


Research Article

Intralayer-Connected Spiking Neural Network with Hybrid Training Using Backpropagation and Probabilistic Spike-Timing Dependent Plasticity

Long Chen,¹ Xuhang Li,² Yaqin Zhu,² Haitao Wang,² Jiayong Li,² Yu Liu,² and Zijian Wang ²

¹Safety Technology R&D Department, China Telecom Research Institute, Shanghai 200000, China

²School of Computer Science and Technology, Donghua University, Shanghai 200000, China

Correspondence should be addressed to Zijian Wang; wang.zijian@dhu.edu.cn

Received 11 November 2022; Revised 21 May 2023; Accepted 4 July 2023; Published 25 July 2023

Academic Editor: Alexander Hošovský

Copyright © 2023 Long Chen et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Spiking neural networks (SNNs) are highly computationally efficient artificial intelligence methods due to their advantages in having a biologically plausible computational framework. Recent research has shown that SNN trained using backpropagation (SNN-BP) exhibits excellent performance and has shown great potential in tasks such as image classification and security detection. However, the backpropagation method limits the dynamics and biological plausibility of the neural models in SNN, which will limit the recognition and simulation performance of SNN. In order to make neural models more similar to biological neurons, this study proposes a leaky integrate-and-fire (LIF) neuron model with dense intralayer connections, as well as efficient forward and backward processes in BP training. The new model will make the interaction between neurons within the layer more frequent, enhancing the intrinsic information exchange capability of SNN. An effective probabilistic spike-timing dependent plasticity (STDP) method is also proposed to reduce the overweighted connections between neurons, as well as a hybrid training method using BP and probabilistic STDP. The training method combines the advantages of BP and STDP to improve the performance of SNN models. An intralayer-connected SNN with hybrid training (ISNN-HY) is proposed with the combination of these improvements. The proposed model was evaluated on three static image datasets and one neuromorphic dataset. The results showed that the performance of ISNN-HY is superior to that of other SNN-BP models. The proposed method also makes it possible to accurately simulate biological neural systems.

1. Introduction

Spiking neural networks (SNNs) are a framework inspired by the brain, designed for artificial intelligence and neural system simulation. They have gained significant interest [1]. Compared to the artificial neural network (ANN) framework [2], SNNs can more efficiently process complex signals using biologically plausible spatiotemporal neuron models and learning methods.

The significant advancement of the SNN algorithm lies in its training methods. Currently, there are three main training methods used for SNNs: spike-timing dependent plasticity (STDP) [3], the conversion method [4], and the

direct training method using error backpropagation (SNN-BP) [5–7]. The STDP learning rule is an unsupervised method discovered in neurophysiological experiments to regulate the connection strength of biological neurons. It can dynamically adjust the local neuronal connection pattern based on the temporal relationship of neuronal firing [8, 9]. However, the STDP rule, being an unsupervised learning method, is difficult to propagate errors to deeper layers. The performance of SNNs based on STDP often fails to meet expectations in practical applications [10].

The conversion method converted a pretrained ANN model to an SNN model. The converted SNN model has the same structure as the ANN model with transformed

parameters. Therefore, the converted SNN lacks biological rationality compared to directly trained SNNs [7]. SNN-BP uses the backpropagation (BP) method to train the connection weights between neurons to reduce the difference between predictions and true labels. However, due to the nondifference of spike events, the BP algorithm could not be transferred to SNNs directly.

Some SNN-BP methods have been proposed and successfully used in various tasks [5–7] and achieved good performance on some application or cognitive tasks, such as STBP [11], ST-RSBP [12], TSSL-BP [6], and ASF-BP [7]. Although SNN-BP has been developed by leaps and bounds in recent years, these improvements seem to focus too much on the derivation of gradients and ignore the simulation of biological neurons by spiking neurons [10, 13]. Nowadays, more research focuses on how to make spiking neural networks trained with the BP algorithm have recurrent connections like biological neurons. Some methods have been proposed to achieve spiking neural network models with recurrent connections, such as improved spiking neural networks with lateral interactions (LISNN) [14], laterally inhibited self-recurrent unit (LISR) [15], skip-connected self-recurrent SNN (ScSr-SNN) [16], and sparsely connected recurrent motif layer (SC-ML) [17]. These methods achieve recurrent connections within a layer or within a single neuron through fixed-weight intralayer connections, single-neuron self-recurrent connections, and sparsely connected intralayer neural connections. However, there is still a lack of efficient spiking neural network models that can achieve dense intralayer neural connections, which may enhance effective information interaction within a layer and improve model training efficacy.

Now, some research is focusing on innovative model training methods. Some studies have proposed probability STDP rule methods for training SNNs, such as the fusion of Bayesian probability inference and STDP [18] or the conversion of STDP rule synaptic prepotential differences into probability distributions [19]. Although these methods attempt to improve the STDP method and have better performance than models trained with the original STDP method, they still have a large gap compared to SNN models trained with supervised learning methods. In addition, some studies have proposed training algorithms based on the combination of STDP and BP, for example, BP-STDP, SSTDP, and an STDP-based supervised learning rule [20–23]. These methods approximate the backpropagation method into STDP style or convert the weight increase and decrease in backpropagation into corresponding STDP rules to achieve the training algorithm combining the advantages of both. However, these methods mainly realize the fusion of the two by converting BP rules into STDP rules or vice versa, resulting in a lower degree of utilization of the supervised BP algorithm, which leads to poor training effects and performance of SNN models.

Therefore, two key issues remain to be addressed in the development of SNN-BP: (1) SNN-BP models trained using BP can only simulate simple interlayer recurrent connections between neurons, which is far less flexible than biological neurons. This limits the application range of SNN. (2) The error BP algorithm would adjust all the connection weights in each training, easily leading to overfitting. Moreover, whether error propagation was the way of biological brain learning is still controversial and lacks biological rationality [24, 25].

In this paper, we proposed a leaky integrate-and-fire (LIF) neuron model [26] with optimized intralayer connections, which could be trained by the BP method. A probabilistic STDP method based on firing probability was also proposed to efficiently modulate local connection weights in between neurons complying with the STDP rules, which could reduce the overfitting of the model. Combining the novel neuron model and learning rule, we proposed an intralayer-connected spiking neural network with hybrid training using BP and probabilistic STDP (ISNN-HY). We trained the SNN model using the TSSL-BP algorithm [6]. We evaluated the SNN model with other SNN-BPs with the same structures in three static image datasets and one neuromorphic dataset. Experiment results demonstrate that the ISNN-HY achieved excellent performances on these datasets. Through time and neural activity analysis, we also proved that the proposed method has high efficiency and more flexible simulation ability and has the potential to apply accurate SNN-BP in the simulation, modeling, and analysis of biological neural activity.

The important contributions in this paper are summarized as follows:

- (i) We proposed an LIF neuron model with optimized intralayer connections, which enable the neurons in one layer to connect with each other in SNN-BP. The proposed neuron model could form a multi-layer reservoir computing model, which could be trained using BP.
- (ii) We proposed a probabilistic STDP method that can effectively adjust and reduce the local connection weights between neurons and reduce overfitting.
- (iii) Based on these methods, we proposed an ISNN-HY model. This model outperformed other SNN-BP models on both nonneuromorphic and neuromorphic datasets and has the potential to simulate biological neural activity.

2. Related Works

2.1. STDP and BP Training for SNNs. STDP is guided by the temporal correlation of presynaptic and postsynaptic neuronal spikes. Like other forms of synaptic plasticity, it is widely believed to underlie learning and information storage

in the brain and the development and refinement of neural circuits during brain development [27].

The classical unsupervised STDP rule is shown in Figure 1, calculated as

$$\Delta w = \sum_{t_{\text{pre}}} \sum_{t_{\text{post}}} f_w(t_{\text{post}} - t_{\text{pre}}),$$

$$f_w(\Delta t) = \begin{cases} A_{\text{pre}} e^{-\Delta t/\tau_{\text{pre}}} & \Delta t > 0, \\ A_{\text{post}} e^{\Delta t/\tau_{\text{post}}} & \Delta t < 0. \end{cases} \quad (1)$$

Δw is the change of connection weight over all pre-synaptic spike times t_{pre} and postsynaptic spike times t_{post} . $f_w(\Delta t)$ is the function calculating the change of weights. A_{pre} , A_{post} , τ_{pre} , and τ_{post} are the parameters that regulate the strength of the weight change, as shown in Figure 1(b).

