Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 9097868, 11 pages
https://doi.org/10.1155/2022/9097868

Research Article

Single Neuron for Solving XOR like Nonlinear Problems

Ashutosh Mishra, Jaekwang Cha, and Shiho Kim

School of Integrated Technology, YICT, Yonsei University, Seoul, Republic of Korea

Correspondence should be addressed to Shiho Kim; shiho@yonsei.ac.kr

Received 16 October 2021; Revised 24 March 2022; Accepted 13 April 2022; Published 28 April 2022

Academic Editor: José Alfredo Hernández-Pérez

XOR is a special nonlinear problem in artificial intelligence (AI) that resembles multiple real-world nonlinear data distributions. A multiplicative neuron model can solve these problems. However, the multiplicative model has the indigenous problem of backpropagation for densely distributed XOR problems and higher dimensional parity problems. To overcome this issue, we have proposed an enhanced translated multiplicative single neuron model. It can provide desired tessellation surface. We have considered an adaptable scaling factor associated with each input in our proposed model. It helps in achieving optimal scaling factor value for higher dimensional input. The efficacy of the proposed model has been tested by randomly increasing input dimensions for XOR-type data distribution. The proposed model has crisply classified even higher dimensional input in their respective class. Also, the computational complexity is the same as that of the previous multiplicative neuron model. It has shown more than an 80% reduction in absolute loss as compared to the previous neuron model in similar experimental conditions. Therefore, it can be considered as a generalized artificial model (single neuron) with the capability of solving XOR-like real problems.

1. Introduction

Minski and Perpert deduced that the XOR problem requires more than one hyperplane [1]. They provide a more generalized artificial neuron model by introducing the concept of weights and proved the inability of a single perceptron for solving 'Exclusive-OR (XOR)' [2]. The XOR problem is symmetrical to other popular and real-world problems such as XOR type nonlinear data distribution in two classes, N-bit parity problems. [3]. Therefore, many researchers tried to find a suitable way out to solve the XOR problem [4–15]. Although, most of the solutions are for the classical XOR problem. They either use more than one layer or provide a complex solution for two-bit logical XOR only. Few of these used the complex value neuron model, eventually creating one more layer (i.e., hidden layer). Because the complex value neuron model requires representing the real input in a complex domain, one approach is based on the multiplicative neuron model. This is translated multiplicative neuron (π_t-neuron) approach [16, 17]. They have modified the π-neuron model (which generates the decision surfaces centered at the origin of input) to an extended multiplicative neuron, i.e., a π_t-neuron model for solving the N-bit parity problems by creating tessellation surfaces. However, it has limitations for higher dimensional N-bit parity problems. It is suitable for up to six dimensions. For seven and higher dimensional inputs, it has reported poor accuracy [17]. In other words, it has a convergence problem for higher dimensional inputs. It is merely because of the multiplicative nature of the model. More clearly, the infinitesimal errors in the model obtain a much smaller value after getting multiplied in case of higher dimensional inputs, consequently vanishing the gradient. Therefore, a convergence problem occurs in this model for higher-dimensional inputs.

To overcome the issue of the π_t-neuron model, we have proposed an enhanced translated multiplicative model neuron (π_t-neuron) model in this paper. It helps in achieving mutually orthogonal separation in the case of two-bit classical XOR data distribution. Also, the proposed model has shown the capability for solving the higher-order N-bit parity problems. Therefore, it is a generalized artificial model for solving real XOR problems. To examine this claim, we have tested our model on different XOR data distributions and N-bit parity problems. For parity problems, we have varied the input dimension for a higher dimensional dataset. Our proposed model has no vanishing gradient issues and convergence issues for higher dimensional inputs. The proposed model has accurately classified the considered dataset. Table 1 presents the list of variables used in this article with their meaning.

2. Understanding the XOR Problem

XOR is a classical problem in the artificial neural network (ANN) [18]. The digital two-input XOR problem is represented in Figure 1. By considering each input as one dimension and mapping the digital digit '*0*' as the negative axis and '*1*' as the positive axis, the same two-digit XOR problem becomes XOR type nonlinear data distribution in two-dimensional space. It is obvious here that the classes in two-dimensional XOR data distribution are the areas formed by two of the axes 'X_1' and 'X_2' (Here, X_1 is input *1*, and X_2 is input *2*). Furthermore, these areas represent respective classes simply by their sign (i.e., negative area corresponds to class *1*, positive area corresponds to class *2*).

There are many other nonlinear data distributions resembling XOR. N-bit parity problem is one such typical example. Both these problems are popular in the AI research domain and require a generalized single neuron model to solve them. We have seen that these problems require a model which can distinguish between positive and negative quantities. Interestingly, addition cannot easily separate positive and negative quantities, whereas multiplication has the basic property to distinguish between positive and negative quantities. Therefore, previous researchers suggested using a multiplicative neuron model for solving XOR and similar problems.

3. Translated Multiplicative Neuron (Π_T-NEURON) Model

The idea of the multiplicative neuron model was initiated by Durbin et al. in 1989 [19]. They named this model the '*Product Units (PUs)*' and used this model to deal with the generalized polynomial terms in the input. It can learn higher-order inputs easily as compared to the additive units. This is because of its increased information capacity as compared to the additive units [19]. Though, PU has shown the capability for N-bit parity problems. However, it has issues in training with the standard backpropagation (BP) algorithm especially for higher-order inputs (more than three-dimensional input) [20]. According to Leerink et al., it is because of nonglobal minima trapping in the case of higher dimensional inputs [20]. Later, in 2004, Iyoda et al. proposed a single neuron based on a multiplicative neuron model, aka π_t-neuron model, to solve the XOR and parity bit problems [16, 17]. They have modified the previous multiplicative π-neuron model to find a suitable tessellation decision surface. They incorporated a scaling factor, a threshold value, and used the sigmoid as an activation function to solve the N-bit parity problems using a single translated multiplicative neuron (the model is defined by equations (1) and (2)) [16].

$$v_{\pi-t} = b_{\pi-t} \times \prod_{i=1}^{N} (x_i - t_i);\ Here,\ (b_{\pi-t})_i \in \mathbb{R}, t_i \in \mathbb{R}, \tag{1}$$

$$y = f(v_{\pi-t}). \tag{2}$$

Here, '$v_{\pi-t}$' represents the π_t-neuron model mathematically, 'y' is the final output through the activation function 'f', '$b_{\pi-t}$' is scaling factor, and 't_i' represent the coordinates of the center of the decision surfaces [16]. Mathematically, Iyoda et al. have shown the capability of the model for solving the logical XOR and N-bit parity problems for $\forall\ N \geq 1$. However, this model also has a similar issue in training for higher-order inputs.

3.1. Limitations of Translated Multiplicative Neuron.

The π_t-neuron model has shown the appropriate research direction for solving the logical XOR and N-bit parity problems [16]. The reported success ratio is '1' for two-bit to six-bit inputs in [17]. However, in the case of seven-bit input, the reported success ratio is '0.6' only. Success ratio has been calculated by considering averaged values over ten simulations [17]. Also, for successful training in the case of seven-bit, it requires adjusting the trainable parameter (scaling factor $b_{\pi-t}$) [17]. This is also indicating the training issue in the case of higher dimensional inputs. Moreover, Iyoda et al. have suggested increasing the range of initialization for scaling factors in case of a seven-bit parity problem [17]. Although, after the suggested increment as well, the reported success ratio is '0.6' only [17]. It indicates the problem of training in the π_t-neuron model for higher dimensional input.

