Fully Spiking Actor Network With Intralayer Connections for Reinforcement Learning

Ding Chen, Peixi Peng, Tiejun Huang, *Senior Member, IEEE*, and Yonghong Tian, *Affiliate, IEEE*

***Abstract*— With the help of special neuromorphic hardware, spiking neural networks (SNNs) are expected to realize artificial intelligence (AI) with less energy consumption. It provides a promising energy-efficient way for realistic control tasks by combining SNNs with deep reinforcement learning (DRL). In this article, we focus on the task where the agent needs to learn multidimensional deterministic policies to control, which is very common in real scenarios. Recently, the surrogate gradient method has been utilized for training multilayer SNNs, which allows SNNs to achieve comparable performance with the corresponding deep networks in this task. Most existing spike-based reinforcement learning (RL) methods take the firing rate as the output of SNNs, and convert it to represent continuous action space (i.e., the deterministic policy) through a fully connected (FC) layer. However, the decimal characteristic of the firing rate brings the floating-point matrix operations to the FC layer, making the whole SNN unable to deploy on the neuromorphic hardware directly. To develop a fully spiking actor network (SAN) without any floating-point matrix operations, we draw inspiration from the nonspiking interneurons found in insects and employ the membrane voltage of the nonspiking neurons to represent the action. Before the nonspiking neurons, multiple population neurons are introduced to decode different dimensions of actions. Since each population is used to decode a dimension of action, we argue that the neurons in each population should be connected in time domain and space domain. Hence, the intralayer connections are used in output populations to enhance the representation capacity. This mechanism exists extensively in animals and has been demonstrated effectively. Finally, we propose a fully SAN with intralayer connections (ILC-SAN). Extensive experimental results demonstrate that the proposed method outperforms the state-of-the-art performance on continuous control tasks from OpenAI gym. Moreover, we estimate the theoretical energy consumption when deploying ILC-SAN on neuromorphic chips to illustrate its high energy efficiency.**

***Index Terms*— Brain-inspired intelligence, intralayer connections, neuromorphic engineering, nonspiking neurons, reinforcement learning (RL), spiking neural networks (SNNs).**

I. Introduction

RECENTLY, guided by the brain, neuromorphic computing has emerged as one of the most promising types of computing architecture, which could realize energy-efficient artificial intelligence (AI) through spike-driven communication [1], [2], [3]. The research efforts of neuromorphic computing not only facilitate the emergence of large-scale neuromorphic chips [4], [5], [6], but also promote the development of spiking neural networks (SNNs) [7], [8], [9]. In this context, the field of neuromorphic computing is a close cooperation organic whole between hardware and algorithm.

An accumulating body of research studies shows that SNNs can be used as energy-efficient solutions for robot control tasks with limited on-board energy resources [10], [11], [12]. To overcome the limitations of SNNs in solving high-dimensional control problems, it would be natural to combine the energy-efficiency of SNNs with the optimality of deep reinforcement learning (DRL), which has been proven effective in extensive control tasks [13], [14]. Since rewards are regarded as the training guidance in reinforcement learning (RL), several works [15], [16] employ reward-based learning using three-factor learning rules. However, these methods only apply to shallow SNNs and low-dimensional control tasks, or require manual tuning of the network architecture and numerous hyperparameters related to neurons and synapses for each scenario [17]. Besides, the surrogate gradient method [18] provides a promising way to train deep SNNs. In a typical control task, the agent needs to learn multidimensional deterministic policies. In the deterministic policy learning, the RL methods [19], [20], [21] often utilize the actor-critic framework, where the actor network maps the state to the continuous action space and the critic network is used to represent state-action value (i.e., Q value). Note that the critic network is just used for training, and only the actor network is deployed on the hardware of the application, hence, it is necessary to develop a spiking actor network (SAN) as the energy-efficient solution for control tasks. Such network structure includes a deep critic network and a SAN, also known as the hybrid framework. However, the existing methods based on hybrid

Manuscript received 21 November 2022; revised 29 April 2023; accepted 1 November 2023. This work was supported in part by the Key-Area Research and Development Program of Guangdong Province under Grant 2020B0101380001; in part by the National Natural Science Foundation of China under Grant 62372010, Grant 62027804, Grant 61825101, and Grant 62088102; and in part by the major key project of the Peng Cheng Laboratory under Grant PCL2021A13. *(Corresponding authors: Peixi Peng; Yonghong Tian.)*

Ding Chen is with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China, and also with the Network Intelligence Research, Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: lucifer1997@sjtu.edu.cn).

Peixi Peng is with the Department of Computer Science and Technology, Peking University, Beijing 100871, China, and also with the Network Intelligence Research, Peng Cheng Laboratory, Shenzhen 518066, China (e-mail: pxpeng@pku.edu.cn).

Tiejun Huang is with the Department of Computer Science and Technology, Peking University, Beijing 100871, China (e-mail: tjhuang@pku.edu.cn).

Yonghong Tian is with the Department of Computer Science and Technology, Peking University, Beijing 100871, China, also with the Network Intelligence Research, Peng Cheng Laboratory, Shenzhen 518066, China, and also with the School of Electronics Computer Engineering, Peking University, Shenzhen 518055, China (e-mail: yhtian@pku.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2024.3352653

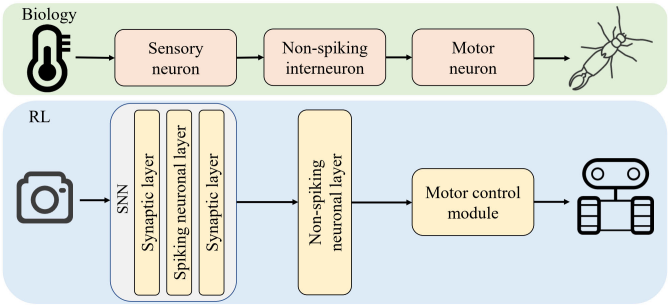

Fig. 1. Correspondence diagram between our method and the sensory motor neuron pathway.

framework [22], [23] take the firing rate[1] as the output of SNNs, and convert it to represent continuous action space (i.e., the deterministic policy) through a fully connected (FC) layer, which bring the floating-point matrix operations to the FC layer, making the whole SNN unable to deploy on the neuromorphic hardware directly.

To develop a fully SAN which could be deployed on neuromorphic chips without any floating-point matrix operations, the key issue is to design a novel neural coding method to decode the spike-train into the continuous value in RL algorithms, such as value estimate or continuous action space, realizing end-to-end spike-based RL. In nature, sensory neurons receive information from the external environment and transmit it to nonspiking interneurons through action potentials [24], and then change the membrane voltage of motor neurons through graded signals to achieve effective locomotion. As a translational unit, nonspiking interneurons could affect the motor output according to the sensory input. Inspired by biological research on sensorimotor neuron path, we propose a novel neural coding method to train SNNs for continuous control, where the membrane voltage of nonspiking neurons is used to represent the continuous value in RL algorithms. As shown in Fig. 1, the SNN receives the state from the environment and encodes them as spike-train by spiking neurons. Then, the nonspiking neurons are subsequently introduced to calculate the membrane voltage from the spike-train, which is further used to select the specific action to be performed. Finally, the agent controls the motor to adjust the motion according to the action.

In addition to the membrane voltage coding method, we also integrate the intralayer connections into the output populations to learn better action representation. The intralayer connections mainly contain self-connections and lateral connections, which widely exist in various brain areas [25]. This mechanism can cause generalization across actions [15] and retain the information in the neuron population [26], which is applied to the neuron population encoding the same action dimension. Therefore, we propose a fully SAN with intralayer connections, named ILC-SAN, which is a novel method based on hybrid framework. The experiment results show that our method achieves the start-of-the-art performance on OpenAI gym tasks. The main contributions of this article are summarized as follows.

1) A novel coding method to decode the spike-train for RL algorithms is proposed, which could represent the continuous value by the membrane voltage of nonspiking neurons. The method could be integrated into any RL algorithms, realizing end-to-end spike-based RL. The results of the experiment show that optimal performance could be achieved using the last membrane voltage of nonleaky integrate-and-fire (IF) neurons.
2) A novel intralayer connection mechanism is proposed, which has been proved by experiments that it can effectively improve the representation capacity of the output populations.
3) A novel SAN (ILC-SAN) is proposed. To the best of our knowledge, our ILC-SAN is the first fully SNN to achieve the same level of performance as the mainstream DRL algorithms, which ensures that all matrix operations can be completed on the neuromorphic hardware after deploying the trained actor network. The evaluation of continuous control tasks from OpenAI gym demonstrates the effectiveness of ILC-SAN in performance and energy efficiency. Under the same experimental configurations, our ILC-SAN achieves the start-of-the-art performance.

II. Related Work

A. Reward-Based Learning by Three-Factor Learning Rules

To bridge the gap between the time scales of behavior and neuronal action potential, modern theories of synaptic plasticity assume that the coactivation of presynaptic and postsynaptic neurons sets a flag at the synapse, called eligibility trace [27]. Only if a third factor, signaling reward, punishment, surprise, or novelty, exists while the flag is set, the synaptic weight will change. Although the theoretical framework of three-factor learning rules has been developed in the past few decades, experimental evidence supporting eligibility trace has only been collected in the past few years [28]. Through the derivation of the RL rule for continuous time, the existing approaches have been able to solve the standard control tasks [15] and robot control tasks [10]. Moreover, Vlasov et al. [29] demonstrate the in-principle possibility to build SNNs with memristor-based synapses trained by inherent local plasticity rules. However, these methods are only suitable for shallow SNNs and low-dimensional control tasks. To solve these problems, Bellec et al. [17] propose a learning method for recurrently connected networks of spiking neurons, which is called e-prop. Although the agent learned by reward-based e-prop successfully wins Atari games, the need that the network architecture and numerous hyperparameters related to neurons and synapses should be manually adjusted between different tasks limits the application of this method.

