LETTER Communicated by Jian Liu

Improving the Antinoise Ability of DNNs via a Bio-Inspired Noise Adaptive Activation Function Rand Softplus

Yunhua Chen*
yhchen@gdut.edu.cn
Yingchao Mai
444301201@qq.com
School of Computers, Guangdong University of Technology, Guangzhou 51006, China

Jinsheng Xiao*
xiaojs@whu.edu.cn
School of Electronic Information, Wuhan University, Wuhan 430072, China

Ling Zhang
june4567@163.com
School of Computers, Guangdong University of Technology, Guangzhou 51006, China

Although deep neural networks (DNNs) have led to many remarkable results in cognitive tasks, they are still far from catching up with human-level cognition in antinoise capability. New research indicates how brittle and susceptible current models are to small variations in data distribution. In this letter, we study the stochasticity-resistance character of biological neurons by simulating the input-output response process of a leaky integrate-and-fire (LIF) neuron model and proposed a novel activation function, rand softplus (RSP), to model the response process. In RSP, a scale factor η is employed to mimic the stochasticity-adaptability of biological neurons, thereby enabling the antinoise capability of a DNN to be improved by the novel activation function. We validated the performance of RSP with a 19-layer residual network (ResNet) and a 19-layer visual geometry group (VGG) on facial expression recognition data sets and compared it with other popular activation functions, such as rectified linear units (ReLU), softplus, leaky ReLU (LReLU), exponential linear unit (ELU), and noisy softplus (NSP). The experimental results show that RSP is applied to VGG-19 or ResNet-19, and the average recognition accuracy under five different noise levels exceeds the other functions on both of the two facial expression data sets; in other words, RSP outperforms the other activation functions in noise resistance. Compared with the application in ResNet-19, the application of RSP in VGG-19 can

*Corresponding authors.

Neural Computation **31, 1215–1233** (2019)
doi:10.1162/neco_a_01192

improve a network's antinoise performance to a greater extent. In addition, RSP is easier to train compared to NSP because it has only one parameter to be calculated automatically according to the input data. Therefore, this work provides the deep learning community with a novel activation function that can better deal with overfitting problems.

1 Introduction

In the past few years, thanks to the development of deep learning algorithms (Hinton, Osindero, & Teh, 2006) and their specific neural network accelerators built in hardware, deep neural networks (DNNs) have achieved many remarkable results in cognitive tasks (Liu, Li, Shan, Wang, & Chen, 2014; Kahou et al., 2013; Kim, Lee, Roh, & Lee, 2015). Research (Sun, Chen, Wang, Liu, & Liu, 2016; Simonyan & Zisserman, 2014; Szegedy et al., 2015) has shown that recognition accuracy increases as the network grows deeper, until He, Zhang, Ren, and Sun (2016) discovered the degradation problem and proposed a solution for it—adding residual learning in very deep neural—and built the deep residual networks (ResNets) structure. As a result, a 152-layer ResNet was trained on ImageNet (Deng, Dong, Socher, & Li, 2009), which decreased the top 5 error rate to 5.71%. It showed a hugely improved performance compared to conventional convolutional neural networks (CNNs) with the same depth of simply stacked layers. Thus, ResNet has become one of the most prominent network architectures to solve AI problems.

Although great outcomes have been achieved by brain-inspired deep learning, it is still far from catching up with human-level cognition in noise robustness. New research indicates that current accuracy numbers are brittle and susceptible to even minute natural variations in the data distribution because the same test sets have been used to select these models for multiple years (Recht, Roelofs, Schmidt, & Shankar, 2018).

A regular artificial neuron comprises a weighted summation of input data, $\sum x_i w_i$, and an activation function, f, applied to the sum. Rectified linear units (ReLU) (Nair & Hinton, 2010) were proposed to replace sigmoid neurons and surpassed the performance of other popular activation units thanks to its advantage of sparsity (Glorot, Bordes, & Bengio, 2011) and robustness toward the vanishing gradient problem. At the same time, it also faces the problem of dead ReLU. Leaky ReLU (LReLU; Maas, Hannun, & Ng, 2013) was proposed to solve this problem: the negative part of the activation function is replaced with a linear function, which also causes a small increase in overfitting risks. In addition, Clevert, Unterthiner, and Hochreiter (2015) presented exponential linear units (ELUs), leading to faster learning and better antinoise performance than the rectified unit family on deep networks. Except for the popular, extended version of ReLU, $\max(0, \sum wx)$, the other implementation of $\log(1 + e^x)$, softplus (Dugas, Bengio, Bélisle, Nadeau, & Garcia, 2001), is more biologically realistic (Liu, Chen, & Furber, 2017). Hunsberger and Eliasmith (2015) proposed the soft LIF response

function for training spiking neural networks (SNNs), which is equivalent to softplus activation of DNNs. However, its nonsparsity contradicts the sparse nature of the brain in expressing information.

The inputs of a biological neuron are spike trains generated by presynaptic neurons, which create postsynaptic potentials (PSPs) on the postsynaptic neuron and trigger a spike train. As the output of this spiking neuron, the stochasticity is intrinsic to the event-based spiking process and threshold-controlled firing mechanism of spikes. The neural dynamics of the membrane potentials, PSPs, and spike trains are all time dependent and event driving, while the neurons of DNNs (e.g., sigmoid units), cope only with numerical values representing spiking rate, without timing information and event driving. The fundamental disparities between an abstract artificial neuron and a time-based spiking neuron with physical properties lead to the big difference between DNNs and the human brain in many cognitive tasks, especially in generalized and stochastic data processing.

