

Hyperparameter on-line learning of stochastic resonance based threshold networks

Weijin Li(李伟进), Yuhao Ren(任昱昊), and Fabing Duan(段法兵)
Citation: Chin. Phys. B, 2022, 31 (**8**): 080503. DOI: 10.1088/1674-1056/ac5886
Journal homepage: http://cpb.iphy.ac.cn; http://iopscience.iop.org/cpb

What follows is a list of articles you may be interested in

A new simplified ordered upwind method for calculating quasi-potential

Qing Yu(虞晴) and Xianbin Liu(刘先斌)
Chin. Phys. B, 2022, 31 (**1**): 010502. DOI: 10.1088/1674-1056/ac140f

Chaotic signal denoising algorithm based on sparse decomposition

Jin-Wang Huang(黄锦旺), Shan-Xiang Lv(吕善翔), Zu-Sheng Zhang(张足生), Hua-Qiang Yuan(袁华强)
Chin. Phys. B, 2020, 29 (**6**): 060505. DOI: 10.1088/1674-1056/ab8a3b

Stochastic bifurcations of generalized Duffing-van der Pol system with fractional derivative under colored noise

Wei Li(李伟), Mei-Ting Zhang(张美婷), Jun-Feng Zhao(赵俊锋)
Chin. Phys. B, 2017, 26 (**9**): 090501. DOI: 10.1088/1674-1056/26/9/090501

Pattern dynamics of network-organized system with cross-diffusion

Qianqian Zheng(郑前前), Zhijie Wang(王直杰), Jianwei Shen(申建伟)
Chin. Phys. B, 2017, 26 (**2**): 020501. DOI: 10.1088/1674-1056/26/2/020501

Modeling random telegraph signal noise in CMOS image sensor under low light based on binomial distribution

Yu Zhang(张钰), Xinmiao Lu(逯鑫淼), Guangyi Wang(王光义), Yongcai Hu(胡永才), Jiangtao Xu(徐江涛)
Chin. Phys. B, 2016, 25 (**7**): 070503. DOI: 10.1088/1674-1056/25/7/070503

Hyperparameter on-line learning of stochastic resonance based threshold networks

Weijin Li(李伟进), Yuhao Ren(任昱昊), and Fabing Duan(段法兵)†

College of Automation, Qingdao University, Qingdao 266071, *China*

(Received 6 January 2022; revised manuscript received 16 February 2022; accepted manuscript online 25 February 2022)

Aiming at training the feed-forward threshold neural network consisting of nondifferentiable activation functions, the approach of noise injection forms a stochastic resonance based threshold network that can be optimized by various gradient-based optimizers. The introduction of injected noise extends the noise level into the parameter space of the designed threshold network, but leads to a highly non-convex optimization landscape of the loss function. Thus, the hyperparameter on-line learning procedure with respective to network weights and noise levels becomes of challenge. It is shown that the Adam optimizer, as an adaptive variant of stochastic gradient descent, manifests its superior learning ability in training the stochastic resonance based threshold network effectively. Experimental results demonstrate the significant improvement of performance of the designed threshold network trained by the Adam optimizer for function approximation and image classification.

Keywords: noise injection, adaptive stochastic resonance, threshold neural network, hyperparameter learning

PACS: 05.40.Ca, 02.50.–r, 84.35.+i **DOI:** 10.1088/1674-1056/ac5886

1. Introduction

Gradient-based optimizers are commonly used for training neural networks, but a necessary condition for successfully implementing the network performance optimization is the continuously differentiable activation functions therein. However, the introduction of the piecewise-linear (e.g., ReLU)[1,2] or the hard-limited (e.g., binarization) activations[3,4] has gained increasing attention in recent years due to the fact of training much deeper networks or largely saving memory storage and computation.[1–8] Therefore, training the neural network with nondifferentiable or zero-gradient activation functions becomes a tricky problem for applications of available gradient-based optimizers.

A natural way of training threshold neural networks by the gradient-descent based back-propagation algorithm is approximating the nondifferentiable threshold neuron to a smoothed activation function,[6,9–13] but performing the network testing with threshold activation functions and the well trained network weights. For instance, a conventional method is to substitute the sigmoid function $1/(1+\mathrm{e}^{-\lambda u})$ with a large parameter $\lambda > 0$ (e.g., $\lambda = 10$) for the threshold neuron.[9–11] Unfortunately, the generalization performance of the threshold neural network is unsatisfactory in testing phase, because network weights are inadequately trained for the saturated regimes of the sigmoid function.[6,9–13]

Recently, the approach of noise injection becomes a useful alternative option to optimize the artificial neural network.[4,6,8,12–22] It is interesting to notice that the benefits of injecting noise in threshold neural networks can be viewed as a type of stochastic resonance effect,[22] because there is also a nonzero noise level for improving the performance of nonlinear systems.[6,12–14,23–33] By injecting artificial noise into the saturated regime of activation function[4] or smoothing the input-output characteristic of hard-limiting neurons with an ensemble of mutually independent noise components,[6,12,13] the non-differentiable threshold networks own a proper definition of the gradients. Then, the gradient-based back-propagation algorithm can successfully perform in finely transformed threshold neural networks. Meanwhile, a stochastic resonance based threshold neural network on the distribution of injected noise into the hidden layer is proposed for effectively recycling the back-propagation training algorithm.[6,12,13] Furthermore, using a stochastic gradient descent (SGD) optimizer, we actually realize a noise-boosted back-propagation training algorithm in this kind of stochastic resonance based threshold networks by adaptively learning of both weights and noise levels.[13,29]

