International Journal of Systems Engineering
2023; 7(1): 1-8
http://www.sciencepublishinggroup.com/j/ijse
doi: 10.11648/j.ijse.20230701.11
ISSN: 2640-4222 (Print); ISSN: 2640-4230 (Online)

Time-Reduced Model for Multilayer Spiking Neural Networks

Yanjing Li

Institute of Education Science Research, Heilongjiang University, Harbin, China

Email address:
yanjing_li01@163.com

To cite this article:
Yanjing Li. Time-Reduced Model for Multilayer Spiking Neural Networks. *International Journal of Systems Engineering*. Vol. 7, No. 1, 2023, pp. 1-8. doi: 10.11648/j.ijse.20230701.11

Received: January 15, 2023; **Accepted**: February 3, 2023; **Published**: February 16, 2023

Abstract: Spiking neural networks (SNNs) is a type of biological neural network model, which is more biologically plausible and computationally powerful than traditional artificial neural networks (ANNs). SNNs can achieve the same goals as ANNs, and can build a large-scale network structure (i.e. deep spiking neural network) to accomplish complex tasks. However, training deep spiking neural network is difficult due to the non-differentiable nature of spike events, and it requires much computation time during the training period. In this paper, a time-reduced model adopting two methods is presented for reducing the computation time of a deep spiking neural network (i.e. approximating the spike response function by the piecewise linear method, and choosing the suitable number of sub-synapses). The experimental results show that the methods of piecewise linear approximation and choosing the suitable number of sub-synapses is effective. This method can not only reduce the training time but also simplify the network structure. With the piecewise linear approximation method, the half of computation time of the original model can be reduced by at least. With the rule of choosing the number of sub-synapses, the computation time of less than one-tenth of the original model can be reduced for XOR and Iris tasks.

Keywords: Spiking Neural Network, Computation Time, Linear Approximation, Sub-Synapses

1. Introduction

Recently, multilayer neural networks have been successfully applied to many fields including pattern recognition [1, 2], time series forecasting [3], and bioinformatics [4], etc. The basic unit of conventional artificial neural networks (ANNs) is called a node (or non-spiking neuron). The node is merely represented by non-linear activation functions without biological interpretability, whereas a spiking neural network (SNN) [5] is composed of spiking neurons that can transmit and receive substantial amounts of information through the relative timing of spikes. The characteristics of biological neurons are the basis of spiking neuron models. Spiking neurons can simulate the full process of biological neurons from receiving stimuli to firing spikes. This property of an SNN is particularly suitable for applications where the timing of input signals carries important information. It has been shown that the SNN can be applied to problems that can be solved by non-spiking neural networks and more importantly, the SNN is more powerful than conventional neural networks [6].

As the complexity of data increases, the number of layers will increase. It causes the network to consume much time for training. Accordingly, deep neural networks (DNNs) [7] can be optimized to work effectively. For instance, a binarization scheme is used to train DNNs [8]. It consists in training the DNNs with binary weights during the forward and backward propagations while retaining the precision of the stored weights in which gradients are accumulated. A fixed-point factorized network (FFN), which was proposed in [9], is used for pre-train models to reduce the computational complexity as well as the storage requirement of networks. The resulting networks have only weights of -1, 0, and 1, which significantly eliminates the most resource-consuming multiply-accumulate operations [9]. The hash trick can also be used for compressing neural networks [10]. It exploits inherent redundancy in neural networks to achieve drastic reductions in model size. Different from these approaches, a key goal of our work is to reduce the time consumption of the multilayer SNN. A time-reduced model is employed for this paper.

The main contributions of this work are as follows: a) Based on the approximation of piecewise linear manner, this paper proposes a set of simple spike response models. It can get reduce much time during the training period. b) Based on the simple spike response model, an SNN with multiple sub-synapses can be obtained. Yet the number of sub-synapses is not explicit. To obtain the explicit sub-synapses, a method of choosing the suitable number of sub-synapses is proposed. The manners not only simplify the SNN structure but can get satisfactory results.

The rest of this paper is organized as follows. Section 2 discusses the spike neuron model and learning algorithm of multilayer SNN briefly. The time-reduce model is derived in Section 3, it contains a set of piecewise linear models and a rule for choosing the number of sub-synapses. Simulation experiment results are provided in Section 4, followed by the conclusion and the future work in Section 5.