This inspired us to construct an abstract activation function to model the response process of a biological neuron to improve the noise robustness of current DNNs. Liu and Furber (2016) proposed the activation function noisy softplus (NSP) to model the response process of LIF neurons and extended it by adding a scale factor S to make it fit into training layered-up SNNs (Liu et al., 2017). In the extended NSP, the parameter pair of (k, S) is curve fitted with the triple data points of (λ, x, σ), in the response area of LIF neuron, where x, σ, and λ represent the input current, the noise level of the input current, and the firing rate of LIF neuron, respectively. However, the training of DNNs applying the extended NSP is complicated due to the fitting of (k, S). Therefore, we simplified NSP by proposing a new biologically plausible activation function: rand softplus (RSP).

After simulating the input-output response process of an LIF neuron and comparing it with the DNN activation functions, a scale factor η, which is defined and calculated according to the input randomness of each neuron, is applied as the weight coefficient, to mimic the stochasticity-adaptability of a biological neuron and to construct RSP.

All of the experimental results on the two data sets and with two network structures, VGG-19 and ResNet-19, show that RSP outperforms the other activation functions in noise resistance. For example, the average recognition accuracies of RSP under five different levels of noise on the data set KDEF are 1.75%, 18.25%, 2.44%, 14.33%, and 2.25% higher than that of ReLU, softplus, LReLU, ELU, and NSP. Compared with ReLU, the difference in recognition accuracy between ResNet-19 and VGG-19 falls from 10.63% (ReLU) to 5.25% (RSP), while the recognition accuracy of VGG-19 increases by 5.89% (RSP) on GENKI-4K. In addition, RSP is easier to train because it has only one parameter to be calculated automatically according to the input data, while NSP has two more parameters to be curve fitted with the triple data points composed of input data, input noise, and the firing rate. Therefore, this work provides the deep learning community with a novel activation function that can better deal with overfitting problems.

The rest of the letter is organized as follows. Section 2 provides closer insight into the biological background of RSP, puts forward the activation function, and demonstrates how to get the scale factor η. In section 3, we describe the network structures, data sets, training approaches, and results of the experiments. Section 4 concludes the study and points out the future work.

2 Methods

2.1 Neural Science Background. The evolution of membrane potential V can be simplified to a resistor-capacitor (RC) circuit, which consists of a membrane capacitor C_m and a membrane resistance R_m, both driven by an input current flow I. In the resting state without any input, the membrane potential V stays at the same potential as the battery V_{rest}. When current flows into the neuron, it will charge the capacitor with current $I_C(t)$ and discharge through the resistance with current $I_R(t)$. When the input current stops, the capacitive charge will decay back to V_{rest} by leaking through the resistance:

$$\begin{aligned} I(t) &= I_R(t) + I_C(t) \\ &= \frac{V - V_{rest}}{R_m} + C_m \frac{dV}{dt}. \end{aligned} \tag{2.1}$$

The standard form of the LIF model describes the subthreshold membrane potential evolution as

$$\tau_m \frac{dV}{dt} = -(V - V_{rest}) + R_m I(t), \tag{2.2}$$

where $\tau_m = C_m R_m$ is called the membrane time constant. As soon as the membrane potential reaches the threshold V_{th}, it is set to a reset potential V_{reset}, and a spike is generated:

$$V = V_{reset}. \tag{2.3}$$

So far, we have seen how a spiking neuron models the physical properties of a biological neuron. The special model descriptions are quite different from the activation functions used in DNNs, which is an abstract rate-based model and ignores the internal states of the physical properties, focusing only on the spike rates. To estimate the output firing rate of a spiking neuron, given a current injection I, we can use the response function of the LIF neuron,

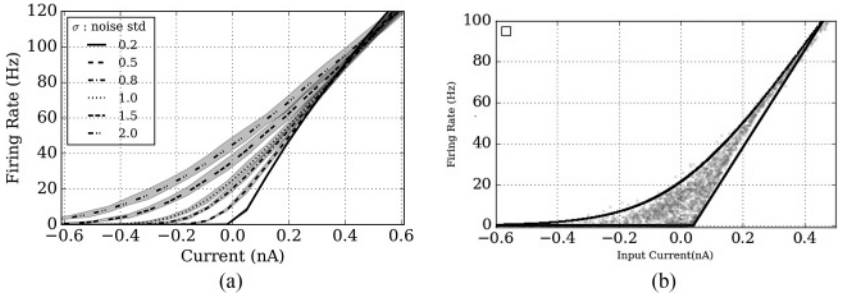

Figure 1: The simulated and recorded firing rates of LIF neuron(s). (a) Firing rates measured by simulations of an LIF neuron driven by different input currents and discrete noise levels. Bold lines show the average, and the gray shading fills the range between the minimum and the maximum. (b) The recorded firing rates of LIF neurons on MNIST.

