Human-Level Control Through Directly Trained Deep Spiking Q-Networks

Guisong Liu, Wenjie Deng, Xiurui Xie, Li Huang, and Huajin Tang, *Senior Member, IEEE*

***Abstract*—As the third-generation neural networks, spiking neural networks (SNNs) have great potential on neuromorphic hardware because of their high energy efficiency. However, deep spiking reinforcement learning (DSRL), that is, the reinforcement learning (RL) based on SNNs, is still in its preliminary stage due to the binary output and the nondifferentiable property of the spiking function. To address these issues, we propose a deep spiking Q-network (DSQN) in this article. Specifically, we propose a directly trained DSRL architecture based on the leaky integrate-and-fire (LIF) neurons and deep Q-network (DQN). Then, we adapt a direct spiking learning algorithm for the DSQN. We further demonstrate the advantages of using LIF neurons in DSQN theoretically. Comprehensive experiments have been conducted on 17 top-performing Atari games to compare our method with the state-of-the-art conversion method. The experimental results demonstrate the superiority of our method in terms of performance, stability, generalization and energy efficiency. To the best of our knowledge, our work is the first one to achieve state-of-the-art performance on multiple Atari games with the directly trained SNN.**

***Index Terms*—Atari games, deep reinforcement learning (DRL), directly training, spiking neural networks (SNNs).**

I. Introduction

In RECENT years, spiking neural networks (SNNs) have attracted widespread interest because of their low power consumption [1] on neuromorphic hardware. In contrast to artificial neural networks (ANNs) that use continuous values to represent information, SNNs use discrete spikes to represent information, which is inspired by the behavior of biological neurons in both spatiotemporal dynamics and communication methods. This makes SNNs have been implemented successfully on dedicated neuromorphic hardware, such as SpiNNaker at Manchester, U.K. [2], IBM's TrueNorth [3], and Intel's Loihi [4], which are reported to be 1000 times more energy efficient than conventional chips. Furthermore, recent studies demonstrated competitive performance of SNNs compared with ANNs on image classification [5]; object recognition [6], [7]; speech recognition [8], [9]; and other fields [10], [11], [12], [13], [14], [15].

The present work focuses on combining SNNs with deep reinforcement learning (DRL), that is, deep spiking reinforcement learning (DSRL), on Atari games. Compared to image classification, DSRL on Atari games involve additional complexity due to the pixel image as input and the partial observability of the environment. The development of DSRL lags behind DRL, while DRL has made tremendous successes, achieved and even surpassed human-level performance in many reinforcement learning (RL) tasks [16], [17], [18], [19], [20], [21]. The main reason is that, training SNNs is a challenge, as the event-driven spiking activities are discrete and nondifferentiable. In addition, the activities of spiking neurons are propagated not only in the spatial domain layer by layer but also along the temporal domain [22]. It makes the training of SNNs in RL more difficult.

To avoid the difficulty of training SNNs, Pérez-Carrasco *et al.* [23] proposed an alternative approach of converting ANNs to SNNs. Patel *et al.* [24] extended existing conversion methods [25], [26], [27] to the domain of deep Q-learning, and improved the robustness of SNNs in input image occlusion. After that, Tan *et al.* [28] proposed a more robust and effective conversion method which converts a pretrained deep Q-network (DQN) to SNN, and achieved state-of-the-art performance on multiple Atari games. Nevertheless, the existing conversion methods rely on pretrained ANNs heavily. Besides, they require very long simulation time window (at least hundreds of timesteps) for convergence, which is demanding in terms of computation.

To maintain the energy-efficiency advantage of SNNs, the direct training methods have been widely studied recently [29], [30], [31], [32], [33], [34]. For instance, Zheng *et al.* [35] proposed a threshold-dependent batch normalization method to train deep SNNs directly. It first explored the directly trained deep SNNs with high performance on ImageNet. Besides, surrogate gradient learning (SGL) is proposed to address the nondifferentiable issue in spiking function by designing a surrogate gradient function that approximates the

Manuscript received 21 April 2022; revised 5 July 2022; accepted 7 August 2022. Date of publication 5 September 2022; date of current version 17 October 2023. This work was supported in part by the Natural Science Foundation of Guangdong Province under Grant 2021A1515011866; in part by the Sichuan Province under Grant 2021YFG0018 and Grant 2022YFG0314; and in part by the Social Foundation of Zhongshan Sci-Tech Institute under Grant 420S36. This article was recommended by Associate Editor B. Ribeiro. *(Corresponding author: Xiurui Xie.)*

Guisong Liu is with the School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu 611130, China, and also with the School of Computer Science, Zhongshan Institute, University of Electronic Science and Technology of China, Zhongshan 528400, China. (e-mail: gliu@swufe.edu.cn).

Wenjie Deng and Xiurui Xie are with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: xiexiurui@uestc.edu.cn).

Li Huang is with the School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu 611130, China.

Huajin Tang is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China, and also with the Institute of Artificial Intelligence, Zhejiang Laboratory, Hangzhou 311122, China (e-mail: htang@zju.edu.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2022.3198259.

Digital Object Identifier 10.1109/TCYB.2022.3198259

spiking back-propagation behavior [36]. The flexibility and efficiency make it more promising in overcoming the training challenges of SNNs compared to existing conversion methods. However, most of the existing direct training methods only focus on image classification but not RL tasks.

Besides the training methods, there is another challenge in DSRL, that is, how to distinguish optimal action from highly similar Q-values [28]. It has been proved that, in the process of optimizing a ANN for image classification, the value of the correct class is always significantly higher than the wrong classes. In contrast to image classificatoin, the Q-values of different actions are often very similar even for a well-trained network in RL [28]. Actually, the confusing Q-value issue is not really a problem in RL, because of the continues information representation of traditional ANNs. But in DSRL, how to make the discrete spikes outputed by SNNs represent these highly similar Q-values well is a challenging problem.

To address these issues, we propose a deep spiking Q-network (DSQN) in this article. Specifically, we use leaky integrate-and-fire (LIF) neurons in DSQN with firing rate coding and appropriate but extremely short simulation time window (64 timesteps) to address the issue of the confusing Q-values. In addition, we adapt a spiking SGL algorithm to achieve the direct training for DSQN. Whereafter, we demonstrate the advantages of using LIF neurons in DSQN theoretically.

In the end, comprehensive experiments have been conducted on 17 top-performing Atari games. The experimental results show that DSQN completely surpasses the conversion-based SNN [28] in terms of performance, stability, generalization, and energy efficiency. At the same time, DSQN reaches the same performance level of the vanilla DQN [17]. Our method provides another way to achieve high performance on Atari games with SNNs while avoiding the limitations of conversion methods. To the best of our knowledge, our work is the first one to achieve state-of-the-art performance on multiple Atari games with the directly trained SNN. It paves the way for further research on solving RL problems with directly trained SNNs.