2. Multilayer Spiking Neural Network

The continuously valued activations are used for non-spiking DNNs communication. On the contrary, the SNNs transmit information between neurons depending on sending a series of action potentials (i.e. spike trains). A one-layered SNN can solve a simple non-linear problem, however, it may not be able to represent all possible mappings [11]. Therefore, the multilayer SNN is worth researching. The spiking neuron model and multilayer SNN learning rule are discussed in this section.

2.1. SRM Neuron Model

The SRM neuron model [12] can give a good approximation of the synaptic response for the neuron. Therefore, it is used as the target neuron model for the investigation of the anti-noise learning rule in this work. Figure 1 shows the connection of SRM-based neurons via one synapse.

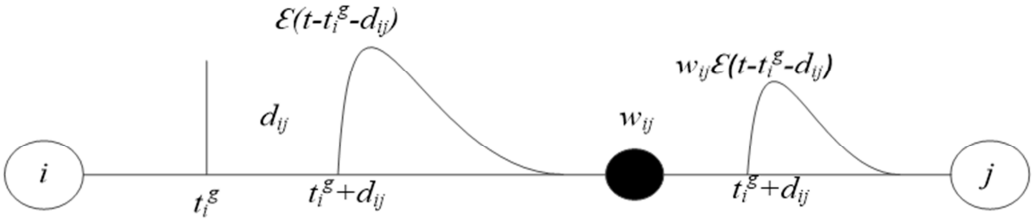

Figure 1. *The connection of SRM-based neurons via one synapse.*

The state of the post-synaptic neuron j is described by a single variable u_j in the framework of the SRM. The neuron is at its resting value when u_j equals to zero if there is no input spikes and it can be denoted by $u_{rest} = 0$. If u_j reaches the threshold after summing the effects of several incoming spikes, an output spike is triggered. After firing, the evolution of u_j is described by

$$u_j(t) = \eta\left(t - t_j^{(f)}\right) \times \sum_i \sum_{t_i^{(g)}} \sum_{k=1}^{l} w_{ji}^k \varepsilon(t - t_i^{(g)} - d^k) \quad (1)$$

where w_{ji}^k is the synapse weight from pre-synaptic neuron i to post-synaptic neuron j with a delay of d^k. The $t_j^{(f)}$ and $t_i^{(g)}$ denote the spike-times of neuron j and neuron i, respectively. If $1 \le g < f \le n$ (n denotes the total number of firing spikes), then $t_j^{(g)} < t_i^{(f)}$. The function ε describes the time course of the response to an incoming spike, which is given by

$$\varepsilon(s) = \frac{s}{\tau}\exp(1 - \frac{s}{\tau})H(s) \quad (2)$$

where τ is the post synaptic membrane potential time constant. $H(s)$ is the Heavy-side step function. H(s)=0 if s≤0 and H(s)=1 if s>0. The return of the membrane potential to baseline after an action potential is described by a function η, which is named refractoriness. The refractoriness is characterized experimentally by the observation immediately after a first action potential. It is impossible (absolute refractoriness) or more difficult (relative refractoriness) to excite a second spike. The function η is given by

$$\eta(s) = -\eta_0 exp(-\frac{s}{\tau_r})H(s) \quad (3)$$

where τ_r is a slow time constant and η_0 is a scale factor for the refractory function.

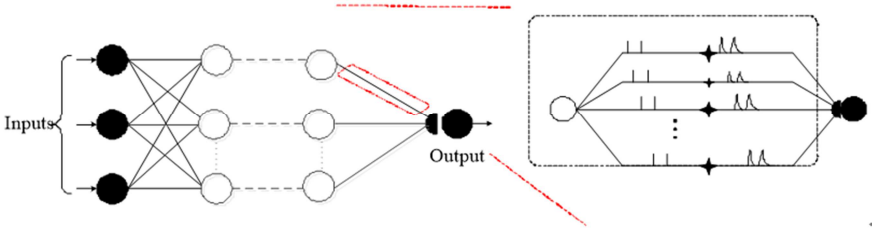

Figure 2. *The multilayer network architecture. The right part indicates that sub-synapses are included in each synaptic junction.*

The SNN is composed of neurons connecting with each other via synapses. Figure 2 shows a multilayer network architecture. It consists of a feedforward fully connected network of spiking neurons with multiple delayed synaptic

terminals. The different multiple delayed synaptic terminals are used in this work.

2.2. Neural Network Architecture and Algorithm

The SNNs rely on synapses to complete learning tasks. A synapse can be either excitatory or inhibitory. Excitatory synapse increases the membrane potential of the neuron upon receiving input spikes, whereas inhibitory synapse decreases the membrane potential of the neuron. The strength of a synapse is modeled by adjustable scalar weights [13]. It can change over time to express the synapse efficacy in SNNs. This plasticity of synapses represents a brain-inspired learning mechanism in neural networks, and the learning rule is aiming for adjusting the strength of interconnecting synapses.