Several studies have attempted to improve traditional STDP methods, with a focus on methods that combine STDP algorithms with probability. For instance, Tavanaei and Maida introduced probability to adjust the STDP rule. This means that each synapse has a probability value to determine whether the STDP rule should be used to adjust the weight [18]. If the probability value is less than a certain threshold, the STDP rule will not be applied to adjust the weight. In addition, Tavanaei and Maida utilized an STDP rule based on the difference in presynaptic potential and introduced a probability factor to control the update of synapse weight [19]. This probability factor is achieved by converting the difference in presynaptic potential into a probability distribution. Although this probabilistic STDP unsupervised learning methods performed better than the model trained by the original STDP algorithm, there is still a significant gap compared to the model trained by the supervised algorithm.

Some recent research has explored combining supervised BP and STDP algorithms. For example, Zhang et al. proposed the BP-STDP algorithm, which trains a multilayer fully connected SNN using an integrate-and-fire (IF) neuron model. This algorithm combines efficient backpropagation with biologically plausible timing-dependent plasticity training rules, providing hardware-friendly local training rules [20]. This method has also been applied to various SNN models with different structures [22]. Tavanaei et al. also proposed a BP and STDP combined training method, which uses STDP and anti-STDP rules to transform back-propagation weight change rules and finally applies STDP to train neuron connection weights at each time point [21]. This achieved a classification accuracy of 97.2% on the MNIST dataset. Liu et al. proposed the supervised spike-timing dependent plasticity (SSTDP) algorithm, which uses time-based encoding and IF neurons as neuron models and combines BP and STDP training methods to provide both global optimization and efficient weight updates [23].

However, these methods mostly rely on the conversion between STDP and BP algorithms, with limited use of supervised algorithms. They are still difficult to achieve the performance of SNN models trained based on BP algorithms

in experiments. Therefore, this work proposes a probabilistic STDP method, which converts firing time differences into probabilities of weight adjustment and combines it with the supervised BP algorithm to achieve rough weight learning through the BP algorithm. The weights are then fine-tuned based on the probabilistic STDP algorithm, resulting in an SNN model with better performance.

2.2. SNN with Recurrent Connections. SNN models with recurrent connections were demonstrated to have good performance and larger application scope than SNNs only with forward connections, such as RSNN [28], ST-RSBP [12], RDS-BP [13], LISNN [14], LISR [15], ScSr-SNN [16], SC-ML [17], and SCRNN [29]. For example, in the LISNN model [14], lateral interaction is achieved through a trainable interaction kernel function, which is used to calculate the interaction weights between neighboring neurons. Specifically, in the calculation of neuron membrane potential, lateral interaction is achieved by multiplying the axonal current of each neuron's adjacent neurons by the interaction kernel function. Zhang and Peng [15] proposed a new neuron structure called laterally inhibited self-recurrent units. This structure consists of an excitatory neuron and an inhibitory neuron, where the excitatory neuron has a self-feedback connection with fixed weights, and the inhibitory neuron provides lateral inhibition to the corresponding excitatory neuron through excitatory and inhibitory synapses. This structure can maintain long-term memory of neurons and reduce information loss caused by excitation and reset mechanisms. The inhibitory neuron regulates the firing activity of the excitatory neuron and serves as a forget gate for memory clearance. Zhang and Li also proposed the ScSr-SNN model [16], which introduces cyclicity by adding recurrent connections to spiking neurons themselves, and the recurrent connections also realize local storage and skip connections, which can achieve cyclic behavior similar to more complex RSNNs; the error gradient can be more directly calculated since most of the network is feedforward. A proposed SCRNN achieves recurrence by integrating convolutional operations and recurrent connections to maintain spatiotemporal relationships [29]. However, these methods are limited by the proposed recurrent connections, which are only implemented as fixed-weight intralayer neuron recurrent connections, single-neuron self-recurrent connections, and sparse interconnectivity between neurons.

In this study, an LIF neuron model with intralayer recursive connections is proposed, which can be effectively trained using the BP algorithm. In addition, a method is proposed to decompose the complex higher order intralayer connection weight matrix into two simple lower-order weight matrices, enabling neurons in the same layer to achieve fully adaptive dense connections at a low cost and increasing the frequency of interaction between neurons in the same layer. Information interaction is enhanced, and the performance of the model is improved. A detailed

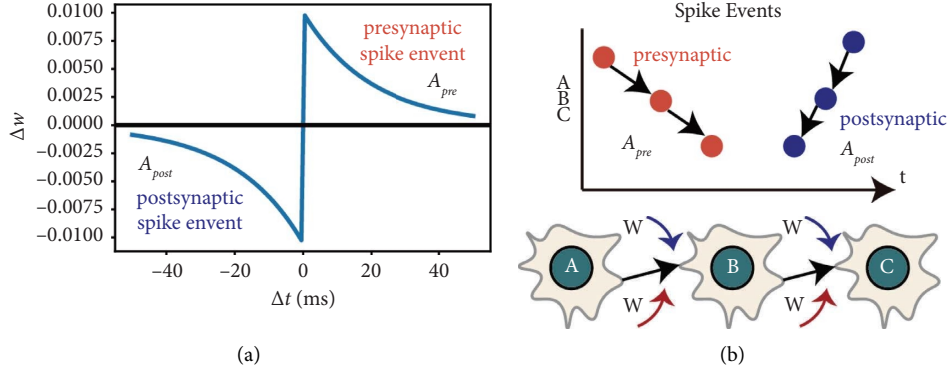


FIGURE 1: The STDP learning rule. (a) The relation between the modulated weight Δw and spike time interval Δt . (b) The presynaptic (orange) and postsynaptic (blue) spike events. The postsynaptic spike events would lead to the reduction of connection weights. The presynaptic spike events would lead to the increment of connection weights.

description of the proposed method is provided in the following sections.

3. Method

The proposed intralayer-connected spiking neural network with hybrid training using backpropagation and probabilistic spike-timing dependent plasticity (ISNN-HY) improved the SNNs trained with BP with two efforts. First, as we all know, since the backpropagation algorithm needs to calculate the gradients of connection weights, ANNs arrange neurons into layer-by-layer connections to avoid mutual connections or loops, which are common in biological neural systems. We proposed a novel LIF neuron model with intralayer connections and designed the forward and backward processes for BP training. Second, the STDP method was found to be engaged in the connection adjustment between neurons in biological neural systems, which are unsupervised, biologically plausible, and able to learn weights flexibly. However, it is difficult for SNNs to get good pattern recognition performance after training with the STDP algorithm. We proposed a hybrid training method for SNNs using backpropagation and a novel probabilistic STDP method for effective training.

3.1. LIF Neuron Model with Intralayer Connections

3.1.1. The Forward Process. The LIF neuron model simulates the biological neurons, which receive neurotransmitters through the dendrites and then increase the membrane potential until reaching a threshold, finally emitting neurotransmitters to the postsynaptic neurons through the axons and synaptic terminals [26]. The biological neural system allows intralayer (intrabrain area) connections. Therefore, to enhance the biological plausibility and computational ability of SNNs, we added intralayer connections to the dynamics of the LIF neuron model (red dashed line in Figure 2).