II. Related Works

Mnih *et al.* [17] introduced deep neural networks into the Q-Learning, a traditional RL algorithm, and formed the DQN algorithm, creating the field of DRL. They used convolutional neural networks to approximate the Q function of Q-learning, as the result, DQN achieved and even surpassed human-level on 49 Atari games. After that, [24] is the first work to introduce spiking conversion methods to the domain of deep Q-learning. They demonstrated that both shallow and deep ReLU networks can be converted to SNNs without performance degradation on Atari game Breakout. Then, they showed that the converted SNN is more robust to input perturbations than the original neural network. However, it only focuses on improving the robustness rather than the performance of SNNs on Atari games. To further improve the performance, Tan *et al.* [28] proposed a more effective conversion method based on the more accurate approximation of the spiking firing rates. It reduced the conversion error based on a pretrained DQN, and achieved state-of-the-art performance on multiple Atari games. Although these conversion-based researchers have led to further development of DSRL, there are still some limitations that remain unsolved, for example, the heavy dependence on pretrained ANNs and demand for very long simulation time window. Other related works are [37], [38], [39], [40], [41].

In contrast to the existing methods, our method is directly trained by spiking SGL on LIF neurons. This makes it more flexible and reduces the training cost because it has no dependence on pretrained ANNs and only requires extremely short simulation time window.

III. Methods

In this section, we describe the DSQN in detail, including the directly trained DSRL architecture, direct learning method, and theoretically demonstration of the advantages of using LIF neurons in DSQN.

A. Architecture of DSQN

The DSQNs consist of three convolution layers and two fully connected layers. We use LIF neurons in DSQN to from a directly trained DSRL architecture. With firing rate encoding and appropriate length of simulation time window, this architecture can achieve sufficient accuracy to handle the confusing Q-value issue mentioned in Section I, while maintaining the energy-efficiency advantage of SNNs with direct learning method. Fig. 1 concretely shows the architecture and environmental interaction of DSQN.

For a network with L layers, let $V^{l,t}$ denote the membrane potential of neurons in layer l at simulation time t. The LIF neuron integrates inputs until the membrane potential exceeds a threshold $V_{\text{th}} \in \mathbb{R}^+$, and a spike correspondingly generated. Once the spike is generated, the membrane potential will be reset by *hard reset* or *soft reset*. Note that the *hard reset* means resetting the membrane potential back to a baseline, typically 0. The *soft reset* means subtracting the threshold V_{th} from the membrane potential when it exceeds the threshold.

The neuronal dynamics of the LIF neurons in layer $l \in \{1, \ldots, L-1\}$ at simulation time t could be described as follows:

$$U^{l,t} = V^{l,t-1} + \frac{1}{\tau_{\text{m}}}\left(W^l S^{l-1,t} - V^{l,t-1} + V_{\text{r}}\right) \tag{1}$$

$$V^{l,t} = \begin{cases} U^{l,t}\left(1 - S^{l,t}\right) + V_{\text{r}} S^{l,t}, & \text{hard reset} \\ U^{l,t} - V_{\text{th}} S^{l,t}, & \text{soft reset.} \end{cases} \tag{2}$$

Equation (1) describes the subthreshold membrane potential of neurons, that is, when the membrane potential does not exceed the threshold potential V_{th}, in which τ_{m} denotes the membrane time constant, W^l denotes the learnable weights of the neurons in layer l, and V_{r} denotes the initial membrane potential. Equation (2) describes the membrane potential of neurons when reached V_{th}.

The output of the LIF neurons in layer $l \in \{1, \ldots, L-1\}$ at simulation time t could be expressed as follows:

$$S^{l,t} = \Theta\left(U^{l,t} - V_{\text{th}}\right) \tag{3}$$

$$\Theta(x) = \begin{cases} 1, & x \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where $\Theta(x)$ is the spiking function of neurons.

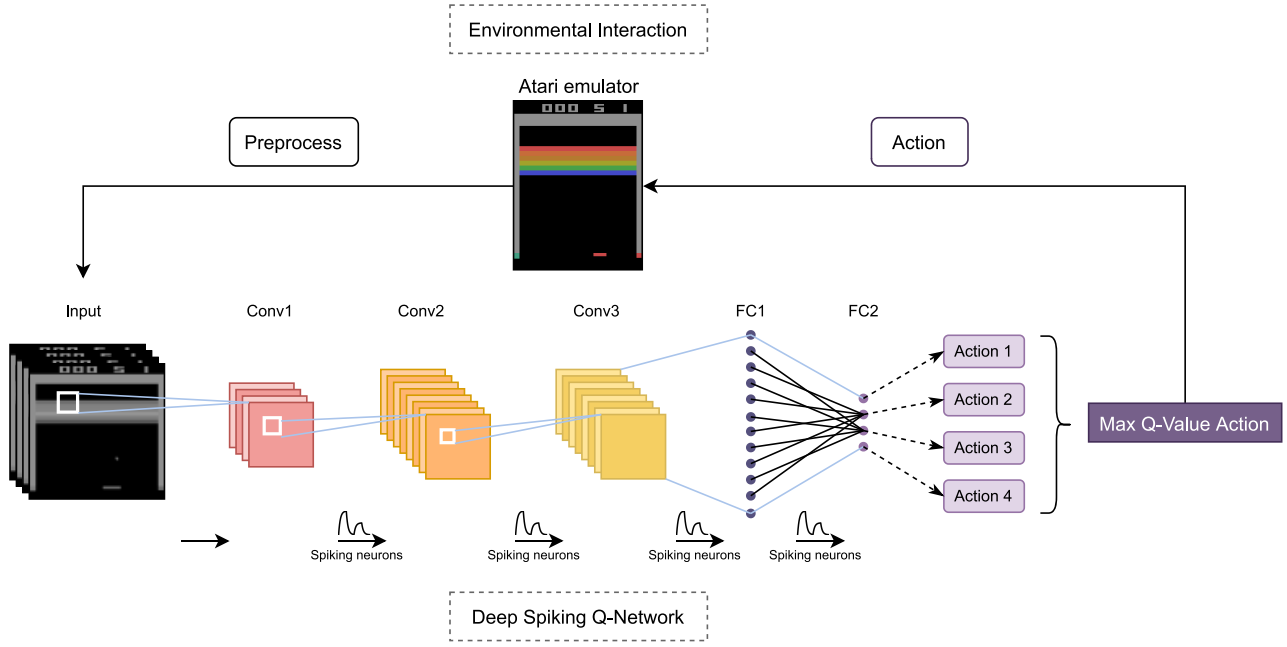


Fig. 1. Architecture and environmental interaction of DSQN which consists of three convolution layers and two fully connected layers.

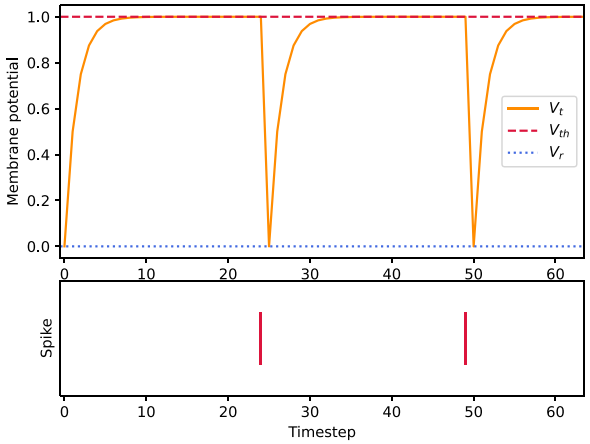


Fig. 2. Neuronal dynamics of the LIF neuron which resets by *hard reset* when the input is constant at 1 for the length of simulation time window $t = 64$, the membrane time constant $\tau_{\mathrm{m}} = 2$, the initial membrane potential $V_{\mathrm{r}} = 0$, and the threshold potential $V_{th} = 1$.

