\eta. \tag{30}$$

The partial derivatives of (30) with respect to the weight $[\boldsymbol{W}]_{kn}$, the threshold parameter θ_k and noise-0level σ can be calculated as

$$\frac{\partial h_k^{\infty}}{\partial[\boldsymbol{W}]_{kn}}=\frac{x_n}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\infty}\frac{\exp\Big(\theta_k-\upsilon_k-\eta-\dfrac{\eta^2}{2\sigma^2}\Big)}{\big[1+\exp\big(-(\upsilon_k+\eta-\theta_k)\big)\big]^2}d\eta \tag{31}$$

$$\frac{\partial h_k^{\infty}}{\partial\theta_k}=\frac{-1}{\sqrt{2\pi}\sigma}\int_{-\infty}^{\infty}\frac{\exp\Big(\theta_k-\upsilon_k-\eta-\dfrac{\eta^2}{2\sigma^2}\Big)}{\big[1+\exp\big(-(\upsilon_k+\eta-\theta_k)\big)\big]^2}d\eta \tag{32}$$

$$\frac{\partial h_k^{\infty}}{\partial\sigma}=\frac{1-\sigma^2}{\sqrt{2\pi}\sigma^4}\int_{-\infty}^{\infty}\frac{(1+\eta^2)\exp\Big(-\dfrac{\eta^2}{2\sigma^2}\Big)}{1+\exp\big(-(\upsilon_k+\eta-\theta_k)\big)}d\eta. \tag{33}$$

If the noise level σ is zero, the hidden unit h_k^{∞} in (3) reduces to the sigmoid function $\phi(\upsilon_k)$ in (16). The partial derivatives (16) with respect to $[\boldsymbol{W}]_{kn}$ and the threshold parameter θ_k become

$$\frac{\partial\phi(\upsilon_k)}{\partial[\boldsymbol{W}]_{kn}}=\phi(\upsilon_k)[1-\phi(\upsilon_k)]x_n \tag{34}$$

$$\frac{\partial\phi(\upsilon_k)}{\partial\theta_k}=-\phi(\upsilon_k)[1-\phi(\upsilon_k)]. \tag{35}$$

Substituting (27)–(35) into (19)–(21), the learning rule can be implemented.

Appendix C
Weak Convergence of the Learning Rule for Training Threshold Neural networks

From (3) and letting $t=\theta_k-\upsilon_k$, it is noticed that the hidden unit $h_k^{\infty}(t,\sigma)$ is bounded within the range from 0 to 1, because

it equals one minus the cumulative distribution function of the standardized Gaussian random variable. The partial derivatives of $h_k^{\infty}(t,\sigma)$ with respect to variables t and σ are given by

$$\frac{\partial h_k^{\infty}(t,\sigma)}{\partial t} = \frac{1}{\sqrt{2\pi}\sigma}\exp\Big(-\frac{t^2}{2\sigma^2}\Big) \tag{36}$$

$$\frac{\partial h_k^{\infty}(t,\sigma)}{\partial \sigma} = \frac{-t}{\sqrt{2\pi}\sigma^2}\exp\Big(-\frac{t^2}{2\sigma^2}\Big). \tag{37}$$

Since $\exp(-t^2/2\sigma^2)$ is infinitesimal of higher order than t as $t \to \pm\infty$, and the same for σ^{-2} as $\sigma \to 0$ and $\sigma \to \infty$, then $\partial h_k^{\infty}(t,\sigma)/\partial t$ and $\partial h_k^{\infty}(t,\sigma)/\partial \sigma$ are uniformly bounded for the domain of definition $-\infty < t < \infty$ and $0 \leqslant \sigma < \infty$. Likewise, we can deduce the exact expressions of the second-order partial derivatives of $h_k^{\infty}(t,\sigma)$ with respect to variables t and σ, and the tedious manipulation is not included here for simplicity. Similarly, the second-order partial derivatives can be also proven to be uniformly bounded in the domain of definition. Therefore, the first-order partial derivatives $\partial h_k^{\infty}(t,\sigma)/\partial t$ and $\partial h_k^{\infty}(t,\sigma)/\partial \sigma$ are Lipschitz continuous, which guarantees the convergence of the proposed noise-boosted algorithm based on gradients [49], [52].

Furthermore, for the approximated optimal noise PDF in (12), the hidden unit in (13) is a linear combination of the expectations $\mathrm{E}_g[\phi(\upsilon_k+\eta)]$, therefore the partial derivatives of $h_k^{\infty}(t,\sigma)$ with respect to variables t and σ are also uniformly bounded in the domain of definition. Thus, the modified noise-boosted backpropagation Algorithm 2 of searching the optimal noise type is still weakly convergent.