The framework of the LIF neuron model with intralayer connections is shown in Figure 2. The connected neurons on the left are a schematic representation of biological neurons with intralayer connections. Neuron A1 could emit spikes to

neurons A3 and A4 in the layer $l + 1$ and neuron A2 in the same layer. The right part describes the dynamics of the proposed LIF neuron model with intralayer connections. The presynaptic potential of a neuron is calculated by

$$c_i^{(l)}(t) = \sum_j w_{j,i} a_j^{(l-1)}(t). \quad (2)$$

$c_i^{(l)}(t)$ is the presynaptic potential of the neuron i in the layer l at time point t . $w_{j,i}$ is the connection weight between the neuron i and the neuron j in the previous layer $l - 1$. $a_j^{(l-1)}(t)$ is the postsynaptic potential of the neuron j at time point t . In the LIF neuron model, the presynaptic potentials could affect the membrane potential by

$$m_i^{(l)}(t) = \frac{\tau_m - 1}{\tau_m} m_i^{(l)}(t - 1) + R c_i^{(l)}(t) + \xi(t). \quad (3)$$

$m_i^{(l)}(t)$ is the membrane potential of the neuron i . It is calculated by the sum of the leaky membrane potential from the previous time point $\frac{\tau_m - 1}{\tau_m} m_i^{(l)}(t - 1)$, presynaptic potential $R c_i^{(l)}(t)$, and reset function $\xi(t)$. τ_m is the membrane potential constant, which defines the leaking speed of the membrane potential. R is the resistance of the presynaptic potential. The reset function $\xi(t)$ was used to reset the membrane potential when it reached the threshold of spike emission θ , defined by

$$\xi(t) = \begin{cases} 0, & m_i^{(l)}(t) < \theta, \\ -\frac{\tau_m - 1}{\tau_m} m_i^{(l)}(t - 1) - R c_i^{(l)}(t), & m_i^{(l)}(t) \geq \theta. \end{cases} \quad (4)$$

The reset function $\xi(t)$ would reset the membrane potential to rest potential 0 mV. The membrane potential was affected by the presynaptic potentials from the neurons in the previous layer. We added an intralayer connection part in the calculation of the membrane potential in the proposed neuron model:

$$m_i^{(l)}(t) = m_i^{(l)}(t) + U \sum_j v_{j,i} a_j^{(l)}(t - 1). \quad (5)$$

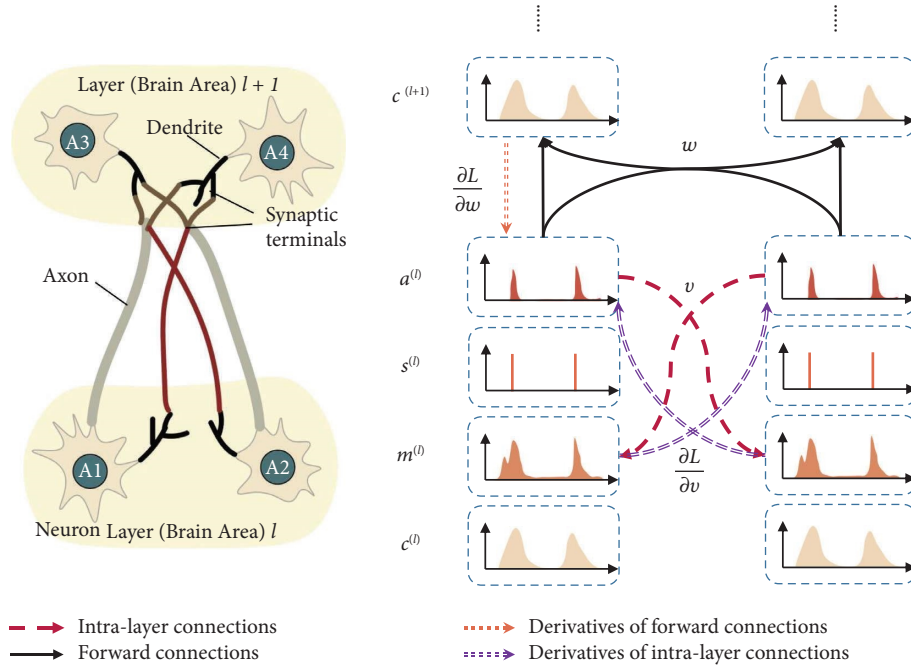


FIGURE 2: The proposed LIF neuron model with intralayer connections.

$m_i^{(l)}(t)$ is the membrane potential in the proposed neuron model. $v_{j,i}$ is the weight of intralayer connection from the neuron j to the neuron i in the same layer. $a_j^{(l)}(t-1)$ is the postsynaptic potential of the neuron j at the previous time point. U is the resistance of the intralayer presynaptic potential. Then, the neuron could emit spikes and postsynaptic potential to postsynaptic neurons by

$$s_i^{(l)}(t) = \begin{cases} 0 \text{ (rest state)}, & m_i^{(l)}(t) < \theta, \\ 1 \text{ (spike state)}, & m_i^{(l)}(t) \geq \theta. \end{cases} \quad (6)$$

$s_i^{(l)}(t)$ is the spike emission state of the neuron i , which was controlled by the relationship between the membrane potential and threshold θ :

$$a_i^{(l)}(t) = \left(\frac{\tau_s - 1}{\tau_s} \right) a_i^{(l)}(t-1) + \frac{1}{\tau_s} s_i^{(l)}(t). \quad (7)$$

In the forward process's final state, a neuron emits postsynaptic potentials to the postsynaptic neurons in the next layer or the same layer. The postsynaptic $a_i^{(l)}(t)$ of the neuron i at time point t was controlled by the leaking postsynaptic potential $(\tau_s - 1/\tau_s)a_i^{(l)}(t-1)$ and spike emission state $1/\tau_s s_i^{(l)}(t)$. τ_s is the postsynaptic potential constant modifying the leaking speed.

However, in the batch calculation, equation (5) was converted to the form of matrix calculation:

$$\mathbf{M}'^{(l)}(t) = \frac{\tau_m - 1}{\tau_m} \mathbf{M}'^{(l)}(t-1) + \mathbf{R} \mathbf{A}^{(l-1)} \times (t) \mathbf{W} + \mathbf{U} \mathbf{A}^l(t-1) \times \mathbf{V} + \xi(t). \quad (8)$$

$\mathbf{M}'^{(l)}(t)$ is the membrane potential matrix at time point t , $\mathbf{M}'^{(l)}(t) \in \mathbb{R}^{b \times N^{(l)}}$. b is the batch size. $N^{(l)}$ is the number of

neurons in the layer l . $\mathbf{A}^{(l-1)}(t)$ is the postsynaptic potential matrix at time point t , $\mathbf{A}^{(l-1)}(t) \in \mathbb{R}^{b \times N^{(l-1)}}$. \mathbf{W} is the weight matrix of forward connections, $\mathbf{W} \in \mathbb{R}^{N^{(l-1)} \times N^{(l)}}$. \mathbf{V} is the weight matrix of intralayer connections, $\mathbf{V} \in \mathbb{R}^{N^{(l)} \times N^{(l)}}$.

In equation (8), the parts with most computational complexity are $\mathbf{A}^{(l-1)}(t) \times \mathbf{W}$ and $\mathbf{A}^l(t-1) \times \mathbf{V}$. The computational complexity of them is $O(bN^{(l-1)}N^{(l)})$ and $O(bN^{(l)}N^{(l)})$. Since all the independent variables in the first part have been calculated at the $l-1$ layer, it can be accelerated by matrix parallel computation. The actual computation speed is much higher than the theoretical computation complexity. However, $\mathbf{A}^l(t-1)\mathbf{V}$ must be calculated time point by time point. High computation complexity is unacceptable. We optimized the calculation by converting equation (8) to the following equation:

$$\mathbf{M}'^{(l)}(t) = \frac{\tau_m - 1}{\tau_m} \mathbf{M}'^{(l)}(t-1) + \mathbf{R} \mathbf{A}^{(l-1)} \times (t) \mathbf{W} + \mathbf{U} \mathbf{A}^l(t-1) \times \mathbf{V}_1 \times \mathbf{V}_2 + \xi(t). \quad (9)$$

\mathbf{V}_1 and \mathbf{V}_2 are both trainable weight matrices, $\mathbf{V}_1 \in \mathbb{R}^{N^{(l)} \times r}$ and $\mathbf{V}_2 \in \mathbb{R}^{r \times N^{(l)}}$. r is the reduction constant for reducing computational complexity, which could be set to a small number. \mathbf{V}_1 and \mathbf{V}_2 are subject to assumptions:

$$\mathbf{V} = \mathbf{V}_1 \times \mathbf{V}_2. \quad (10)$$

After optimization, the forward computational complexity of intralayer connections is $O(2brN^{(l)})$. For most SNNs, the number of neurons in most layers $N^{(l)}$ is much larger than r . Therefore, by optimization, we could avoid the huge computational resource consumption caused by extra intralayer connections.