A supervised learning algorithm, the tempotron, has been explored in [14], where neurons learn to distinguish between spatial and temporal sequences of spike patterns [15]. The gradient descent approach is used to train the tempotron. However, this learning rule can only be employed in one-layered networks. The leaky integrated-and-fire neuron model [12, 16] in the tempotron is trained to only solve binary classification. Because this learning rule discriminates the patterns by neurons firing spikes or not.

A backpropagation learning rule, named multi-spikeprop [17, 18], is used in our work. It is also on the basis of a gradient descent algorithm, and the network can converge gradually to the minimum error via many training epochs [13]. The error function is an important criterion to measure the error between desired and actual output spikes. It can be calculated by

$$E = \frac{1}{2}\sum_{j=1}^{N}\sum_{f=1}^{F_j}(t_j^{(f)} - \hat{t}_j^{(f)})^2 \tag{4}$$

where $\hat{t}_j^{(f)}$ is the desired time of output spikes. Once the error is not a desired value, the synaptic weights will be updated. The updating rule of weights is the basis of gradient descent, and it can be calculated by

$$\Delta w = -\alpha \nabla E = -\alpha \frac{\partial E}{\partial w}, \tag{5}$$

where α is the earning rate. The gradient computation of synapses between neurons in the last layer and neurons in the hidden layer should be calculated firstly. The weights updating for the kth synapse between the presynaptic neuron i and postsynaptic neuron j can be calculated by

$$\Delta w_{ij}^k = -\alpha \nabla E_{ij}^k. \tag{6}$$

According to the chain rule, ∇E_{ij}^k can be calculated by

$$\nabla E_{ij}^k = \sum_{f=1}^{F_j} \frac{\partial E}{\partial t_j^{(f)}} \frac{\partial t_j^{(f)}}{\partial w_{ij}^k}. \tag{7}$$

The weights updating between neurons in the input (or hidden) layer and neurons in the hidden layer can be computed by error backpropagation. It can be described as

$$\Delta w_{hi}^k = -\alpha \nabla E_{hi}^k. \tag{8}$$

According to the chain rule, ∇E_{hi}^k can be calculated by

$$\nabla E_{hi}^k = \frac{\partial E}{\partial w_{hi}^k} = \sum_{t_i^{(f)} \in \Gamma_i} \frac{\partial E}{\partial t_j^{(f)}} \frac{\partial t_j^{(f)}}{\partial w_{hi}^k}. \tag{9}$$

The detailed learning rule is researched in [17, 19]. Although the multi-spikeprop learning rule can effectively handle classification tasks, it consumes much time to training the SNN.

3. The Time-Reduced Model

To solve the problem of time-consuming in the multilayered SNN, a set of piecewise linear spiking neural models and a rule for choosing the number of sub-synapses are proposed in this section. There is usually a trade-off between model accuracy and computational complexity for the piecewise linear spiking neuron model. For example, if the neural behavior is required to understand, the Hodgkin-Huxley model [20] will be more suitable. However, it is computationally expensive and cannot be simulated in large numbers. Therefore it is essential to simplify the SNN model.

3.1. SRM Linear Approximation

The original spike response function ε(s) is an exponential function, and the computation cost can be decreased by approximation. To reduce the computation time for multilayer SNN, a piecewise linear model is used to approximate the spike response function.

Five piecewise linear models are used to approximate the spike response function in our work, i.e. two segments, three segments, four segments, and five segments and six segments respectively. The two-segment piecewise linear model is shown in Figure 3. The blue line is the curve of the original spike response function, and the black line is the approximated curve. It can be seen that this segment is not precise enough.

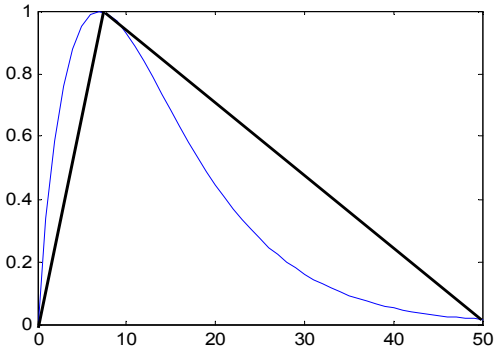

Figure 3. *The two-segment piecewise linear model.*

The detailed function of the two-segment model can be calculated by

$$\varepsilon(s) = \begin{cases} a_1 s + b_1, 0 < s \le 8 \\ a_2 s + b_2, 8 < s \le 50 \end{cases} \tag{10}$$

The parameters of a_1, a_2, b_1 and b_2 are coefficients, and

$a_1 = 0.1367$, $a_2 = -0.0232$, $b_1 = 0.2086$, $b_2 = 0.9833$. The three-segment piecewise linear model is shown in Figure 4. The blue line refers to the original curve of spike response function, and the black line refers to the curve of the approximated function.