B. ANN to SNN Conversion for RL

By matching the firing rate of spiking neurons with the graded activation of analog neurons, trained ANNs can be converted into corresponding SNNs with few accuracy loss [30].

[1]It should be noted that the firing rate mentioned here and later refers to the ratio of the number of spikes to the discrete simulation time.

For the SNNs converted from the ANNs trained by the DQN algorithm, the firing rate of spiking neurons in the output layer is proportional to the Q value of the corresponding action, which makes it possible to select actions according to the relative size of the Q value [31], [32]. But there is a trade-off between accuracy and efficiency, which tells us that longer inference latency is needed to improve accuracy. As far as we know, for RL tasks, the converted SNNs cannot achieve better results than ANNs.

C. RL Methods Using Spike-Based BP

Following the surrogate gradient method [18], the spike-based backpropagation (BP) algorithm has quickly become the mainstream solution for training multilayer SNNs [8], [9]. Tang et al. [33] first propose the hybrid framework, composed of a SAN and a deep critic network. Through the colearning of the two networks, the hybrid framework avoids the problem of value estimation using SNNs. However, the method of scaling the firing rate makes it difficult to accurately characterize the continuous action space, which greatly limits the choice of actions. To improve the representation capacity of SNNs, Tang et al. [22] propose a population-coded SAN (PopSAN), which achieves the same level of performance as the deep network. Based on this work, a multiscale dynamic coding improved SAN (MDC-SAN) [23] is proposed and performs better than its counterpart deep actor network (DAN). In the decoding stage, the two methods use the FC layer to convert the firing rate into continuous action. Recently, Liu et al. [34] propose a direct spiking learning algorithm for the deep spiking Q-network (DSQN), using a FC layer to decode the firing rate into Q value (i.e., the state-action value). Sun et al. [35] implement the Q value in the same way. Nevertheless, these methods also bring new problems. Since the neuromorphic hardware only accepts the spike input in the form of 0 or 1, matrix operations of decoders need to be completed by other traditional hardware (i.e., CPU, GPU, or embedded AI chip). The resulting energy consumption problem is not reflected in the analysis of previous work [22], which makes the energy consumption analysis have great defects. Different from these methods, we use the membrane voltage of nonspiking neurons to represent the continuous value in RL algorithms, such as value estimate or continuous action space, so that our method can be directly applied to neuromorphic hardware. Take Loihi as an example, the internal structure of the neuromorphic core mainly includes four primary operating modes: input spike handling, neuron compartment updates, output spike generation, and synaptic updates [5]. By skipping the process of output spike generation that simulates axons, Loihi supports nonspiking compartments without voltage reset.

Besides, Akl et al. [36] use the membrane potential readout as the decoding method. However, it lacks a detailed analysis of membrane potential readout methods. Moreover, they encode the observation into a two-neuron input scheme first and then multiply the encoded vector by the weight matrix. Qin et al. [37] use a learnable matrix to compress spike-trains in the temporal dimension through matrix multiplication, which is also a feasible decoding method. Both of these two methods require floating-point matrix multiplication, and cannot be completed only using the neuromorphic hardware. Compared with them, the proposed method employs the population encoder to realize full spike-based RL. In addition, we propose a novel intralayer connection mechanism to enhance the action decoding.

D. Nonspiking Neurons

As shown in previous studies [38], [39], [40] and open-source frameworks [41], the membrane voltage of nonspiking neurons is feasible to represent a continuous value in spike-based BP methods. However, how to use it to effectively train SNNs for RL has not been systematically studied and remains unsolved, which is a goal of this article. Moreover, inspired by astrocytes, a Loihi-run central pattern generator (CPG) [42] is proposed, which uses nonspiking neurons to generate bursting spikes to control the motor. Their work illustrates the feasibility of nonspiking neurons on Loihi and complements the lack of specific motor control in our work, which constitutes a complete simulation of the sensor motor neuron path (Fig. 1).

E. Intralayer Connections

Intralayer connections play an essential role in the brain and are widely used in SNNs. Generally, intralayer connections can be divided into self-connections and lateral connections. The former is widely used in spiking neural models to provide more complex and adaptive dynamics. The latter, also known as lateral interactions, was previously used to build different receptive fields in visual tasks.

Caused by the reset mechanism, spiking neural models have the characteristics of self-connection. Recurrent SNN (RSNN) fully exploits the self-connection characteristics of spiking neural models, making them more adaptive [17], [43], [44]. However, most of them are based on soft reset or hard reset, and cannot describe hyperpolarization, which reduces the readiness of biological neurons to fire again after recent firing activity. To solve this problem, after-hyperpolarizing (AHP)-neurons [45] model the AHP currents, leading to spike frequency adaptation. Although the update to the AHP-current in response to an output spike is fixed by a hyperparameter, AHP-neurons develop a new way of self-connection and provide a principled alternative to LSTM units for sequence processing tasks.

In the field of neuroscience, the work on lateral interactions mainly focuses on the retina [46], which makes most of the work on lateral interactions in SNNs related to visual tasks [26], [47], [48]. Because of its ability to sharpen edges and highlight contours, lateral interactions are used to extract features from images. Furthermore, lateral interactions are widespread in various regions of the brain [25]. Frémaux and Gerstner [15] use population coding to represent actions, and each neuron encodes a different motion vector. They adopt fixed lateral interactions in the neuron population to implement the N-winner-take-all mechanism, leading to generalization across actions. However, there are several problems in this method, such as depending on the relatively low-dimension

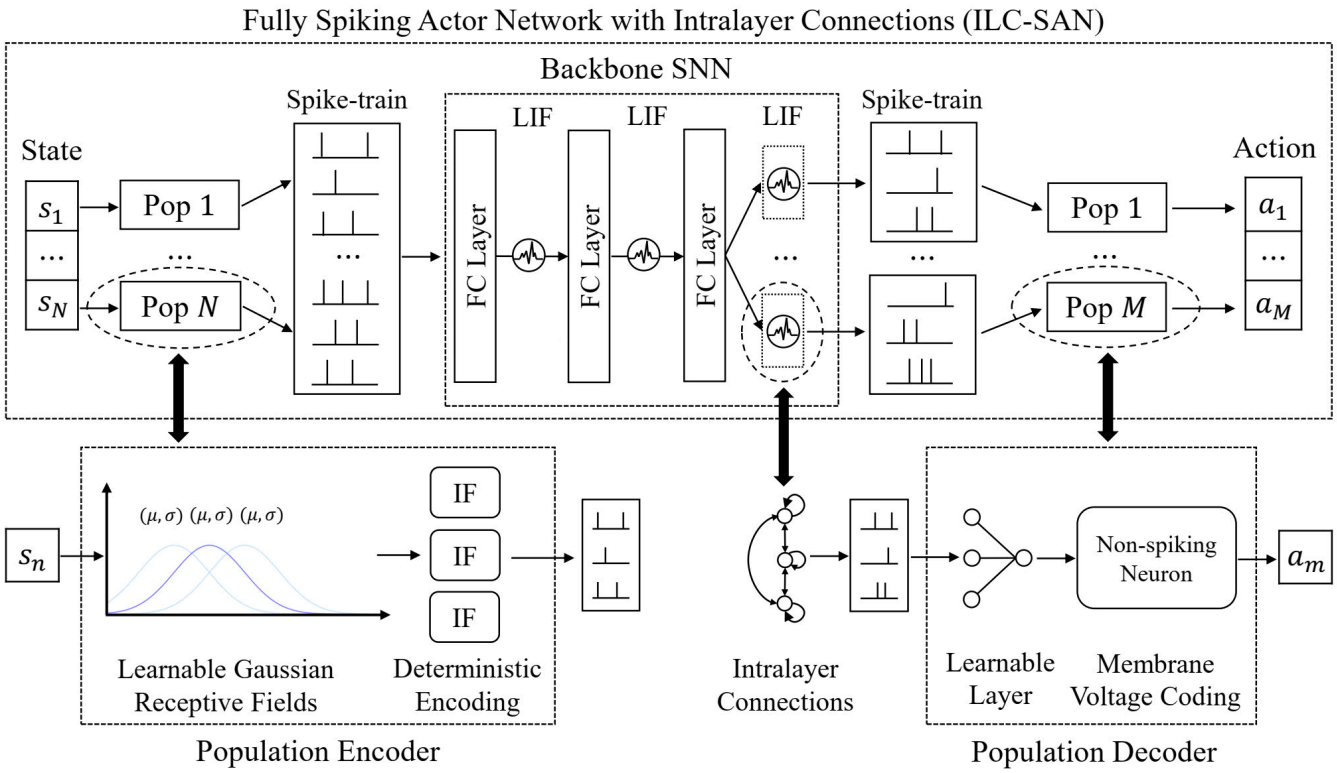

Fig. 2. Overall framework of the proposed ILC-SAN. The state is transformed into spike-trains by the population encoder. Each state dimension s_n is encoded by the corresponding input population, which consists of learnable Gaussian receptive fields and IF neurons. Each neuron in the input population has a different Gaussian kernel (μ, σ). Through these Gaussian kernels, s_n is first encoded into the stimulation strength for each neuron in the input population, and then transformed into the spike-trains using deterministic encoding. After that, the spike-trains are transmitted through the backbone SNN to the population decoder. The spiking neurons in the last layer of the backbone SNN are evenly divided into M output populations, and the intralayer connections are applied in each output population. Each output population has a corresponding population decoder, where the spike-trains are first integrated into a single nonspiking neuron and then decoded into the corresponding action dimension using membrane voltage coding.

action space, a large number of hyperparameters need to be adjusted for different action spaces, which limits its application in complex tasks.