3.2. Causes of Failure in Π_t-NEURON Model.

In the backpropagation algorithm, the local gradient '$\delta(n)$' accounts for the required changes in the trainable parameter at 'n^{th}' iteration to obtain desired output [21]. It is equal to the product of the corresponding error signal for that neuron and the derivative of the associated activation function [21]. Backpropagation requires that the activation function should be bounded, continuous, and monotonic. Also, it should be continuously differentiable for the entire domain of the input to get optimization [22]. Sigmoid activation function '$\phi(x)$' is preferred in the classification problem because it has met all of the aforementioned requirements [23]. Also, it is an appropriate activation function for training multiplicative neuron models [23]. Iyoda et al. have demonstrated the error gradient ($\nabla\mathcal{E}$) associated with the π_t-neuron model [17]. Here, '$\mathcal{E}(n)$' is the error energy, i.e., the instantaneous sum of the error squares at 'n^{th}' iteration. The error gradient has two components, one is due to the scaling factor '$(b_{\pi-t})$,' given by equation (3), and the other is due to the thresholds 't_i', given by equation (4) [17].

Table 1: List of variables used in this article with their meaning.

Variables	Meaning
π-neuron	Multiplicative neuron model
π_t-neuron	Symbol of translated multiplicative neuron model
Π	Product operator
$v_{\pi-t}$	The mathematical form of the p_t-neuron model
f	Activation function
x_i	'i^{th}' input to the neuron
y	The final output of the p_t-neuron model through the activation function 'f'
$b_{\pi-t}$	Scaling factor associated with the p_t-neuron model
t_i	The coordinates of the center of the decision surfaces ('i^{th}' threshold value)
$\delta(n)$	The local gradient in the backpropagation algorithm at 'n^{th}' iteration
$\phi(x)$	Sigmoid activation function
$\mathcal{E}(n)$	The error energy (the instantaneous sum of the error squares at 'n^{th}' iteration)
$\nabla\mathcal{E}$	Error gradient
$e(n)$	The error value in the backpropagation algorithm at 'n^{th}' iteration
ϕ'	Derivative of the sigmoid activation function
b	Scaling factor associated with the enhanced p_t-neuron model (proposed)
v	The mathematical form of the proposed model
O	The final output of the proposed model through the activation function 'ϕ'
$\mathbb{R}$	Set of the real numbers
L_1, L_2, L_∞	Respective loss functions
$\mathscr{L}$	L_1 loss function
W	Weight matrix
G	Gaussian probability distribution
p	Number of samples required in an XOR dataset for appropriate training
μ	Mean or average value
σ	Standard deviation value

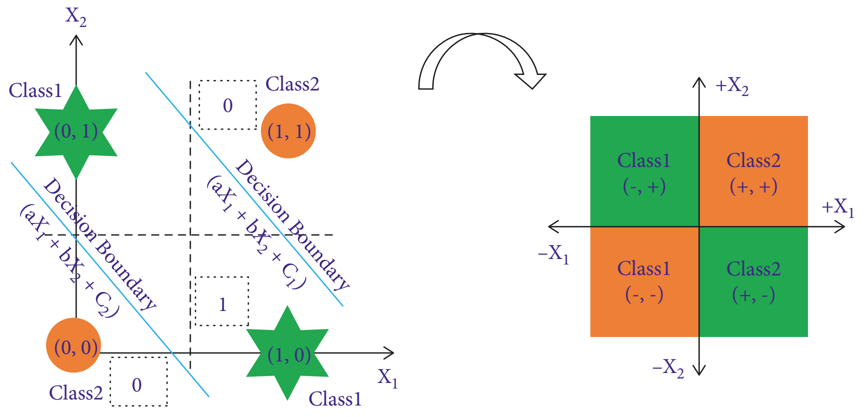

Figure 1: XOR problem is illustrated by considering each input as one dimension and mapping the digital digit '*0*' as negative axis and '*1*' as the positive axis. Therefore, XOR data distribution is the areas formed by two of the axes 'X_1' and 'X_2', such that the negative area corresponds to class *1*, and the positive area corresponds to class *2*.

$$\frac{\partial \varepsilon(n)}{\partial b_{\pi-t}(n)} = \delta(n) \times \prod \left(x_k(n) - t_k(n)\right), \tag{3}$$

$$\frac{\partial \varepsilon(n)}{\partial t_i(n)} = -\delta(n) \times b_{\pi-t} \times \prod \left(x_k(n) - t_k(n)\right); \forall (\mathrm{i} \neq k), \tag{4}$$

$$\delta(n) = e(n) \times \phi'\left(v_{\pi-t}(n)\right). \tag{5}$$

Here, 'n' represents 'n^{th}' iteration, $\forall$ ($k = 1, 2, 3, \ldots, N$). '$x_k(n)$' is the 'k^{th}' input for 'n^{th}' iteration, and '$v_{\pi-t}(n)$' represents π_t-neuron model. Therefore, the error's gradient obtains a much smaller value after getting multiplied for higher dimensional inputs and becomes an infinitesimally small value. Consequently, vanishing the gradient. Therefore, a convergence problem occurs in this model.

It is inferred from Figure 1 and equation (1) that the π_t-neuron model has ranged between [−1, 1] for XOR and N-bit parity problems. Here, '*−1*' corresponds to digit '*0*', and '*+1*' corresponds to digit '*1*'. Sigmoid function has a basic issue of vanishing gradient near the extremes as shown in Figure 2(a). However, about the XOR and N-bit parity problems, the input varies between [−1, 1] only, as explained earlier. Therefore, the main region of interest is incorporated

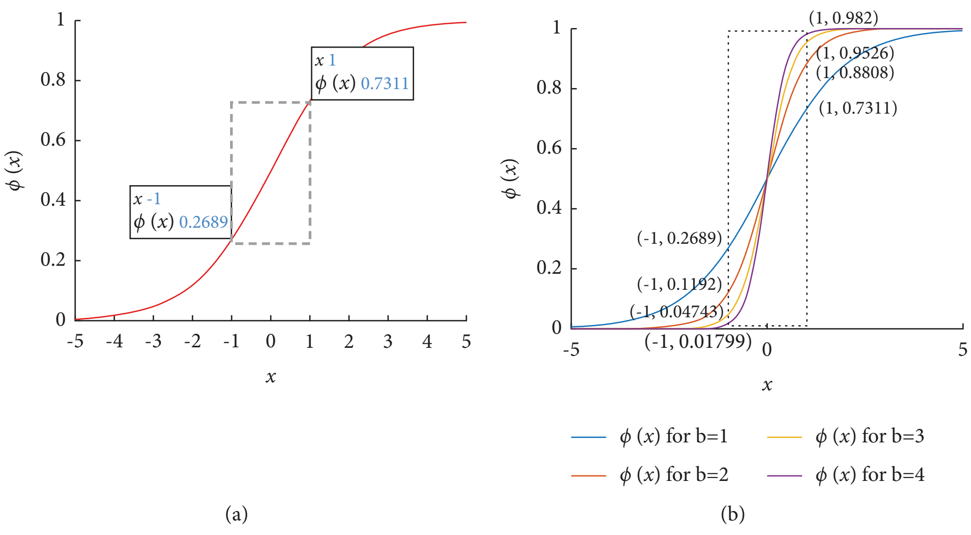

FIGURE 2: (a) Sigmoid function $\phi(x)$; (b) effect of scaling on $\phi(x)$.