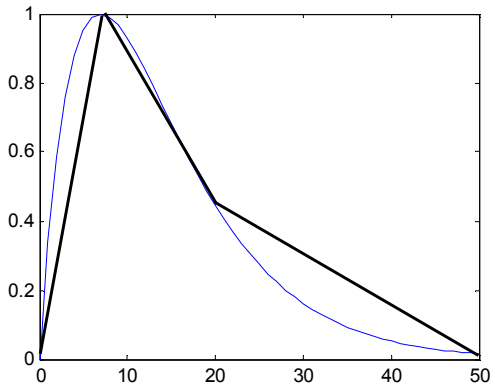

Figure 4. *The three-segment piecewise linear model.*

The detailed function of the three-segment model can be calculated by

$$\varepsilon(s) = \begin{cases} a_3 s + b_3, 0 < s \leq 8 \\ a_4 s + b_4, 8 < s \leq 20 \\ a_5 s + b_5, 20 < s \leq 50 \end{cases} \quad (11)$$

The values of a_3, a_4, a_5, b_3, b_4 and b_5 are respectively 0.1367, -0.0477, -0.013, 0.2086, 1.399 and 0.5939. Figure 5 shows the four-segment piecewise linear model. The blue line shows the curve of the original spike response function, and the black line shows the curve of the approximated function.

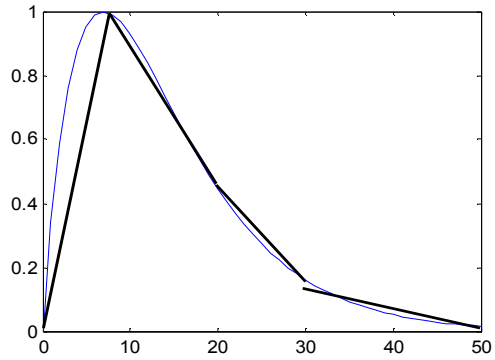

Figure 5. *The four-segment piecewise linear model.*

The detailed function of the four-segment model can be obtained by

$$\varepsilon(s) = \begin{cases} a_6 s + b_6, 0 < s \leq 8 \\ a_7 s + b_7, 8 < s \leq 20 \\ a_8 s + b_8, 20 < s \leq 30 \\ a_9 s + b_9, 30 < s \leq 50 \end{cases} \quad (12)$$

The values of a_6, a_7, a_8, a_9, b_6, b_7, b_8 and b_9 are respectively 0.1367, -0.0477, -0.0296, -0.0068, 0.2086, 1.399, 1.0226 and 0.3376. Figure 6 shows the five-segment piecewise linear model. The blue line shows the curve of original spike response function, and the black line shows the curve of approximated function.

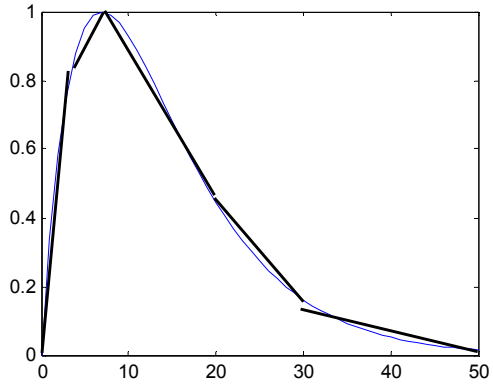

Figure 6. *The five-segment piecewise linear model.*

The detailed function of the five-segment model can be obtained by

$$\varepsilon(s) = \begin{cases} a_{10} s + b_{10}, 0 < s \leq 5 \\ a_{11} s + b_{11}, 5 < s \leq 8 \\ a_{12} s + b_{12}, 8 < s \leq 20 \\ a_{13} s + b_{13}, 20 < s \leq 30 \\ a_{14} s + b_{14}, 30 < s \leq 50 \end{cases} \quad (13)$$

The values of a_{10}, a_{11}, a_{12}, a_{13}, a_{14}, b_{10}, b_{11}, b_{12}, b_{13} and b_{14} are respectively 0.2918, 0.0594, -0.0477, -0.0296, -0.0068, 0.0149, 0.6182, 1.399, 1.0226, and 0.3376. The six-segment piecewise linear model is shown in Figure 7. The blue line shows the original curve of spike response function, and the black line shows the curve of approximated function.