Zhang and Li [49] use a method similar to ours to model intralayer connections. The difference is that the spike-train level postsynaptic potentials (S-PSPs) are transmitted between neurons from the same layer, rather than the spike signals in our method. However, S-PSPs rely on a relatively long spike-train, which makes its simulation time much longer than the methods using back-propagation through time (e.g., ours) [8], resulting in relatively high energy consumption and inference time.

III. Method

In this article, we focus on the typical control task where the agent needs to learn multidimensional deterministic policies. Note the RL methods [19], [20], [21] in this field often utilize the actor-critic framework, where the critic network is just used for training, and only the actor network is deployed on the hardware of the application. Hence, our method aims to develop a fully SAN that could be deployed on neuromorphic chips without any floating-point matrix operations. In this section, we first introduce the spiking neural model and its discrete dynamics. Then, we propose the nonspiking neurons and analyze the membrane voltage coding. Finally, we present the implemented details of our ILC-SAN (Fig. 2), which can be trained in conjunction with a deep critic network using the DRL algorithms. Moreover, the trained ILC-SAN provides an energy-efficient solution for continuous control tasks due to its fully spiking architecture.

A. Spiking Neural Model

The basic computing unit of SNNs is the spiking neuron. In the open-source deep learning framework for SNNs, spiking neural models are usually simulated in discrete time-steps. The dynamics of most kinds of discrete spiking neurons can be described as follows:

$$H_t = f(V_{t-1}, X_t) \tag{1}$$

$$S_t = \Theta(H_t - V_{\text{th}}) \tag{2}$$

$$V_t = H_t(1 - S_t) + V_{\text{reset}} S_t \tag{3}$$

where H_t and V_t denote the membrane voltage after neural dynamics and the trigger of a spike at time-step t, respectively. X_t denotes the external input, and S_t means the output spike at time-step t, which equals 1 if there is a spike and 0 otherwise. V_{th} denotes the threshold voltage and V_{reset} denotes the membrane reset voltage. As shown in Fig. 3, (1)–(3) establish a general mathematical model to describe the discrete dynamics of spiking neurons, which include charging, firing, and resetting. The dynamics described by (3) are called the hard reset. In addition, spiking neurons can also adopt soft

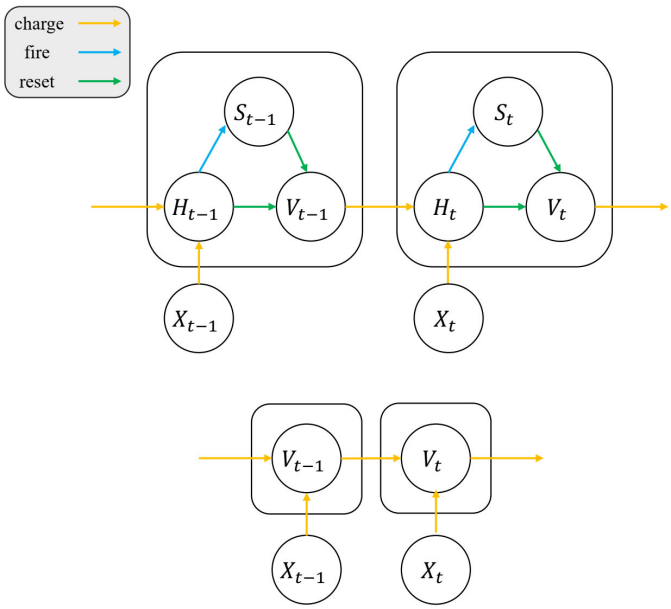

Fig. 3. General discrete neural model (Top) Spiking neural model. (Bottom) nonspiking neural model.

reset, whose dynamics can be described as follows:

$$V_t = H_t - V_{\text{th}} S_t. \tag{4}$$

Specifically, (1) describes the subthreshold dynamics, which vary with the type of neuron models. Here, we consider the IF model [50] and the leaky IF (LIF) model [51], the two most commonly used spiking neuron models. The function $f(\cdot)$ of the LIF neuron is defined as follows:

$$f(V_{t-1}, X_t) = \alpha_V (V_{t-1} - V_{\text{reset}}) + V_{\text{reset}} + X_t \tag{5}$$

where α_V is the voltage decay factor. When $\alpha_V = 1$, this equation represents the function $f(\cdot)$ of the IF neuron. The spike generative function $\Theta(x)$ is the Heaviside step function, which is defined by $\Theta(x) = 1$ for $x \geq 0$ and $\Theta(x) = 0$ for $x < 0$. Note that $V_0 = V_{\text{reset}}$, $S_0 = 0$.

B. Nonspiking Neural Model

Nonspiking neurons can be regarded as a special case of spiking neurons. If we set the threshold voltage V_{th} of spiking neurons to infinity, the dynamics of neurons will always be under the threshold, which is so-called nonspiking neurons. Therefore, the application of nonspiking neurons does not affect our network to be a fully SNN. Since nonspiking neurons do not have the dynamics of firing and resetting, we could simplify the neural model to the following equation (see Fig. 3):

$$V_t = f(V_{t-1}, X_t). \tag{6}$$

Here we consider the nonspiking LIF model, which can also be called the leaky integrate (LI) model. The dynamics of LI neurons are described by the following equation:

$$V_t = \alpha_V (V_{t-1} - V_{\text{reset}}) + V_{\text{reset}} + X_t. \tag{7}$$

When $\alpha_V = 1$, this equation represents the Integrate model.

C. Membrane Voltage Coding

According to the definition of SNNs, the output is a spike-train. However, the output of the functions in RL algorithms are all continuous values. To bridge the difference between these two data forms, we need a spike decoder to complete the data conversion. Inspired by the nonspiking interneurons found in insects, we propose to use nonspiking neurons as a bridge between perception and motion for decision-making. In the sensorimotor neuron path, spike signals transmitted by sensory neurons are integrated into nonspiking neurons. Then, the agents use the membrane voltage of nonspiking neurons to make decisions, which determines the input current of motor neurons.

In the whole simulation time T, the nonspiking neurons take the spike-train as the input sequence, and then the membrane voltage V_t at each time-step t can be obtained. To represent the final output O, we need to choose an optimal statistic according to the membrane voltage at all times, i.e., $O = \text{Stat}(V_1, \ldots, V_T)$.

To meet the needs of the algorithm, we finally design three statistics as candidates.

1) *Last Membrane Voltage D_{last}:* It is a natural idea to use the last membrane voltage after the simulation time as the characterization to make full use of all simulations. The formula of D_{last} is as follows:

$$O = V_T. \tag{8}$$

2) *Membrane Voltage With Maximum Absolute Value D_{max}:* By recording the membrane voltage of nonspiking neurons at each time-step in the whole simulation time, we can get the sequence of the membrane voltage. In previous work [38], the maximum membrane voltage is used to represent the probability of each category. However, this statistic is more suitable for representing nonnegative numbers. Due to the unknown sign of continuous values, the membrane voltage with the maximum absolute value is undoubtedly a better statistic. The formula of D_{max} is as follows:

$$O = V_{\underset{1 \leq t \leq T}{\arg\max} |V_t|}. \tag{9}$$

3) *Mean Membrane Voltage D_{mean}:* Similar to the maximum membrane voltage, we can obtain the mean value by collecting the membrane voltage at each time-step, which is also a meaningful statistic. The formula of D_{mean} is as follows:

$$O = \frac{1}{T} \sum_{t=1}^{T} V_t. \tag{10}$$

For these statistics, we empirically evaluate them in experiments, respectively, in Section IV-C. We find both D_{last} and D_{max} are effective and D_{last} performs better. In addition, it is obvious that D_{last} is easier to implement and deploy than D_{max}. Hence, we choose D_{last} as the default membrane voltage coding method in Section IV. It should be noted that the final output O in this article represents the action of any dimension.

D. Fully SAN With Intralayer Connections

In this section, we present the implemented details of our ILC-SAN. With the help of the membrane voltage of nonspiking neurons, we incorporate all matrix operations of SAN into the neuromorphic chips, solving the problem of floating-point matrix operations in the decoding stage. By applying intralayer connections in the output populations, the learned action representation could be better.

1) Population Encoder: For the N-dimensional state $\boldsymbol{s} \in \mathbb{R}^N$, we use a population of neurons E with different Gaussian receptive fields $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ to encode each state dimension s_i into a spike-train, where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are task-specific trainable parameters. Suppose P_{in} represents the input population sizes per state dimension, the shape of the spike-train $\boldsymbol{ST}$ is $(N \cdot P_{\text{in}}, T)$. The calculation process of the encoder can be divided into two stages. First, the state $\boldsymbol{s}$ is converted to the stimulation strength of each neuron in the population $\boldsymbol{A}_E$

$$\boldsymbol{A}_E = e^{-\frac{1}{2}(\frac{s_i-\mu}{\sigma})^2}. \tag{11}$$

Second, the computed $\boldsymbol{A}_E$ is used to generate the spike-train. For performance reasons [22], we use deterministic encoding, where the neurons are simulated as soft-reset IF neurons and $\boldsymbol{A}_E$ is the external input ($\boldsymbol{X}_t = \boldsymbol{A}_E$). Therefore, we call this population encoder $E_{\text{pop\_det}}$ for short. For simplicity, we directly back-propagate the gradient with respect to the stimulation strength $\boldsymbol{A}_E$ regardless of whether a spike is fired or not at any time-step t, $(\delta \boldsymbol{S}_t / \delta \boldsymbol{A}_E) = 1$.