REFERENCES

[1] C. M. Bishop, "Training with noise is equivalent to Tikhonov regularization," *Neural Comput.*, vol. 7, no. 1, pp. 108–116, 1995.

[2] R. Reed, R. J. Marks, and S. Oh, "Similarities of error regularization, sigmoid gain scaling, target smoothing, and training with jitter," *IEEE Trans. Neural Netw.*, vol. 6, no. 3, pp. 529–538, May 1995.

[3] G. An, "The effects of adding noise during backpropagation training on a generalization performance," *Neural Comput.*, vol. 8, no. 3, pp. 643–674, Apr. 1996.

[4] J. Sietsma, "Neural net pruning-why and how," in *Proc. IEEE Int. Conf. Neural Netw.*, San Diego, CA, USA, Jul. 1988, pp. 325–333.

[5] L. Holmstrom and P. Koistinen, "Using additive noise in back-propagation training," *IEEE Trans. Neural Netw.*, vol. 3, no. 1, pp. 24–38, Jan. 1992.

[6] K. Matsuoka, "Noise injection into inputs in back-propagation learning," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, no. 3, pp. 436–440, May 1992.

[7] P. L. Barlett and T. Downs, "Using random weights to train multilayer networks of hard-limiting units," *IEEE Trans. Neural Netw.*, vol. 3, no. 2, pp. 202–210, Mar. 1992.

[8] Y. Grandvalet, S. Canu, and S. Boucheron, "Noise injection: Theoretical prospects," *Neural Comput.*, vol. 9, no. 5, pp. 1093–1108, 1997.

[9] C. Gulcehre, M. Moczulski, M. Denil, and Y. Bengio, "Noisy activation functions," 2016, *arXiv:1603.00391*. [Online]. Available: http://arxiv.org/abs/1603.00391

[10] Z. He, A. S. Rakin, and D. Fan, "Parametric noise injection: Trainable randomness to improve deep neural network robustness against adversarial attack," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Long Beach, CA, USA, Jun. 2019, pp. 588–597.

[11] X. Meng, C. Liu, Z. Zhang, and D. Wang, "Noisy training for deep neural networks," in *Proc. IEEE China Summit Int. Conf. Signal Inf. Process. (ChinaSIP)*, Xi'an, China, Jul. 2014, pp. 16–20.

[12] A. Neelakantan *et al.*, "Adding gradient noise improves learning for very deep networks," in *Proc. 4th Int. Conf. Learn. Represent.*, San Juan, Puerto Rico, May 2016, p. 327.

[13] K. Audhkhasi, O. Osoba, and B. Kosko, "Noise-enhanced convolutional neural networks," *Neural Netw.*, vol. 78, pp. 15–23, Jun. 2016.

[14] Y. Li and F. Liu, "Whiteout: Gaussian adaptive noise regularization in deep neural networks," 2016, *arXiv:1612.01490*. [Online]. Available: http://arxiv.org/abs/1612.01490

[15] J. Wang, Q. Chang, Q. Chang, Y. Liu, and N. R. Pal, "Weight noise injection-based MLPs with group lasso penalty: Asymptotic convergence and application to node pruning," *IEEE Trans. Cybern.*, vol. 49, no. 12, pp. 4346–4364, Dec. 2019.

[16] N. Frazier-Logue and S. J. Hanson, "The stochastic delta rule: Faster and more accurate deep learning through adaptive weight noise," *Neural Comput.*, vol. 32, no. 5, pp. 1018–1032, May 2020.

[17] P. Panda and K. Roy, "Implicit adversarial data augmentation and robustness with noise-based learning," *Neural Netw.*, vol. 141, pp. 120–132, Sep. 2021.

[18] L. Xiao, F. Duan, J. Tang, and D. Abbott, "A noise-boosted remaining useful life prediction method for rotating machines under different conditions," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, Art. no. 3512612.

[19] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[20] P. M. Aziz, H. V. Sorensen, and J. van der Spiegel, "An overview of sigma-delta converters," *IEEE Signal Process. Mag.*, vol. 13, no. 1, pp. 61–84, Jan. 1996.

[21] L. Bin, T. W. Rondeau, J. H. Reed, and C. W. Bostian, "Analog-to-digital converters," *IEEE Signal Process. Mag.*, vol. 22, no. 6, pp. 69–77, Nov. 2005.

[22] J.-J. Xiao, R. Ribeiro, Z.-Q. Luo, and G. B. Giannakis, "Distributed compression-estimation using wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 23, no. 4, pp. 27–41, Jul. 2006.

[23] D. Rousseau and F. Chapeau-Blondeau, "Noise-improved Bayesian estimation with arrays of one-bit quantizers," *IEEE Trans. Instrum. Meas.*, vol. 56, no. 6, pp. 2658–2662, Dec. 2007.