$$\lambda_{out} = \left[t_{ref} - \tau_m \log\left(1 - \frac{V_{th} - V_{rest}}{IR_m}\right)\right]^{-1}, \tag{2.4}$$

where t_{ref} is the absolute refractory period, τ_m is the membrane time constant, V_{rest} is the resting membrane potential, and R_m is the membrane resistance. When the membrane potential $IR_m > V_{th} - V_{rest}$, a spike will be triggered; otherwise, the membrane potential cannot reach the threshold V_{th}, and the output firing rate is zero. However, in practice, a noisy current generated by the random arrival of spikes, rather than a constant current, flows into the neurons. The response function of the LIF neuron to a noisy current is

$$\lambda_{out} = \left[t_{ref} + \tau_m \int_{\frac{V_{rest}-u\tau_m}{\sigma\sqrt{\tau_m}}}^{\frac{V_{th}-u\tau_m}{\sigma\sqrt{\tau_m}}} \sqrt{\pi}\exp(u^2)(1 + erf(u))du\right]^{-1}, \tag{2.5}$$

where $erf()$ is the error function and u and σ are, respectively, the mean and variance of the current, which can be estimated by La Camera, Giugliano, Senn, and Fusi (2008):

$$u = \tau_{syn} \sum_i w_i \lambda_i, \sigma^2 = \frac{1}{2}\tau_{syn} \sum_i w_i^2 \lambda_i. \tag{2.6}$$

In equation 2.6 τ_{syn} is the synaptic time constant, and each Poisson spike train connects to the neuron with a strength w_i and a firing rate λ_i.

The theoretic response firing rate of an LIF neuron is shown in Figure 1a, while Figure 1b shows the recorded neural response firing rates of the same

kernel convolved with 10 MNIST (LeCun, Cortes, & Burges, 2010) images in an SNN simulation, where the estimated firing rates of softplus are located roughly on the upper boundary of the area and ReLU on the bottom and right boundaries.

2.2 Rand Softplus. In this letter, the input-output response process of an LIF neuron is simulated employing the NEST simulation platform. The simulation results shown in Figure 1b demonstrate the intrinsic stochasticity of the spike trains generated by biological neurons, and the response firing rates of a biological neuron located inside the area surrounded by the black thick boundaries, where the upper boundary approximates the shape of the softplus function and the lower and right boundaries correspond to the shape of the ReLU function. There is a huge difference between the biological neuron response and the prediction of abstract activation functions applied in DNNs. The biological neuron response process corresponds to the input data with a noise variance of normalized value in interval $[0, 1]$, while ReLU and softplus only approximate to the input data with a fixed noise variance of normalized value zero and one. For the biological neurons, the input data are random, and the threshold-controlled firing mechanism can adaptively compensate for this randomness; for the abstract neurons, the input data are not random, and the abstract activation mechanism will not compensate for any randomness.

Similar to NSP, RSP also models the response process of an LIF neuron by representing the input-output map. Unlike NSP, RSP does not need to get the parameter pair of (k, S) by fitting the triple data points in the response area of the LIF neurons; it mimics the stochasticity-adaptability of LIF neurons by adding a scale factor η to combine the well-verified and widely used activation functions ReLU and softplus, which correspond to the lower and upper border of the response area of LIF neurons, respectively. The value of η in RSP comes from the noise level of the input current σ in NSP and will adapt to the input data of each network layer by adjusting, normalizing, and truncating, therefore, training DNNs to apply RSP is easier than applying NSP.

To model the response process of biological neurons, the activation function should be able to adaptively compensate for the randomness of input data. Thus, the definition is

$$y = (1 - \eta)\max(0, x) + \eta \log(1 + e^x), \tag{2.7}$$

where x refers to the input, y indicates the intensity of the output firing rate, and η, which is determined by the noise level of the input data, controls the shape of the activation function.

Figure 2 shows the activation function corresponding to noise with different variances, which demonstrates that the activation region of RSP can

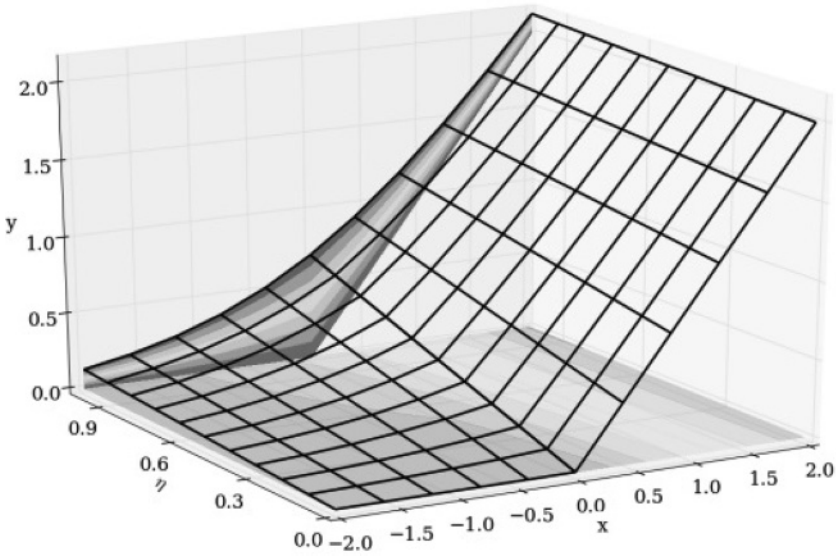

Figure 2: RSP corresponding to different levels of noise.

well mimic the response activities of LIF neurons shown in Figure 1b. The derivative of RSP is

$$y' = \begin{cases} \dfrac{\eta e^x}{1+e^x} & x < 0 \\ 1 - \eta + \dfrac{\eta e^x}{1+e^x} & x > 0 \end{cases}, \tag{2.8}$$

which could be applied easily to backpropagation in any DNN training. From equation 2.8, we can see that the derivative of the RSP changes with η, which is also a scale factor to balance between sparsity activation and gradient ascending.