by a rectangular box of sigmoid activation function in Figure 2. Here, it is important to notice that margin between two points has been reduced by the sigmoid activation function (as shown in Figure 2(a), $\phi\ (-1) = 0.2689$, and $\phi\ (1) = 0.7311$). Therefore, it leads to the smaller local gradient '$\delta(n)$' value (given by Equation (5)) which consequently results in smaller error gradients (given equations (3)–(5)), eventually leading to the gradient vanishing problem.

For higher dimensional input, the error gradient ($\nabla\mathcal{E}$) attains further smaller values because of the presence of the factor ($\prod (x_k(n) + t_k(n))$) in the expression of error gradients (as given by equations (3)–(5)). Therefore, the possibilities of nonconvergence/nonglobal minima problems occur in the previous π_t-neuron model. To overcome this issue, the model should have a larger margin for the extreme values. It is possible by introducing a compensatory scaling factor in the model. It eventually scales the sigmoid activation function, as depicted in Figure 2(b). Therefore, in [17], the author suggested using a scaling factor '$b_{\pi-t}$'. However, it requires an optimized value of the scaling factor to mitigate the effect of multiplication and sigmoid function in higher-dimensional problems. Because the effect of multiplication and sigmoid function is severe in higher-order input, Iyoda et al. recommended initializing the scaling factor only with higher values (not to the threshold factor) for the seven-bit parity problem [17]. Convergence is not possible with a smaller scaling factor for the higher dimensional problem (results given in 'Table 2' of [17] follow this statement). Though, the idea of increasing the learning rate for the scaling factor is worth overcoming the vanishing gradient problem in higher dimensional input. However, an optimized value of the learning rate is not suggested in the previous π_t-neuron model. Also, it is difficult to adjust the appropriate learning rate or range of initialization of scaling factors for variable input dimensions. Therefore, a generalized solution is still required to solve these issues of the previous model. In this paper, we have suggested a generalized model for solving the XOR and higher-order parity problems by enhancing the p_t-neuron model.

TABLE 2: Assessment of Proposed Model (through variation in dimension and no. of training samples).

Dimension (N)	10^3		10^4		10^6	
	Time (*s*)	$\mathscr{L}$ (*Ours*)	*Time* (*s*)	$\mathscr{L}$ (*Ours*)	*Time* (*s*)	$\mathscr{L}$ (*Ours*)
2	0.710	0.0195	0.712	0.0195	1.571	0.0244
5	0.692	0.0105	0.768	0.0091	2.808	0.0088
7	0.686	0.0093	0.759	0.0064	3.822	0.0058
10	0.688	0.0184	0.770	0.0048	5.806	0.0040
13	0.693	0.0577	0.790	0.0045	7.666	0.0023
15	0.703	0.2156	0.793	0.0126	9.435	0.0020
18	0.696	0.9041	0.805	0.1501	8.497	0.0150
20	0.701	0.9883	0.812	0.2441	9.378	0.0679
25	0.701	1.2155	0.811	0.3790	11.982	0.1188

4. Related Works

Robotics, parity problems, and nonlinear time-series prediction are some of the significant problems suggested by the previous researchers where multiplicative neurons are applied. Forecasting involving the time series has been performed using the multiplicative neuron models [24–26]. Yildirim et al. have proposed a threshold single multiplicative neuron model for time series prediction [24]. They utilized a threshold value and used the particle swarm optimization (PSO) and harmony search algorithm (HSA) to obtain the optimum weight, bias, and threshold values. In [25], Yolcu et al. have used autoregressive coefficients to predict the weights and biases for time series modeling. A recurrent multiplicative neuron model was presented in [26] for forecasting time series.

Yadav et al. have also used a single multiplicative neuron model for time series prediction problems [27]. In [28], authors have used the multiplicative neuron model for the prediction of terrain profiles for both air and ground vehicles. Egrioglu et al. have represented forecasting purposes like classical time series forecasting using a single multiplicative neuron model in [29]. In [30], Gao et al. proposed a dendritic neuron model to overcome the limitation of traditional ANNs. It has utilized the nonlinearity of synapses to improve the capability of artificial neurons. A few other recent works are suggested in [31–35].

5. Enhanced Translated Multiplicative Neuron

We have seen the problems associated with the π_t-neuron model. It has an issue with BP training in case of highly dense XOR data distribution and higher dimensional parity problems. In this paper, we have proposed an enhanced translated multiplicative single neuron model which can easily learn the nonlinear problems such as XOR and N-bit parity without any training limitations. We have modified the existing π_t-neuron to overcome its limitations. The proposed enhanced translated multiplicative neuron model is represented in Figure 3 and described as follows:

$$v = (-1)^{N+1} \times \prod_{i=1}^{N} \{(b_i) \times (x_i + t_i)\},$$
$$\textit{Here}, b_i \in \mathbb{R}, t_i \in \mathbb{R}. \quad (6)$$
$$\textit{For } b_1 = b_2 = \cdots = b_i = \cdots = b; \textit{and}, b \in \mathbb{R}, \textit{then}:$$

$$v = (-1)^{N+1} \times \prod_{i=1}^{N} \{(b) \times (x_i + t_i)\}. \quad (7)$$

Therefore, the final output through the proposed model for an N-input neuron is obtained by equation (8) as follows:

$$O = \phi(v). \quad (8)$$

Further simplifying the proposed model (as given by equation (7)), we have the following:

$$v = (-1)^{N+1} \times b^N \times \prod_{i=1}^{N} (x_i + t_i). \quad (9)$$

5.1. Scaling Factor in Proposed Model. The issue of vanishing gradient and nonconvergence in the previous π_t-neuron model has been resolved by our proposed neuron model. It is because of the input dimension-dependent adaptable scaling factor (given in equation (6)). The effect of the scaling factor is already discussed in the previous section (as depicted in Figure 2(b)). We have seen that a larger scaling factor supports BP and results from proper convergence in the case of higher dimensional input. The significance of scaling has already been demonstrated in Figure 2(b). Figure 4 is the demonstration of the optimal value of scaling factor 'b'.

To illustrate the significance of the optimized value of scaling factor 'b', we have plotted the gradient of sigmoid function '$\phi'(x)$' by considering variation in the values of 'b' in Figure 3. It is observed from the plot that the scaling factor, $b = 1$, has poor sensitivity for any change in the input. Also, the sensitivity of the '$\phi'(x)$' increases by increasing the value of scaling factor 'b'. However, as we increase the scaling factor 'b' more than 6, we have poor sensitivity regions again, causing gradient vanishing problems. Vanishing gradient regions are shown by encircled areas in the plot. It shows an optimal value is between (3 ~ 6). For less than three, it has smaller sensitivity, and for more than six, it again shows the gradients vanishing problem. In our experiment, we have empirically found that initializing the scaling factor 'b' with the value '4' for each input results in successful training. However, we require to fine-tune the scaling factor according to the input and its dimension.