The neuronal dynamics of the LIF neuron which is reset by *hard reset* is illustrated in Fig. 2. As the simulation time passes, the LIF neuron integrates the input current, and its membrane potential continues to rise according to (1). Until the membrane potential reaches the membrane potential threshold V_{th}, a spike is emitted by the LIF neuron according to (3) and (4). Then, the membrane potential is reset according to (2).

For the neurons in the final layer L, their output O^L could be described by

$$O^L = W^L \frac{1}{t} \sum_{t'=1}^{t} S^{L-1,t'} \tag{5}$$

where W^L is the learnable weights of the neurons in the final layer. At the same time, O^L denotes the output Q-values of DSQN.

As an RL agent, during the interacting with the environment, DSQN is trained by deep Q-learning algorithm [17]. Thus, the deep spiking Q-learning algorithm uses the following loss function:

$$\mathcal{L}(W) = \mathbb{E}_{(s,a,r,s')\sim U(D)}\left[\left(y_{(r,s')} - Q(s,a;W)\right)^2\right] \tag{6}$$

with

$$y_{(r,s')} = r + \gamma \max_{a'} Q\left(s', a'; W^-\right) \tag{7}$$

where $Q(s,a;W)$ denotes the approximate Q-value function parameterized by DSQN, and W and W^- denote the weights of DSQN at the current and history, respectively. $(s,a,r,s') \sim U(D)$ denotes the minibatches drawn uniformly at random from the experience replay memory D, and γ is the reward discount factor.

According to (5), the loss function could also be simply expressed by

$$\mathcal{L}(W) = \mathbb{E}\left[(y - O^L)^2\right]. \tag{8}$$

B. Direct Learning Method for DSQN

In this section, we only consider the case of LIF neurons which are reset by *hard reset*.

According to (5) and (8) and the chain rule, for the final layer L, we have

$$\frac{\partial \mathcal{L}}{\partial W^L} = \frac{2}{t}\mathbb{E}\left[O^L - y\right] \sum_{t'=1}^{t} S^{L-1,t'} \tag{9}$$

and for the hidden layers $l \in \{1, \ldots, L-1\}$, according to (1), (3), and (8), we have

$$\frac{\partial \mathcal{L}}{\partial W^l} = \sum_{t'=1}^{t} \frac{\partial \mathcal{L}}{\partial S^{l,t'}} \frac{\partial S^{l,t'}}{\partial U^{l,t'}} \frac{\partial U^{l,t'}}{\partial W^l}. \tag{10}$$

The first factor in (10) could be derived as

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial S^{l,t'}} &= \frac{\partial \mathcal{L}}{\partial S^{l+1,t'}} \frac{\partial S^{l+1,t'}}{\partial U^{l+1,t'}} \frac{\partial U^{l+1,t'}}{\partial S^{l,t'}} \\ &= \frac{\partial \mathcal{L}}{\partial S^{l+1,t'}} \frac{\partial \Theta\left(U^{l+1,t'} - V_{\text{th}}\right)}{\partial U^{l+1,t'}} \frac{W^{l+1}}{\tau_{\text{m}}}. \end{aligned} \tag{11}$$

Specially, when $l = L-1$, (11) is different

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial S^{l,t'}} &= \frac{\partial \mathcal{L}}{\partial O^L} \frac{\partial O^L}{\partial S^{l,t'}} \\ &= \frac{2}{t} W^l \mathbb{E}\left[O^L - y\right]. \end{aligned} \tag{12}$$

The spiking function $\Theta(x)$ is nondifferentiable, this leads to that (10) which is also nondifferentiable. To address this issue, we use a differentiable surrogate gradient function [36] $\sigma(x)$ to approximate $\Theta(x)$. Correspondingly, (11) can be rewritten as

$$\frac{\partial \mathcal{L}}{\partial S^{l,t'}} = \frac{\partial \mathcal{L}}{\partial S^{l+1,t'}} \frac{\partial \sigma\left(U^{l+1,t'} - V_{\text{th}}\right)}{\partial U^{l+1,t'}} \frac{W^{l+1}}{\tau_{\text{m}}}. \tag{13}$$

The third factor in (10) could be derived as

$$\begin{aligned} \frac{\partial U^{l,t'}}{\partial W^l} &= \frac{\partial U^{l,t'}}{\partial V^{l,t'-1}} \frac{\partial V^{l,t'-1}}{\partial W^l} + \frac{1}{\tau_{\text{m}}} S^{l-1,t'} \\ &= M^{l,t'} \frac{\partial U^{l,t'-1}}{\partial W^l} + \frac{S^{l-1,t'}}{\tau_{\text{m}}} \end{aligned} \tag{14}$$

where

$$M^{l,t} = \left(1 - \frac{1}{\tau_{\text{m}}}\right)\left[1 - S^{l,t-1} + \frac{\partial \sigma\left(U^{l,t-1} - V_{\text{th}}\right)}{\partial U^{l,t-1}}\left(V_{\text{r}} - U^{l,t-1}\right)\right]. \tag{15}$$

Then, we continuous derive gradients across time dimension. When $t = 1$, we have

$$\frac{\partial U^{l,t'}}{\partial W^l} = \frac{S^{l-1,1}}{\tau_{\text{m}}} \tag{16}$$

when $t > 1$, we obtain

$$\frac{\partial U^{l,t'}}{\partial W^l} = \prod_{\tau=1}^{t'} M^{l,\tau} + \sum_{\tau=1}^{t'-1} \prod_{i=\tau}^{t'} M^{l,i} \frac{S^{l-1,\tau}}{\tau_{\text{m}}} + \frac{S^{l-1,t'}}{\tau_{\text{m}}}. \tag{17}$$

The commonly used surrogate gradient functions are arc-tangent function and sigmoid function, both of them are very similar to $\Theta(x)$. In this article, considering that the complexity of surrogate gradient function would affect the computational efficiency, thus, we use the arc-tangent function in DSQN, which is defined by

$$\sigma_{\text{arctan}}(\alpha x) = \frac{1}{\pi} \arctan\left(\frac{\pi}{2} \alpha x\right) + \frac{1}{2} \tag{18}$$

where α is the factor that controls the smoothness of the function. The gradient of the arc-tangent function could be expressed as

$$\sigma'_{\text{arctan}}(\alpha x) = \frac{\alpha}{2\left[1 + \left(\frac{\pi}{2} \alpha x\right)^2\right]}. \tag{19}$$

Fig. 3 shows the curves of the spiking function $\Theta(x)$, the two commonly used surrogate gradient functions $\sigma_{\text{arctan}}(\alpha x), \sigma_{\text{sigmoid}}(\alpha x)$, and their gradient functions $\sigma'_{\text{arctan}}(\alpha x), \sigma'_{\text{sigmoid}}(\alpha x)$ with different α. Notably, the larger α is, the closer $\sigma_{\text{arctan}}(\alpha x)$ and $\Theta(x)$ will be. Meanwhile, when x is near 0, the gradient will be more likely to explode, and when x moves away from 0, the gradient will be more likely to disappear.

Using the surrogate gradient function allows DSQN to be directly trained well by a deep spiking Q-learning algorithm.

C. Demonstration of Using LIF Neurons in DSQN

In this section, we demonstrate the advantages of using LIF neurons in directly trained DSQNs theoretically. We first explain why existing conversion methods in DSRL use integrate-and-fire (IF) neurons, then provide the theory for using LIF neurons in directly trained DSQN.