Because η is obtained adaptively to the level of noise in the input data, it can be used to compensate the stochasticity of the input data; thus, RSP has strong noise immunity. Since η is the key to making RSP mimic the stochasticity-resistance characteristic of biological neurons, detailed steps for calculating it are described in section 2.3.

2.3 The Scale Factor η. As described in section 2.2, the scale factor η is key to making RSP mimic the stochasticity-resistance characteristic of biological neurons. The detailed steps for calculating it follow:

Step 1: Calculate the standard deviation of noise in the input data. In a spiking neural network, the input current of neuron j can be expressed as

Table 1: Some of the Mean Standard Deviation of Noise.

Layer	First-Layer	Third-Layer	Fifth-Layer	Seventh-Layer
Standard deviation	0.4∼1.25	0.21±0.01	0.085±0.01	0.04±0.01

$$\mu_j = \sum_i w_{ij}(\lambda_i \tau_{syn}), \tag{2.9}$$

where i refers to neurons connected to neuron j. When converted to an artificial neural network, the input of neuron j is

$$x_j = \sum_i w_{ij} x_i \quad \text{and} \quad x_i = \lambda_i \tau_{syn}, \tag{2.10}$$

and the variance of the input noise can be expressed as

$$\sigma_j^2 = \sum_i \frac{1}{2}(w_{ij}^2) x_i, \tag{2.11}$$

which can be used to calculate the standard deviation of noise in the input data in DNNs.

Step 2: Adjust the standard deviation. Because some values of the standard deviation are much larger than the average value of the same layer, we set the standard deviation of noise greater than two times of the mean to two times, which can avoid uneven data distribution after normalization in the next step.

Step 3: Normalization of the standard deviation. The standard deviation of noise will gradually decrease with the increasing network layer; some of the mean standard deviation of noise in ResNet-19 on GENKI-4K is shown in Table 1. To solve this problem, the adjusted standard deviation will be normalized on a layer-by-layer basis to obtain scale factor η. The normalization also helps us to prevent the standard function from approaching 0, which means RSP will approach ReLU and lose its adaptability to randomness.

Step 4: Truncation of η. Neuroscience research indicates that cortical neurons are rarely in their maximum saturation regime; they encode information in a sparse mode (Attwell & Laughlin, 2001), and only 1% to 4% of them are active simultaneously (Lennie, 2003). The sparsity of neuron networks can reduce the energy consumption but also improve the antinoise performance of the network. Due to network sparsity, different inputs may contain different amounts of information and would be more conveniently represented by a variable-size data structure. It will also cause differences in the distribution of activated neurons. The difference between the above

two structures can make the neural network extract better features. In order to make RSP sparse, a threshold η_{th} is set for scale factor η. In this way, the degree of sparsity of the neuron network can be controlled. Therefore, we set the normalized η less than η_{th} to zero as follows,

$$\eta = \begin{cases} \eta & \eta \geq \eta_{th} \\ 0 & \eta < \eta_{th} \end{cases}. \tag{2.12}$$

2.4 The Impact of η_{th} on Sparsity of the Network. Sparsity, a concept of interest in many research fields, was introduced in computational neuroscience in the context of sparse coding in the visual system and has been a key element of DNNs. A sparsity penalty has been used in several computational neuroscience and machine learning models, in particular for deep architectures (Glorot et al., 2011). Using ReLU gives rise to zero activation to prevent the neurons from ending up taking small but nonzero activation for firing probability; meanwhile, when the learning rate is high, it also brings the dead RelU problem, which was partly solved by LReLU (Maas et al., 2013) and ELU (Clevert et al., 2015).

To figure out the impact of η_{th} on the network, we performed several experiments based on different network structures and data sets. First, based on the MNIST data set we constructed a CNN of 28 $\times$ 28-6c5-2s-12c5-2s-fc-10o to test the impact of η_{th} on the sparsity of the network. The proposed activation function is used in the convolutional and fully connected layers. In the experiment, the input data were normalized to interval $[0, 1]$. The network sparsity is defined as the ratio of the number of neurons that output 0 to the total number of neurons. Figure 3 shows the results, from which we can see that when η_{th}, is set to a smaller value, the sparsity of the network can easily reach around 21%, and the network sparsity becomes larger with the increase of η_{th}. Nevertheless, forcing too much sparsity may hurt predictive performance for an equal number of neurons because it reduces the effective capacity of the model. In Figure 3, we can see that a sparsity of 42% is the limit for the current network structure to correctly classify the MNIST data set. The results presented demonstrate the feasibility of applying η_{th} to control network sparsity.

Second, based on the GENKI-4K data set, we constructed a ResNet-19 whose structure is introduced in section 3.1 to further verify the impact of η_{th} on the antinoise performance of the network. We use the original and the images contaminated with gaussian noise with a mean of 0 and a variance of 0.1 in the GENKI-4K data set as the input data, respectively. In the experiment, η_{th} was set to different values within the interval $[0, 1]$. The results are shown in Table 2, from which we can see that as long as the value of η_{th} is *not* within the interval [0.4, 0.5] or [0.9, 1], the value of η_{th} does not have much effect on network performance.