Therefore, we have considered the optimization of the scaling factor depending on the dimension and value of the input in our model. Therefore, we have considered an adaptable scaling factor (b_i) which is associated with each input (x_i) in our proposed model (as given by Equation (6)). Further, it has another advantage in that it helps in rapidly achieving the optimized value of the scaling factor without changing the learning rate in training the model. It eventually helps in achieving convergence using the BP algorithm in training the model. Mathematically, the error gradient ($\nabla\mathcal{E}$) associated with our proposed neuron model (obtained by equations (3)–(5)) is defined as follows:

$$\frac{\partial\varepsilon(n)}{\partial b(n)} = \delta(n) \times \mathrm{N} \times b^{N-1} \times \prod (x_k(n) + t_k(n)), \frac{\partial\varepsilon(n)}{\partial t_i(n)} = \delta(n) \times b^N \times \prod (x_k(n) + t_k(n)); \forall (i \neq k). \quad (10)$$

Here, the larger scaling factor 'b^N' accurately compensates for infinitesimally small gradient problems. Therefore, the larger scaling factor enforces a sharper transition to the sigmoid function and supports easier learning in case of higher dimensional parity problems. In the proposed model, the scaling factor is trainable and depends upon the number of input bits. It has exponent term as the no. of input bits means, for higher input we have sharper transition which compensates for infinitesimally small gradient problems. Therefore, the proposed enhanced π_t-neuron model has no limitation for higher dimensional inputs.

5.2. Sign-Manipulation in the Proposed Model. The enhanced π_t-neuron is based on the multiplicative neuron model. The multiplicative model suffers from a class reversal problem. It is the reversal of class depending upon the number of input bits. It is because of the sign change property of the multiplicative model according to even and odd input dimensions. This leads to severe confusion in classification. To mitigate this issue, we have multiplied a sign-manipulation factor as '$(-1)^{N+1}$'. Therefore, it introduces an extra negative sign for the even number of input bits to maintain the input combinations belonging to the same class. These two (scaling

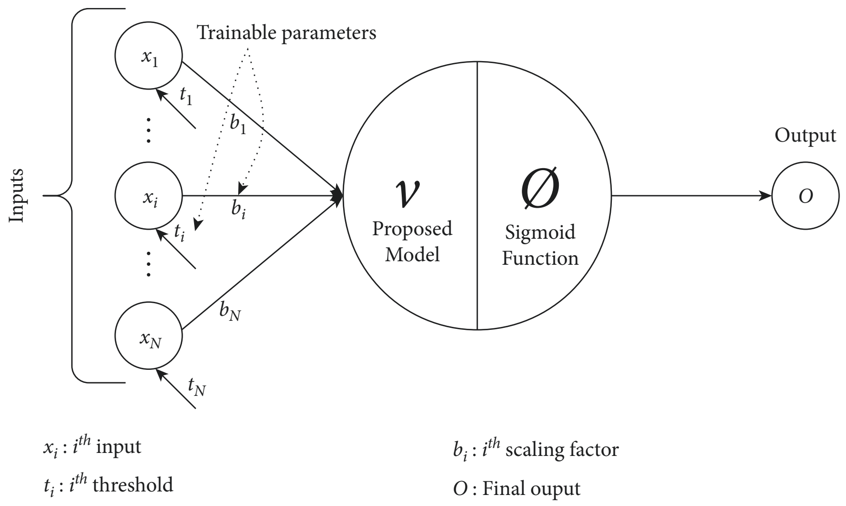

Figure 3: Proposed translated multiplicative neuron architecture.

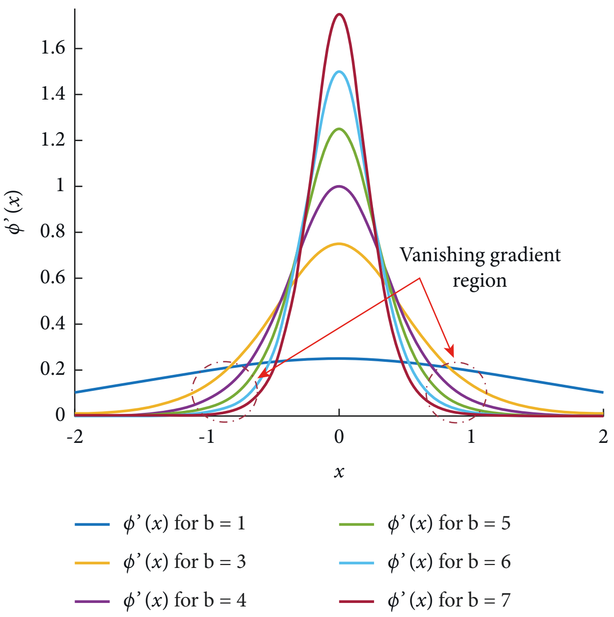

Figure 4: Effect of scaling factor on the gradient of the sigmoid function.

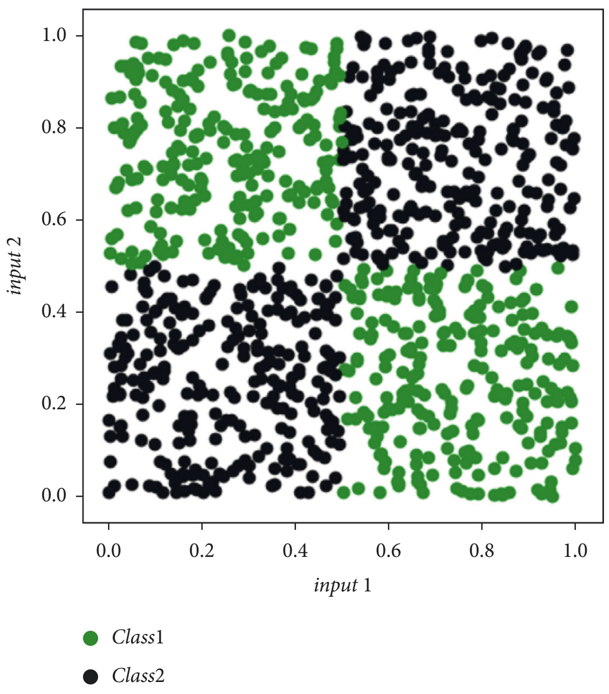

Figure 5: A typical highly dense two-input XOR data distribution.

factor and sign-manipulation) modifications in the existing π_t-neuron model have enhanced its performance for highly dense XOR data distribution and higher-order N-bit parity problems.