3.1.2. *The Backward Process.* The backward process in backpropagation training was based on the TSSL-BP method [6]. We used the loss function for producing the desired firing sequence specified by the input class label:

$$L = \sum_{k=0}^{N_t} \frac{1}{2} [(\varepsilon * d)(t_k) - (\varepsilon * s)(t_k)]^2. \quad (11)$$

L is the loss function. $d(t)$ and $s(t)$ are the desired and actual spike trains for the output layer at time t . ε is a kernel function measuring the van Rossum distance between the actual and desired spike trains. Then, the TSSL-BP method was used to calculate the derivative of the connection weight \mathbf{W} :

$$\frac{\partial L}{\partial \mathbf{W}} = \sum_{k=0}^{N_t} a^{(l-1)}(t_k) \delta^{(l)}(t_k). \quad (12)$$

$\partial L / \partial \mathbf{W}$ is the derivative of the forward connection weight \mathbf{W} in the layer l . $\delta^{(l)}(t_k)$ is the backpropagation error at time point t_k , defined as

$$\delta^{(l)}(t_k) = \begin{cases} \sum_{i=k}^T (a^{(l)}(t_i) - a_d^{(l)}(t_i)) \phi_i^{(l)}(t_i, t_k), & l = N_l, \\ (\mathbf{W}^{(l+1)})^T \sum_{i=k}^{N_l} \phi_i^{(l)}(t_i, t_k) \delta^{(l)}(t_i), & l < N_l. \end{cases} \quad (13)$$

$\phi_i^{(l)}(t_i, t_k)$ is the neuron dependency according to the presynaptic firing times. The details of $\phi_i^{(l)}(t_i, t_k)$ and TSSL-BP method are described in [6]. Although TSSL-BP was defined for classical LIF neuron models, the intralayer connections used in the proposed neuron model would not disturb the calculation of the derivative of \mathbf{W} in the backward process.

The other trainable parameters are \mathbf{V}_1 and \mathbf{V}_2 . The derivatives of them are calculated by the following equations:

$$\frac{\partial L}{\partial \mathbf{V}_2} = \left(\sum_{i=0}^{N_t} (\delta^{(l)}(t_i) \times \mathbf{V}_2^T)^T \times U \mathbf{A}^l (t_i - 1) \right)^T, \quad (14)$$

$$\frac{\partial L}{\partial \mathbf{V}_1} = \left(\sum_{i=0}^{N_t} \delta^{(l)}(t_i)^T \times U \mathbf{A}^l (t_i - 1) \times \mathbf{V}_1 \right)^T. \quad (15)$$

$(\delta^{(l)}(t_i))$ is $\partial L / \partial \mathbf{M}'^{(l)}(t_i)$, which is the derivative of membrane potentials at time point t_i . According to the above equations, the derivatives of all the trainable parameters were attainable.

3.2. *Probabilistic Spike-Timing Dependent Plasticity.* The forward and backward processes have proved the trainability of LIF neuron models with intralayer connections using

backpropagation. ISNN-HY has also proposed a biologically plausible algorithm to fine adjust the connection weights between neurons according to STDP rules.

The STDP rule modulates the connection weights sample by sample, which would be inefficient for most datasets. The proposed probabilistic STDP learning method is expressed as

$$\Delta w = \sum_{t_{\text{pre}}} \sum_{t_{\text{post}}} f_{w,p}(t_{\text{post}} - t_{\text{pre}}), \quad (16)$$

$$f_{w,p}(\Delta t) = \begin{cases} P_{\text{post}} A_{\text{pre}} e^{-\Delta t / \tau_{\text{pre}}} & \Delta t > 0, \\ P_{\text{pre}} A_{\text{post}} e^{\Delta t / \tau_{\text{post}}} & \Delta t < 0. \end{cases}$$

$f_{w,p}$ is the function calculating the change of weights depending on the firing probability of neurons. P_{post} is the postsynaptic spike probability, which is defined as the smaller firing probability of the presynaptic neuron at t_{pre} ($P_{t_{\text{pre}}}^{\text{neuron}_{\text{pre}}}$) and postsynaptic neuron at t_{post} ($P_{t_{\text{post}}}^{\text{neuron}_{\text{post}}}$). It could be expressed as

$$P_{\text{post}} = \min(P_{t_{\text{pre}}}^{\text{neuron}_{\text{pre}}}, P_{t_{\text{post}}}^{\text{neuron}_{\text{post}}}). \quad (17)$$

As shown in Figure 3, P_{post} is the conducting probability of the spiking events along the forward direction of connections between neurons. n_{pre} indicates the presynaptic neuron. n_{post} indicates the postsynaptic neuron. On the contrary, P_{pre} is the presynaptic spike probability, which is defined as the smaller firing probability of the postsynaptic neuron at t_{pre} ($P_{t_{\text{pre}}}^{\text{neuron}_{\text{post}}}$) and presynaptic neuron at t_{post} ($P_{t_{\text{post}}}^{\text{neuron}_{\text{pre}}}$). It could be expressed as

$$P_{\text{pre}} = \min(P_{t_{\text{pre}}}^{\text{neuron}_{\text{post}}}, P_{t_{\text{post}}}^{\text{neuron}_{\text{pre}}}), \quad (18)$$

which means the conducting probability of the spiking events along the backward direction of connections between neurons. The probabilistic STDP learning method could calculate the adjustment strength of connection weights for all samples through the activation probability at once. However, in practical implementation, P_{pre} and P_{post} of all pairs of neurons still need to be sequentially traversed and calculated, which has extremely high computational complexity. Therefore, we optimized its computational efficiency through matrix computation in the implementation to parallelize computation:

$$\mathbf{P}_{\text{post}} = \min \left(\mathbf{P}_{t_{\text{pre}}}^{n_{\text{pre}}} \times \mathbf{1}^{1 \times N_{\text{post}}}, \mathbf{1}^{N_{\text{pre}} \times 1} \times \left(\mathbf{P}_{t_{\text{post}}}^{n_{\text{post}}} \right)^T \right), \quad (19)$$

$$\mathbf{P}_{\text{pre}} = \min \left(\mathbf{P}_{t_{\text{post}}}^{n_{\text{pre}}} \times \mathbf{1}^{1 \times N_{\text{post}}}, \mathbf{1}^{N_{\text{pre}} \times 1} \times \left(\mathbf{P}_{t_{\text{pre}}}^{n_{\text{post}}} \right)^T \right).$$

\mathbf{P}_{post} and \mathbf{P}_{pre} are the postsynaptic and presynaptic spike probability matrices, $\mathbf{P}_{\text{post}}, \mathbf{P}_{\text{pre}} \in \mathbb{R}^{N_{\text{pre}} \times N_{\text{post}}}$. N_{pre} and N_{post} are the number of presynaptic and postsynaptic neurons. $\mathbf{P}_{t_{\text{pre}}}^{n_{\text{pre}}}$ is the spike probability vector for n_{pre} at time point t_{pre} . $\mathbf{1}^{1 \times N_{\text{post}}}$ and $\mathbf{1}^{N_{\text{pre}} \times 1}$ are the row vector and column vector setting all element as 1, respectively. By converting equations

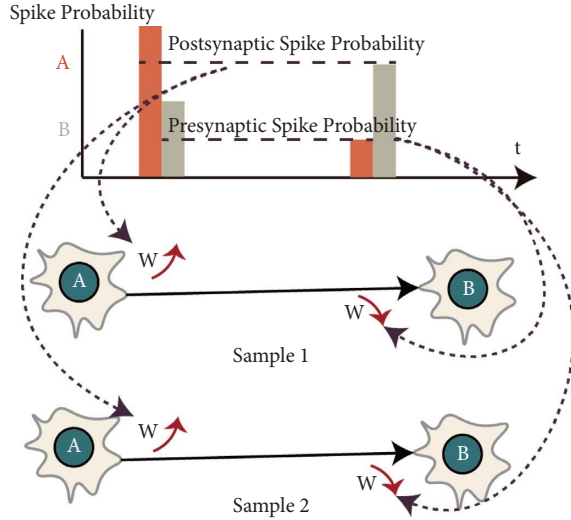


FIGURE 3: The proposed probabilistic STDP learning method.