The introduction of injected noise into the threshold neural network extends the order of dimension of the parameter space, and also poses a challenge to the SGD optimizer for finding a good optima in the non-convex performance landscape of the loss function.[34–38] This is because the SGD optimizer is sometimes trapped in a domain of the flat landscape of the loss function, and the gradients with respect to the noise level and network weights approach zero.[36–38] In this paper, the noise-boosted Adam optimizer is further demonstrated to train the stochastic resonance based threshold neural network more effectively in comparison with the SGD optimizer. It is shown that, with the powerful hyperparameter on-line learning capacity of the Adam optimizer,[34–38] the designed threshold

†Corresponding author. E-mail: fabingduan@qdu.edu.cn

© 2022 Chinese Physical Society and IOP Publishing Ltd

network can attain the much lower mean square error (MSE) for function approximation or the higher accuracy for image classification. Moreover, in the testing phase of the stochastic resonance based threshold neural network, the practical realization of the threshold network trained by the Adam optimizer requires less computational cost than that of the optimized threshold network by the SGD optimizer. The benefits of injected noise manifested by the Adam optimizer in each hidden neuron with different noise levels also conduce to the research of exploiting adaptive stochastic resonance effects in machine learning.

2. Main results

2.1. Stochastic resonance based threshold network

Consider an $N \times K \times M$ feed-forward threshold neural network with three layers.[6,12,13,29,39,40] The input layer receives the data $\boldsymbol{x} \in \mathbb{R}^{N\times 1}$, and the weight matrix $\boldsymbol{W} \in \mathbb{R}^{K\times N}$ connects the hidden layer and the input one. The hidden layer consists of K threshold neurons[41] described as

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

with an adjustable threshold parameter θ_k for $k = 1, 2, \ldots, K$. The weight matrix $\boldsymbol{U} \in \mathbb{R}^{M\times K}$ connects K threshold neurons in the hidden layer with M linear activation functions in the output layer, and $\boldsymbol{y} \in \mathbb{R}^{M\times 1}$ denotes the network output vector.

It is seen from Fig. 1(a) that the threshold activation function $\psi(u)$ of Eq. (1) is nondifferentiable at $u = \theta_k$ and with zero-gradients for $u \neq \theta_k$. Based on the noise injection method,[6,12,13,29,39,40] we can replace the activation function $\psi(u)$ in Eq. (1) with a differentiable one. This process is formed by injecting a sufficiently large number T of mutually independent noise variables η_t with the common probability density function (PDF) $f_\eta(\eta)$, as shown in Fig. 1(b). Then, a continuously differentiable activation function is deduced as the substitution, i.e., the expectation[6,12,13,29]

$$h(x) = \mathrm{E}_\eta[\psi(x+\eta)] = \int_{\theta_k - x}^{\infty} f_\eta(\eta)\,\mathrm{d}\eta \tag{2}$$

with respect to the injected noise PDF $f_\eta(\eta)$. An illustrative example of $h(x)$ with respect to the Gaussian noise PDF $f_\eta(x) = \exp(-x^2/2\sigma_k^2)/\sqrt{2\pi\sigma_k^2}$ and the noise level $\sigma_k = 1$ is plotted in Fig. 1(c).

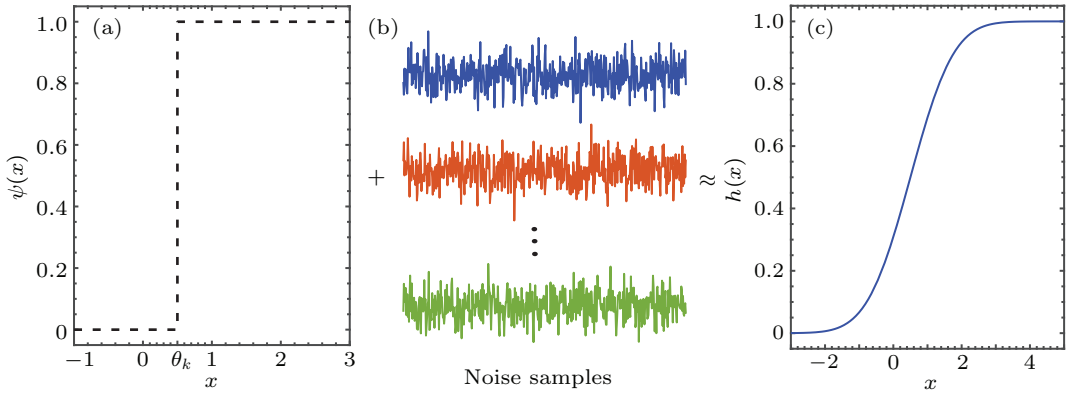


Fig. 1. (a) Threshold activation function $\psi(u)$ in Eq. (1) with the threshold parameter θ_k, (b) an ensemble of T noise samples with a common level σ of the standard deviation and (c) the noise-smoothed activation function $h(x)$ with learnable parameters θ_k and σ.

It is interesting to note that, for the designed threshold neural network with the hidden neuron given by Eq. (2), the aforementioned difficulty of training threshold neural networks can be overcame, because the activation function $h(x)$ in Eq. (2) has its finely defined gradient $\partial h(x)/\partial x = f_\eta(\theta_k - x)$. Furthermore, it is emphasized that the effective learning ability of the designed threshold neural network is also extended by the introduction of the learnable parameters of noise levels σ_k and threshold θ_k in Eq. (2). During the training process, the parameters σ_k and θ_k, as well as the network weight matrices $\boldsymbol{W}$ and $\boldsymbol{U}$, are all optimized by the gradient-based learning rule. Specially, we here represent the corresponding gradients as $\partial h(x)/\partial\sigma_k = \int_{\theta_k - x}^{\infty} \partial f_\eta(\eta)/\partial\sigma_k\,\mathrm{d}\eta$ and $\partial h(x)/\partial\theta_k = -f_\eta(\theta_k - x)$ with respect to σ_k and θ_k, respectively. Thus, the back-propagation learning algorithm can be successfully implemented for training the designed threshold neural network.