6. Results and Discussion

We have used gradient–decedent algorithm for training the proposed neuron model. The binary cross-entropy loss function is used for estimating loss between target and trained threshold vectors training on a single '*Nvidia Geforce eXtreme 1080*' graphic card. The efficacy of the proposed neuron has been evaluated for generalized XOR problems. We have considered a typical highly dense two-input XOR data distribution, as shown in Figure 5. It is applied to both models (i.e., the π_t-neuron model and the proposed model) to compare the efficacy of the model. There are many popular loss functions to visualize the deviation in desired and predicted values, such as L_1 loss, L_2 loss, and L_∞ loss. However, in our situation, data points vary between [0, 1], and L_1 loss renders the best visualization in such cases. Therefore, we have considered the L_1 Loss function, which is the least absolute deviation, and used it to estimate the error. The L_1 loss ($\mathscr{L}$) is defined as follows:

$$\mathscr{L} = \sum_{i=1}^{N} \left| (\text{desired})_i - (\text{predicted})_i \right|. \quad (11)$$

Since random weights and biases are important in the training of the model. That is why we have considered He-initialization [36] in our approach. It is a variant of Xavier-

Initialization [37]. In He-initialization, the biases are initialized with 0.0 (zero value) and the weight is initialized using Gaussian probability distribution (G) given as $(W) \sim G(0, \sqrt{2/r_l})$ for 'l^{th}' layer. Here, 'r' denotes the number of connections. Further, to assess the applicability and generalization of our proposed single neuron model, we have varied the input dimension and no. of input samples in training the proposed model. We have considered three different cases having 10^3, 10^4, and 10^6 samples in the dataset, respectively. Results (in all three cases) have been summarized in Table 2. Results show that the loss depends upon the no. of samples in the dataset. It decreases by increasing the number of samples.

Number of samples required in the XOR dataset for appropriate training depends upon the input dimension. It is given by the following equation:

$$p = 2^N. \tag{12}$$

Here, 'p' is the number of required samples for 'N' dimensional input. To understand this relation, consider two-dimensional datasets (i.e., $N = 2$). Therefore, the no. of the required sample (i.e., p) is obtained by (9) as ($p = 2^2 = 4$). It is the classical exclusive OR (XOR) dataset, represented as {(0, 0), (0, 1), (1, 0), (1, 1)}. Similarly, if ($N = 3$), then ($p = 2^3 = 8$), which indicates a three-input XOR dataset, and so on. Lesser samples in the training dataset cause non-convergence and inaccuracy.

Equation (12) tells the number of samples required in the training dataset. Therefore, for ten-dimensional input, the number of samples required for training should be ($p = 2^{10} = 1024$). Therefore, approximately 1,000 samples are sufficient for a ten-dimensional training dataset. However, if we increase the dimension, it requires more no. of samples to train the model appropriately. Otherwise model fails to get converge. The same is shown in Table 3. To assess the accuracy of our proposed model, we repeated each experiment 25 times and provided accurate results. Here, the success rate signifies the ratio of successful simulation over total simulations for each case. In the case of ten-dimensional input for 1000 training samples, the success rate is 0.96, whereas it is reduced to 0.76 in the case of thirteen-dimensional input because of insufficient training samples. However, if we increase the no. of training samples to 10,000, the model report 100% of success ratio. Similarly, for 20 bits input ($p = 2^{20} = 1{,}048{,}576$), samples are required. Therefore, by training 1,000 samples, the success ratio is 0.0, while for 10,000 samples, it is 0.32. It increases further to 0.64 for one million samples. These results furnish the importance of no. of training samples for solving XOR type nonlinear problems. Also, by observing the results, we can easily understand the capability of the proposed model for generalized XOR type real problems.

Further, the proposed algorithm has been repeated 30 times to assess the performance of its training. The standard statistical indicators such as mean (μ) and standard deviation (σ) are considered the assessment parameters of the predicted values. Table 4 provides the prediction results (in terms of threshold values (t_1, t_2) and scaling factor (b)) obtained by the proposed models. It also showcases the mean and standard deviations of the predicted thresholds and bias values.

Table 5 provide values of the threshold obtained by both the p_t-neuron model and proposed models. In experiment #2 and experiment #3, the p_t-neuron model has predicted threshold values beyond the range of inputs, i.e., [0, 1]. This is because we have not placed any limit on the values of the trainable parameter. It only reflects that the π_t-neuron model has been unable to obtain the desired value in these experiments.

L_1 loss ($\mathscr{L}$) obtained in these three experiments for the π_t-neuron model, and the proposed model is provided in Table 3. This loss function is only used to visualize the comparison in the model. As mentioned earlier, we have used the binary cross-entropy loss function to train our model.

It is observed by the results of Tables 5 and 6 that the π_t-neuron model has a problem in learning highly dense XOR data distribution. However, the proposed neuron model has shown accurate classification results in each of these cases. Also, the loss function discerns heavy deviation as predicted and desired values of the π_t-neuron model.

Further, we have monitored the training process for both models by measuring the binary cross-entropy (BCE) loss versus the number of iterations (as shown in Figure 6). We should remember that it is the cross-entropy loss on a logarithmic scale and not the absolute loss. It supports backpropagation error calculation which is an issue with smaller errors. It is generally considered an appropriate loss metric in classification problems. Therefore, we have used BCE as a measure to observe the trend of training to compare the π_t-neuron model with our proposed model. As observed, the proposed model has achieved convergence which is not obtained by the π_t-neuron model. We have examined the performance of our proposed model over N-bit parity problems. We have considered similar data distribution (as that in Figure 5) for parity problems as well. Further, we have compared the training performance of the π_t-neuron model with our proposed model for the 10-bit parity problem. Training results of both models have been represented in Figure 7 (by plotting binary cross-entropy loss versus the number of iterations).

We have examined the performance of our proposed model for higher dimensional parity problems. It is to assess the applicability and generalization of our model. We have randomly varied the input dimension from 2 to 25 and compared the performance of our model with π_t-neuron. Results are tabulated below. Table 7 provides the scaling factor and loss obtained by both π_t-neuron and proposed neuron models.

As mentioned earlier, we have measured the performance for the N-bit parity problem by randomly varying the input dimension from 2 to 25. L_1 loss function has been considered to visualize the deviations in the predicted and desired values in each case. The proposed model has shown much smaller loss values than that of with π_t-neuron model. Also, the proposed model has easily obtained the optimized value of the scaling factor in each case. Tessellation surfaces

Table 3: Success Rate (through variation in dimension and no. of training samples).

Dimension (N)	10^3	10^4	10^6
2	1.0	1.0	1.0
5	1.0	1.0	1.0
7	1.0	1.0	1.0
10	0.96	1.0	1.0
13	0.76	1.0	1.0
15	0.32	0.96	1.0
18	0.0	0.40	0.92
20	0.0	0.32	0.64
25	0.0	0.0	0.24

Table 4: Predictions Through the Proposed Model (in terms of threshold values (t_1, t_2), and scaling factor (b)).