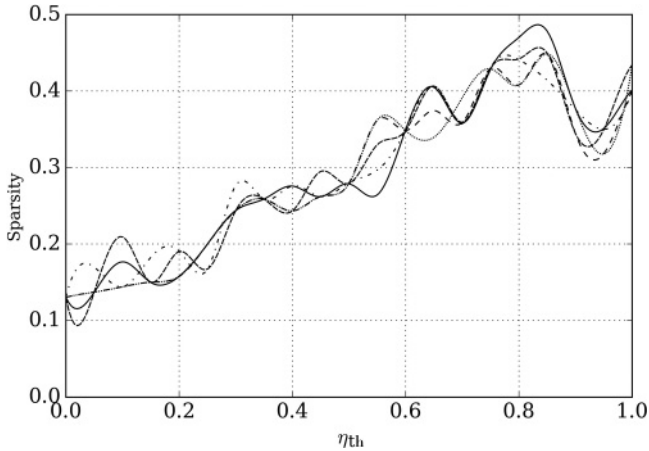

Figure 3: The impact of η_{th} on network sparsity. The four curves correspond to the results of four randomly selected runs.

Table 2: Accuracy on GENKI-4K Corresponding to Different η_{th} and Input Noise with a Variance of 0 and 0.1.

η_{th}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Noise Variance 0	95.1	95.8	95	94.8	94.3	93.5	94.9	95.2	95.1	95.6	94.4
Noise Variance 0.1	91.5	91.7	92.4	93.1	90.4	88.4	92.2	91.9	91.7	88.2	88.9

3 Experiments

3.1 Network Structures. In a ResNet, two or more stacked layers are used as one unit, and a shortcut is added between two neighboring units, which is called identity mapping. Therefore, the unit and the shortcut together form a residual unit (see Figure 4). Suppose the output of a residual unit is expressed as $H(x)$, where x is the input; then $F(x) = H(x) - x$ represents a residual mapping. Compared with direct fitting $H(x)$, the fitting of $F(x)$ makes the output change more sensitively, and the amount of adjustment for the parameters is larger, which speeds up the learning rate and thus improves network performance.

The forward propagation of a residual unit can be defined as

$$y = F(x, \{W_i\}) + x, \tag{3.1}$$

where x and y are the input and output vectors of the residual unit, respectively, and $F(x, \{W_i\})$ is the residual mapping that needs to be learned.

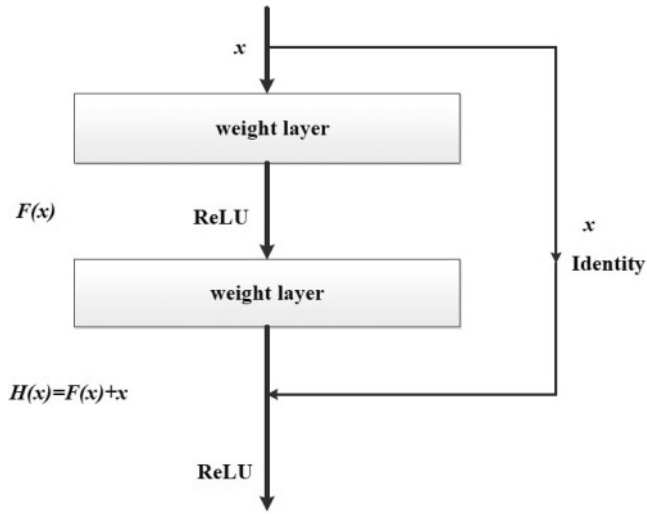

Figure 4: The structure of a residual unit.

The operation $F + x$ is performed by a shortcut connection, followed by an activation $f(y)$. The quick connection in equation 3.1 does not introduce additional parameters or increas computation costs; that is why ResNets can achieve better performance than the conventional CNNs with the same depth.

The dimensions of x and F must be equal in equation 3.1, or a linear projection W_s needs to be performed on the shortcut to match the dimension

$$y = F(x, \{W_i\}) + W_s x. \tag{3.2}$$

Since residual learning can solve the problem of network degradation and improve the performance of conventional CNNs, we built a 19-layer ResNet by adding residual units into VGG-19 (Simonyan & Zisserman, 2014) to evaluate the performance of RSP in more complex visual tasks. The experiments were performed on two public facial expression recognition data sets.

The VGG-19 consists of 16 convolutional layers and 3 fully connected layers. Each convolutional layer has a different number of convolution kernels, from 64 to 512; all the kernels are the same size: 3×3. Each convolutional layer is activated by ReLU after the convolution operation. We removed two fully connected layers and added two convolutional layers with a convolution kernel of 3×3, which can reduce the number of parameters in the network, as the fully connected layer can be viewed as a convolutional layer with a convolution kernel of 1×1. There is no exact rule for choosing

Figure 5: Samples of training images in the data sets. (a) KDEF. (b) GENKI-4K.

the number of convolutional layers included in the residual unit. What can be determined is that more convolutional layers will lead to more network parameters. Conversely, we cannot guarantee that the value of function $F(\cdot)$ is approaching 0. We selected two convolutional layers to form one residual unit after multiple tests; thus, there are seven shortcuts in the 19-layer ResNet, which we call ResNet-19. Compared with VGG-19, ResNet-19 has only one fully connected layer; therefore, it has fewer parameters and less complexity.