2.2. Motivated example

Let $\{\boldsymbol{x}(\ell), \boldsymbol{s}(\ell)\}_{\ell=1}^{L}$ denote L examples of the training set to train the desinged threshold network in a supervised learning manner, and $\boldsymbol{s} \in \mathcal{R}^{M\times 1}$ represents the desired response vector. The loss function of the empirical mean square error (MSE) can be calculated as

$$\mathcal{J}_{\mathrm{mse}} = \frac{1}{L}\sum_{\ell=1}^{L} \mathcal{J}(\ell) = \frac{1}{L}\sum_{\ell=1}^{L} \|\boldsymbol{s}(\ell) - \boldsymbol{y}(\ell)\|^2, \tag{3}$$

where $\|\cdot\|$ denotes the Euclidean norm. Let $\Theta \in \{\boldsymbol{W}, \boldsymbol{U}, \theta_k, \sigma_k\}$ denote the learnable parameter of the designed threshold network, the plain stochastic gradient descent (SGD)

optimizer updates the parameter Θ by the learning rule

$$\Theta(\ell) = \Theta(\ell-1) - \alpha \frac{\partial \mathcal{J}(\ell)}{\partial \Theta}\Big|_{\Theta=\Theta(\ell-1)}, \tag{4}$$

where $\Theta(0)$ is the initial value of Θ and the learning rate $\alpha > 0$. It is noted that the noise levels σ_k, as learnable network parameters, can reconstruct the nonlinearity of each neuron of $h(x)$ across the training phase, and are then expected to improve the performance of the designed threshold network.

For instance, $L = 41$ input examples $x(\ell)$ are generated equally spaced in the interval $[-2,2]$ and the corresponding function values $s(\ell)$ are recorded from an unidimensional function

$$f(x) = \sin(2x) + 2\exp(-10x^2). \tag{5}$$

Then, the $1 \times K \times 1$ ($K = 10$) stochastic resonance based threshold neural network is trained to fit the training set $\{\boldsymbol{x}(\ell), \boldsymbol{s}(\ell)\}_{\ell=1}^{L}$ sampled from the target function of Eq. (5). Here, the learning rate takes $\alpha = 0.01$, the initial noise levels $\sigma_k(0) = 1$, the initial threshold parameters $\theta_k(0)$ and the initial weight vectors $\boldsymbol{W}(0)$ and $\boldsymbol{U}(0)$ are uniformly distributed in the interval $[-1,1]$. The learning process of $\mathcal{J}_{\rm mse}$ of the designed threshold network is shown in Fig. 2 via the SGD optimizer for different initial values of learnable parameters.

It is important to note in Fig. 2 that the SGD optimizer is sensitive to the initial values of parameters. For example, setting the initial weight $W_{1,1} = -5$ and the noise level $\sigma_8 = 1$, it is seen in Fig. 2 that the MSE $\mathcal{J}_{\rm mse}$ ($*$) of the designed threshold network converges to a local minimum of $\mathcal{J}_{\rm mse} = 1.149 \times 10^{-3}$ after 5000 epochs of training. However, when the initial values of $W_{1,1} = -2.3$ and $\sigma_8 = 0.2$ are not properly set, it is shown in Fig. 2 that the learning trajectory of the MSE $\mathcal{J}_{\rm mse}$ ($\bigtriangledown$) is stuck in a flat landscape of the network performance surface, because the gradient herein is almost zero in every direction. Here, besides $W_{1,1}$ and the noise level σ_8, other parameters $\Theta \in \{\boldsymbol{W}, \boldsymbol{U}, \theta_k, \sigma_k\}$ are fixed.

Next, we employ the Adam optimizer to update the parameter Θ of the stochastic resonance based threshold neural network by the learning rule[34]

$$\begin{aligned}
m(\ell) &= \beta_1 m(\ell-1) + (1-\beta_1)\frac{\partial \mathcal{J}}{\partial \Theta}\Big|_{\Theta(\ell-1)}, \\
v(\ell) &= \beta_2 v(\ell-1) + (1-\beta_2)\frac{\partial \mathcal{J}}{\partial \Theta}\Big|_{\Theta(\ell-1)} \odot \frac{\partial \mathcal{J}}{\partial \Theta}\Big|_{\Theta(\ell-1)}, \\
\hat{m}(\ell) &= m(\ell)/(1-{\beta_1}^{\ell}), \quad \hat{v}(\ell) = v(\ell)/(1-{\beta_2}^{\ell}), \\
\Theta(\ell) &= \Theta(\ell-1) - \frac{\alpha \hat{m}(\ell)}{\sqrt{\hat{v}(\ell)+\varepsilon}},
\end{aligned} \tag{6}$$

where $\beta_1 = 0.9$ and $\beta_2 = 0.999$ are attenuation coefficients, $0 < \varepsilon \ll 1$ is the regularization parameter, $m(\ell)$ denotes the first-order moment with its corrected value $\hat{m}(\ell)$ and $v(\ell)$ is the second-order moment estimation with its corrected value $\hat{v}(\ell)$. The operator $\odot$ denotes the Hadamard product. Likewise, starting from the initial values of $W_{1,1} = -2.3$ and $\sigma_8 = 0.2$, it is seen in Fig. 2 that the MSE $\mathcal{J}_{\rm mse}$ ($+$) of the designed network trained by the Adam optimizer can escape from the flat landscape of the network performance surface, and finally converge to the minimum of $\mathcal{J}_{\rm mse} = 1.109 \times 10^{-3}$. Here, other parameters $\Theta \in \{\boldsymbol{W}, \boldsymbol{U}, \theta_k, \sigma_k\}$ (not including $W_{1,1}$ and σ_8) are fixed as the converged values searched by the SGD optimizer. This result clearly shows that the Adam optimizer, compared with the SGD one, is more efficient for optimizing the designed stochastic resonance based threshold network.