(17) and (18) to matrix computation in equation (19), the spike probability of all pairs of neurons could be calculated in a matrix style, which is more efficient.

Finally, we show the details of probabilistic STDP with the pseudocode shown in Algorithm 1. This work used this algorithm to adjust the weights of the forward connections in fully connected layers.

3.3. Hybrid Training with Backpropagation and Probabilistic STDP. In the training phase, the proposed ISNN-HY method adopted the hybrid training method using backpropagation and probabilistic STDP to learn connection weights. Backpropagation was the primary learning method, and each batch of sample data updated the connection weight according to the backpropagation algorithm. After a round of training for all the sample data, the probabilistic STDP method was used to fine adjust the neuron connection weights according to the firing probability of neurons obtained from all samples. The training process is shown in Algorithm 2.

4. Experiments and Results

4.1. Dataset and Experiment Setups. The proposed ISNN-HY method was tested on nonneuromorphic datasets (MNIST [30], Fashion-MNIST [31], and CIFAR10 [32]) and neuromorphic datasets (N-MNIST [33]).

The MNIST, Fashion-MNIST, and CIFAR10 datasets are standard benchmarks used to test the performance of machine-learning algorithms. The MNIST dataset contains 60,000 handwritten digital training images and 10,000 test images. All images are single-channel grayscale images. The resolution of images is 28×28 . The Fashion-MNIST dataset contains ten different styles of clothing images. This dataset has the same structure as the MNIST dataset. CIFAR10

contains images of ten types of objects. Each category has 6,000 samples. The numbers of training and test images are 50,000 and 10,000, respectively. All the images are three-channel color images. The resolution of images is 32×32 . All the datasets' images were normalized before being put into the SNN models. The images in CIFAR10 were also augmented by random cropping and horizontal flipping. All the images were encoded into real-valued spike current inputs within a short time window, using the method proposed in [6].

The N-MNIST dataset is the neuromorphic version of MNIST, which converts pixels into spikes and consists of the same 60,000 training and 10,000 testing samples as the MNIST dataset.

We implemented four network structures in the four datasets, shown in Table 1. "mCn" denotes a convolutional layer with m channels, and the convolution size is $n \times n$. "Pn" denotes a pooling layer with $n \times n$ filters. "nFC" represents a fully connected layer with n neurons. We compared the performance with that of other SNN models in the same structures. We also compared the TSSL-BP model and intralayer-connected spiking neural network without hybrid training (ISNN) in the same structures with the proposed ISNN-HY method to prove the improvement of the intralayer connections and hybrid training. We set the hyperparameters as follows: learning rate $lr = 5e^{-4}$, $N_e = 100$ for all datasets except CIFAR10, $N_e = 150$ for CIFAR10, $N_t = 5$, $\tau_m = 4$, $\tau_s = 2$, $R = 1$, $U = 1$, $r = 8$, $A_{pre} = A_{post} = 5e^{-4}$, and $\tau_{pre} = \tau_{post} = 1$. The optimization for backpropagation used in the experiments is AdamW [34]. The desired output spike trains were set as $[0, 1, 1, 1, 1]$. The parameters such as thresholds and learning rates are empirically and manually tuned. We vary the membrane potential constant from 2 ms to 8 ms. The same performance level has been observed. All the models were trained and tested in different random seeds. All models were trained and tested 10 times, each time using a different random seed to ensure that the performance of the models is not due to randomness. The average accuracy and maximum accuracy of all models are reported.

According to the FLOPs of SNN models used in [35], the FLOPs of the four SNN models in this work could be easily estimated. The FLOPs of a convolutional layer of proposed SNN model could be calculated by $(k^l)^2 \times H_o^l \times W_o^l \times C_o^l \times C_i^l \times \xi^l$. k^l represents both the height and weight of the filter. H_o^l , W_o^l , and C_o^l are the tensor height, weight, and number of input channels and output channels. ξ^l is the firing rate of the neurons in the layer. The FLOPs of an SNN fully connected layer could be calculated by $(f_i^l \times f_o^l + 2 \times f_o^l \times r) \times \xi^l$. f_i^l and f_o^l are the number of input and output features. r is the reduction constant for recurrent intralayer connections. $2 \times f_o^l \times r$ is the additional FLOP for calculation of the proposed intralayer connections. The number of trainable parameters in the SNN models is also calculated. The FLOPs are shown in Table 1. The experiments were implemented using Python 3.6 and PyTorch 1.8 on an NVIDIA RTX 3090 GPU.

Require: Spike probability at each time point \mathbf{p} , number of layers N_l , number of time points N_t , and weight matrix \mathbf{W} parameters

$A_{\text{pre}}, A_{\text{post}}, \tau_{\text{pre}}, \tau_{\text{post}}$

Ensure: Update the weight \mathbf{W}

- (1) **for** $l = 0$ **to** N_l **do**
- (2) **if** l layer is not fully connected **then**
- (3) Continue
- (4) **end if**
- (5) Matrix $\Delta\mathbf{W}^{(l)} = 0$
- (6) **for** $t_{\text{pre}} = 0$ **to** $N_t - 1$ **do**
- (7) **for** $t_{\text{post}} = t_{\text{pre}} + 1$ **to** N_t **do**
- (8) Calculate \mathbf{P}_{post} using equation (19)
- (9) Calculate \mathbf{P}_{pre} using equation (19)
- (10) $\Delta t = t_{\text{post}} - t_{\text{pre}}$
- (11) Calculate the update weight matrix between t_{pre} And t_{post}
- (12) Calculate $\Delta\mathbf{W}$ using equation (16)
- (13) $\Delta\mathbf{W}^{(l)} = \Delta\mathbf{W}^{(l)} + \Delta\mathbf{W}$
- (14) **end for**
- (15) **end for**
- (16) **Update the connection weight for layer!**
- (17) $W^{(l)} = W^{(l)} + \Delta\mathbf{W}^{(l)}$
- (18) **end for**

ALGORITHM 1: Probabilistic STDP.

Require: Number of training epoch N_e , training batches B_s , and the ISNN-HY model Net

Ensure: Train Net

- (1) **for** $e = 0$ **to** N_e **do**
- (2) **for** data, label in B_s **do**
- (3) **Training using backpropagation**
- (4) prediction = Net (data)
- (5) update the spike probability \mathbf{p}
- (6) Calculate loss L using equation (11)
- (7) Calculate $\partial L / \partial \mathbf{W}$ using equations (12) and (13)
- (8) Calculate $\partial L / \partial \mathbf{V}_1$ and $\partial L / \partial \mathbf{V}_2$ using equations (14) and (15)
- (9) update weights of Net
- (10) **end for**
- (11) **Fine adjust using probabilistic STDP**
- (12) update weights of Net using probabilistic STDP
- (13) **end for**

ALGORITHM 2: Hybrid training.

TABLE 1: Network structures used in experiments.

Name	Structures	Datasets	Parameters	FLOP (M)	ANN FLOP
Net1	15C5-P2-40C5-P2-300FC-10FC	MNIST	0.69 M	0.65	3.65 M
Net2	400FC-400FC-10FC	Fashion-MNIST	0.47 M	0.21	0.46 M
Net3	128C3-256C3-P2-512C3-P2-1024C3-512C3-1024FC-512FC-10FC	CIFAR10	0.19 G	53.77	1.19 G
Net4	12C5-P2-64C5-P2-10FC	N-MNIST	3.84 M	0.54	3.84 M

The loss curves of ISNN and ISNN-HY are shown in Figure 4. It can be seen that the training set loss of both models can quickly decrease and stabilize at a low value, indicating that these two models converge in all four network structures.