2) CLIF Neurons: For a fair comparison, we employ the current-based LIF (CLIF) model of spiking neurons, which is used in previous work [22]. The CLIF neurons can integrate presynaptic spikes into the current and subsequently integrate the current into the membrane voltage ($\boldsymbol{X}_t = \boldsymbol{C}_t$), the detailed dynamics of which are described in Algorithm 1. α_C and α_V are the current and voltage decay factors. To distinguish from the voltage decay factors of nonspiking neurons, we express them as α_V^{LI} and α_V^{CLIF}. And the membrane reset voltage also uses different superscripts to distinguish the symbols. Except for the population encoder, all spiking neural models of ILC-SAN adopt hard reset. The rectangular function is used as the surrogate function. The gradient is defined by the following equation:

$$\Theta'(x) = \begin{cases} 1, & |x| < w \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

where w is the threshold window for passing the gradient.

3) Backbone SNN: The backbone SNN connects the population encoder and the population decoder, whose input and output are spike-trains. The typical architecture of SNNs consists of synaptic layers and neuronal layers [8]. The synaptic layers include convolutional layers and FC layers, each of which is followed by a neuronal layer. Since the population encoder will bring extremely high computation costs when handling high-dimensional data such as images or spatiotemporal signals, we extract the state feature as a 1-D vector and only use the FC layer as the synaptic layer. Moreover, the CLIF neurons form the neuronal layers. In addition, all weight parameters are shared at all simulation time-steps. Note that we treat the synaptic layer and its subsequent neuronal layer as one layer in formula derivation. The former plays a similar role to dendrites in neuronal cells, and the latter works like the bodies and axons. For a task with M-dimensional actions, we equally divide the spiking neurons of the last layer into M output population with a size of P_{out}. Each output population has a corresponding population decoder. Suppose that the backbone SNN has L layers, $\boldsymbol{W}^l$ and $\boldsymbol{b}^l$ represent the weights and biases of the lth layer. $\boldsymbol{C}_t^l$ represents the current for CLIF neurons of the lth layer at time-step t. Similarly, $\boldsymbol{H}_t^l$, $\boldsymbol{V}_t^l$, $\boldsymbol{S}_t^l$ represent different variables in CLIF neurons of the lth

Algorithm 1 Forward Propagation Through ILC-SAN

Input: N-dimensional state $\boldsymbol{s}$
Output: M-dimensional action $\boldsymbol{a}$

Initialize encoding means $\boldsymbol{\mu}$ and standard deviations $\boldsymbol{\sigma}$ for the population encoder;
Randomly initialize the L-layer backbone SNN;
Randomly initialize the population decoder;
Compute the spike-train from input populations generated by the encoder: $\boldsymbol{ST} = Encoder(\boldsymbol{s}, \boldsymbol{\mu}, \boldsymbol{\sigma})$;
for $t = 1, \cdots, T$ **do**
  Spikes from input populations at time-step t: $\boldsymbol{S}_t^0 = \boldsymbol{ST}_t$;
  for $l = 1, \cdots, L$ **do**
    Update CLIF neurons in layer l at time-step t:
    if $l = L \wedge t \neq 1$ **then**
      Integrate spikes from layer $l-1$ and layer l into the current:
      $\boldsymbol{C}_t^l = \alpha_C \cdot \boldsymbol{C}_{t-1}^l + \boldsymbol{W}^l \boldsymbol{S}_t^{l-1} + \boldsymbol{b}^l + \boldsymbol{I}_t$;
    else
      Integrate spikes from layer $l-1$ into the current:
      $\boldsymbol{C}_t^l = \alpha_C \cdot \boldsymbol{C}_{t-1}^l + \boldsymbol{W}^l \boldsymbol{S}_t^{l-1} + \boldsymbol{b}^l$;
    end if
    $\boldsymbol{H}_t^l = \alpha_V^{CLIF}(\boldsymbol{V}_{t-1}^l - V_{reset}^{CLIF}) + V_{reset}^{CLIF} + \boldsymbol{C}_t^l$;
    $\boldsymbol{S}_t^l = \Theta(\boldsymbol{H}_t^l - V_{th})$;
    $\boldsymbol{V}_t^l = \boldsymbol{H}_t^l(1 - \boldsymbol{S}_t^l) + V_{reset}^{CLIF} \boldsymbol{S}_t^l$;
  end for
  Divide $\boldsymbol{S}_t^L$ evenly among M output populations: $\{\boldsymbol{S}_t^{L,m}\}$, $m = 1, \cdots, M$;
  for $m = 1, \cdots, M$ **do**
    if $t \neq T$ **then**
      Calculate the current generated by intralayer connections $\boldsymbol{I}_{t+1}^m$ using Eq. (13);
    end if
    Update the LI neuron of the m-th output population:
    $X_t^m = \boldsymbol{W}_D^m \boldsymbol{S}_t^{L,m} + b_D^m$;
    $V_t^m = \alpha_V^{LI}(V_{t-1}^m - V_{reset}^{LI}) + V_{reset}^{LI} + X_t^m$;
  end for
  if $t \neq T$ **then**
    Merge $\{\boldsymbol{I}_{t+1}^m\}$, $m = 1, \cdots, M$ into $\boldsymbol{I}_{t+1}$;
  end if
end for
Generate M-dimensional action $\boldsymbol{a}$:
$a_m = O^m = Stat(V_1^m, \cdots, V_T^m)$, $m = 1, \cdots, M$;

layer at time-step t. Note that we use $\boldsymbol{S}_t^0$ to represent the spike input of the backbone SNN at time-step t, i.e., the output of population encoder at time-step t, $\boldsymbol{ST}_t$.

4) Population Decoder: Each population decoder consists of a learnable layer and an LI neuron. The mth population decoder projects the spike-trains from the mth output population into the mth action dimension, $m \in \{1, \ldots, M\}$. Specifically, in the mth population decoder, the spike-trains from the mth output population are inputted to the learnable layer first, and then the membrane voltage of the LI neuron is updated at each time-step t. After every T simulation time-steps, the input of the mth population decoder is decoded into the specified statistic of membrane voltage O^m, which represents the value of the mth action dimension. For the mth output population, $\boldsymbol{W}_D^m$ and b_D^m represent the weights and biases of the decoder, and X_t^m, V_t^m represent different variables in LI neurons at time-step t.

5) Intralayer Connections: For continuous control tasks from OpenAI gym, actions in different dimensions vary greatly. Therefore, we apply intralayer connections for each output population, that is, intralayer connections only occur between neuron populations encoding the same dimension of continuous action space. For the mth output population, the current received from the intralayer connections at the next time-step can be calculated as follows:

$$\boldsymbol{I}_{t+1}^m = \boldsymbol{W}_{\text{intra}}^m \boldsymbol{S}_t^{L,m} + \boldsymbol{b}_{\text{intra}}^m \tag{13}$$

where $\boldsymbol{W}_{\text{intra}}^m$, $\boldsymbol{b}_{\text{intra}}^m$ represent the weights and biases of the intralayer connections. $\boldsymbol{S}_t^{L,m}$ represent the spike output of the backbone SNN used in the mth output population at the time-step t. Note that $\boldsymbol{W}_{\text{intra}}^m$ is a square matrix of order P_{out}. Since the intralayer connection and the interlayer connection both are connections between neurons, we model them in the same way as FC layers. As similar to the implementation of RNNs,[2] $\boldsymbol{b}_{\text{intra}}^m$ is introduced to improve the fitting ability of $\boldsymbol{W}_{\text{intra}}^m \boldsymbol{S}_t^{L,m}$ by learning the data bias.

E. ILC-SAN Embedded Into TD3

Our ILC-SAN is functionally equivalent to a DAN, which can be trained in conjunction with a deep critic network using TD3 algorithms [20]. During training, the ILC-SAN establishes a mapping between states and actions to represent the policy of agents. The deep critic network estimates the corresponding Q value, which can guide the ILC-SAN to learn a better policy. During the evaluation, the trained ILC-SAN can predict the action with the maximum Q value produced by the trained critic network. More details can be found in the forward propagation of ILC-SAN (Algorithm 1).

IV. Experiments

In this section, we first evaluate the performance of ILC-SAN on eight continuous control tasks from OpenAI gym. Then, we analyze the effects of different decoders on performance, and evaluate the influence of each component in the intralayer connections. Finally, we compare the theoretical energy consumption of different models according to the power efficiency of their advanced hardware. In addition, our experiments are built upon the open-source code of PopSAN [22].

[2]https://pytorch.org/docs/stable/generated/torch.nn.RNN.html

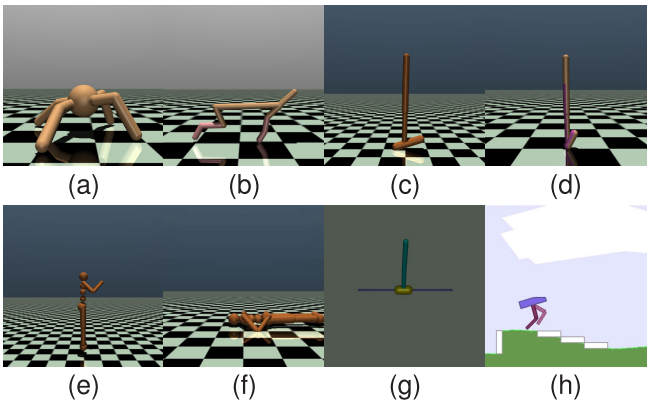

Fig. 4. Eight continuous control tasks from OpenAI gym. (a) Ant-v3: make a 3-D four-legged robot move forward as fast as possible by applying torques on the eight hinges. (b) HalfCheetah-v3: make a 2-D robot run forward as fast as possible by applying a torque on the joints. (c) Hopper-v3: make a 2-D one-legged figure hop forward as fast as possible by applying torques on the three hinges. (d) Walker2d-v3: make a 2-D two-legged figure move forward as fast as possible by applying torques on the six hinges. (e) Humanoid-v3: make a 3-D bipedal robot walk forward as fast as possible without falling over by applying torques on the seventeen hinges. (f) HumanoidStandup-v2: make a 3-D bipedal robot stand up and then keep it standing by applying torques on the seventeen hinges. (g) InvertedDoublePendulum-v2: balance the second pole on top of the first pole by applying continuous forces on the cart. (h) BipedalWalker-v3: make a 2-D walker robot walk forward without falling over by controlling the motor speed for each of the four joints at both hips and knees.