Experiment no.	t_1	t_2	b
1	0.4982	0.5097	3.9651
2	0.5049	0.5079	3.97
3	0.4958	0.5113	3.9869
4	0.5099	0.4967	3.98
5	0.5015	0.4967	3.969
6	0.4903	0.5014	3.9693
7	0.5002	0.5076	3.9652
8	0.504	0.511	3.9638
9	0.4973	0.5138	3.9872
10	0.4995	0.4887	3.9773
11	0.5031	0.497	3.9777
12	0.4945	0.497	3.9802
13	0.5019	0.4989	3.9773
14	0.4974	0.4895	3.9688
15	0.4921	0.5024	3.9807
16	0.4943	0.5106	3.9858
17	0.5074	0.5101	3.9697
18	0.5075	0.4975	3.9686
19	0.4914	0.4959	3.9702
20	0.4974	0.4888	3.9772
21	0.5019	0.5138	3.9628
22	0.5084	0.5005	3.9631
23	0.5085	0.5055	3.964
24	0.5138	0.4993	3.9742
25	0.5063	0.5026	3.9634
26	0.5022	0.4901	3.9702
27	0.503	0.5147	3.9866
28	0.4992	0.4831	3.9636
29	0.512	0.4983	3.9696
30	0.4968	0.4975	3.9706
Mean (μ)	0.5014	0.5013	3.9726
Standard deviation (σ)	0.00614	0.00849	0.00781

Table 5: Comparison of Π_t-Neuron Model and Proposed Model (in terms of threshold values).

Experiment no.	Desired		Predicted (p_t)		Predicted (ours)	
	t_1	t_2	t_1	t_2	t_1	t_2
1	0.5	0.5	0.6023	0.1546	0.5024	0.5037
2	0.580	0.672	0.5567	2.6124	0.579	0.674
3	0.460	0.084	0.4679	−15.9951	0.459	0.085

Table 6: L_1 loss ($\mathscr{L}$) Obtained by Π_t-Neuron Model and Proposed Model.

Experiment no.	$\mathscr{L}$ (π_t)	$\mathscr{L}$ (Ours)
1	0.1744	0.003
2	0.9818	0.0015
3	8.0435	0.001

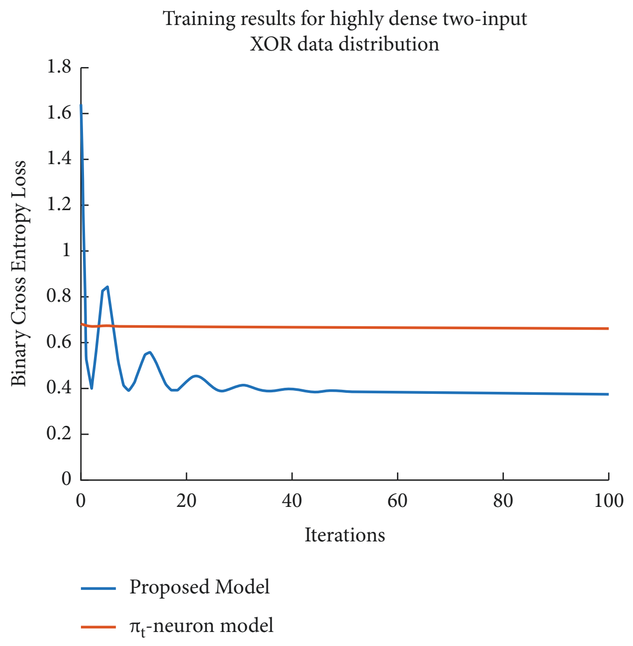

Figure 6: Training progress of both models (i.e., π_t-neuron model and proposed model). Here, we have considered a typical highly dense two-input XOR data distribution. The result shows that the π_t-neuron model has an issue in training while the proposed model has achieved convergence.

formed by the π_t-neuron model and the proposed model have been compared in Figure 8 to compare the effectiveness of the models (considering two-dimensional input).

This is observed here that the proposed model has formed an enhanced tessellation surface than that of the π_t-neuron model. It is merely because of the optimal scaling. In the case of the π_t-neuron model, the scaling factor is ($b_{\pi-t} = -1.7045$), whereas our model has obtained the scaling factor as

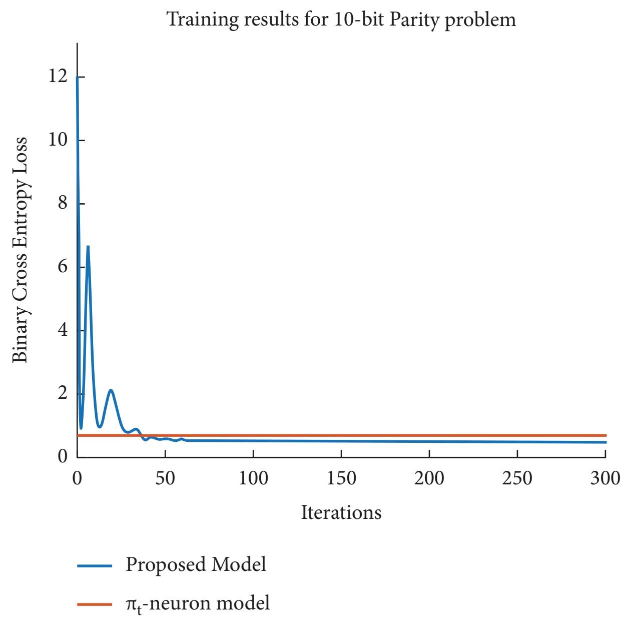

FIGURE 7: Results show the training progress of both models (i.e., π_t-neuron model and proposed model) for the 10-bit parity problem. The proposed model has achieved convergence while the π_t-neuron model has not.

TABLE 7: Scaling Factor and loss Obtained by Π_t-Neuron and Proposed Models with Increasing Input Dimension of N-bit Parity Problem.

N	Scaling factor		L_1 loss ($\mathscr{L}$)	
	π_t-neuron ($b_{\pi-t}$)	Ours (b)	π_t-neuron	Ours
2	−1.7045	4.6900	0.3281	0.0093
5	0.3911	4.2760	2.1059	0.0093
7	72.1092	4.2208	0.8367	0.0058
10	−42.6576	4.0720	0.5690	0.0047
13	33.9146	4.0548	1.0349	0.0037
15	64.4870	3.9516	0.8966	0.0079
18	−20.1572	3.8317	0.9393	0.0072
20	−79.3860	3.9426	0.8311	0.0072
25	83.9456	3.7275	0.9855	0.1431

($b = 4.6900$). As we have discussed earlier, the value of the scaling factor associated with input should be around (4) for each input (described in Figure 4). Further, because of the two-dimensional problem, the effective scaling factor in our case is ($b^N = 21.9961$). We have plotted the effective values of the scaling factor in our proposed model and the π_t-neuron model on a logarithmic scale to visualize the effect of scaling with increasing input dimension in Figure 9.