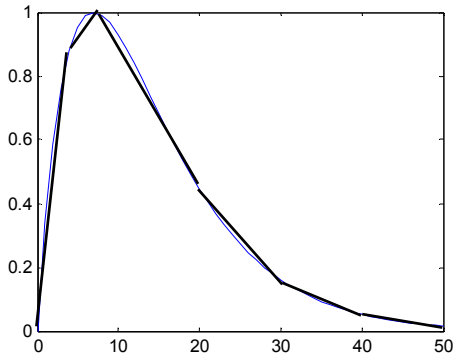

Figure 7. *The six-segment piecewise linear model.*

The detailed function of the six-segment model can be obtained by

$$\varepsilon(s) = \begin{cases} a_{15} s + b_{15}, 0 < s \leq 5 \\ a_{16} s + b_{16}, 5 < s \leq 8 \\ a_{17} s + b_{17}, 8 < s \leq 20 \\ a_{18} s + b_{18}, 20 < s \leq 30 \\ a_{19} s + b_{19}, 30 < s \leq 40 \\ a_{20} s + b_{20}, 40 < s \leq 50 \end{cases} \quad (14)$$

The values ofa_{15}, a_{16}, a_{17}, a_{18}, a_{19}, a_{20}, b_{15}, b_{16}, b_{17}, b_{18}, b_{19} and b_{20} are respectively 0.2918, 0.0594, -0.0477,

-0.0296, -0.0113, -0.0035, 0.0149, 0.6182, 1.399, 1.0226, 0.4920, and 0.1893. The calculation time and approximated error between original and approximated function are shown in Table 1. It can be seen that the more the number of segments, the smaller the error, and the less the number of segments, the less time consuming.

Table 1. *The Time consumption of different piecewise linear models.*

Different functions	Approximated error	Time-consuming
Original	0	0.003735
Two segment	0.003	0.000817
Three segment	0.0036	0.00088
Four segment	0.0025	0.000900
Five segment	2.1633e-4	0.001637
Six segment	1.6045e-4	0.001702

3.2. Method of Choosing Sub-Synapses

In conventional neural networks, there is usually one synapse between two nodes. It contains multiple synapses between presynaptic neurons and postsynaptic neurons (i.e. sub-synapses) in SNNs. A suitable number of sub-synapses is crucial to the performance of the SNNs. For instance, too many sub-synapses will cause the SNN computationally expensive and much time-consuming. On the contrary, the network with fewer sub-synapses cannot work effectively. Therefore, a method of choosing a suitable number of sub-synapses is an important task.

The number of sub-synapses is usually set to a big value by experience [11, 19, 21]. However, it is not suitable for all tasks. Thus, a rule of choosing the number of sub-synapses is employed for our work. It can be calculated by

$$n = \begin{cases} n_0 + 1, \frac{E}{n} > E_d \\ n_0, \frac{E}{n} \leq E_d \end{cases} \tag{15}$$

where n is the current number of sub-synapses, n_0 is the final number of sub-synapses, and E_d is the minimum acceptable error of multilayered SNN. It shows that the number of synapses increases gradually from n_0. If the value of $\frac{E}{n}$ more than the given threshold (E_d), continue to increase until it meets the conditions.

4. Experimental Results

The exclusive-or (XOR) [11] and Iris [15] datasets are employed to test the performance of the time-reduced model in this work. The temporal encoding scheme is chosen for encoding the data. Particularly, the iris data is encoded to integral spike firing time using a direct mapping method.

4.1. Experimental Settings

A fully connected feedforward multilayered SNN with multiple delays per connection is used in our work. The time range and time step of the network are respectively set to 50 ms and 0.1 ms. The learning rate is set to 1. The mean squared error (E) is set to 0.1 as a stop criterion, and the maximum epoch is 1000. If iterations of the network are more than 1000 and E>0.1, the learning process will be failed.

All the experimental simulations are done under the MATLAB 2018a platform on a Windows 10 operating system with Intel (R) Core (TM) i5-9400 CPU 2.90GHz processor with 16 GB of random-access memory.