1) Limitation of Conversion Methods in DSRL: Rueckauer *et al.* [26] demonstrated the IF neuron which is reset by *soft reset* is an unbiased estimator of the ReLU activation function over time. The neuronal dynamics of the IF neuron could be described as

$$V^{l,t} = \begin{cases} \left(V^{l,t-1} + z^{l,t}\right)\left(1 - S^{l,t}\right) + V_{\text{r}} S^{l,t}, & \text{hard reset} \\ V^{l,t-1} + z^{l,t} - V_{\text{th}} S^{l,t}, & \text{soft reset.} \end{cases} \tag{20}$$

With $V_{\text{r}} = 0$ and the fact that the input of the first layer $z^1 = V_{\text{th}} a^1$ constantly, Rueckauer *et al.* [26] provided the relationship between the firing rate $r^{1,t}$ of IF neurons in the first layer and the output of ReLU activation function a^1 when receiving the same inputs as

$$r^{1,t} = \begin{cases} a^1 r_{\max} \frac{V_{\text{th}}}{V_{\text{th}} + \epsilon^1} - \frac{V^{1,t}}{t(V_{\text{th}} + \epsilon^1)}, & \text{hard reset} \\ a^1 r_{\max} - \frac{V^{1,t}}{t V_{\text{th}}}, & \text{soft reset} \end{cases} \tag{21}$$

where $r_{\max}$ denotes the maximum firing rate, and ϵ denotes the residual charge which is discarded at resetting.

According to (21), when the length of simulation time window t tends to infinity and the inputs are in $[0, 1]$, the IF neuron which is reset by *soft reset* is an unbiased estimator of ReLU activation function over time.

Fig. 4 shows the relationship between the firing rate of the IF neuron which is reset by *soft reset* and the output of ReLU activation function when receiving the same inputs. When $x \in [0, 1]$, the firing rate of the IF neuron is highly similar to the output of ReLU activation function. But when input $x \geq 1$, the firing rate of the IF neuron no longer increase, because $r_{\max} = 1$.

Therefore, the existing conversion methods in DSRL must introduce a normalization technique to normalize the inputs beyond 1 into $[0, 1]$, and then ANNs can be successfully converted to SNNs. This results in the existing conversion methods in DSRL requiring very long simulation time window,

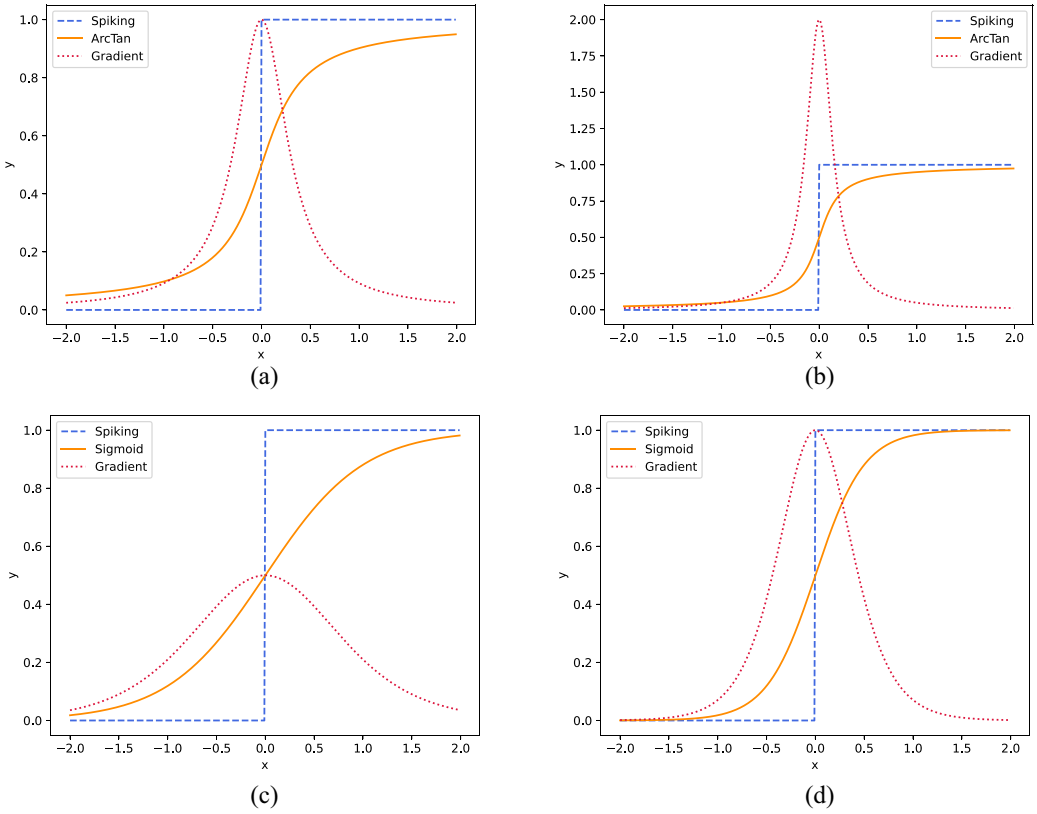


Fig. 3. Curves of the spiking function $\Theta(x)$, the two commonly used surrogate gradient functions $\sigma_{\arctan}(\alpha x)$, $\sigma_{\text{sigmoid}}(\alpha x)$, and their gradient functions $\sigma'_{\arctan}(\alpha x)$, $\sigma'_{\text{sigmoid}}(\alpha x)$ with different α. (a) Arc-tangent function with $\alpha = 2$. (b) Arc-tangent function with $\alpha = 4$. (c) Sigmoid function with $\alpha = 2$. (d) Sigmoid function with $\alpha = 4$.

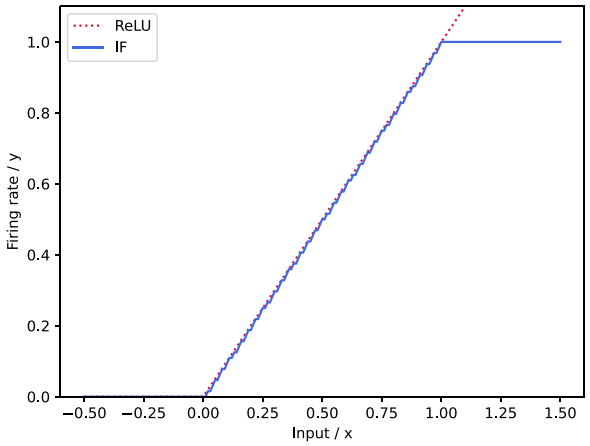


Fig. 4. Firing rate of the IF neuron which is reset by *soft reset* and the output of the ReLU function.

typically hundreds of timesteps, to achieve sufficient accuracy, so as to minimize the error generated during the converting process. In addition, the inherent problem of the conversion methods, that is, the heavy dependence on pretrained ANNs, remains unsolved.

2) Advantages of Using LIF Neurons in DSQN: Similar to the process of deriving (21), we could derive an equation describing the firing rate of the LIF neurons in the first layer from (1) and (2). To simplify the notation, we drop the layer and neuron indices, let z denote the input and $V_r = 0$.