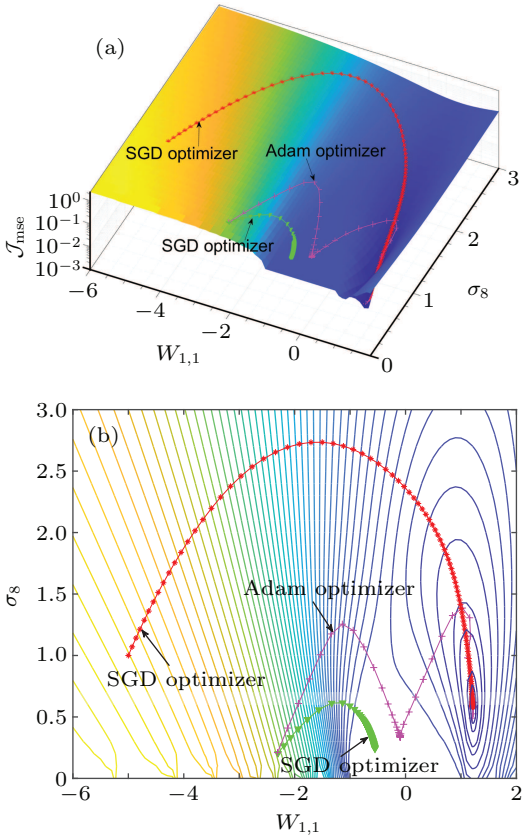


Fig. 2. (a) Performance surface and (b) the corresponding contour of the MSE $\mathcal{J}_{\rm mse}$ versus the weight $W_{1,1}$ and the noise level σ_8 of the eighth hidden neuron. The learning curves of $\mathcal{J}_{\rm mse}$ of the designed threshold network are also illustrated for different optimizers.

Furthermore, it is shown in Fig. 3 that the Adam optimizer can optimize the designed threshold network with a smaller MSE $\mathcal{J}_{\rm mse} = 1.137 \times 10^{-5}$ after 1600 epochs of training. Here, the performance surface is still illustrated versus the weight $W_{1,1}$ and the injected noise level σ_8, but the optimum value of $\mathcal{J}_{\rm mse} = 1.137 \times 10^{-5}$ is searched over the the whole space of the learnable network parameters by the Adam optimizer. The parameters $\Theta \in \{\boldsymbol{W}, \boldsymbol{U}, \theta_k, \sigma_k\}$ including $W_{1,1}$ and the noise level σ_8 are simultaneously updated by the learning rule of Eq. (6).

It is emphasized that each hidden neuron in Eq. (2) is associated with one noise level σ_k for $k = 1, 2, \ldots, K$. Thus, in the training process of the designed threshold network, the injected noise will manifest its beneficial role as the convergence

of the noise level σ_k reaches a non-zero (local) optimum. It is seen in Figs. 2 and 3 that, for a given weight $W_{1,1}$ nearby 1.2, the conventional stochastic resonance phenomenon can be observed as the noise level σ_8 increases. When the noise level σ_8 reaches the corresponding non-zero (local) optimum, the MSE $\mathcal{J}_{\rm mse}$ attains its minimum. It is noted that the resonant phenomenon of the designed threshold networks characterized by the MSE $\mathcal{J}_{\rm mse}$ is induced by a bundle of K noise level σ_k in the hidden neurons. The stochastic resonance phenomenon shown in Fig. 2 or Fig. 3 only describes a slice of the MSE $\mathcal{J}_{\rm mse}$ versus the noise level σ_8 for visualization. In view of the learning curves of the K noise levels σ_k, Fig. 4 also presents the distinct characteristic of the adaptive stochastic resonance effect in the training phase. The learning curves of the injected noise levels σ_k in Fig. 4 show that the noise levels σ_k start from the same initial value of unity, and converge to different but non-zero optima. This fact also validates the practicability of the proposed noise-boosted Adam optimizer in Eq. (6) by adaptively optimizing the noise level in the training phase.

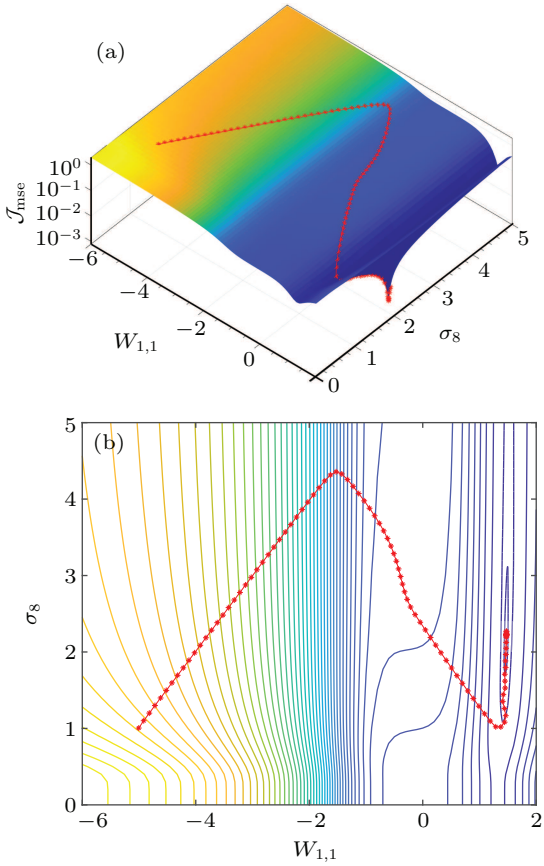


Fig. 3. (a) Performance surface and (b) the corresponding contour of the MSE $\mathcal{J}_{\rm mse}$ versus the weight coefficient $W_{1,1}$ and the noise level σ_8 of the eighth hidden neuron. The learning curve ($*$) of $\mathcal{J}_{\rm mse}$ is also illustrated for the Adam optimizer.