4.2. XOR Task

Table 2. *The encoding scheme of the XOR task.*

The input data (ms)		The output data (ms)
0	0	16
0	6	10
6	0	10
6	6	16

Table 2 shows the temporal encoding scheme of the XOR task. For the input signals, an input spike at 0 ms represents logic 0 while a spike at 6 ms represents logic 1. For the output, a spike at 16 ms represents logic 0 while a spike at 10 ms represents logic 1. The time constant of membrane potential should be slightly larger than the interval of encoding. Therefore the initial value of τ is set to 7 ms. The network consists of three neurons (two coding neurons and one bias neuron) in the input layer, and a single neuron in the output layer.

Table 3. *The performance of different spike response models for the XOR task.*

Different models	Average epochs	Time-consuming (s)	Successful trials (%)
2	160	3.54	34%
3	145	6.32	61%
4	182	6.94	80%
5	165	8.54	96%
6	193	10.31	96%
Original	179	22.64	100%

Table 3 shows the performance of multilayer SNN using the model of piecewise linear approximation. It can be seen that the original multilayer SNN model runs 22.64s and achieves a hundred percent successful convergence rate. The two-segment piecewise linear model runs with the least time (3.54s), however, it only achieves a thirty-four percent rate of successful convergence. The five-segment and six-segment piecewise linear models achieve the same rate of successful convergence (96%), and the five-segment piecewise linear model consumes less time than the six-segment piecewise linear model. Therefore, the five-segment piecewise linear model is suitable for the XOR task.

Table 4. *The performance of multilayered SNN with the different number of sub-synapses for the XOR task.*

The number of sub-synapses	Average epochs	Time-consuming (s)	Successful trials (%)
2	N	N	Cannot converge
3	N	N	Cannot converge
4	N	N	Cannot converge
5	N	N	Cannot converge
6	30	3.60	100%
7	68	5.99	100%

The number of sub-synapses	Average epochs	Time-consuming (s)	Successful trials (%)
8	29	2.87	100%
9	31	3.01	100%
10	69	6.77	100%
11	49	5.09	100%
12	33	3.57	100%
13	34	4.03	100%
14	17	2.55	100%
15	91	11.14	100%
16	179	22.64	100%

Table 4 shows the performance of multilayer SNN with the different number of sub-synapses. It can be seen that the network with 2, 3, 4 and 5 sub-synapses cannot converge to a target value. The network with 14 sub-synapses carries out 17 epochs and consumes 2.55s to converge to the target value. Therefore, the multilayer SNN with 14 sub-synapses is the best choice for the XOR task.

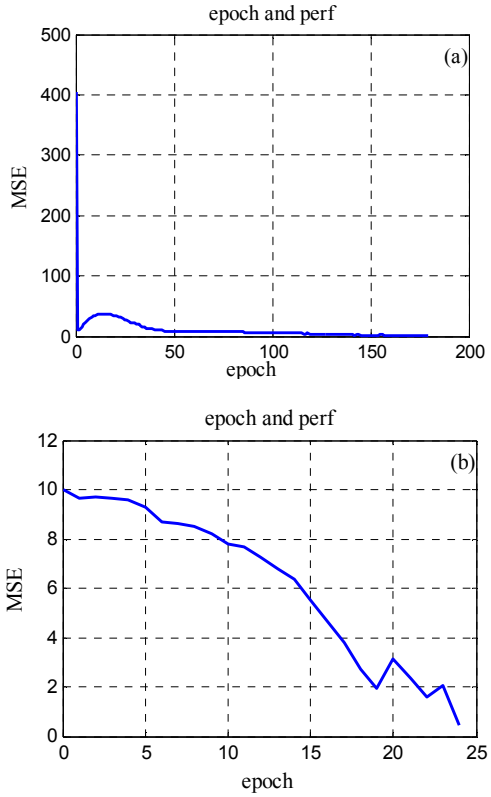

Figure 8. *The relationships between epochs and mean squared error under different spike response functions. (a). The performance of the original multilayer SNN. (b). The performance of multilayer SNN with the time-reduce model.*

The relationships between epochs and mean squared error under different spike response functions are shown in Figure 8. (a) refers to the original multilayer SNN, it carries 179 epochs and consumes 22.64s. (b) refers to the multilayer SNN using the five-segment piecewise linear model and the rule of choosing the number of sub-synapses, it carries out 24 epochs and consumes 2.47s to converge to the target value. The experimental result of the XOR task can prove that the method is effectively proposed in this paper.