4.2. Performance

4.2.1. MNIST Dataset. We compared the accuracies of ISNN and ISNN-HY with other SNN training results on the MNIST dataset. The results can be found in Table 2.

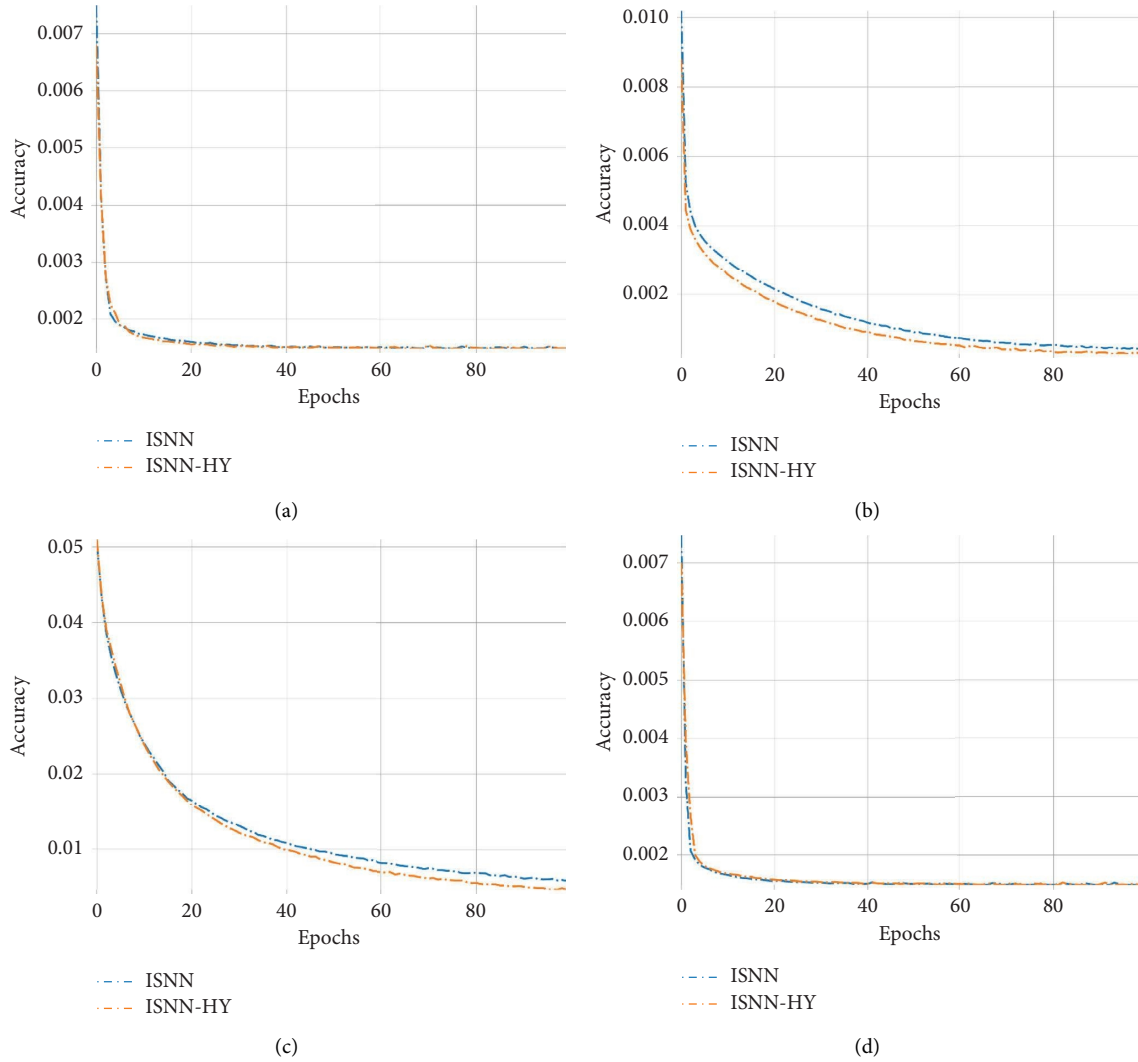


FIGURE 4: Training set loss curves for ISNN and ISNN-HY with four different network structures: (a) loss curve for Net1, (b) loss curve for Net2, (c) loss curve for Net3, and (d) loss curve for Net4.

TABLE 2: Comparison of different models on the MNIST dataset.

Models	Structure	N_t	Average accuracy (%)	Max accuracy (%)
Wu et al. [11]	Net1	> 100	—	99.42
Jin et al. [36]	Net1	400	99.42	99.49
Zhang and Peng [6]	Net1	5	99.50	99.53
Wang et al. [13]	Net1	5	99.48	99.52
Zhang and Peng [12]	Net1	400	99.57	99.62
Amiri et al. [37]	—	—	—	97.8
ISNN	Net1	5	99.53	99.55
ISNN-HY	Net1	5	99.55	99.58

The bold values in the first column are the names of the proposed models. The bold values in the last two columns are the best average and max accuracies.

The proposed ISNN method achieved an average accuracy of 99.53% and a maximum accuracy of 99.55%. ISNN-HY achieved a maximum accuracy of 99.58% and an average accuracy of 99.55%. ISNN-HY outperformed other SNN models with the same structure, improving the accuracy by 0.05%–1.78%, except for ST-RSBP [12]. The proposed

model also has a higher average accuracy than other models. Although the proposed model performed worse than ST-RSBP, ST-RSBP required 400 time steps to achieve 99.62% accuracy. The proposed method only used five time steps, which are much shorter than those used by ST-RSBP.

TABLE 3: Comparison of different models on the Fashion-MNIST dataset.

Models	Structure	N_t	Average accuracy (%)	Max accuracy (%)
ANN	Net2	—	—	89.01
Zhang and Peng [6]	Net2	5	89.75	89.80
Wang et al. [13]	Net2	5	89.47	89.56
Jin et al. [36]	Net2	400	—	88.99
ISNN	Net2	5	89.91	89.99
ISNN-HY	Net2	5	90.00	90.03

The bold values in the first column are the names of the proposed models. The bold values in the last two columns are the best average and max accuracies.

TABLE 4: Comparison of different models on the CIFAR10 dataset.

Models	Structure	N_t	Average accuracy (%)	Max accuracy (%)
ANN in Wu et al. [38]	Net3	—	—	90.49
Zhang and Peng [6]	Net3	5	—	91.41
Wang et al. [13]	Net3	10	91.03	91.11
Wu et al. [38]	Net3	400	—	90.53
Rathi et al. [39]	ResNet20	250	—	92.22
Rathi and Roy [40]	VGG16	5	—	92.70
Sengupta et al. [41]	VGG16	2500	—	91.55
Amiri et al. [37]	Brain-inspired structure	—	—	93.1
ISNN	Net3	5	91.40	91.45
ISNN-HY	Net3	5	91.43	91.47

The bold values in the first column are the names of the proposed models. The bold values in the last two columns are the best average and max accuracies.

TABLE 5: Comparison of different models on the N-MNIST dataset.

Models	Structure	N_t	Average accuracy (%)	Max accuracy (%)
Sumit and Orchard [42]	Net4	300	99.20	99.22
Zhang and Peng [6]	Net4	100	99.35	99.40
ISNN	Net4	100	99.36	99.42
ISNN-HY	Net4	100	99.39	99.44

The bold values in the first column are the names of the proposed models. The bold values in the last two columns are the best average and max accuracies.