3.2 Data Sets. We selected the Karolinska directed emotional faces (KDEF; Lundqvist, Flykt, & Öhman, 1998) and GENKI-4K (Whitehill & Movellan, 2008) as the experimental data sets to evaluate our method.

The KDEF data set contains seven kinds of basic expressions (anger, neutrality, disgust, fear, happiness, sadness, and surprise) of 70 subjects. Each subject is shot twice from five angles, so KDEF contains 4900 images. Figure 5a shows some example images of one subject. The data set consists of seven categories of samples according to the basic expressions. We randomly selected 2 images from one angle in 10 images from five angles for each expression of one subject as the test set, which is composed of 2(shots) $\times$ 1(angle) $\times$ 7(expressions) $\times$ 70(subjects) $=$ 980 images. The ratio of the number of images in the training and test set is just 4:1.

The GENKI-4K data set contains 4000 images with different faces, head poses, and illumination. It has two categories of expression: happiness and neutrality. Each category contains 2162 and 1838 images, respectively. Figure 5b shows some samples of images of happiness. For the images of

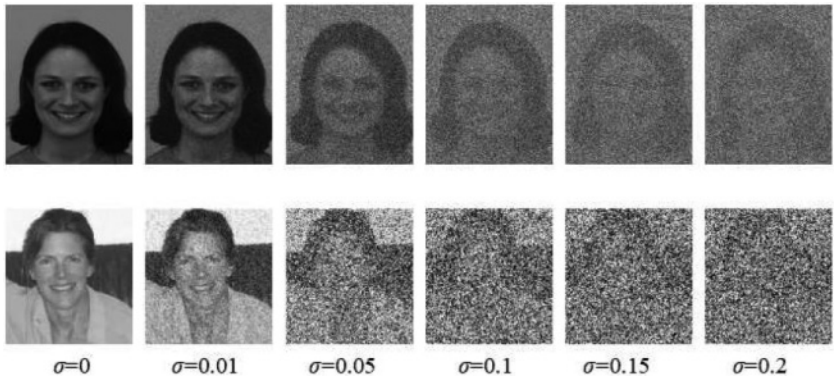

Figure 6: Samples of test images with different variance of noise (σ normalized).

happiness, we selected 1622 to form the training set, and the remaining 540 formed the test set. We selected 1378 images of neutrality as the training set and the remaining 460 as the test set. Thus, the ratio of the number of images in the training and test set is 3:1. The images in GENKI-4K are all color images; therefore, we normalized the values in the three color channels (R,G,B) to interval [0,1] separately according to the input requirement of a ResNet. Thus, the input for each pixel is a three-dimensional vector.

3.3 Results and Analysis. In the experiments, all of the data in the data sets were normalized to interval [0,1]. The training sets contain no noise, while the test sets were contaminated by the gaussian noise with a mean value of 0 and a variance of different values (Xiao, Tian, Zhang, Zhou, & Lei, 2018). Some samples of the test images are shown in Figure 6.

3.3.1 Recognition Accuracy and Antinoise Performance. We applied multiple activation functions to VGG-19 and ResNet-19 for comparison.

Performance at different noise level. Table 3 shows the accuracy of different activation functions corresponding to different levels of test noise in ResNet-19. We can see that in most of the cases, RSP and NSP perform the best in terms of recognition accuracy. With increased noise intensity, the advantage of RSP in accuracy becomes more obvious (see Figure 7).

Performance in different network structures. The recognition accuracy of the six activation functions across GENKI-4K in two different network structures, VGG-19 and ResNet-19, is shown in Table 4. From the table, we can see that ResNet-19 still performs better than VGG-19 in recognition accuracy regardless of which activation function is used and at any noise level.

Table 3: Accuracy of Different Activation Functions in ResNet-19 Corresponding to Different Levels of Test Noise (%).

Data Set	Activation	Variance = 0	Variance = 0.01	Variance = 0.05	Variance = 0.10	Variance = 0.15	Average
KDEF	ReLU	93.72	94.27	92.55	80.65	57.52	83.74
	Softplus	94.12	93.84	84.73	42.9	20.52	67.22
	LReLU	94.19	93.78	92.56	79.09	55.62	83.05
	ELU	92.71	92.56	86.85	56.02	27.66	71.16
	NSP	94.08	**94.49**	**93.87**	79.24	54.50	83.24
	RSP	**94.63**	94.42	93.22	**82.00**	**63.20**	**85.49**
GENKI-4K	ReLU	94.67	95.20	94.70	93.27	90.51	93.67
	Softplus	94.93	95.06	93.96	90.86	86.26	92.21
	LReLU	94.82	94.98	94.63	92.9	90.45	93.56
	ELU	94.92	95.08	94.25	89.75	78.52	90.50
	NSP	**95.16**	**95.43**	94.80	93.30	90.36	93.81
	RSP	95.07	95.35	**95.00**	**93.80**	**91.67**	**94.18**

Note: The numbers in bold correspond to the highest recognition accuracy obtained by different functions under the same noise variance.