For *hard reset*, starting from (2), the average firing rate could be simply computed by summing over the simulation time t as

$$\sum_{t'=1}^{t} V^{t'} = \frac{1}{\tau_m} \sum_{t'=1}^{t} \Big[(\tau_m - 1) V^{t'-1} + z \Big] \Big(1 - S^{t'} \Big). \quad (22)$$

Under the assumption of constant input z to the first layer, and using $N^t = \sum_{t'=1}^{t} S^{t'}$, we obtain

$$\tau_m \sum_{t'=1}^{t} V^{t'} = (\tau_m - 1) \sum_{t'=1}^{t} V^{t'-1} \Big(1 - S^{t'} \Big) + z \Big(\frac{t}{\Delta t} - N^t \Big). \quad (23)$$

The time resolution Δt enters (23) when evaluating the time-sum over a constant: $\sum_{t'=1}^{t} 1 = (t/\Delta t)$. It will be replaced by the definition of the maximum firing rate $r_{\max} = (1/\Delta t)$ in the following.

After rearranging (23) to yield the total number of spikes N^t at simulation time t, dividing by the simulation time t, setting $V^0 = 0$, and reintroducing the dropped indices, we obtain the average firing rate r of the LIF neurons in the first layer

$$r^{1,t} = r_{\max} - \frac{1}{t z^{1,t}} \left\{ \tau_m V^{1,t} + \sum_{t'=1}^{t} \Big[V^{1,t'-1} + (\tau_m - 1) V^{1,t'-1} S^{1,t'} \Big] \right\}. \quad (24)$$

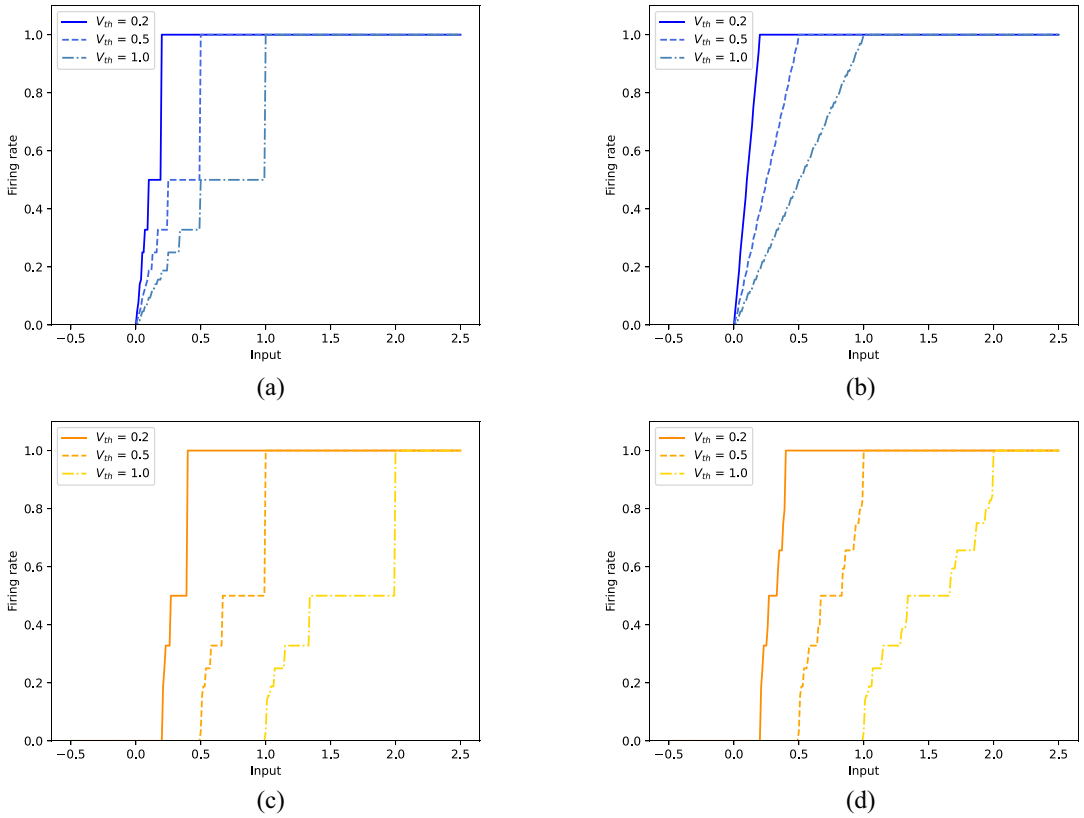


Fig. 5. Relationship between the firing rate and inputs of the IF neuron and LIF neuron which are reset by *hard reset* and *soft reset*. V_{th} is set to 0.2, 0.5, and 1, respectively. $V_{\text{r}} = 0$ constantly. (a) IF neuron which is reset by *hard reset*. (b) IF neuron which is reset by *soft reset*. (c) LIF neuron which is reset by *hard reset*. (d) LIF neuron which is reset by *soft reset*.

For *soft reset*, averaging (2) over the simulation time t yields

$$\frac{1}{t}\sum_{t'=1}^{t} V^{t'} = \frac{\tau_{\text{m}}-1}{t\tau_{\text{m}}}\sum_{t'=1}^{t} V^{t'-1} + \frac{z}{\tau_{\text{m}}} r_{\max} - V_{\text{th}}\frac{N^t}{t}. \quad (25)$$

Using $z^1 = V_{\text{th}} a^1$, solving for $r = N/t$, setting $V^0 = 0$, and reintroducing the indices yields

$$r^{1,t} = \frac{1}{\tau_{\text{m}}}\left(a^1 r_{\max} - \frac{1}{tV_{\text{th}}}\sum_{t'=1}^{t} V^{1,t'-1}\right) - \frac{V^{1,t}}{tV_{\text{th}}}. \quad (26)$$

Fig. 5 shows the relationship between the firing rate and inputs of the IF and LIF neurons which are reset by *hard reset* and *soft reset* with different V_{th}.

In practice, only one particular threshold could be used in the training and testing stage, and the ranges of the neuron inputs with fixed threshold are similar. However, in the hyperparameter tuning stage, the LIF neuron provides wider ranges for different thresholds. Thus, LIF neurons have more potential to obtain optimal results in our DSQN and could be directly trained without relying on the normalization technique in conversion methods.

IV. Experimental Results

In this section, we evaluated the DSQN on 17 top-performing Atari games. Then, we compared the experimental results of DSQN and the conversion-based SNN [28] using the vanilla DQN [17] as a benchmark, and analyzed the experimental results in several aspects.

TABLE I
Experimental Hardware Environment

Item	Detail
CPU	Intel Xeon Silver 4116
GPU	NVIDIA GeForce RTX 2080Ti
OS	Ubuntu 16.04 LTS
Memory	At least 40 GB for a single experiment

A. Experimental Setup

1) Environments: We performed the experiments based on OpenAI Gym.[1] Each experiment ran on a single GPU. The specific experimental hardware environments are shown in Table I.

2) Parameters: The proposed DSQN consists of three convolution layers and two fully connected layers. The specific parameters of the three convolution layers are shown in Table II, while the two fully connected layers use different paramaters. The first fully connected layer FC1 has 512 neurons, and the final layer FC2 has different neurons on different Atari games, from 4 to 18, depending on the number of validate actions in the game.