A. Experimental Settings

We evaluate our approach on the OpenAI gym tasks, which are often used as a benchmark for continuous control algorithms. All the tasks used are shown in Fig. 4. To save computational resources, we only use all the tasks for the comparative experiments in Section IV-B, while for other experiments, we use the first four most commonly used tasks. Table I shows the state dimension and action dimension for different tasks, which affects the network structure and the number of operations. To ensure the reproducibility, each of our models is trained for ten rounds, corresponding to ten random seeds. In each round, the task is trained for 1 million steps and evaluated every 10k steps, where each evaluation reported the average reward over ten episodes using the deterministic policy. Each episode can last up to 1000 execution steps. In all experiments, we use TD3 algorithm to train the ILC-SAN.

Hyperparameter configurations for the methods used subsequently are as follows: the deep critic network is ($N+M$, 256, relu, 256, relu, 1); the DAN is (N, 256, relu, 256, relu, M, tanh); the ILC-SAN is ($N \cdot P_{\text{in}}$, 256, CLIF, 256, CLIF, $M \cdot P_{\text{out}}$), the population size for each state and action dimension is 10; the learning rate of the DAN and the deep critic network is 1e-3; the learning rate of the ILC-SAN is 1e-4; the reward discount factor is 0.99; the Gaussian exploration noise with stddev is 0.1; the Gaussian smoothing noise for target policy with stddev is 0.2; the maximum length of replay buffer is

TABLE I
STATE DIMENSION AND ACTION DIMENSION OF DIFFERENT TASKS

Task	N	M
Ant-v3	111	8
HalfCheetah-v3	17	6
Hopper-v3	11	3
Walker2d-v3	17	6
Humanoid-v3	376	17
HumanoidStandup-v2	376	17
InvertedDoublePendulum-v2	11	1
BipedalWalker-v3	24	4

1e6; the soft target update factor is 0.005; the batch size is 100; the noise clip is 0.5; the policy delay factor is 2.

For CLIF neurons, we use the same hyperparameters as the open-source code of PopSAN, so the membrane reset voltage $V_{\text{reset}}^{\text{CLIF}}$ is 0.0, the threshold voltage V_{th} is 0.5, the current decay factor α_C is 0.5, the voltage decay factor α_V^{CLIF} is 0.75, and the threshold window w is 0.5. For LI neurons, we set $V_{\text{reset}}^{\text{LI}}$ to 0.0. Since α_V^{CLIF} is a fixed value in all experiments, we use α_V to represent the voltage decay factor of nonspiking neurons in the rest of Section IV.

B. Comparison With the State-of-the-Art

According to the input of SNNs, we can divide the existing methods into two categories, namely, spike input [22] and floating-point input [23], [36], [37]. The former considers both energy consumption and performance to facilitate the deployment of neuromorphic hardware. And the latter focuses on performance improvement, so its performance is usually considered higher [23]. When comparing with such methods, we need to modify our population encoder $E_{\text{pop\_det}}$. Here, we tested two alternative input coding methods, raw population coding E_{pop} [23] and learnable layer coding E_{layer} [36], [37]. For E_{pop}, we remove the second stage of the population encoder and use $\boldsymbol{A}_E$ calculated by (11) as the input of the backbone SNN directly at each time-step t, i.e., $\boldsymbol{S}_t^0 = \boldsymbol{A}_E$. For E_{layer}, we remove the population encoder and take the state $\boldsymbol{s}$ as the input directly at each time-step t, i.e., $\boldsymbol{S}_t^0 = \boldsymbol{s}$. Therefore, the size of $\boldsymbol{S}_t^0$ is $N \cdot P_{\text{in}}$ for $E_{\text{pop\_det}}$ and E_{pop}, N for E_{layer}. Note that only the full method $E_{\text{pop\_det}}$ meets the requirements of fully SNNs. E_{pop} and E_{layer} are only used to compare with other methods with the same input coding scheme.

We compare the performance of our ILC-SANs with other SANs, taking the average performance ratio (APR) of different SANs to the corresponding DANs across all the tasks as the measurement standard. This measure can be described as the following equation:

$$\text{APR} = \frac{1}{N_{\mathcal{T}}} \sum_{\text{task} \in \mathcal{T}} \frac{\text{AN}_{\text{task}}}{\text{DAN}_{\text{task}}} \tag{14}$$

where $\mathcal{T}$ represents a set of tasks, and $N_{\mathcal{T}} = |\mathcal{T}|$. AN_{task} is the max average rewards of the given actor network over 10 random seeds on the corresponding task, where AN can be any actor network, such as DAN, PopSAN, MDC-SAN, and ILC-SAN.

To avoid the impact of software package versions and random seeds, almost all the experiments are under the same experimental setting, except for AC-BCQ [37]. Since AC-BCQ has not been officially published and its source code has not been released, we directly use the data provided in the article. We rerun DAN, PopSAN, and the method proposed by Akl et al. [36] using their open-source code. Using the best dynamic parameters of dynamic neurons given by the authors, we reproduce the MDC-SAN and rerun it. As Table II shows, our models achieve better performance than PopSAN on all three different input coding methods, which demonstrates that our method has a marked effect for different encoders. When considering ILC-SAN under different encoders, $E_{\text{pop\_det}}$ achieves the best performance, which is different from the experimental conclusions obtained by previous methods [23]. Considering the significant advantage of energy consumption brought by spike input, we focus on the ILC-SAN using $E_{\text{pop\_det}}$, which is the first fully SAN. Meanwhile, its performance outperforms the state-of-the-art methods and the corresponding DANs, which demonstrates that our method has achieved a win-win situation in terms of performance and energy efficiency. Finally, the performance variance of our method is relatively small in most games, which indicates the stability of our method.

The complete learning curves of PopSAN and ILC-SAN using $E_{\text{pop\_det}}$ across the first four tasks are shown in Fig. 5. As we can see, ILC-SAN achieves better performance on Ant-v3 and Hopper-v3. Especially on Hopper-v3, ILC-SAN successfully solves the learning problems encountered by PopSAN. In addition, ILC-SAN achieves the same level of performance on HalfCheetah-v3 and Walker2d-v3.

In Fig. 6, we show the complete learning curves of MDC-SAN and ILC-SAN using E_{pop} across the first four tasks. It can be seen that ILC-SAN achieves better performance on Walker2d-v3. In addition, ILC-SAN achieves the same level of performance on Ant-v3, HalfCheetah-v3, and Hopper-v3. Although the performance of ILC-SAN on HalfCheetah-v3 and Hopper-v3 is slightly inferior at the early stage of learning, it still converges to a considerable performance after around 1 million training steps.

C. Analysis of Decoding Methods

We begin by evaluating the influence of various decoding methods on performance while removing the intralayer connections in the ILC-SAN. Due to the relationship between the performance of membrane voltage coding methods and the voltage decay factor of nonspiking neurons, we choose three representative values of the voltage decay factor, $\alpha_V = 1.0$, 0.75, 0.5. As shown in Table III, we compare our membrane voltage coding methods with the mainstream decoding method based on firing rate (D_{fr}) [23], [33] using $E_{\text{pop\_det}}$ and demonstrate that both D_{last} and D_{max} are superior to D_{fr} in performance. Compared with D_{last} and D_{max}, the performance of D_{mean} is relatively poor. When we deploy the model to the neuromorphic chips, D_{last} only needs to retain the membrane voltage at the last time-step after executing T time-steps. Due to the advantages of deployment and performance, we finally adopt D_{last} as the decoding scheme.

TABLE II

MAX AVERAGE REWARDS OVER TEN RANDOM SEEDS FOR DAN, ILC-SAN, AND OTHER SANS

Task	DAN [20]	PopSAN [22]	MDC-SAN [23]	Akl et al. [36]	AC-BCQ [37]	ILC-SAN		
						E_{layer}	E_{pop}	E_{pop_det}
Ant-v3	5472±653	4848±1023	5410±748	3820±942	4070±34	5752±258	5387±529	5653±533
HalfCheetah-v3	10471±1695	10523±1142	10472±1146	3220±1857	10610±63	10120±1864	10422±909	10376±1163
Hopper-v3	3520±105	517±1057	3554±153	2713±1424	2527±455	3518±182	3547±163	3583±86
Walker2d-v3	4999±842	4199±1426	4604±809	3208±1514	2551±932	5263±869	4919±627	4511±462
Humanoid-v3	3681±2258	5821±440	5879±225	135±123	–	5872±236	5241±1633	5874±395
HumanoidStandup-v2	129271±13021	155387±21632	149815±10404	73620±33564	–	156062±18252	161889±31302	160974±22132
InvertedDoublePendulum-v2	8425±2950	9327±3	9323±6	9338±8	–	9335±10	9336±13	9327±5
BipedalWalker-v3	194±146	267±93	267±98	190±86	–	202±151	272±100	301±10
APR	100.00%	101.81%	114.48%	63.90%	74.63%	112.77%	114.47%	**118.05%**

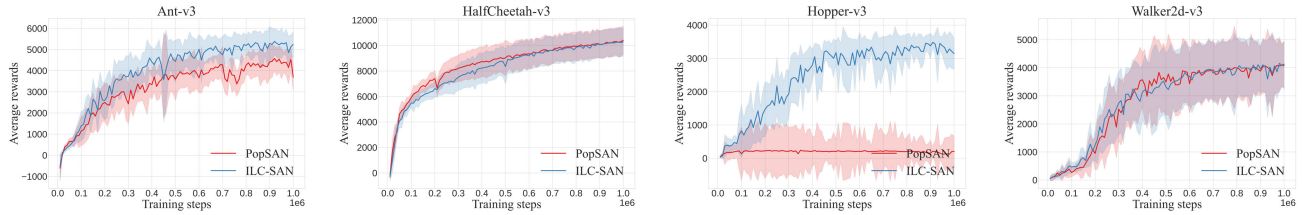

Fig. 5. Comparison of average rewards for PopSAN and ILC-SAN using $E_{\text{pop\_det}}$ over ten random seeds. The shaded area represents half the value of the standard deviation, and the curves are smoothed for clarity.