[24] Y. Zhou, Y. Huang, J. Pang, and K. Wang, "Remaining useful life prediction for supercapacitor based on long short-term memory neural network," *J. Power Sources*, vol. 440, Nov. 2019, Art. no. 227149.

[25] S. Lu, Q. He, F. Hu, and F. Kong, "Sequential multiscale noise tuning stochastic resonance for train bearing fault diagnosis in an embedded system," *IEEE Trans. Instrum. Meas.*, vol. 63, no. 1, pp. 106–116, Jan. 2014.

[26] S. Ikemoto, F. DallaLibera, and K. Hosoda, "Noise-modulated neural networks as an application of stochastic resonance," *Neurocomputing*, vol. 277, pp. 29–37, Feb. 2018.

[27] M. G. Xibilia, M. Latino, Z. Marinkovic, A. Atanaskovic, and N. Donato, "Soft sensors based on deep neural networks for applications in security and safety," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 10, pp. 7869–7876, Oct. 2020.

[28] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal reprentations by error propagation," in *Parallel Distributed Processing*, vol. 1, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA, USA: MIT Press, 1986, pp. 318–362.

[29] R. Benzi, A. Sutera, and A. Vulpiani, "The mechanism of stochastic resonance," *J. Phys. A, Math. Gen.*, vol. 14, no. 11, p. L453, Nov. 1981.

[30] B. Ando, S. Baglio, A. R. Bulsara, and V. Marletta, "A nonlinear energy harvester operated in the stochastic resonance regime for signal detection/measurement applications," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 8, pp. 5930–5940, Aug. 2020.

[31] B. Kosko, K. Audhkhasi, and O. Osoba, "Noise can speed backpropagation learning and deep bidirectional pretraining," *Neural Netw.*, vol. 129, pp. 359–384, Sep. 2020.

[32] N. G. Stocks, "Suprathreshold stochastic resonance in multilevel threshold systems," *Phys. Rev. Lett.*, vol. 84, no. 11, pp. 2310–2313, Mar. 2000.

[33] N. Mtetwa and L. S. Smith, "Precision constrained stochastic resonance in a feedforward neural network," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 250–262, Jan. 2005.

[34] M. D. McDonnell, N. G. Stocks, C. E. M. Pearce, and D. Abbott, *Stochastic Resonance: From Suprathreshold Stochastic Resonance to Stochastic Signal Quantization*. New York, NY, USA: Cambridge Univ. Press, 2008.

[35] F. Duan, Y. Pan, F. Chapeau-Blondeau, and D. Abbott, "Noise benefits in combined nonlinear Bayesian estimators," *IEEE Trans. Signal Process.*, vol. 67, no. 17, pp. 4611–4623, Sep. 2019.

[36] Y. Fu, Y. Kang, and R. Liu, "Novel bearing fault diagnosis algorithm based on the method of moments for stochastic resonant systems," *IEEE Trans. Instrum. Meas.*, vol. 70, 2021, Art. no. 6500610.

[37] A. Patel and B. Kosko, "Stochastic resonance in continuous and spiking neuron models with Levy noise," *IEEE Trans. Neural Netw.*, vol. 19, no. 12, pp. 1993–2008, Dec. 2008.
[38] O. Osoba, S. Mitaim, and B. Kosko, "The noisy expectation-maximization algorithm," *Fluctuation Noise Lett.*, vol. 12, no. 3, Mar. 2013, Art. no. 1350012.
[39] O. Adigun and B. Kosko, "Noise-boosted bidirectional backpropagation and adversarial learning," *Neural Netw.*, vol. 120, pp. 9–31, Dec. 2019.
[40] O. Adigun and B. Kosko, "Using noise to speed up video classification with recurrent backpropagation," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Anchorage, AK, USA, May 2017, pp. 108–115.
[41] X. Liu, L. Duan, F. Duan, F. Chapeau-Blondeau, and D. Abbott, "Enhancing threshold neural network via suprathreshold stochastic resonance for pattern classification," *Phys. Lett. A*, vol. 403, Jul. 2021, Art. no. 127387.
[42] S. Uhlich, "Bayes risk reduction of estimators using artificial observation noise," *IEEE Trans. Signal Process.*, vol. 63, no. 20, pp. 5535–5545, Oct. 2015.
[43] G.-B. Huang, Q.-Y. Zhu, K. Z. Mao, C.-K. Siew, P. Saratchandran, and N. Sundararajan, "Can threshold networks be trained directly?" *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 3, pp. 187–191, Mar. 2006.
[44] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.
[45] D. Dua and C. Graff. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[46] M. H. Rafiei and H. Adeli, "A novel machine learning model for estimation of sale prices of real estate units," *J. Construct. Eng. Manage.*, vol. 142, no. 2, Feb. 2016, Art. no. 04015066.
[47] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decis. Support Syst.*, vol. 47, no. 4, pp. 547–553, 2009.
[48] M. Cassotti, D. Ballabio, R. Todeschini, and V. Consonni, "A similarity-based QSAR model for predicting acute toxicity towards the fathead minnow (Pimephales promelas)," *SAR QSAR Environ. Res.*, vol. 26, no. 3, pp. 217–243, Mar. 2015.
[49] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, 2006.
[50] R. M. Neal, "Slice sampling," *Ann. Statist.*, vol. 31, no. 3, pp. 705–767, Jun. 2003.
[51] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980
[52] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.
[53] K. Suresh, *Design Optimization Using MATLAB and SOLIDWORKS*. Cambridge, U.K.: Cambridge Univ. Press, 2021.
[54] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