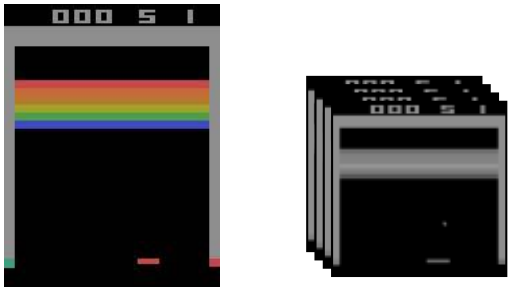
[1]The open-source code of OpenAI Gym could be accessed at https://github.com/openai/gym.

TABLE II
PARAMETERS OF THE CONVOLUTION LAYERS IN DSQN

Layer	Number of kernels	Kernel size	Stride
Conv1	32	8×8	4
Conv2	64	4×4	2
Conv3	64	3×3	1

TABLE III
VALUES AND DESCRIPTIONS OF THE DSQN-SPECIFIC HYPERPARAMETERS

Hyperparameter	Value	Description
t	64	Simulation time window
τ_{m}	2	Membrane time constant
V_{th}	1	Membrane potential threshold
V_{r}	0	Initial membrane potential
α	2	Smoothness factor
lr	0.0001	Learning rate



(a) (b)

Fig. 6. Preprocess raw Atari images and stack the m most recent frames as a state. (a) Raw image of Atari game Breakout. (b) State produced by preprocessing.

Both DSQN and the vanilla DQN were trained for 50M timesteps with the same architecture and hyperparameters [17]. The lengths of simulation time window of the conversion-based SNN and DSQN were set to 500 and 64 timesteps, respectively. Other DSQN-specific hyperparameters are illustrated in Table III.

It is important to point out that, different from the results reported in [28] which were conducted with the best *percentile* in [99.9, 99.99], we set the percentile to 99.9 constantly when reproducing the conversion-based SNN.

3) Preprocessing: Following [17], to reduce the amount of computation and memory required for training, we preprocess a single raw Atari game frame and stack the m most recent frames to produce the input of DSQN, in which $m = 4$. Fig. 6 shows the raw Atari image and preprocessed images which were stacked as a state. Same as the conversion-based SNN and vanilla DQN, we directly use raw real-valued pixel images as the inputs of the neural network without any neural encoding method.

4) Testing: To demonstrate the superiority of DSQN, we choose 17 top-performing Atari games as same as [28], which are also the games where the vanilla DQN has achieved very high performances [17]. In order to fairly compare DSQN with the conversion-based SNN and vanilla DQN, we reproduced these two methods.

TABLE IV
METRICS FOR NORMALIZED SCORES AND STANDARD DEVIATIONS

	Inferior	Equal	Outperform
Score	$< 95\%$	$\in [95\%, 105\%]$	$> 105\%$
Standard Deviation	$> 105\%$	$\in [95\%, 105\%]$	$< 95\%$

We evaluated these three RL agents by playing 30 rounds at each game with the ε-greedy policy ($\varepsilon = 0.05$). For each round, agents start with different initial random conditions by taking random times (at most 30 times) of *no-op* action, and play for up to 5 min (18 000 timesteps). The mean and standard deviation of the scores obtained in 30 rounds were used as the final scores and standard deviations of these three RL agents.

5) Metrics: We evaluated the performance and stability of the three RL agents over multiple Atari games by the scores and standard deviations they obtained, respectively. We normalized the scores of DSQN and the conversion-based SNN by the scores of the vanilla DQN. The standard deviations of the three RL agents were first normalized by their corresponding scores, and then the normalized standard deviations of DSQN and the conversion-based SNN were again normalized by the standard deviations of the vanilla DQN.

In this way, we can easily compare the performance and stability of DSQN with that of the conversion-based SNN by the normalized scores and standard deviations. Due to the generally poor stability of DRL algorithms, we developed Table IV to comparing DSQN and the conversion-based SNN with the vanilla DQN based on the normalized scores and standard deviations.

6) Code: We implemented DSQN based on SpikingJelly [42], which is an opensource[2] deep learning framework for SNNs based on PyTorch. The trained vanilla DQN was converted to SNN through the method proposed by Tan *et al.* [28] based on their opensource code.[3]

The source code could be accessed at our Github repository.[4]

B. Comparison Results

Table V reports the raw experimental results of the three RL agents on 17 top-performing Atari games, including scores and standard deviations. The sorting in Table V is based on the lexicographical order of game names. By using the metrics illustrated in the previous section, we obtained Table VI from Table V, which shows the performance and stability difference between DSQN and the conversion-based SNN by normalizing the raw experimental results in Table V with the scores and standard deviations of the vanilla DQN. The sorting in Table VI is based on the score differences between DSQN and Conversion-based SNN.

1) Performance: According to the differences of the scores shown in Table VI, DSQN achieved higher scores than the

[2]The opensource code of SpikingJelly could be accessed at https://github.com/fangwei123456/spikingjelly.

[3]The opensource code of [28] could be accessed at https://github.com/WeihaoTan/bindsnet-1.

[4]https://github.com/AptX395/Deep-Spiking-Q-Networks

TABLE V
RAW EXPERIMENTAL RESULTS OF THE VANILLA DQN, CONVERSION-BASED SNN, AND DSQN ON 17 TOP-PERFORMING ATARI GAMES

Game	Vanilla DQN			Conversion-based SNN [1]($t = 500$) [2]			Deep Spiking Q-Network [1]($t = 64$) [2]		
	Score	±std	(% Score)	Score	±std	(% Score)	Score	±std	(% Score)
Atlantis	493343.3	21496.9	(4.4%)	460700.0	17771.0	(3.9%)	**487366.7**	14400.6	(**3.0%**)
BeamRider	7414.1	1943.3	(26.2%)	6041.2	2423.2	(40.1%)	**7226.9**	2348.7	(**32.5%**)
Boxing	96.1	3.1	(3.2%)	91.3	7.0	(7.6%)	**95.3**	3.7	(**3.8%**)
Breakout	425.4	74.2	(17.5%)	364.6	108.0	(29.6%)	**386.5**	61.1	(**15.8%**)
Crazy Climber	120516.7	16126.6	(13.4%)	113133.3	28441.9	(25.1%)	**123916.7**	19142.0	(**15.4%**)
Gopher	10552.7	3935.1	(37.3%)	**10670.7**	4189.9	(**39.3%**)	10107.3	4303.2	(42.6%)
Jamesbond	906.7	982.2	(108.3%)	621.7	147.0	(**23.6%**)	**1156.7**	2699.4	(233.4%)
Kangaroo	4140.0	1605.5	(38.8%)	4520.0	1691.8	(**37.4%**)	**8880.0**	4041.3	(45.5%)
Krull	9309.3	1072.5	(11.5%)	7425.3	2588.9	(34.9%)	**9940.0**	999.5	(**10.1%**)
Name This Game	11004.0	1257.5	(11.4%)	10541.0	1653.9	(15.7%)	**10877.0**	1581.2	(**14.5%**)
Pong	20.2	1.0	(4.9%)	18.5	1.4	(7.3%)	**20.3**	0.9	(**4.6%**)
Road Runner	54596.7	6082.0	(11.1%)	43160.0	16322.1	(37.8%)	**48983.3**	5903.1	(**12.1%**)
Space Invaders	2274.5	808.2	(35.5%)	1387.3	791.6	(57.1%)	**1832.2**	735.4	(**40.1%**)
Star Gunner	51070.0	9513.2	(18.6%)	1176.7	2997.9	(254.8%)	**57686.7**	6296.3	(**10.9%**)
Tennis	−1.0	0.0	(0.0%)	−1.0	0.0	(0.0%)	−1.0	0.0	(0.0%)
Tutankham	187.4	60.3	(32.2%)	190.9	34.5	(**18.1%**)	**194.7**	51.4	(26.4%)
Video Pinball	316428.0	223159.8	(70.5%)	266940.1	192004.0	(71.9%)	**275342.8**	177157.0	(**64.3%**)

[1] The bold part represents the better among DSQN and the conversion-based SNN. Each game was run for 30 rounds.
[2] The length of simulation time window t of the conversion-based SNN and DSQN were set to **500** and **64** timesteps, respectively. The *percentile* of the conversion-based SNN was set to **99.9** constantly.