4.3. Iris Classification Task

The iris dataset is used for this testing. It contains three classes (i.e. Iris Setosa, Iris versicolor, Iris Virginia) of 150 samples, where each class corresponds to a type of iris plant. There are four attributes in this dataset. Each attribute corresponds to an input neuron. Thus, the network has four input neurons in the first layer and one output neuron in the last layer. Due to the value of each attribute being a decimal value, it is encoded to integral spike firing time first. The encoding interval is set to 1, and the time constant of membrane potential is set to 3 ms. 120 samples are used to train the multilayer SNN, and the rest samples are used to test the performance.

Table 5. *The performance of different spike response models for the Iris task.*

Different models	Average epochs	Time-consuming (s)	Average accuracy (%)
2	13	224.82	63.33%
3	39	661.21	73.33%
4	75	1268.70	93.33%
5	6	106.61	93.33%
6	10	175.09	96.67%
Original	46	794.78	96.67%

Table 5 shows the performance of multilayer SNN using the model of piecewise linear approximation. It can be seen that the original multilayer SNN model runs 794.78s and achieves a 96.67 percent successful convergence rate. The five-segment piecewise linear model runs with the least time (106.61s), and it achieves a higher rate of successful convergence (93.33%). The five-segment piecewise linear model consumes less time than the six-segment piecewise linear model, however, the six-segment piecewise linear model can achieve the same successful convergence rate as the original model. Therefore, considering the time consumption, the five-segment piecewise linear model is more suitable for the Iris task.

Table 6. *The performance of multilayered SNN with the different number of sub-synapses for the Iris task.*

The number of sub-synapses	Average epochs	Time-consuming (s)	Average accuracy (%)
2	N	N	Cannot converge
3	N	N	Cannot converge
4	N	N	Cannot converge
5	19	69.95	56.67%
6	32	460.67	63.33%
7	35	439.09	56.67%
8	28	150.80	70%
9	26	158.76	66.67%
10	57	191.91	70%
11	20	72.27	73.33%
12	17	67.93	60%
13	16	63.27	73.33%
14	9	38.34	93.33%
15	21	85.08	73.33%
16	46	794.78	96.67%

Table 6 shows the performance of multilayer SNN with the different number of sub-synapses. It can be seen that the network with 2, 3, and 4 sub-synapses cannot converge to a target value. The network with 14 sub-synapses carries out 9

epochs and consumes 38.34s to converge to the target value. Therefore, the multilayer with 14 sub-synapses is the best choice for the Iris task.

Table 7. *Comparison of SNN training results for Iris task.*

Algorithms	Time-consuming (s)	Average accuracy (%)
SpikeProp	794.78	96.1%
MatlabBP	912.42	95.5%
MatlabLM	1022	95.7%
Weight limit learning	260.62	92.6%
SWAT	1243.14	95.3%
This work	175.09	96.67%

Table 7 shows the performance of this work with other SNN algorithms. It can be seen that our work can get an effective accuracy on Iris dataset with less time consumption.

Figure 9 depicts the curves of learning performance in the interval of epochs and mean squared error under different spike response functions. (a) depicts the original multilayer SNN, it carries 46 epochs and consumes 794.78s. (b) depicts the multilayer SNN using the five-segment piecewise linear model and the rule of choosing the number of sub-synapses, it carries out 14 epochs and consumes 33.56s to converge to the target value. The experimental result of the Iris task is further explained that the method is effectively proposed in this paper.

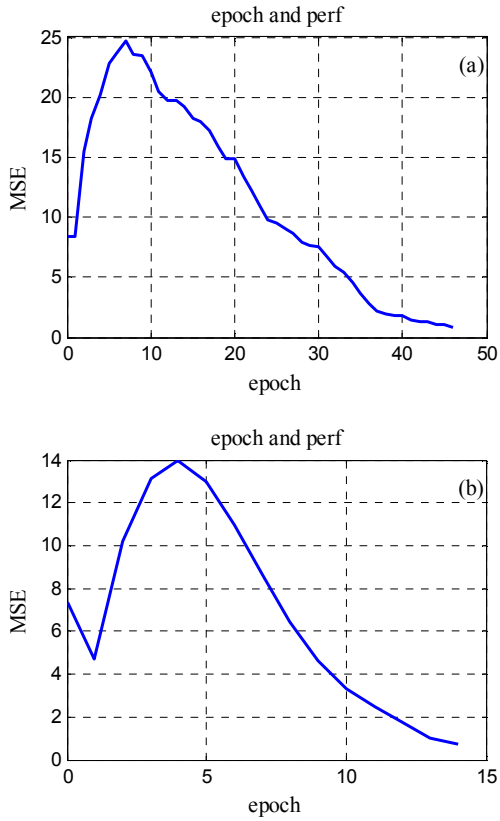

Figure 9. *The relationships between epochs and mean squared error under different spike response functions. (a). The performance of the original multilayer SNN. (b). The performance of multilayer SNN with the time-reduce model.*