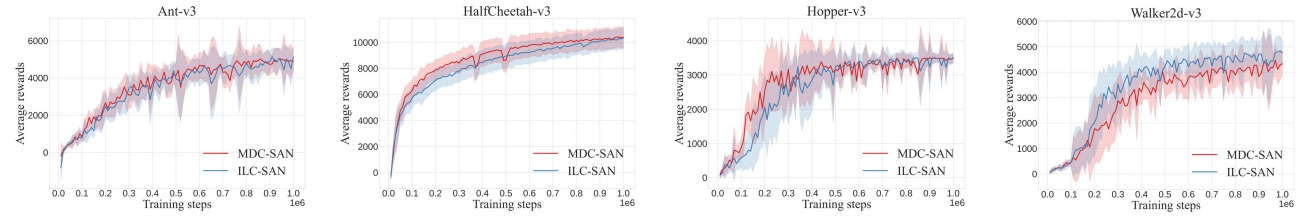

Fig. 6. Comparison of average rewards for MDC-SAN and ILC-SAN using E_{pop} over ten random seeds. The shaded area represents half the value of the standard deviation, and the curves are smoothed for clarity.

TABLE III

MAX AVERAGE REWARDS OVER TEN RANDOM SEEDS FOR VARIOUS DECODING METHODS WITHOUT INTRALAYER CONNECTIONS

Decoding Method	Ant-v3	HalfCheetah-v3	Hopper-v3	Walker2d-v3	APR
D_{fr}	4848±1023	10523±1142	517±1057	4199±1426	71.94%
D_{last} ($\alpha_V = 1.0$)	5433±619	9932±1238	3249±1082	4668±702	**94.95%**
D_{last} ($\alpha_V = 0.75$)	5001±667	9944±1215	3190±1075	5049±426	94.50%
D_{last} ($\alpha_V = 0.5$)	5183±903	9936±1308	3217±1089	4910±654	94.81%
D_{max} ($\alpha_V = 1.0$)	5222±464	9540±1564	3506±158	4609±604	94.59%
D_{max} ($\alpha_V = 0.75$)	5021±910	8775±1661	3453±156	4723±439	92.03%
D_{max} ($\alpha_V = 0.5$)	5268±752	9694±1827	3200±1085	4843±797	94.16%
D_{mean} ($\alpha_V = 1.0$)	5297±544	9995±1169	2183±1712	4691±770	87.03%
D_{mean} ($\alpha_V = 0.75$)	5635±350	9181±1301	2514±1410	4537±1388	88.21%
D_{mean} ($\alpha_V = 0.5$)	5278±809	9394±1351	512±1001	4309±553	71.73%

TABLE IV

MAX AVERAGE REWARDS OVER TEN RANDOM SEEDS FOR DIFFERENT DECODING METHODS WITH INTRALAYER CONNECTIONS

Decoding Method	Ant-v3	HalfCheetah-v3	Hopper-v3	Walker2d-v3	APR
D_{fr}	5279±508	10111±1065	1394±1394	4794±407	82.13%
D_{last} ($\alpha_V = 1.0$)	5653±533	10376±1163	3583±86	4511±462	**98.61%**
D_{last} ($\alpha_V = 0.75$)	4796±751	10304±803	3527±163	4563±835	94.38%
D_{last} ($\alpha_V = 0.5$)	5341±532	10200±1399	3479±128	4955±664	98.24%

Then, we compare the decoding methods (D_{last} and D_{fr}) on performance after adding intralayer connections into ILC-SAN using $E_{\text{pop\_det}}$. As Table IV shows, D_{last} has significant performance advantages over D_{fr}. When $\alpha_V = 1.0$, D_{last} achieves the optimal performance, which is taken as the final configuration of our ILC-SAN.

TABLE V
MAX AVERAGE REWARDS OVER TEN RANDOM SEEDS FOR ABLATION STUDY ON INTRALAYER CONNECTIONS

Intralayer Connections	Ant-v3	HalfCheetah-v3	Hopper-v3	Walker2d-v3	APR
No I_{intra}	5433±619	9932±1238	3249±1082	4668±702	94.95%
I_{self}	5245±769	10168±1465	3512±146	4843±757	97.40%
$I_{lateral}$	5389±747	9384±987	3521±145	4702±728	95.55%
I_{bias}	4935±862	9906±1208	3141±1034	4409±1527	90.56%
$I_{lateral} + I_{bias}$	5549±404	9685±998	3464±151	4851±907	97.34%
$I_{self} + I_{bias}$	5114±729	8431±1410	2645±1439	4699±504	85.76%
$I_{self} + I_{lateral}$	5067±727	9117±2923	3525±133	4788±745	93.90%
I_{intra}	5653±533	10376±1163	3583±86	4511±462	**98.61%**

TABLE VI
MAX AVERAGE REWARDS OVER TEN RANDOM SEEDS FOR COMPARATIVE EXPERIMENTS ON THE BACKBONE SNN

N_{hidden}	Ant-v3	HalfCheetah-v3	Hopper-v3	Walker2d-v3	APR
1	4205±809	7455±549	3014±719	4026±621	78.55%
2	5653±533	10376±1163	3583±86	4511±462	**98.61%**
3	5402±722	10580±1727	3192±951	4623±724	95.73%
4	4760±997	10057±1081	3223±776	4221±1281	89.76%

D. Analysis of Intralayer Connections

In this section, we analyze the impact of intralayer connections. As shown in (13), all the elements on the main diagonal of $\boldsymbol{W}^m_{\text{intra}}$ represent the weight of self-connections and other elements represent the weight of lateral connections. To evaluate the contribution of these two parts as well as $\boldsymbol{b}^m_{\text{intra}}$, we introduce I_{intra}, I_{self}, I_{lateral}, and I_{bias} here. I_{intra} represents our full method which means the current received from the intralayer connection of any neuron in the output population, as well as the intralayer bias current. That is, $\boldsymbol{I}^m_{t+1}$ is formed by I_{intra} of each neuron in the mth output population at time-step $t+1$. I_{self} means only the elements on the main diagonal of $\boldsymbol{W}^m_{\text{intra}}$ are retained, while I_{lateral} means only the elements that are not on the main diagonal are retained. I_{bias} represents $\boldsymbol{W}^m_{\text{intra}}\boldsymbol{S}^{L,m}_t$ is removed and only $\boldsymbol{b}^m_{\text{intra}}$ is retained. Overall, $I_{\text{intra}} = I_{\text{self}} + I_{\text{lateral}} + I_{\text{bias}}$. We further evaluate each part of I_{intra} on all four tasks while keeping the input coding methods fixed to $E_{\text{pop\_det}}$ and the decoding methods fixed to D_{last} with $\alpha_V = 1.0$.

As shown in Table V, we conduct the ablation study on intralayer connections, which proves that each component is indispensable and complementary in performance. Using the complete I_{intra}, our model achieves the best performance, which increases the APR by 3.66% compared with that without I_{intra}. This result verifies that the spatial-temporal information brought by intralayer connections effectively improves the representation capacity of the network and led to better action representation. For individual components, I_{self} can bring the greatest performance improvement as it enriches the dynamics of spiking neurons. However, when I_{self} is combined with either I_{lateral} or I_{bias}, the performance of the model degrades greatly. These results show that I_{lateral} and I_{bias} have the strongest complementary effect and cannot be used alone, which is also supported by the experimental results of $I_{\text{lateral}} + I_{\text{bias}}$. Therefore, the performance improvement brought by intralayer connections mainly comes from two parts, namely, the current based on the neuron's dynamics I_{self} and the intralayer current from outside the neuron $I_{\text{lateral}} + I_{\text{bias}}$.

E. Analysis of the Backbone SNN

In previous experiments, we adopt the mainstream setting of TD3 algorithms, which involves two hidden layers with 256 nodes in the backbone SNN [22], [23]. The hidden layers are located between the input layer and the output layer. To evaluate the effects of network depth, we have conducted an additional experiment on the number of hidden layers in the backbone SNN (N_{hidden}). In this experiment, the number of nodes in each hidden layer is set to 256. As shown in Table VI, the performance of ILC-SANs reaches its peak when $N_{\text{hidden}} = 2$. In general, the larger network could obtain higher performance. However, in RL tasks, the larger network could bring overfitting due to the sensitive learning targets of the temporal-difference loss and the unstable training data [52], [53].