Lingling Duan was born in China in 1982. She received the master's degree in computational mathematics from Xiamen University, Xiamen, China, in 2008. She is currently pursuing the Ph.D. degree in system science with Qingdao University, Qingdao, China.

Since 2009, she has been with Jining University, Jining, China. Her research interest includes noise-enhanced effects in neural networks.

Fabing Duan was born in China in 1974. He received the master's degree in engineering mechanics from the China University of Mining and Technology, Beijing, China, in 1999, and the Ph.D. degree in solid mechanics from Zhejiang University, Hangzhou, China, in 2002.

From 2002 to 2003, he was a Post-Doctoral Fellow with the University of Angers, Angers, France. Since 2004, he has been with Qingdao University, Qingdao, China, where he is currently a Professor of system science. His research interests include nonlinear signal processing and machine learning.

François Chapeau-Blondeau was born in France in 1959. He received the Engineer Diploma from ESEO, Angers, France, in 1982, the Ph.D. degree in electrical engineering from the University Pierre of Marie Curie, Paris, France, in 1987, and the Habilitation degree from the University of Angers, Angers, in 1994.

In 1988, he was a Research Associate with the Department of Biophysics at the Mayo Clinic, Rochester, MN, USA, where he was working on biomedical ultrasonics. Since 1990, he has been with the University of Angers, Angers, where he is currently a Professor of electrical and electronic engineering. His research interests include information theory, signal processing and imaging, and the interactions between physics and information sciences.

Derek Abbott (Fellow, IEEE) was born in South Kensington, London, U.K., in 1960. He received the B.Sc. degree (Hons.) in physics from Loughborough University, Leicestershire, U.K., in 1982, and the Ph.D. degree in electrical and electronic engineering from The University of Adelaide, Adelaide, SA, Australia, in 1997, under the supervision of K. Eshraghian and B. R. Davis.

From 1978 to 1986, he was a Research Engineer with the GEC Hirst Research Center, London, U.K. From 1986 to 1987, he was a VLSI Design Engineer with Austek Microsystems, Australia. Since 1987, he has been with The University of Adelaide, where he is currently a Full Professor with the School of Electrical and Electronic Engineering. His research interests include multidisciplinary physics and electronic engineering applied to complex systems, networks, game theory, energy policy, stochastics, and biophotonics.

Dr. Abbott is a fellow of the Institute of Physics (IoP), U.K., and Honorary Fellow of Engineers Australia. He has received a number of awards, including the Tall Poppy Award for Science in 2004, an Australian Research Council Future Fellowship in 2012, the David Dewhurst Medal in 2015, the Barry Inglis Medal in 2018, and the M. A. Sargent Medal in 2019 for eminence in engineering. He has coedited *Quantum Aspects of Life* (Imperial College Press, 2008), and coauthored *Stochastic Resonance* (Cambridge University Press, 2008) and *Terahertz Imaging for Biomedical Applications* (Springer-Verlag, 2012). He has been an Editor and/or Guest Editor for a number of journals, including the IEEE JOURNAL OF SOLID-STATE CIRCUITS, *Journal of Optics B*, *Microelectronics Journal*, *PLOS ONE*, PROCEEDINGS OF THE IEEE, and the IEEE PHOTONICS JOURNAL. He is currently on the Editorial Boards of IEEE ACCESS, *Scientific Reports* (Nature), *Royal Society Open Science*, and *Frontiers in Physics*. He has served on the Editorial Boards of PROCEEDINGS OF THE IEEE from 2009 to 2014, the Editorial Board of IEEE ACCESS from 2015 to present. He currently serves for the IEEE Publication Services and Products Board (PSPB) from 2019 to present.