The trend of variation of the effective scaling factor with an increasing dimension of input discerns that the proposed model can rapidly increase the required value of the scaling factor to compensate for the effect of miniaturization of errors within higher dimensional input. However, the previous π_t-neuron model has no such ability. This is possible in our model by providing the compensation to each input (as given in our proposed enhanced π_t-neuron model by equation (6)). We have considered the input distribution similar to Figure 5 (i.e., the input varies between

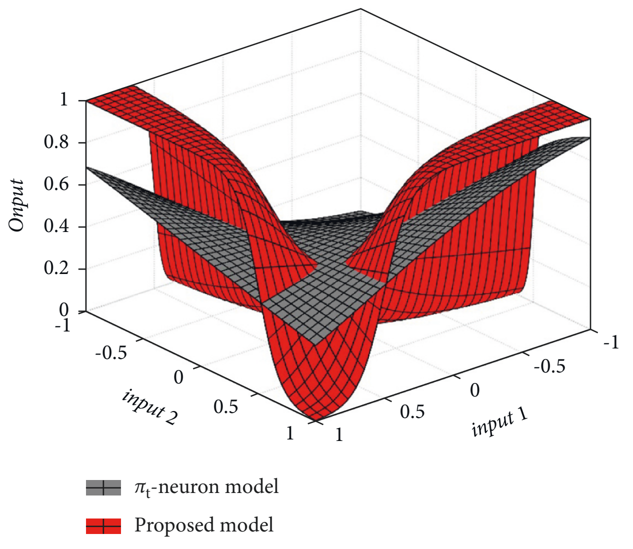

FIGURE 8: Tessellation surface formed by π_t-neuron model and proposed model for two-dimensional input.

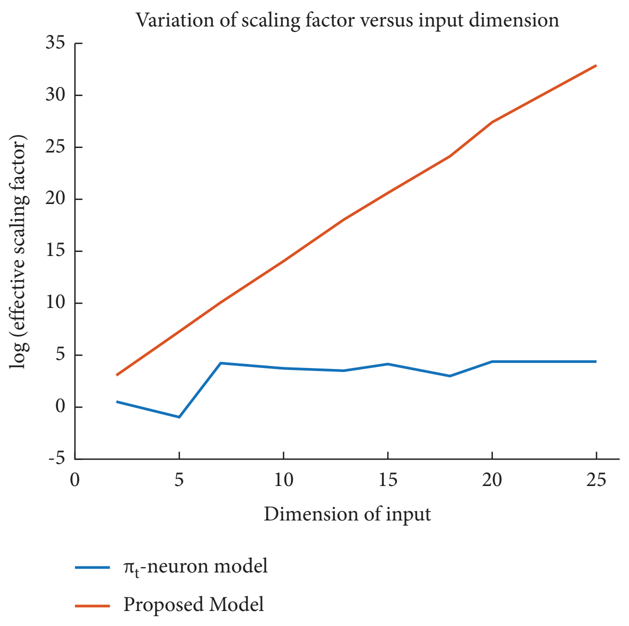

FIGURE 9: Trend of scaling factor variation for (N)-bit parity problem is compared in both of the models (i.e., π_t-neuron model and proposed model). Here, the effective scaling factor for a π_t-neuron model is '($b_{\pi-t}$)', whereas for the proposed model 'b^N.'

[0, 1]) for each dimension. Results show that the effective scaling factor depends upon the dimension of input as well as the magnitude of the input. Therefore, our proposed model has overcome the limitations of the previous π_t-neuron model.

Further, the computational complexity of the proposed model is obtained from the investigation of Schmitt in [38]. Schmitt has investigated the computational complexity of multiplicative neuron models. They have used the Vapnik-Chervonenkis (VC) dimension and the pseudo dimension to analyze the computational complexity of the multiplicative neuron models. The VC dimension is a theoretical tool that quantifies the computational complexity of neuron models. According to their investigation for a single product unit the VC dimension of a product unit with N-input variables is equal to N.

7. Discussion and Conclusions

Translated multiplicative (π_t) neuron model has been suggested by past researchers to solve the XOR and N-bit parity problems. However, it has an issue in backpropagation for densely distributed XOR and higher dimensional parity problems. It is an indigenous problem associated with multiplicative neuron models. Though the π_t-neuron model has a scaling factor in subduing this problem, however, without suitable initialization, it is unable to obtain the appropriate scaling factor for higher-dimensional input. Therefore, a generalized solution is still required to overcome these issues. In this paper, an enhanced translated multiplicative neuron modeling has been proposed to enhance the performance of the π_t-neuron model. The proposed model can obtain the optimized value of the scaling factor for any input dimension. It has solved the existing backpropagation issue of the π_t-neuron model. We have considered an adaptable scaling factor associated with each input in our proposed model. This helps in achieving optimal scaling factor value for higher dimensional input. We have assessed the efficacy of our model by randomly increasing input dimensions and considered a magnitude variation between [0, 1] for each input. The proposed model has outperformed the π_t-neuron model in each case. It has shown more than an 80% reduction in absolute loss as compared to the previous neuron model in similar experimental conditions. Also, the proposed model has formed a more accurate tessellation surface as compared to the previous model for two-dimensional input. Further, there are multiple real-world implementations involving the time series forecasting and classification such as trends analysis, seasonal (weather) predictions, cycle, and irregularity predictions. These real-world problems are associated with forecasting and classifications of time-series data. A multiplicative neuron model is commonly employed in such predictions and renders superior results. Our proposed single multiplicative neuron model has overcome the limitations of dimensionalities. Therefore, it can be easily employed in such prediction tasks as well.

Data Availability

The data used to support the findings of this study are included in the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Authors' Contributions

Ashutosh Mishra conceptualized the study. Ashutosh Mishra and Jaekwang Cha developed the methodology. Ashutosh Mishra performed a formal analysis and investigated the study. Ashutosh Mishra wrote, reviewed, and edited the paper. Ashutosh Mishra and Jaekwang Cha provided the software and performed validation, visualization, and data curation. Ashutosh Mishra provided the resources and prepared the manuscript. Shiho Kim supervised the study and was responsible for project administration and and funding acquisition. All authors have read and agreed to the published version of the paper. For correspondence, any of the authors can be addressed (Ashutosh Mishra; ashutoshmishra@yonsei.ac.kr; Jaekwang Cha; chajae42@yonsei.ac.kr, and Shiho Kim; shiho@yonsei.ac.kr).

Acknowledgments

This work was partially supported by Brain Pool Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2019H1D3A1A01071115) and by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00966, Development of AI processor with a Deep Reinforcement Learning Accelerator adaptable to Dynamic Environment).

References

[1] M. L. Minsky and S. A. P.. Perceptrons, *An Introduction to Computational Geometry*, MIT Press, Cambridge, Massachusetts, United States, 2017.

[2] H. E. Amir and M. Hamdy, "Deep learning fundamentals," in *Deep Learning Pipeline*Apress, Berkeley, CA, 2020.

[3] A. Özdemir and M. M. İnal, "Only one neuron either N-bit parity rule based modified translated multiplicative or McCulloch-pitts models for some machine learning problems," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 4, pp. 67–72, 2016, https://dergipark.org.tr/en/download/article-file/233567.

[4] B. E. Rosen, J. M. Goodwin, and J. J. Vidal, "Transcendental functions in backward error propagation," in *Proceedings of the 1990 IEEE International Conference on Systems, Man, and Cybernetics Conference*, pp. 239–241, Los Angeles, CA, USA, 1990.

[5] A. Kaveh and J. Vogh, "Influence of different nonlinearity functions on Perceptron performance," in *Proceedings of the SPIE Proceedings,Applications of Optical Engineering: Proceedings of OE/Midwest '90*, pp. 215–225, Rosemont, IL, USA, 1991.