F. Analysis of Energy Consumption

Since low energy consumption is the main advantage of SNNs, we estimate the energy consumption of different networks in this section. Due to the floating-point input, the first FC layer of MDC-SAN cannot be directly deployed to the neuromorphic chips. In addition, due to the high proportion of the computation of the first FC layer in the entire network, considering the simulation time T, the energy consumption of MDC-SAN will be even higher than that of the original DAN. Therefore, we only calculate the theoretical energy consumption of DAN, PopSAN and our ILC-SAN.

Taking the hardware selection in [54] as a reference, we employ the FPGA of Intel Stratix 10 TX and the neuromorphic chip of ROLLS [55] for estimation. As reported in [54], the FPGA operates at a cost of 12.5 pJ per FLOP (floating-point operation) and the ROLLS consumes 77 fJ per SOP (synaptic operation [4]). For a FC layer, assuming that

TABLE VII
AVERAGE FIRING RATE OF THE NEURONAL LAYER BEFORE EACH FC LAYER FOR POPSAN AND ILC-SAN

Task	Actor Network	FC1	FC2	FC3	Group FC	Intra FC
Ant-v3	PopSAN	3.0%	38.8%	58.3%	83.9%	–
	ILC-SAN	2.8%	35.5%	49.0%	41.3%	31.5%
HalfCheetah-v3	PopSAN	19.7%	33.5%	51.0%	69.4%	–
	ILC-SAN	18.7%	25.9%	39.1%	40.0%	30.7%
Hopper-v3	PopSAN	29.8%	68.7%	56.7%	45.9%	–
	ILC-SAN	16.5%	20.8%	36.9%	42.1%	32.8%
Walker2d-v3	PopSAN	18.7%	51.2%	67.8%	52.8%	–
	ILC-SAN	15.9%	27.4%	47.5%	47.1%	37.0%

TABLE VIII
NUMBER OF OPERATIONS OF EACH FC LAYER FOR POPSAN AND ILC-SAN

Task	Actor Network	FC1	FC2	FC3	Group FC	Intra FC	Total
Ant-v3	PopSAN	42624 SOP	127140 SOP	59699 SOP	336 FLOP	–	229463 SOP+336 FLOP
	ILC-SAN	40351 SOP	116326 SOP	50176 SOP	165 SOP	1260 SOP	208278 SOP
HalfCheetah-v3	PopSAN	42867 SOP	109773 SOP	39168 SOP	208 FLOP	–	191808 SOP+208 FLOP
	ILC-SAN	40691 SOP	84869 SOP	30029 SOP	120 SOP	921 SOP	156630 SOP
Hopper-v3	PopSAN	41958 SOP	225116 SOP	21773 SOP	69 FLOP	–	288847 SOP+69 FLOP
	ILC-SAN	23232 SOP	68157 SOP	14170 SOP	63 SOP	492 SOP	106114 SOP
Walker2d-v3	PopSAN	40604 SOP	167772 SOP	52070 SOP	158 FLOP	–	260447 SOP+158 FLOP
	ILC-SAN	34598 SOP	89784 SOP	36480 SOP	141 SOP	1110 SOP	162114 SOP

TABLE IX
ENERGY CONSUMPTION OF DIFFERENT TASKS PER INFERENCE FOR POPSAN AND ILC-SAN. THE UNIT OF ENERGY IS NANOJOULE (nJ)

Task	DAN	PopSAN	ILC-SAN
Ant-v3	1200.0	18.7	16.0
HalfCheetah-v3	892.8	15.5	12.1
Hopper-v3	864.0	22.6	8.2
Walker2d-v3	892.8	20.8	12.5

the size of the input sample and the output sample is $\mathrm{Dim}_{\mathrm{in}}$ and $\mathrm{Dim}_{\mathrm{out}}$, the number of operations required is $\mathrm{Dim}_{\mathrm{in}}\mathrm{Dim}_{\mathrm{out}}$ FLOPs for ANN and $\mathrm{Tfr}_{\mathrm{in}}\mathrm{Dim}_{\mathrm{in}}\mathrm{Dim}_{\mathrm{out}}$ SOPs for SNN, where T is the simulation time and $\mathrm{fr}_{\mathrm{in}}$ is the average firing rate of the previous neuronal layer. Note that we use ($\mathrm{Dim}_{\mathrm{in}}$, $\mathrm{Dim}_{\mathrm{out}}$) to represent the FC layer later.

First, we need to determine the number of operations required for each task (FLOP for floating-point input and SOP for spike input). For DAN, the number of operations is equal to $256(256 + N + M)$, which goes as follows: 96 000 FLOP, 71 424 FLOP, 69 120 FLOP, 71 424 FLOP, respectively, for Ant-v3, HalfCheetah-v3, Hopper-v3, and Walker2d-v3. For PopSAN, the first three FC layers can be expressed as FC1 ($N \cdot P_{\mathrm{in}}$, 256), FC2 (256, 256), FC3 (256, $M \cdot P_{\mathrm{out}}$), and the population decoder, which consists of a group of M FC layers (P_{out}, 1), can be expressed as Group FC. For our ILC-SAN, we have an additional calculation of intralayer connections, which consists of a group of M FC layers (P_{out}, P_{out}) and can be expressed as Intra FC. As shown in Fig. 2, the learnable layer represents the Group FC, and the intralayer connections represent the Intra FC. As shown in Table VII, we run all the trained models for one episode to obtain the average firing rate of each neuronal layer. Although the neuronal layer before Group FC is the same as that before Intra FC, the average firing rate is different due to the time-steps involved in the calculation. It can be seen that compared with PopSAN, the firing rate of spiking neurons in each layer of ILC-SAN is significantly reduced.

Then, we estimate the number of operations for each FC layer (Table VIII), based on the formula of SOP for the FC layer mentioned in this section. Since the input of Group FC in PopSAN is the floating-point firing rate, it needs to be calculated on traditional hardware, using FLOP to count the number of operations.

Concerning each platform's energy efficiency, we calculate the energy consumption required by DAN, PopSAN and ILC-SAN per inference. As Table IX shows, the energy consumption of DAN is more than 71 times greater than the energy consumed by ILC-SAN across all four tasks. In addition, compared with PopSAN, ILC-SAN can reduce 35.1% of the energy consumption per inference on average.

V. CONCLUSION

In this article, we present ILC-SAN, a fully ILC-SAN for the RL. With the help of membrane voltage coding, we solve the deployment problem of current spike-based RL methods, and provide a better alternative for the representation of floating-point values, such as continuous action and Q value. In addition, we add intralayer connections to the output population, and analyze the effects of self-connections and lateral connections, both of which have improved the performance of the model. In terms of energy efficiency, our model

effectively reduces the firing rate of spiking neurons for each layer, thus greatly reducing the energy consumption running on neuromorphic hardware. Our model is evaluated on eight continuous control tasks from OpenAI gym, and outperforms the start-of-the-art methods. We provide a comparison of our method with the corresponding deep network in terms of performance, stability, and energy consumption, comprehensively demonstrating the superiority of our method.

As an energy-efficient alternative for real-time robot control tasks, our ILC-SAN has great application potential, which could be taken as the foundation for the follow-up work. For example, one of the future directions is to combine ILC-SAN with neuromorphic sensors to achieve robot control in real-world scenes, which requires addressing the limitations of vector input in our method and extending it to higher order tensor inputs such as images or spatiotemporal signals. In the future, more biologically plausible rules could be applied to SNNs to improve performance and energy efficiency for different tasks. We believe that neuroscience will be an important source of inspiration for AI.

ACKNOWLEDGMENT

Computing support was provided by Pengcheng Cloudbrain.

REFERENCES

[1] D. Kuzum, S. Yu, and H.-S. P. Wong, "Synaptic electronics: Materials, devices and applications," *Nanotechnology*, vol. 24, no. 38, Sep. 2013, Art. no. 382001.

[2] G. Indiveri and S.-C. Liu, "Memory and information processing in neuromorphic systems," *Proc. IEEE*, vol. 103, no. 8, pp. 1379–1397, Aug. 2015.

[3] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, Nov. 2019.

[4] P. A. Merolla et al., "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[5] M. Davies et al., "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[6] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

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

[8] W. Fang, Z. Yu, Y. Chen, T. Masquelier, T. Huang, and Y. Tian, "Incorporating learnable membrane time constant to enhance learning of spiking neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2641–2651.

[9] W. Fang, Z. Yu, Y. Chen, T. Huang, T. Masquelier, and Y. Tian, "Deep residual learning in spiking neural networks," in *Proc. 35th Conf. Neural Inf. Process. Syst.*, 2021, pp. 1–18.

[10] A. Mahadevuni and P. Li, "Navigating mobile robots to target in near shortest time using reinforcement learning with spiking neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 2243–2250.

[11] Z. Bing et al., "End to end learning of spiking neural network based on R-STDP for a lane keeping vehicle," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 4725–4732.

[12] Z. Bing, C. Meschede, F. Röhrbein, K. Huang, and A. C. Knoll, "A survey of robotics control based on learning-inspired spiking neural networks," *Frontiers Neurorobot.*, vol. 12, p. 35, Jul. 2018.

[13] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[14] D. Silver et al., "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, Dec. 2018.

[15] N. Frémaux, H. Sprekeler, and W. Gerstner, "Reinforcement learning using a continuous time actor-critic framework with spiking neurons," *PLoS Comput. Biol.*, vol. 9, no. 4, Apr. 2013, Art. no. e1003024.

[16] N. Frémaux and W. Gerstner, "Neuromodulated spike-timing-dependent plasticity, and theory of three-factor learning rules," *Frontiers Neural Circuits*, vol. 9, p. 85, Jan. 2016.

[17] G. Bellec et al., "A solution to the learning dilemma for recurrent networks of spiking neurons," *Nature Commun.*, vol. 11, no. 1, pp. 1–15, Jul. 2020.