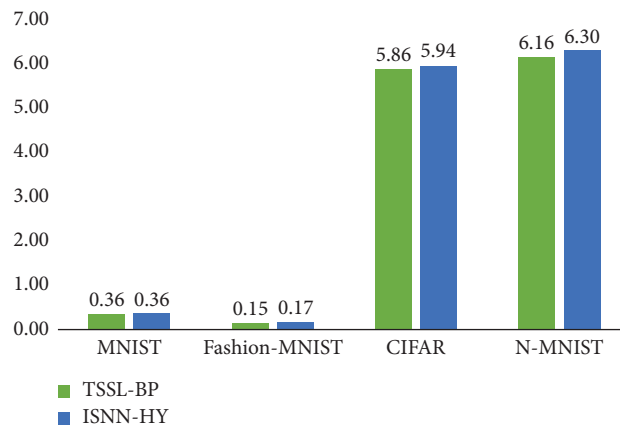


FIGURE 5: Training time of TSSL-BP and ISNN-HY.

4.2.2. Fashion-MNIST Dataset. The performance of ISNN and ISNN-HY on the Fashion-MNIST dataset was evaluated using three-layer fully connected structures, with Table 3 showing the results. An average accuracy of 89.91% and a maximum accuracy of 89.99% were achieved by the

proposed ISNN method, while ISNN-HY achieved an average accuracy of 90.90% and a maximum accuracy of 90.03%. Other SNN and ANN models with the same structure were outperformed by the proposed ISNN-HY model, with an accuracy improvement of 0.23%–1.02%.

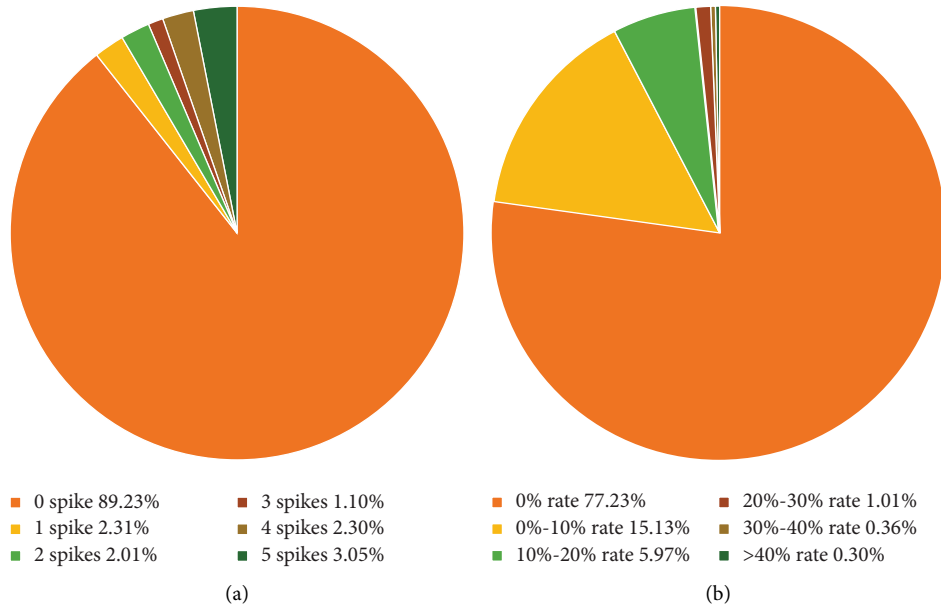


FIGURE 6: Firing activity on CIFAR10 and N-MNIST.

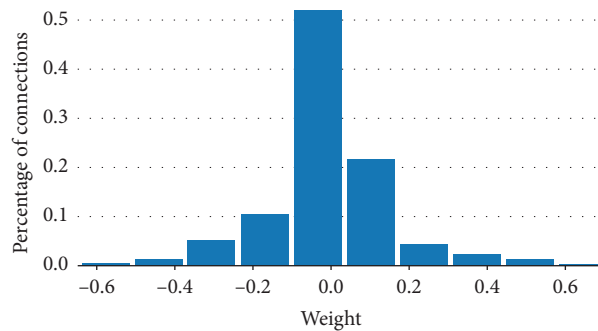


FIGURE 7: The histogram of intralayer connection weights on CIFAR.

4.2.3. CIFAR10 Dataset. In addition, the more challenging CIFAR10 dataset was used to apply the proposed method. As shown in Table 4, an average accuracy of 91.40% and a maximum accuracy of 91.45% were achieved by the proposed ISNN model. The ISNN-HY model achieved an average accuracy of 91.43% and a maximum accuracy of 91.47%. While the proposed model did not outperform SNN models with VGG16, ResNet20, or brain-inspired structures [37, 39–41], it did surpass methods such as TSSL-BP [6] and RDS-BP [13] with the same structure and achieved a maximum accuracy improvement of 0.06–0.94%. Although the proposed model performed worse than more complex models, this is likely due to the complexity of model structures. As long as overfitting does not occur, the more complex the model structure, whether it is ANN or SNN, the stronger the model’s memory and reasoning ability, resulting in better performance. Therefore, although the proposed ISNN-HY model did not perform as well as SNN models with more complex structures such as DIET-SNN reported in other studies, it still outperformed other ANN and SNN models with the same structure.

4.2.4. N-MNIST Dataset. We evaluated the proposed SNN model with a convolutional structure on the N-MNIST dataset. Table 5 compares the results obtained by previous SNN models. The ISNN-HY and ISNN models processed spike patterns and achieved 99.44% and 99.42% maximum accuracies, respectively. The proposed model outperformed results from the previous models.

Results show that ISNN-HY demonstrates outstanding performance on all four datasets, especially surpassing other SNN models with the same structure. The performance of ISNN is slightly inferior to ISNN-HY, possibly due to the use of probability-based STDP for fine-tuning local connection weights in ISNN-HY, which improves performance. The performance of ISNN is superior to that of TSSL-BP with the same structure. The difference between the two models lies in whether the neurons in the fully connected layer have intralayer connections. This difference suggests that the proposed intralayer connections can increase interactions between neurons within the same layer and improve model performance.

4.3. Experiment Analysis

4.3.1. Time Analysis. Compared with TSSL-BP, the neuron model and the training method of ISNN-HY are more complex. Therefore, it is necessary to analyze the training time to evaluate whether these proposed methods could reduce the computational efficiency of the model too much.

The training time of TSSL-BP and ISNN-HY is shown in Figure 5. It can be found from the results that the training time difference between TSSL-BP and ISNN-HY for a single sample is only 0.14 seconds/sample. This result shows that although the proposed ISNN-HY method increases a large number of intralayer neuronal connections and additional probabilistic STDP training, it does not increase too much additional computational resource consumption after optimization. In Table 1, the computational complexity (number of FLOPs) of the SNN model and the same structured ANN model is compared, and it can be observed that the proposed ISNN-HY model, although increases intralayer connections, has significantly lower computational complexity than the same structured ANN model, thereby demonstrating the high efficiency advantage of the proposed SNN model.

4.3.2. Neuron Activity Analysis. The proposed ISNN-HY method can train the exact SNN with the same layer connection. At the same time, the firing activity of the well-trained network also tends to be sparse. To prove emission sparsity, we choose two well-trained SNN models, one for CIFAR10 and the other for N-MNIST, as shown in Figure 6. It can be found that the firing rate of neurons on both CIFAR10 and N-MNIST datasets is sparse. Our proposed intralayer connection does not alter the sparsity of the spiking neuronal activity.

The weight distribution of intralayer connections is shown in Figure 7. It can be found that most connections had a weight close to 0 ranging from -0.1 to 0.1 , and a small number of connections had large weights. It indicates that the weights of the intralayer connections will be automatically adjusted during training to achieve the biological rationality of sparse connections. Therefore, the biological rationality of sparse neuronal connections and firing rates and accurate inference ability give ISNN-HY the potential to simulate biological neural activity, such as modeling the functional magnetic resonance imaging (fMRI) and electroencephalogram (EEG) [43, 44].

5. Conclusion

This work proposed an LIF neuron model with intralayer connections and probabilistic STDP for fast training in different types of datasets. Combining these methods, we proposed an intralayer-connected spiking neural network with hybrid training using BP and probabilistic STDP, called ISNN-HY. ISNN-HY achieved excellent performances on nonneuromorphic and neuromorphic datasets and has sufficient biological rationality and potential to simulate biological neural activities.