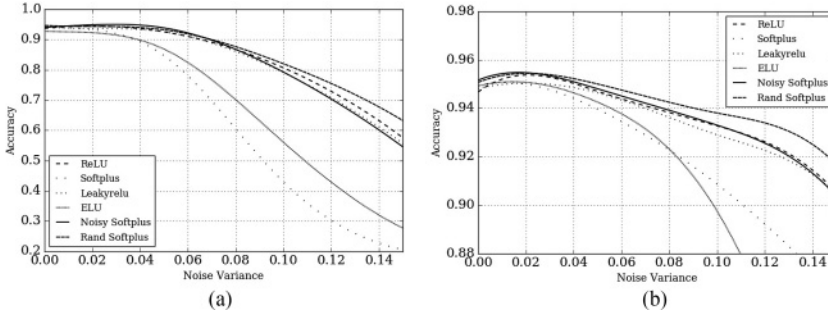

Figure 7: The antinoise performance of different activation functions across data sets: (a) KDEF. (b) GENKI-4K.

Activation function applicability to different networks. In Table 5, the average recognition accuracies of five levels of noise in ResNet-19 and VGG-19 corresponding to six activation functions are shown in the first and second rows, respectively. The third row shows the difference of the average recognition accuracies between ResNet-19 and VGG-19, from which we can see that RSP, NSP, and ELU have significantly smaller differences than the other three activation functions; in particular, RSP has the smallest difference. This demonstrates that ResNet-19 has better antinoise performance than VGG-19 on the whole. But it also indicates that RSP, NSP, and ELU can better improve the antinoise ability of VGG-19 than the other activation functions can. From this, we may infer that the proposed activation function RSP is more applicable in VGG-19 than ResNet-19.

3.3.2 Speed of Convergence and Computational Complexity. Because both VGG-19 and Res Net-19 use the pretraining results on ImgNet applying ReLU as the initial weights on the facial expression data sets, they are not suitable for comparing the speed of convergence and computational complexity of the activation functions. Consequently, we chose to compare the results of the CNN (28×28-6c5-2s-12c5-2s-fc-10o) mentioned in section 2.4 on MNIST. Experimental results show that the six activation functions have approximately the same speed of convergence, and the computational complexity of RSP is slightly higher than that of other functions. All six activation functions achieve stable convergence after approximately 30 epochs. The execution times are (in seconds) 105 (RSP without truncation of η), 119 (with truncation of η), 102 (NSP), 102 (ReLU), 101 (ELU), 102 (softplus), and 101 (LRelU). It can be seen that the increase in computational complexity is largely due to the truncation of η. Although the truncation of η increases the computational complexity, it also achieves better sparsity and noise robustness.

Table 4: Accuracy of Different Activation Functions in Different Network Structures (%).

Network	Activation	Var = 0	Variance = 0.01	Variance = 0.05	Variance = 0.10	Variance = 0.15	Average
VGG-19	ReLU	94.56	94.4	92.58	75.32	58.33	**83.04**
	Softplus	93.73	93.05	87.63	68.28	50.88	**78.71**
	LReLU	94.31	94.16	91.85	78.65	46.43	**81.80**
	ELU	94.55	94.23	90.88	77.7	66.3	**84.73**
	NSP	94.8	94.6	93.03	79.96	76.61	**87.80**
	RSP	95.31	95.3	93.73	83.47	76.83	**88.93**
ResNet-19	ReLU	94.67	95.20	94.70	93.27	90.51	**93.67**
	Softplus	94.93	95.06	93.96	90.86	86.26	**92.21**
	LReLU	94.82	94.98	94.63	92.9	90.45	**93.56**
	ELU	94.92	95.08	94.25	89.75	78.52	**90.50**
	NSP	95.16	5.43	94.80	93.30	90.36	**93.81**
	RSP	95.07	95.35	95.00	93.80	91.67	**94.18**

Note: The numbers in bold correspond to the average recognition accuracy of different levels of noise data obtained by each activation function.

Table 5: Average Recognition Accuracy and Difference Applying Different Activation Functions under Different Network Structures and Levels of Noise (%).

Network	ReLU	Softplus	LReLU	ELU	NSP	RSP
ResNet-19	93.67	92.21	93.56	90.50	93.81	94.18
VGG-19	83.04	78.71	81.08	84.73	87.80	88.93
Difference	10.63	13.5	12.48	**5.77**	**6.01**	**5.25**

Note: The numbers in bold correspond to the smallest three of the six differences.

4 Conclusion

In this letter, we studied the stochasticity-resistance ability of biological neurons by simulating the input-output response process of an LIF neuron model, according to which we proposed a biologically plausible activation function that we named RSP. In this activation function, a scale factor η is employed to mimic the stochasticity-adaptability of a biological neuron to improve the antinoise capability of an artificial neuron.

RSP was compared with the following popular activation functions: ReLU (Nair & Hinton, 2010), softplus (Dugas et al., 2001), LReLU (Maas et al., 2013), ELU (Clevert et al., 2015), and another biologically plausible activation function NSP (Liu & Furber, 2016; Liu et al., 2017) on the facial expression recognition tasks. Experimental results demonstrate that RSP and NSP outperform the other activation functions in noise resistance. In most cases, RSP performs the best among these activation functions thanks to the scale factor adapting to the input noise involved in the test data. The results also indicate that ResNet-19 has better antinoise performance than VGG-19, and RSP can improve the antinoise ability of VGG-19 better.