TABLE VI
PERFORMANCE AND STABILITY DIFFERENCE BETWEEN DSQN AND THE CONVERSION-BASED SNN

Game	Conversion-based SNN ($t = 500$)		Deep Spiking Q-Network ($t = 64$) [1]		DSQN - Conversion-based SNN [2]	
	Score (% DQN)	±std (% DQN)	Score (% DQN)	±std (% DQN)	Score Difference	±std Difference
Star Gunner	2.3%	1367.7%	**113.0%**	**58.6%**	**110.7%**	**−1309.1%**
Kangaroo	109.2%	96.5%	**214.5%**	117.4%	**105.3%**	20.9%
Jamesbond	68.6%	21.8%	**127.6%**	215.4%	**59.0%**	193.6%
Krull	79.8%	302.6%	**106.8%**	**87.3%**	**27.0%**	**−215.3%**
Space Invaders	61.0%	160.6%	80.6%	113.0%	**19.6%**	**−47.6%**
BeamRider	81.5%	153.0%	97.5%	124.0%	**16.0%**	**−29.0%**
Road Runner	79.1%	339.5%	89.7%	108.2%	**10.7%**	**−231.3%**
Pong	91.7%	151.3%	100.7%	95.6%	**9.0%**	**−55.7%**
Crazy Climber	93.9%	187.9%	102.8%	115.4%	**8.9%**	**−72.5%**
Atlantis	93.4%	88.5%	98.8%	**67.8%**	**5.4%**	**−20.7%**
Breakout	85.7%	169.8%	90.9%	**90.6%**	**5.1%**	**−79.2%**
Boxing	95.1%	238.4%	99.2%	120.0%	**4.1%**	**−118.4%**
Name This Game	95.8%	137.3%	98.8%	127.2%	**3.1%**	**−10.1%**
Video Pinball	84.4%	102.0%	87.0%	**91.2%**	**2.7%**	**−10.8%**
Tutankham	101.9%	56.1%	103.9%	**82.0%**	**2.0%**	25.9%
Tennis	100.0%	100.0%	100.0%	100.0%	0.0%	0.0%
Gopher	101.1%	105.3%	95.8%	114.2%	−5.3%	8.9%
Average	83.8%	222.3%	106.3%	107.5%	22.5%	−114.8%

[1] The bold part in the two column of *Deep Spiking Q-Network* represents that DSQN outperforms the vanilla DQN in terms of performance and stability.
[2] The bold part in the two column of *DSQN - Conversion-based SNN* represents that DSQN achieved higher scores and lower standard deviations than the conversion-based SNN.

conversion-based SNN on 15 out of 17 Atari games. This illustrates the superiority of our direct learning method for DSQN compared to the conversion method.

At the same time, based on the scores shown in Table VI and the judging criteria in Table IV, DSQN outperforms the vanilla DQN on four games, is equal to it on nine games, and is inferior to it on four games. This illustrates that our directly trained DSRL architecture has the same-level performance of the vanilla DQN in solving DRL problems.

2) Stability: According to the differences of the standard deviations shown in Table VI, DSQN achieved lower standard deviations than the conversion-based SNN on 12 out of 17 Atari games. This illustrates that DSQN is stronger than the conversion-based SNN in terms of stability.

At the same time, based on the standard deviations shown in Table VI and the judging criteria in Table IV, DSQN outperforms the vanilla DQN on six games, is equal to it on two games, and is inferior to it on nine games. This illustrates that DSQN has the same-level stability of the vanilla DQN.

Furthermore, DSQN shows stronger generalization than the conversion-based SNN. With the respective hyperparameters of DSQN and the conversion-based SNN being fixed, the experimental results show that DSQN outperforms the conversion-based SNN on most games. This indicates that

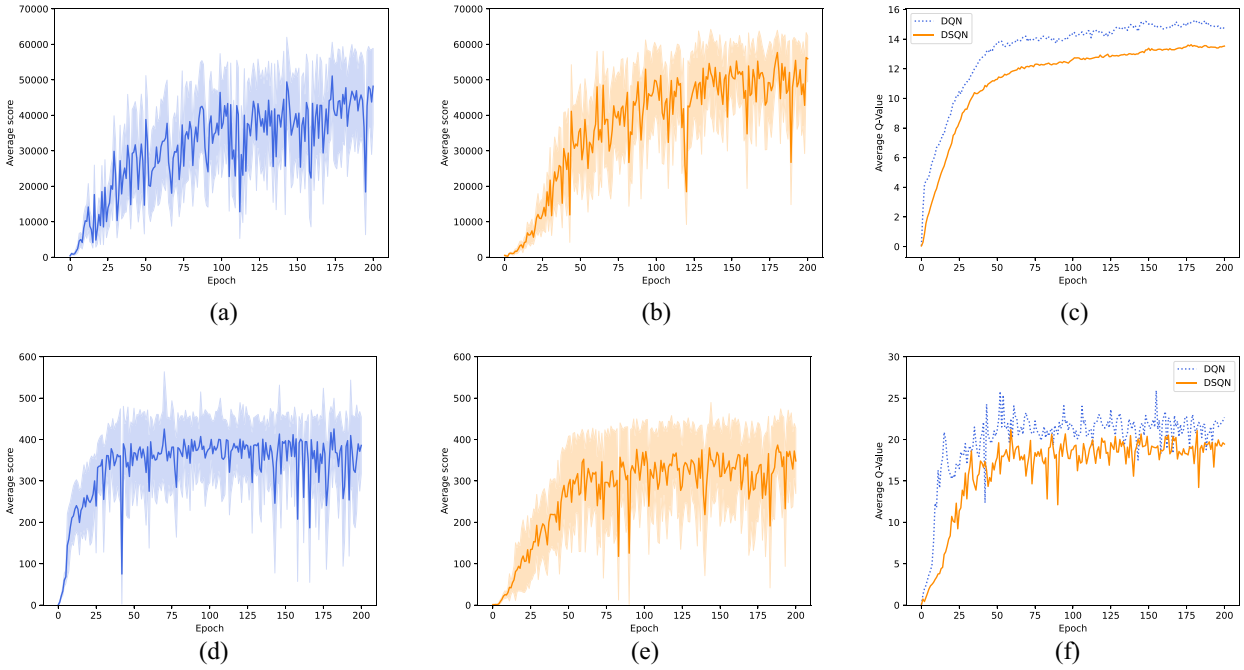


Fig. 7. Learning curves of DSQN and the vanilla DQN on Atari game Star Gunner and Breakout. During the training process, each episode was run for 30 rounds. In (a), (b), (d), and (e), each point is the average score achieved per episode. In (c) and (f), each point is the average max Q-value achieved per episode (epoch = 250 000 timesteps).