Data Availability

The research data used to support the findings of this study are all publicly available, and the sources of them are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the Shanghai Sailing Program (23YF1401100) and the Fundamental Research Funds for the Central Universities (Grant no. 2232021D-26).

References

- [1] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons Populations Plasticity*, Cambridge University Press, Cambridge, United Kingdom, 2002.
- [2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
- [3] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature Neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.
- [4] Q. Yu, C. Ma, S. Song, G. Zhang, J. Dang, and K. C. Tan, "Constructing accurate and efficient deep spiking neural networks with double-threshold and augmented schemes," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 4, pp. 1714–1726, 2022.
- [5] F. Zenke and S. Ganguli, "Superspike: supervised learning in multilayer spiking neural networks," *Neural Computation*, vol. 30, no. 6, pp. 1514–1541, 2018.
- [6] W. Zhang and L. Peng, "Temporal spike sequence learning via backpropagation for deep spiking neural networks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., pp. 12022–12033, Curran Associates, Inc, Red Hook, NY, USA, 2020.
- [7] H. Wu, Y. Zhang, W. Weng et al., "Training spiking neural networks with accumulated spiking flow," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, pp. 10320–10328, 2021.
- [8] C. Hur, B. Ibromkhimov, and S. Kang, "N3-cpl: neuroplasticity-based neuromorphic network cell proliferation learning," *Neurocomputing*, vol. 411, pp. 193–205, 2020.
- [9] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "Stdp-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56–67, 2018.
- [10] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in Neuroscience*, vol. 12, p. 774, 2018.
- [11] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in Neuroscience*, vol. 12, p. 331, 2018.
- [12] W. Zhang and L. Peng, "Spike-train level backpropagation for training deep recurrent spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [13] Z. Wang, Y. Zhang, H. Shi, L. Cao, C. Yan, and G. Xu, "Recurrent spiking neural network with dynamic presynaptic

- currents based on backpropagation,” *International Journal of Intelligent Systems*, vol. 37, no. 3, pp. 2242–2265, 2022.
- [14] X. Cheng, H. Yunzhe, J. Xu, and B. Xu, “Lisnn: improving spiking neural networks with lateral interactions for robust object recognition,” in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence (IJCAI)*, Virtual, Montreal, August 2021.
- [15] W. Zhang and L. Peng, “Spiking neural networks with laterally-inhibited self-recurrent units,” in *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Shenzhen, China, July 2021.
- [16] W. Zhang and P. Li, “Skip-connected self-recurrent spiking neural networks with joint intrinsic parameter and synaptic weight training,” *Neural Computation*, vol. 33, no. 7, pp. 1886–1913, 2021.
- [17] W. Zhang and L. Peng, “Composing recurrent spiking neural networks using locally-recurrent motifs and risk-mitigating architectural optimization,” 2021, <https://arxiv.org/abs/2108.01793>.
- [18] A. Tavanaei and A. S. Maida, “Bio-inspired spiking convolutional neural network using layer-wise sparse coding and stdp learning,” 2016, <https://arxiv.org/abs/1611.03000>.
- [19] A. Tavanaei and A. S. Maida, “Multi-layer unsupervised learning in a spiking convolutional neural network,” in *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Anchorage, AK, USA, July 2017.
- [20] J. Zhang, R. Wang, X. Pei, D. Luo, S. Hussain, and G. Zhang, “A fast spiking neural network accelerator based on bp-stdp algorithm and weighted neuron model,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 4, pp. 2271–2275, 2022.
- [21] A. Tavanaei, Z. Kirby, and A. S. Maida, “Training spiking convnets by stdp and gradient descent,” in *Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Rio de Janeiro, Brazil, July 2018.
- [22] W. Guo, X. Lin, and X. Yang, “A supervised learning algorithm for recurrent spiking neural networks based on bp-stdp,” in *Proceedings of the Neural Information Processing: 28th International Conference, ICONIP 2021*, pp. 583–590, Springer, Sanur, Bali, Indonesia, December 2021.
- [23] F. Liu, W. Zhao, Y. Chen, Z. Wang, T. Yang, and L. Jiang, “Sstdp: supervised spike timing dependent plasticity for efficient spiking neural network training,” *Frontiers in Neuroscience*, vol. 15, Article ID 756876, 2021.
- [24] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, “Backpropagation and the brain,” *Nature Reviews Neuroscience*, vol. 21, no. 6, pp. 335–346, 2020.
- [25] P. R. Roelfsema and A. Holtmaat, “Control of synaptic plasticity in deep cortical networks,” *Nature Reviews Neuroscience*, vol. 19, no. 3, pp. 166–180, 2018.
- [26] A. N. Burkitt, “A review of the integrate-and-fire neuron model: I. homogeneous synaptic input,” *Biological Cybernetics*, vol. 95, no. 1, pp. 1–19, 2006.
- [27] H. Markram, W. Gerstner, and P. J. Sjöström, “Spike-timing-dependent plasticity: a comprehensive overview,” *Frontiers in Synaptic Neuroscience*, vol. 4, no. 2, p. 2, 2012.
- [28] V. Demin and D. Nekhaev, “Recurrent spiking neural network learning based on a competitive maximization of neuronal activity,” *Frontiers in Neuroinformatics*, vol. 12, p. 79, 2018.
- [29] Y. Xing, G. Di Caterina, and J. Soraghan, “A new spiking convolutional recurrent neural network (scrnn) with applications to event-based hand gesture recognition,” *Frontiers in Neuroscience*, vol. 14, Article ID 590164, 2020.
- [30] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [31] X. Han, K. Rasul, and V. Roland, “Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms,” 2017, <https://arxiv.org/abs/1708.07747>.
- [32] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features from Tiny Images*, Toronto, ON, Canada, Canada, 2009.
- [33] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, “Converting static image datasets to spiking neuromorphic datasets using saccades,” *Frontiers in Neuroscience*, vol. 9, p. 437, 2015.
- [34] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *Proceedings of the International Conference on Learning Representations*, Vancouver CANADA, May 2018.
- [35] G. Datta, S. Kundu, and P. A. Beerel, “Training energy-efficient deep spiking neural networks with single-spike hybrid input encoding,” in *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Shenzhen, China, May 2021.
- [36] Y. Jin, W. Zhang, and L. Peng, “Hybrid macro/micro level backpropagation for training deep spiking neural networks,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [37] M. Amiri, A. H. Jafari, B. Makkiabadi, S. Nazari, and M. M. Van Hulle, “A novel un-supervised burst time dependent plasticity learning approach for biologically pattern recognition networks,” *Information Sciences*, vol. 622, pp. 1–15, 2023.
- [38] Y. Wu, L. Deng, G. Li, J. Zhu, X. Yuan, and L. Shi, “Direct training for spiking neural networks: faster, larger, better,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1311–1318, 2019.
- [39] N. Rathi, G. Srinivasan, P. Panda, and K. Roy, “Enabling deep spiking neural networks with hybrid conversion and spike timing dependent backpropagation,” 2020, <https://arxiv.org/abs/2005.01807>.
- [40] N. Rathi and K. Roy, “Diet-snn: a low-latency spiking neural network with direct input encoding and leakage and threshold optimization,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, 2021.
- [41] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, “Going deeper in spiking neural networks: vgg and residual architectures,” *Frontiers in Neuroscience*, vol. 13, p. 95, 2019.
- [42] B. S. Sumit and G. Orchard, “Slayer: spike layer error reassignment in time,” *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [43] C. Tan, M. Šarlija, and N. Kasabov, “Spiking neural networks: background, recent development and the neucube architecture,” *Neural Processing Letters*, vol. 52, no. 2, pp. 1675–1701, 2020.
- [44] N. Kasabov, L. Zhou, M. Gholami Doborjeh, Z. Gholami Doborjeh, and J. Yang, “New algorithms for encoding, learning and classification of fmri data in a spiking neural network architecture: a case on modeling and understanding of dynamic cognitive processes,” *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 4, pp. 293–303, 2017.