Both NSP and RSP count on the noise of the current influx generated by Poissonian arriving spikes and therefore have noise immunity. However, NSP has two parameters to be curve fitted with the triple data points of input data, input noise, and the firing rate. RSP has just one parameter to be calculated automatically according to the input data. Thus, it is easier to train networks by applying RSP than NSP.

Although the antinoise capability of RSP is better than the other five activation functions, it is computationally more complex.

We will consider transformation of RSP-trained DNN to SNN to pave the way to vision applications with stringent size, weight, and energy requirements. We will also gain insight into the practical side of RSP, especially in challenging tasks with highly variable data.

Acknowledgments

The research leading to the results presented in this letter has received funding from the National Natural Science Foundation of China (no. 61471272),

the Natural Science Foundation of Guangdong Province, China (no. 2016A030313713), the Natural Science Foundation of Guangdong Province, China (no. 2014A030310169), and Science and Technology Projects of Guangdong Provincial Transportation Department, China (Science and Technology-2016-02-030).

References

Attwell, D., & Laughlin, S. B., (2001). An energy budget for signaling in the grey matter of the brain. *Journal of Cerebral Blood Flow and Metabolism*, *21*(10), 1133–1145.

Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). *Fast and accurate deep network learning by exponential linear units (ELUS)*. arXiv:1511.07289.

Deng, J., Dong, W., Socher, R., & Li, L. J. (2009). Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 248–255). Piscataway, NJ: IEEE.

Dugas, C., Bengio, Y., Bélisle, F., Nadeau, C., & Garcia, R. (2001). Incorporating second-order functional knowledge for better option pricing. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems, 13* (pp. 472–478). Cambridge, MA: MIT Press.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 315–323). Piscataway, NJ: IEEE.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). Piscataway, NJ: IEEE.

Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*(7), 1527–1554.

Hunsberger, E., & Eliasmith, C. (2015). *Spiking deep networks with LIF neurons*. arXiv:1510.08829.

Kahou, S. E., Pal, C., Bouthillier, X., Froumenty, P., Memisevic, R., Vincent, P., . . . Mirza, M. (2013). Combining modality specific deep neural networks for emotion recognition in video. In *Proceedings of the International Conference on Multimodal Interaction* (pp. 543–550). New York: ACM.

Kim, B. K., Lee, H., Roh, J., & Lee, S. Y. (2015). Hierarchical committee of deep CNNs with exponentially-weighted decision fusion for static facial expression recognition. In *Proceedings of the International Conference on Multimodal Interaction* (pp. 427–434). New York: ACM.

La Camera, G., Giugliano, M., Senn, W., & Fusi, S. (2008). The response of cortical neurons to in vivo–like input current: theory and experiment. *Biological Cybernetics*, *99*(4–5), 279–301.

LeCun, Y., Cortes, C., & Burges, C. J. (2010). *MNIST handwritten digit database*. AT&T Labs. http://yann.lecun.com/exdb/mnist.

Lennie, P. (2003). The cost of cortical computation. *Current Biology*, *13*(6), 493–497.

Liu, M., Li, S., Shan, S., Wang, R., & Chen, X. (2014). *Deeply learning deformable facial action parts model for dynamic expression analysis*. Berlin: Springer.

Liu, Q., Chen, Y., & Furber, S. (2017). *Noisy softplus: An activation function that enables SNNs to be trained as ANNs*. arXiv:1706.03609.

Liu, Q., & Furber, S. (2016). Noisy softplus: A biology inspired activation function. In *Neural Information Processing: 23rd International Conference Proceedings* (pp. 405–412). Berlin: Springer.

Lundqvist, D., Flykt, A., & Öhman, A. (1998). The Karolinska directed emotional faces—KDEF [CD-ROM] Stockholm: Karolinska Institute, Department of Clinical Neuroscience, Psychology Section. www.facialstimuli.com/index_files/Page369.htm.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. *Proceedings of the ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. Piscataway, NJ: IEEE.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the International Conference on International Conference on Machine Learning* (pp. 807–814). Berlin: Springer.

Recht, B., Roelofs, R., Schmidt, L., & Shankar, V. (2018). *Do cifar-10 classifiers generalize to cifar-10?* arXiv:1806.00451v1.

Simonyan, K., & Zisserman, A. (2014). *Very deep convolutional networks for large-scale image recognition*. arXiv:1409.1556.

Sun, S., Chen, W., Wang, L., Liu, X., & Liu, T. Y. (2016). On the depth of deep neural networks: A theoretical view. In *Proceedings of the 30th International Conference of the Association for the Advancement of Artificial Intelligence* (pp. 2066–2072). Palo Alto, CA: AAAI Press.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–9). Piscataway, NJ: IEEE.

Whitehill, J., & Movellan, J. R. (2008). A discriminative approach to frame-by-frame head pose tracking. In *Proceedings of the Eighth IEEE International Conference on Automatic Face and Gesture Recognition* (pp. 1–7). Piscataway, NJ: IEEE.

Xiao, J., Tian, H., Zhang, Y., Zhou, Y., & Lei, J. (2018). Blind video denoising via texture-aware noise estimation. *Computer Vision and Image Understanding*, *169*, 1–13.

Received September 11, 2018; accepted February 11, 2019.