[18] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers Neurosci.*, vol. 10, p. 508, Nov. 2016.

[19] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. 4th Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds. San Juan, Puerto Rico, May 2016, pp. 1–14.

[20] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[22] G. Tang, N. Kumar, R. Yoo, and K. Michmizos, "Deep reinforcement learning with population-coded spiking neural network for continuous control," in *Proc. Conf. Robot Learn.*, 2021, pp. 2016–2029.

[23] D. Zhang, T. Zhang, S. Jia, and B. Xu, "Multiscale dynamic coding improved spiking actor network for reinforcement learning," in *Proc. 36th AAAI Conf. Artif. Intell.*, 2022, pp. 59–67.

[24] S. S. Bidaye, T. Bockemühl, and A. Büschges, "Six-legged walking in insects: How CPGs, peripheral feedback, and descending signals generate coordinated and adaptive motor rhythms," *J. Neurophysiol.*, vol. 119, no. 2, pp. 459–475, Feb. 2018.

[25] J. Satel, "Mechanisms of inhibition of return: Brain, behavior, and computational modeling," Dalhousie Univ., Halifax, NS, Canada, Tech. Rep. 947930103, 2013.

[26] A. Renner, M. Evanusa, and Y. Sandamirskaya, "Event-based attention and tracking on neuromorphic hardware," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2019, pp. 1709–1716.

[27] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[28] W. Gerstner, M. Lehmann, V. Liakoni, D. Corneil, and J. Brea, "Eligibility traces and plasticity on behavioral time scales: Experimental support of NeoHebbian three-factor learning rules," *Frontiers Neural Circuits*, vol. 12, p. 53, Jul. 2018.

[29] D. Vlasov, R. Rybka, A. Sboev, A. Serenko, A. Minnekhanov, and V. Demin, "Reinforcement learning in a spiking neural network with memristive plasticity," in *Proc. 6th Sci. School Dyn. Complex Netw. their Appl. (DCNA)*, Sep. 2022, pp. 300–302.

[30] B. Rueckauer, I.-A. Lungu, Y. Hu, M. Pfeiffer, and S.-C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers Neurosci.*, vol. 11, p. 682, Dec. 2017.

[31] D. Patel, H. Hazan, D. J. Saunders, H. T. Siegelmann, and R. Kozma, "Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to Atari breakout game," *Neural Netw.*, vol. 120, pp. 108–115, Dec. 2019.

[32] W. Tan, D. Patel, and R. Kozma, "Strategy and benchmark for converting deep Q-networks to event-driven spiking neural networks," 2020, *arXiv:2009.14456*.

[33] G. Tang, N. Kumar, and K. P. Michmizos, "Reinforcement co-learning of deep and spiking neural networks for energy-efficient mapless navigation with neuromorphic hardware," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 6090–6097.

[34] G. Liu, W. Deng, X. Xie, L. Huang, and H. Tang, "Human-level control through directly trained deep spiking Q-networks," *IEEE Trans. Cybern.*, vol. 53, no. 11, pp. 7187–7198, Sep. 2022.

[35] Y. Sun, Y. Zeng, and Y. Li, "Solving the spike feature information vanishing problem in spiking deep Q network with potential based normalization," 2022, *arXiv:2206.03654*.

[36] M. Akl, D. Ergene, F. Walter, and A. Knoll, "Toward robust and scalable deep spiking reinforcement learning," *Frontiers Neurorobotics*, vol. 16, p. 314, Jan. 2023.

[37] L. Qin, R. Yan, and H. Tang, "A low latency adaptive coding spiking framework for deep reinforcement learning," 2022, *arXiv:2211.11760*.
[38] B. Cramer, Y. Stradmann, J. Schemmel, and F. Zenke, "The Heidelberg spiking data sets for the systematic evaluation of spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2744–2757, Jul. 2022.
[39] C. Lee, A. K. Kosta, A. Z. Zhu, K. Chaney, K. Daniilidis, and K. Roy, "Spike-FlowNet: Event-based optical flow estimation with energy-efficient hybrid neural networks," in *Proc. Eur. Conf. Comput. Vis.*, Glasgow, U.K. Cham, Switzerland: Springer, Aug. 2020, pp. 366–382.
[40] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "STDP-based spiking deep convolutional neural networks for object recognition," *Neural Netw.*, vol. 99, pp. 56–67, Mar. 2018.
[41] W. Fang et al. (2020). *Spikingjelly*. Accessed: Dec. 1, 2021. [Online]. Available: https://github.com/fangwei123456/spikingjelly
[42] I. Polykretis, G. Tang, and K. P. Michmizos, "An astrocyte-modulated neuromorphic central pattern generator for hexapod robot locomotion on Intel's loihi," in *Proc. Int. Conf. Neuromorphic Syst.*, Jul. 2020, pp. 1–9.
[43] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 795–805.
[44] B. Yin, F. Corradi, and S. M. Bohté, "Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks," *Nature Mach. Intell.*, vol. 3, no. 10, pp. 905–913, Oct. 2021.
[45] A. Rao, P. Plank, A. Wild, and W. Maass, "A long short-term memory for AI applications in spike-based neuromorphic hardware," *Nature Mach. Intell.*, vol. 4, no. 5, pp. 467–479, May 2022.
[46] F. Ratliff, H. K. Hartline, and D. Lange, "The dynamics of lateral inhibition in the compound eye of limulus. I," in *Studies on Excitation and Inhibition in the Retina: A Collection of Papers From the Laboratories*, H. K. Hartline, Ed. New York, NY, USA: Rockefeller Univ. Press, 1974, p. 463.
[47] P. U. Diehl and M. Cook, "Unsupervised learning of digit recognition using spike-timing-dependent plasticity," *Frontiers Comput. Neurosci.*, vol. 9, p. 99, Aug. 2015.
[48] X. Cheng, Y. Hao, J. Xu, and B. Xu, "LISNN: Improving spiking neural networks with lateral interactions for robust object recognition," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1519–1525.
[49] W. Zhang and P. Li, "Spike-train level backpropagation for training deep recurrent spiking neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, pp. 7800–7811.
[50] N. Brunel and M. C. W. van Rossum, "Lapicque's 1907 paper: From frogs to integrate-and-fire," *Biol. Cybern.*, vol. 97, nos. 5–6, pp. 337–339, Dec. 2007.
[51] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge, U.K.: Cambridge Univ. Press, 2014.
[52] K. Ota, D. K. Jha, and A. Kanezaki, "Training larger networks for deep reinforcement learning," 2021, *arXiv:2102.07920*.
[53] A. Kumar, R. Agarwal, D. Ghosh, and S. Levine, "Implicit under-parameterization inhibits data-efficient deep reinforcement learning," 2020, *arXiv:2010.14498*.
[54] Y. Hu, H. Tang, and G. Pan, "Spiking deep residual networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 5200–5205, 2023.
[55] N. Qiao et al., "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Frontiers Neurosci.*, vol. 9, p. 141, Apr. 2015.

Ding Chen is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China.

His current research interests include spiking neural networks, reinforcement learning, and brain-inspired computing.

Peixi Peng received the Ph.D. degree from Peking University, Beijing, China, in 2017.

He is currently an Associate Researcher with the School of Computer Science, Peking University, and also an Assistant Researcher with the Artificial Intelligence Research Center, Peng Cheng Laboratory, Shenzhen, China. He is the author or coauthor of more than 30 technical articles in refereed journals, such as IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), PR and conferences, such as CVPR/ECCV/IJCAI/ACMMM/AAAI. His research interests include computer vision, multimedia big data, and reinforcement learning.

Tiejun Huang (Senior Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from the Wuhan University of Technology, Wuhan, China, in 1992 and 1995, respectively, and the Ph.D. degree in pattern recognition and image analysis from the Huazhong (Central China) University of Science and Technology, Wuhan, in 1998.

He is currently a Professor with the School of Computer Science, Peking University, Beijing, China, and the Director of the Beijing Academy for Artificial Intelligence, Beijing. His research areas include visual information processing and neuromorphic computing. He published more than 300 peer-reviewed papers on leading journals and conferences and also the coeditor of four ISO/IEC standards, five national standards, and four IEEE standards. He holds more than 100 granted patents.

Prof. Huang is a fellow of CAAI, CCF, and CSIG. He received National Award for Science and Technology of China (Tier-2) for three times (2010, 2012, and 2017). He the Vice Chair of the China National General Group on AI Standardization.

Yonghong Tian (Affiliate, IEEE) is currently the Dean of the School of Electronics and Computer Engineering, a Boya Distinguished Professor with the School of Computer Science, Peking University, Beijing, China, and the Deputy Director of the Artificial Intelligence Research Department, Peng Cheng Laboratory, Shenzhen, China. He is the author or coauthor of over 300 technical articles in refereed journals and conferences. His research interests include neuromorphic vision, distributed machine learning, and multimedia big data.

Prof. Tian is a Senior Member of CIE and CCF and a member of ACM. He was a recipient of the Chinese National Science Foundation for Distinguished Young Scholars in 2018, two National Science and Technology Awards, and three ministerial level awards in China, the 2015 EURASIP Best Paper Award for Journal on Image and Video Processing, and the Best Paper Award of IEEE BigMM 2018 and the 2022 IEEE SA Standards Medallion and SA Emerging Technology Award. He was/is an Associate Editor of IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (TCSVT), from January 2018 to December 2021, IEEE TRANSACTIONS ON MULTIMEDIA (TMM), from August 2014 to August 2018, IEEE *Multimedia Magazine*, from January 2018 to August 2022, and IEEE ACCESS from January 2017 to December 2021.