However, the hidden neuron $h(x)$ in Eq. (2) is a limit expression, which ensures the success of training, but cannot be realized in practical testing of the threshold neural network. In testing experiments, the hidden neuron $h(x)$ needs to be realized as

$$h(x) = \mathrm{E}_\eta[\psi(x+\eta)] \approx \frac{1}{T}\sum_{t=1}^{T}\psi(x+\eta_t), \tag{7}$$

where a finite number T of threshold activation functions in Eq. (1) are activated by T mutually independent and identically distributed noise components η_t, respectively. For 10^3 testing data $x(\ell)$ equally spaced in the interval $[-2,2]$, the trained threshold neural network with the hidden neurons $h(x)$ indicated by Eq. (7) is simulated for 10^2 times. In each trial, for the k-th hidden neuron $h(x)$ ($k = 1,2,\ldots,K$), T independent noise components η_t are randomly generated at the converged noise level σ_k after the training phase. Then, the outputs of the designed threshold network are averaged as an approximation to the target function of Eq. (5).

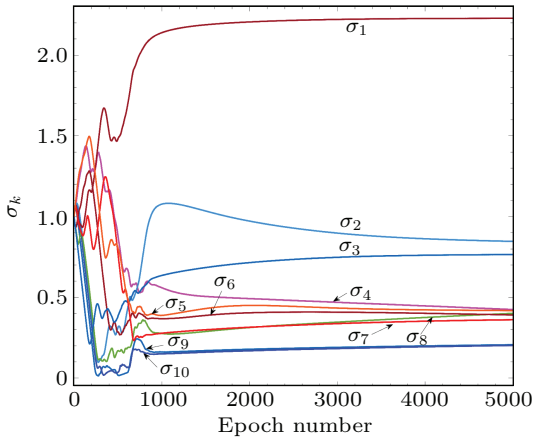


Fig. 4. Learning curves of $K = 10$ noise levels σ_k in the hidden layer of the designed stochastic resonance based threshold network. Other parameters are the same as those in Fig. 3.

For different numbers of T, the experimental results of the MSE $\mathcal{J}_{\rm mse}$ are shown in Fig. 5(a) for testing the trained threshold networks by both optimizers. It is seen from Fig. 5(a) that, in order to achieve the same MSE $\mathcal{J}_{\rm mse}$, the designed threshold network trained by the Adam optimizer needs a smaller number T of threshold activation functions in Eq. (1) as well as the noise components η_t. For instance, when the statistical mean value of the MSE $\mathcal{J}_{\rm mse} = 4\times10^{-4}$, $T = 3\times10^3$ threshold functions in Eq. (1) are required for the designed threshold network trained by the Adam optimizer, whereas $T = 10^4$ threshold functions are required for the SGD optimizer. In Fig. 5(b), using $T = 10^4$ threshold functions assisted by the injected noise with the same size, the output (blue dashed line) of the well trained threshold network by the Adam optimizer is plotted. For comparison, the target function (red solid line) of Eq. (5) is also illustrated. It is seen in Fig. 5(b) that the trained threshold neural network performs well on approximating the target unidimensional function in the testing phase. It is emphasized that, in the practical realization of the designed threshold network, the beneficial role of injected noise

is accomplished by averaging an ensemble of threshold functions driven by the same size of injected noise samples in each hidden neuron, whereby the application of adaptive stochastic resonance in threshold neural networks is evidently confirmed.

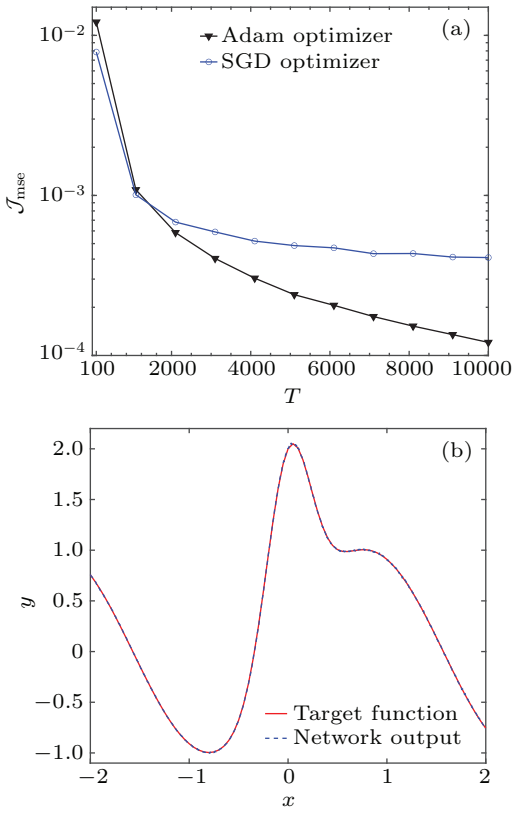


Fig. 5. (a) Experimental results of the MSE $\mathcal{J}_{\rm mse}$ for 10^3 testing points of the target unidimensional function in Eq. (5) versus the number T. Here, 10^2 trials are realized for each point of experimental results, and the ensemble of outputs of T threshold functions activated by T noise samples is regarded as the realization of the hidden neuron of Eq. (7). (b) The output of the trained threshold network by the Adam optimizer as the approximation (blue dash line) of the target function of Eq. (5). For comparison, the target function (red solid line) of Eq. (5) is also plotted.