[6] Z. Zhang and M. Sarhadi, "A modified neuron activation function which enables single layer perceptrons to solve some linearly inseparable problems," in *Proceedings of the 1993 International Conference on Neural Networks*, pp. 2723–2726, Nagoya, Japan, 1993.

[7] R. Labib, "New single neuron structure for solving nonlinear problems," in *Proceedings of the IJCNN'99. International Joint Conference on Neural Networks. (Cat. No.99CH36339)*, pp. 617–620, Washington, DC, USA, 1999.

[8] Y. Wu, M. Zhao, and X. Ding, "A new kind of neuron model with a tunable activation function and its applications," *Science in China - Series E: Technological Sciences*, vol. 40, no. 1, pp. 105–112, 1997.

[9] Y. Wu and M. Zhao, "A neuron model with trainable activation function (TAF) and its MFNN supervised learning," *Science in China, Series A: Information Sciences*, vol. 44, no. 5, pp. 366–375, 2001.

[10] Z. Yanling, D. Bimin, and W. Zhanrong, "Analysis and study of perceptron to solve XOR problem," in *Proceedings of the The 2nd International Workshop on Autonomous Decentralized System, 2002*, pp. 168–173, Beijing, China, 2002.

[11] T. Nitta, "Solving the XOR problem and the detection of symmetry using a single complex-valued neuron," *Neural Networks*, vol. 16, no. 8, pp. 1101–1105, 2003.

[12] Y. Shen, B. Wang, F. Chen, and L. Cheng, "A new multi-output neural model with tunable activation function and its applications," *Neural Processing Letters*, vol. 20, no. 2, pp. 85–104, 2004.

[13] M. F. Amin and K. Murase, "Single-layered complex-valued neural network for real-valued classification problems," *Neurocomputing*, vol. 72, pp. 945–955, 2009.

[14] N. Tsapanos, A. Tefas, N. Nikolaidis, and I. Pitas, "Neurons with paraboloid decision boundaries for improved neural network classification performance," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 1, pp. 284–294, 2018.

[15] A. Sagheer, M. Zidan, and M. M. Abdelsamea, "A novel autonomous perceptron model for pattern classification applications," *Entropy*, vol. 21, no. 8, p. 763, 2019.

[16] E. M. Iyoda, H. Nobuhara, and K. Hirota, "A solution for the *N*-bit parity problem using a single translated multiplicative neuron," *Neural Processing Letters*, vol. 18, no. 3, pp. 233–238, 2003.

[17] E. M. Iyoda, H. Nobuhara, and K. Hirota, "Translated multiplicative neuron: an extended multiplicative neuron that can translate decision surfaces," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 8, no. 5, pp. 460–468, Apr. 2004.

[18] R. Bland, *Learning XOR: Exploring the Space of a Classic Problem*, Stirling: Department of Computing Science and Mathematics, University of Stirling, Stirling, Scotland, 1998.

[19] R. Durbin and D. E. Rumelhart, "Product units: a computationally powerful and biologically plausible extension to backpropagation networks," *Neural Computation*, vol. 1, pp. 133–142, 1989.

[20] L. R. Leerink, C. L. Giles, B. G. Horne, and M. A. Jabri, "Learning with product units," *Advances in Neural Information Processing Systems*, vol. 7, pp. 537–544, 1995, https://clgiles.ist.psu.edu/papers/NIPS94.product.units.pdf.

[21] S. Haykin, "Multilayer perceptrons," in *Neural Networks: A Comprehensive Foundation*, Prentice-Hall, Hoboken, New Jersey, United States, 2nd ed edition, 1999.

[22] M. H. Bakr and M. H. Negm, "Modeling and design of high-frequency structures using artificial neural networks and space mapping," in *Advances in Imaging and Electron Physics,* vol. 174, pp. 223–260, Elsevier, 2012.

[23] B. Karlik and A. V. Olgac, "Performance analysis of various activation functions in generalized MLP architectures of neural networks," *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, pp. 111–122, 2011.

[24] A. N. Yildirim, E. Bas, and E. Egrioglu, "Threshold single multiplicative neuron artificial neural networks for nonlinear time series forecasting," *Journal of Applied Statistics*, vol. 48, no. 13-15, pp. 2809–2825, 2021.

[25] O. C. Yolcu, E. Bas, E. Egrioglu, and U. Yolcu, "Single multiplicative neuron model artificial neural network with autoregressive coefficient for time series modelling," *Neural Processing Letters*, vol. 47, no. 3, pp. 1133–1147, 2018.

[26] E. Egrioglu, U. Yolcu, C. H. Aladag, and E. Bas, "Recurrent multiplicative neuron model artificial neural network for nonlinear time series forecasting," *Neural Processing Letters*, vol. 41, no. 2, pp. 249–258, 2015.

[27] R. N. Yadav, P. K. Kalra, and J. John, "Time series prediction with single multiplicative neuron model," *Applied Soft Computing*, vol. 7, no. 4, pp. 1157–1163, 2007.

[28] W. Pan, L. Zhang, and C. Shen, "Data-driven time series prediction based on multiplicative neuron model artificial neuron network," *Applied Soft Computing*, vol. 104, Article ID 107179, 2021.

[29] E. Egrioglu and E. Bas, "A new automatic forecasting method based on a new input significancy test of a single multiplicative neuron model artificial neural network," *Network: Computation in Neural Systems*, pp. 1–16, 2022.

[30] S. Gao, M. Zhou, Y. Wang, J. Cheng, H. Yachi, and J. Wang, "Dendritic neuron model with effective learning algorithms for classification, approximation, and prediction," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 2, pp. 601–614, 2018.

[31] Z. H. Zhang, F. Min, G. S. Chen, S. P. Shen, Z. C. Wen, and X. B. Zhou, "Tri-partition state alphabet-based sequential pattern for multivariate time series," *Cognitive Computation*, pp. 1–19, 2021.

[32] X. Ran, X. Zhou, M. Lei, W. Tepsan, and W. Deng, "A novel k-means clustering algorithm with a noise algorithm for capturing urban hotspots," *Applied Sciences*, vol. 11, no. 23, p. 11202, 2021.

[33] H. Cui, Y. Guan, H. Chen, and W. Deng, "A novel advancing signal processing method based on coupled multi-stable stochastic resonance for fault detection," *Applied Sciences*, vol. 11, no. 12, p. 5385, 2021.

[34] W. Deng, X. Zhang, X. Zhou et al., "An enhanced fast non-dominated solution sorting genetic algorithm for multi-objective problems," *Information Sciences*, vol. 585, pp. 441–453, 2022.

[35] E. Q. Wu, M. Zhou, D. Hu et al., "Self-paced dynamic infinite mixture model for fatigue evaluation of pilots' brains," *IEEE Transactions on Cybernetics*, pp. 1–16, 2020, https://ieeexplore.ieee.org/document/9285173.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, NW, Washington, DC; United States, 2015.

[37] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, Sardinia, Italy, May 2010.

[38] M. Schmitt, "On the complexity of computing and learning with multiplicative neural networks," *Neural Computation*, vol. 14, no. 2, pp. 241–301, 2002.