5. Conclusion

In this paper, methods for reducing the time consumption of multilayer SNN were proposed. The methods include a piecewise linear approximation of the spike response function and a rule of choosing the suitable number of sub-synapses. The experiment of the XOR task and Iris dataset classification task were used to verify the performance of the proposed time-reduced model. The experimental results showed that the proposed methods can reduce time consumption and have a better performance. Future work includes the theoretical proof of the proposed methods and further exploration of spiking neuron models and deep SNNs.

References

[1] J. Wu, Y. Chua, and H. Li, "A biologically plausible speech recognition framework based on spiking neural networks," in 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–8.

[2] J. Liu and G. Zhao, "A bio-inspired SOSNN model for object recognition," in 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–8.

[3] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," IEEE Signal Process. Mag., vol. 29, no. 6, pp. 82–97, 2012.

[4] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification.," IEEE Access, vol. 6, pp. 1662–1669, 2018.

[5] S. Min, B. Lee, and S. Yoon, "Deep learning in bioinformatics," Brief. Bioinform., vol. 18, no. 5, pp. 851–869, 2017.

[6] W. Maass, "Networks of spiking neurons: The third generation of neural network models," Neural Networks, vol. 10, no. 9, pp. 1659–1671, 1997.

[7] J. P. Dominguez-morales, Q. Liu, R. James, D. Gutierrez-Galan, A. Jimenez-Fernandez, S. Davidson, and S. Furber, "Deep spiking neural network model for time-variant signals classification : a real-time speech recognition approach," in 2018 International Joint Conference on Neural Networks (IJCNN), 2018, pp. 1–8.

[8] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[9] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," arXiv Prepr. arXiv1511.00363, pp. 1–9, 2016.

[10] P. Wang and J. Cheng, "Fixed-point factorized networks," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 3966–3974.

[11] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing neural networks with the hashing trick," in International Conference on International Conference on Machine Learning, 2015, vol. 37, pp. 2285–2294.

[12] O. Booij and H. Tat Nguyen, "A gradient descent rule for spiking neurons emitting multiple spikes," Inf. Process. Lett., vol. 95, no. 6, pp. 552–558, 2005.

[13] Y. C. Yoon, "LIF and simplified SRM neurons encode signals into spikes via a form of asynchronous pulse sigma-delta modulation," IEEE Trans. Neural Networks Learn. Syst., vol. 28, no. 5, pp. 1192–1205, 2017.

[14] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep learning in spiking neural networks," arXiv Prepr. arXiv1804.08150, pp. 1–18, 2018.

[15] M. Zhang, J. Li, Y. Wang, and G. Gao, "R-tempotron: A robust tempotron learning rule for spike timing-based decisions," in International Computer Conference on Wavelet Active Media Technology and Information Processing, 2016, pp. 139–142.

[16] I. Sporea and A. Gruning, "Supervised learning in multilayer spiking neural networks," Neural Comput., vol. 25, no. 2, pp. 473–509, 2013.

[17] N. Soltani and A. J. Goldsmith, "Directed information between connected leaky integrate-and-fire neurons," IEEE Trans. Inf. Theory, vol. 63, no. 9, pp. 5954–5967, 2017.

[18] Y. Xu, X. Zeng, L. Han, and J. Yang, "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks," Neural Networks, vol. 43, no. 4, pp. 99–113, 2013.

[19] Q. Yu, H. Tang, K. C. Tan, and H. Yu, "A brain-inspired spiking neural network model with temporal encoding and learning," Neurocomputing, vol. 138, no. 11, pp. 3–13, 2014.

[20] Y. Luo, Q. Fu, J. Liu, J. Harkin, L. McDaid, and Y. Cao, "An extended algorithm using an adaptation of momentum and learning rate for spiking neurons emitting multiple spikes," Int. Work. Artif. Neural Networks, pp. 569–579, 2017.

[21] Q. Kang, B. Huang, and M. Zhou, "Dynamic behavior of artificial Hodgkin-Huxley neuron model subject to additive noise," IEEE Trans. Cybern., vol. 46, no. 9, pp. 2083–2093, 2016.