2.3. Two-dimensional function approximation

Furthermore, we consider the test of a two-dimensional benchmark function

$$\begin{aligned} f(x_1,x_2) \;=\; & 3(1-x_1)^2\mathrm{e}^{-x_1^2-(x_2+1)^2} \\ & -10\Big(\frac{x_1}{5}-x_1^3-x_2^5\Big)\mathrm{e}^{-x_1^2-x_2^2} \\ & -\frac{1}{3}\mathrm{e}^{-(x_1+1)^2-x_2^2}. \end{aligned} \tag{8}$$

The stochastic resonance based threshold neural network is designed with the size $2\times 50\times 1$. The 16×16 training set of $\{\boldsymbol{x}(\ell),\boldsymbol{s}(\ell)\}_{\ell=1}^{L}$ sampled from the target function of Eq. (8) is equally spaced in the range $[-3,3]\times[-3,3]$. After 2×10^4 training epochs, the MSEs $\mathcal{J}_{\rm mse}$ are achieved as 0.1003 and 7×10^{-4} for the SGD optimizer and the Adam optimizer, respectively. The Adam optimizer is still superior to the SGD optimizer in training the designed threshold network to approximate the two-dimensional function in Eq. (8). For 32×32 testing set sampled from the target function of Eq. (8) over the range $[-3,3]\times[-3,3]$, Fig. 6(a) illustrates the outputs of the trained threshold network as the approximation (patched surface) of the two-dimensional function $f(x_1,x_2)$ in Eq. (8). The experimental results of the MSE $\mathcal{J}_{\rm mse}$ are obtained as 0.1012 and 7.5×10^{-4} for the SGD optimizer and the Adam optimizer, respectively. Especially, Fig. 6(b) illustrates the relative errors $|y-f(x_1,x_2)|/\Delta$ between the output y of the trained threshold network by the Adam optimizer and the testing data. Here, the maximum relative error is obtained as $\max|y-f(x_1,x_2)|/\Delta = 5.188\times 10^{-3}$ and the maximum difference $\Delta = \max f(x_1,x_2)-\min f(x_1,x_2) = 13.4628$ for the target function in Eq. (8) in the range $[-3,3]\times[-3,3]$.

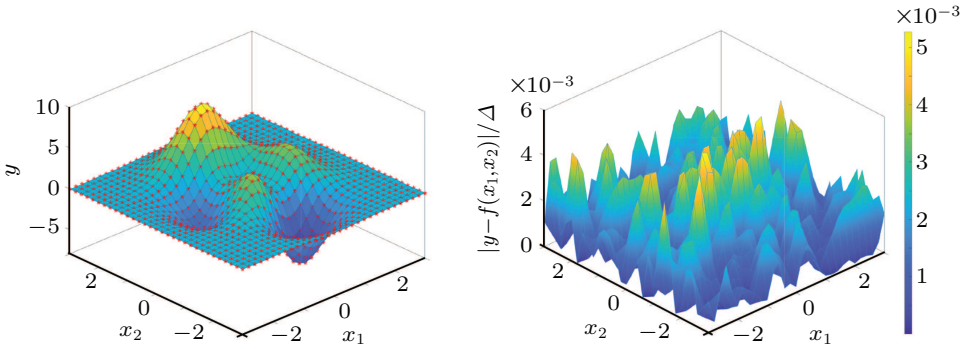


Fig. 6. (a) Network outputs y of the trained threshold neural network by the Adam optimizer as the approximation (patched surface) to the 32×32 testing data ($\star$) of the two-dimensional function $f(x_1,x_2)$ in Eq. (8). (b) The corresponding relative error $|y-f(x_1,x_2)|/\Delta$ between the threshold network output and the testing data. For reference, the maximum difference $\Delta=\max f(x_1,x_2)-\min f(x_1,x_2)=13.4628$ is taken for the target function in Eq. (8) in the range $[-3,3]\times[-3,3]$.

2.4. Real world data set of the function regression

We also validate the trained $N\times 20\times 1$ threshold network on nine real-world data sets.[42–50] The dimension N and the length L of data sets are listed in Table 1, and the computer is equipped with CPU of Intel Core i7-6700@3.40 GHz and 16G RAM DDR4@2133 MHz. 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 2 reports the training and testing results of the MSE $\mathcal{J}_{\rm mse}$ of the stochastic

resonance based threshold neural network for two considered optimizers. It is seen in Table 2 that the Adam optimizer can optimize the designed threshold network with a lower MSE $\mathcal{J}_{\rm mse}$ than the SGD optimizer does in both training and testing data sets. The hyperparameter on-line learning of stochastic resonance based threshold networks via the Adam optimizer enables more precise control of injected noise levels in the hidden layer, leading to significantly improved performance of the designed threshold network.

Table 1. Feature dimension and length of data sets.

Data set	Training data length	Testing data length	Dimension
Airfoil noise	1202	301	5
Auto MPG	319	79	7
Boston house prices	404	102	13
Wine quality	3918	980	11
QSAR fish toxicity	727	181	6
Real estate valuation	331	83	5
Concrete compressive strength	824	206	5
Concrete slump test	80	23	9
Yacht hydrodynamics	246	62	6

Table 2. Experimental results of the MSE $\mathcal{J}_{\rm mse}$ of designed threshold threshold networks.

Data set	$\mathcal{J}_{\rm mse}$ for SGD (10^{-2})		$\mathcal{J}_{\rm mse}$ for Adam (10^{-2})	
	Training	Testing	Training	Testing
Airfoil noise	1.2650	1.2650	0.1275	0.1629
Auto MPG	0.0528	0.0682	0.0430	0.0631
Boston house prices	0.7080	0.4971	0.1297	0.2269
Wine quality	1.5006	1.4625	1.2325	1.2825
QSAR fish toxicity	0.9467	1.1113	0.6561	0.9843
Real estate valuation	0.9082	1.0081	0.0340	0.2881
Concrete compressive strength	1.5925	1.6628	0.2915	0.7102
Concrete slump test	0.2340	0.3385	0.0001	0.0094
Yacht hydrodynamics	0.0504	0.0807	0.0012	0.0027

2.5. Recognition of handwritten digits

We further incorporate the expectation expression of Eq. (2) into the deep convolution neural network for image classification. The architecture of the deep convolution neural network contains 20 convolutional filters with the size 9×9 and the same padding, the hidden layer with 20 neurons indicated by Eq. (2), the factor-2 pooling layer, and the fully connected layers with 10 neurons. The benchmark data set of MNIST consists of a training set of 6×10^4 and a testing set of 10^4 gray-scale images (28×28) representing the handwritten digits from 0 to 9. Here, we employ 10^4 images with the training set and the test set in a ratio 4 : 1. Using the SGD and Adam optimizers, the training accuracies of the designed convolution neural network versus the training epoch number are shown in Fig. 7. It is seen in Fig. 7 that the designed convolution neural network optimized by the Adam optimizer generally achieves a higher accuracy for recognizing handwritten digits. For the well trained convolution neural network after 50 training epochs, using $T = 10$ threshold functions assisted by the injected noise with the same size, the experimental accuracy rate in the test set is 98.41% for the Adam optimizer and 97.63% for the SGD optimizer, respectively. For comparison, it is noted that the full-connected backward propagation network[29] and the support vector machine[51] achieve the accuracy rates of 97% and 94.1%, respectively. The accuracy rate 98.41% obtained by the trained convolution threshold neural network by the Adam optimizer indicates a very satisfactory efficiency of the deep learning method. These results demonstrate that the injection of noise into threshold neurons facilitates the optimization of the designed deep convolution neural network, and the hyperparameter on-line learning of the Adam optimizer can also train the deeper stochastic resonance based threshold networks for image classification with a competitive performance.

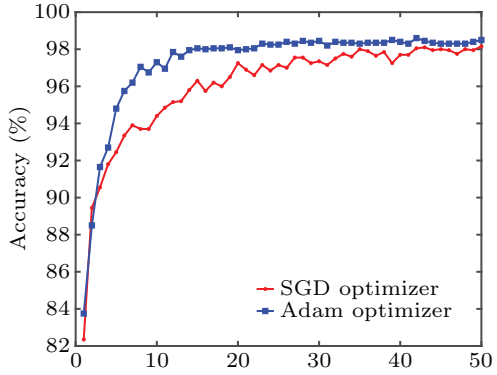


Fig. 7. Learning curves of accuracies of the designed convolution neural network on the MNIST data set.

3. Conclusion

In summary, in order to optimize the threshold neural network in the training phase, we replace the zero-gradient activation function with the continuously differentiable function that is based on the PDF of the injected noise. However, this substitution strategy introduces a group of noise parameters related to the noise PDF, and poses challenges for the training algorithm. For function approximation and image classification, it is shown that, due to the hyperparameter on-line learning capacity, the Adam optimizer can speed up the training of

the designed threshold neural network and overcome the local convergence problem existing in the SGD optimizer. More interestingly, the injected noise not only extends the dimension of the parameter space wherein the designed threshold network is optimized, but also converges to a nonzero noise level in each hidden neuron. This distinguishing feature is closely related to the adaptive stochastic resonance effect, and also indicates a meaningful application of the stochastic resonance phenomenon in artificial neural networks.

However, it is noted that the Adam optimizer can train the artificial neural network to fit the target function well or recognize labels in classification more correctly. Then, for noisy observations of a target function or the uncorrected labels, the overfitting problem may occur for the training process of the designed threshold network by the Adam optimizer. Noting the the regularization in the loss function contributed by the injected noise,[16,17] it is much more meaningful to further investigate the generalization performance of the stochastic resonance based threshold neural network, especially for the low signal-to-noise ratio of the acquired signals by sensors. In Eq. (2), only the Gaussian noise injected into the designed feed-forward threshold neural network is considered. It will be interesting to find the optimal injected noise type to achieve the improved performance of the designed threshold network with respect to the noise PDF. In addition, we test the convolution threshold neural network with the injection of noise on image classification of the MNIST data set, and the potential applications of this kind of deep convolution threshold neural networks on more challenging data set, e.g., CIFAR-10 and ImageNet, also deserve to be explored.

Acknowledgement

Project supported by the Natural Science Foundation of Shandong Province, China (Grant No. ZR2021MF051).

References

[1] Nair V and Hinton G E 2010 *Proceeding of the 27th International Conference on Machine Learning*, June 21–24, 2010, Haifa, Israel, pp. 807–814
[2] Glorot X, Bordes A and Bengio Y 2011 *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, May 13–15, 2011, Sardinia, Italy, pp. 315–323
[3] Courbariaux M and Bengio Y 2016 arXiv:1602.02830v3 [cs]
[4] Gulcehre C, Moczulski M, Denil M and Bengio Y 2016 *Proceedings of the 33rd International Conference on Machine Learning*, June 19–24, 2016, New York, pp. 3059–3068
[5] Rastegari M, Ordonez V, Redmon J and Farhadi A 2016 *The 14th European Conference on Computer Vision*, October 8–16, 2016, Amsterdam, pp. 525–542
[6] Ikemoto S, DallaLibera F and Hosoda K 2018 *Neurocomputing* **277** 29
[7] Ignatov D and Ignatov A 2020 *Pattern Recognition Lett.* **138** 276
[8] Qin H T, Gong R H, Liu X L, Bai X, Song J K and Sebe N 2020 *Pattern Recognition* **105** 107281
[9] Toms D J 1990 *Electron. Lett.* **26** 1745
[10] Corwin E M, Logar A M and Oldham W J B 1994 *IEEE Trans. Neural Networks* **5** 507
[11] Wilson E and Rock S M 2002 *Int. J. Robust Nonlin. Control* **12** 1009
[12] Ikemoto S 2021 *Neurocomputing* **448** 1
[13] Duan L L, Duan F B, Chapeau-Blondeau F and Abbott D 2021 *IEEE Trans. Instrum. Measurement* **70** 1010612
[14] Qiao Z J and Shu X D 2021 *Chaos, Solitons & Fractals* **145** 110813
[15] Bishop C M 1995 *Neural Comput.* **7** 108
[16] An G 1996 *Neural Comput.* **8** 643
[17] Grandvalet Y, Canu S and Boucheron S 1997 *Neural Comput.* **9** 1093
[18] Grandvalet Y, Canu S and Boucheron S 2016 *Neural Networks* **78** 15
[19] Adigun O and Kosko B 2019 *Neural Networks* **120** 9
[20] Frazier-Logue N and Hanson S J 2020 *Neural Comput.* **32** 1018
[21] Frazier-Logue N and Hanson S J 2020 *Neural Comput.* **32** 1
[22] Benzi R, Sutera A and Vulpiani A 1981 *J. Phys. A: Math. Gen.* **14** L453
[23] Kosko B, Audhkhasi K and Osoba O 2020 *Neural Networks* **129** 359
[24] Jiang L, Lai L, Yu T and Luo M K 2021 *Chin. Phys. B* **30** 060502
[25] Jin Y F 2018 *Chin. Phys. B* **27** 050501
[26] He L F, Cui Y Y, Zhang T Q, Zhang G and Song Y 2016 *Chin. Phys. B* **25** 060501
[27] Liu J, Wang Y G, Zhai Q Q and Liu J 2016 *Chin. Phys. B* **25** 100501
[28] Li J L 2009 *Chin. Phys. B* **18** 5196
[29] Liu X J, Duan L L, Duan F B, Chapeau-Blondeau F and Abbott D 2021 *Phys. Lett. A* **403** 127387
[30] Han J and Bao J D 2014 *Chin. Phys. Lett.* **31** 120502
[31] Qiao Z J, Elhattab A, Shu X D and He C B 2021 *Nonlin. Dyn.* **106** 707
[32] Qiao Z J, Liu J, Ma X and Liu J L 2021 *J. Franklin Institute* **358** 2194
[33] Song Y L 2011 *Chin. Phys. Lett.* **28** 120502
[34] Kingma D P and Ba J 2015 *International Conference on Learning Representations*, May 7–9, 2015, San Diego, CA, pp. 1–15
[35] Bock S and Weiβ M 2019 *International Conference on Learning Representations*, May 6–9, 2019, Bugdapest, Hungary, pp. 14–19
[36] Cong G and Buratti L 2018 *IEEE/ACM Machine Learning in HPC Environments (MLHPC)*, Nov 12–12, 2018, Dallas, TX, pp. 85–94
[37] Chaudhury S and Yamasaki T 2021 *IEEE Access* **9** 37039
[38] Liu Z, Shen Z, Li S, Helwegen K, Huang D and Cheng K T 2021 arXiv: 2106.11309 [cs]
[39] Stocks N G 2000 *Phys. Rev. Lett.* **84** 2310
[40] Sanjuán M A F 2010 *Contemp. Phys.* **51** 448
[41] McCulloch, Warren S and Pitts W 1943 *Bull. Math. Biophys.* **5** 115
[42] Lopez R, Balsa-Canto E and Oñate E 2008 *Int. J. Numer. Methods Engin.* **75** 1341
[43] Quinlan J R 1993 *Proceedings of The Tenth International Conference on Machine Learning*, June 27–29, 1993, University of Massachusetts, Amherst, pp. 236–243
[44] Rafiei M H and Adeli H 2016 *J. Construct. Engin. Manag.* **142** 04015066
[45] Cortez P, Cerdeira A, Almeida F, Matos T and Reis J 2009 *Decision Support Systems* **47** 547
[46] Cassotti M, Ballabio D, Todeschini R and Consonni V 2015 *SAR and QSAR Environ. Res.* **26** 217
[47] Yeh I C and Hsu T K 2018 *Appl. Soft Comput.* **65** 260
[48] Yeh I C 1998 *Cement Concrete Res.* **28** 1797
[49] Yeh I C 2008 *Comput. Concrete* **5** 559
[50] Gerritsma J, Onnink R and Versluis A 1981 *Int. Shipbuild. Prog.* **28** 276
[51] Liu W F, We J C and Meng Q M 2020 *IEEE International Conference on Advances in Electrical Engineering and Computer Applications*, August 25–27, 2020, Dalian, China, pp. 587–590