DSQN is more adaptable to different game environments, and its generalization is stronger than that of the conversion-based SNN.

3) Learning Capability: In addition to the comparison of DSQN and the conversion-based SNN, we also analyzed the gap between DSQN and the vanilla DQN.

According to Table VI, the average score of DSQN slightly exceeds that of the vanilla DQN (i.e., 100%), and the average standard deviation of DSQN also slightly exceeds that of the vanilla DQN. This indicates that DSQN achieves the same level of learning capability as the vanilla DQN.

For instance, Fig. 7 shows the learning curves of DSQN and the vanilla DQN on Atari game Star Gunner and Breakout during the training process. As shown in Fig. 7(a) and (b), the scores of DSQN are higher overall than that of the vanilla DQN. As shown in Fig. 7(d) and (e), although the scores of DSQN are slightly lower overall than that of the vanilla DQN, the standard deviations of DSQN are lower overall than that of the vanilla DQN. These examples can confirm the learning capability of DSQN. The curves of average Q-values shown in Fig. 7(c) and (f) can also confirm this.

4) Energy Efficiency: Besides, DSQN shows higher energy efficiency than the conversion-based SNN as the fact that, DSQN achieved better performance than the conversion-based SNN while its simulation time window length is only 64 timesteps, which is one order of magnitude lower than that of the conversion-based SNN (500 timesteps).

To compare the energy efficiency of DSQN and the conversion-based SNN more precisely, we calculate their computational costs in the inference stage based on the number of synaptic operations using the method in [43], and we count the spiking times of them each time a decision is made on average.

TABLE VII
AVERAGE SPIKING TIMES OF DSQN AND THE CONVERSION-BASED SNN

Game	Conversion-based SNN	Deep Spiking Q-Network
Breakout	185.8k	**159.6k**
Star Gunner	183.6k	**157.0k**

For DSQN, let N_D denote the number of neurons in the neural network. For each neuron, there are three addition and one multiplication operations, for a total of four synaptic operations according to (1) and (2). Since the simulation time window length is 64 timesteps, the computational cost is $N_D \times 4 \times 64 = 256N_D$.

For the conversion-based SNN, let N_C denotes the number of neurons in the neural network. For each neuron, there are one addition, that is, a total of one synaptic operation according to (20). Since the simulation time window length is 500 timesteps, the computational cost is $N_C \times 1 \times 500 = 500\ N_C$.

On the other hand, from the perspective of energy transfer, we take the two games shown in Fig. 7 (Breakout and Star Gunner) as examples, and count the average spiking times of DSQN and the conversion-based SNN each time a decision is made, which are shown in Table VII. From the results, the average spiking times of DSQN are only 85.9% and 85.5% of that of the conversion-based SNN on the two games, respectively.

According to the above analysis, when the structure and scale of DSQN and the conversion-based SNN are the same (i.e., $N_D = N_C$), the energy efficiency of DSQN is higher than that of the conversion-based SNN.

5) In Summary: Combining the experimental results of DSQN and the conversion-based SNN in terms of both the scores and standard deviations, DSQN not only achieves

TABLE VIII
COMPARISON RESULTS OF DOUBLE DQN AND DOUBLE DSQN

Game	Double DQN	Double DSQN	Score(% Double DQN)
Beam Rider	3617.3	4611.1	**127.5**%
Breakout	137.9	360.9	**261.7**%
Jamesbond	748.0	691.5	92.4%
Kangaroo	4302.0	8620.0	**200.4**%
Krull	10825.1	10054.1	92.9%
Road Runner	34275.0	43925.0	**128.2**%
Space Invaders	694.9	1089.8	**156.8**%
Average	-	-	**151.4**%

TABLE IX
COMPARISON RESULTS OF CDQN AND CDSQN

Game	CDQN	CDSQN	Score(% CDQN)
Beam Rider	8084.0	6913.5	85.5%
Breakout	337.8	275.5	81.6%
Jamesbond	235.0	491.7	**209.2**%
Kangaroo	2360.0	7000.0	**296.6**%
Krull	8404.3	8938.6	**106.4**%
Road Runner	28200.0	30537.5	**108.3**%
Space Invaders	1178.1	1238.3	**105.1**%
Average	-	-	141.8%

higher scores than the conversion-based SNN on the vast majority of Atari games but also achieves lower standard deviations at the same time. This indicates that DSQN outperforms the conversion-based SNN in both performance and stability. In addition, according to the comparison of DSQN and the vanilla DQN, even though DSQN is an SNN, its learning capability is not inferior to the vanilla DQN.

These experimental results demonstrated the superiority of DSQN over the conversion-based SNN in terms of performance, stability, generalization, and energy efficiency. Meanwhile, DSQN reaches the same level as DQN in terms of performance and surpasses DQN in terms of stability.

C. Further Comparison

To further explore the performance of our method, we compare our method with the latest variants of DQN, the Double DQN [44], and the state-of-the-art DRL method CDQN [45]. For fair comparison, we adapt our proposed DSRL strategy to the comparison baselines, called Double DSQN and CDSQN.

The Double DQN and DSQN are trained on 1M timesteps, and the CDQN and CDSQN are trained on 10M timesteps. The hyperparameters of the Double DQN and CDQN follow their [44] and [45], respectively. Other experimental setups are the same as previous experiments.

The code of baselines are in the open source github.[5,6] The experiments are conducted based on the OpenAI Gym.[7]

The comparison results of the Double DQN and CDQN are shown in Tables VIII and IX, respectively. From which we see that our method achieves the averaged percent score of 151.4% in the Double DQN, and achieves 141.8% in the CDQN comparison. This further demonstrates the effectiveness of our method. Specifically, in the Kangaroo game, our method obtains scores of 8620 and 7000 in the Double DSQN and CDSQN, respectively, which significantly outperforms conventional scores of 4302 and 2360. The conclusions are consistent with previous experiments.

[5]https://github.com/thu-ml/tianshou
[6]https://github.com/Z-T-WANG/ConvergentDQN
[7]https://github.com/openai/gym

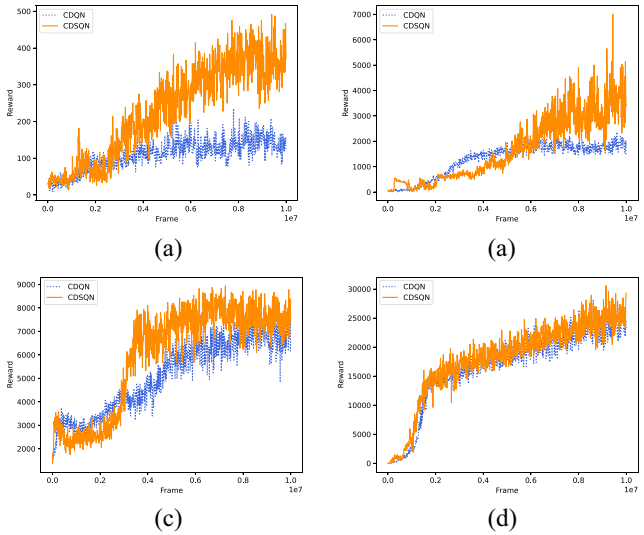


Fig. 8. Learning curves of CDQN and CDSQN on Atari game (a) Jamesbond, (b) Kangaroo, (c) Krull, and (d) Road runner.

For further analysis, we plot the converge learning curves of the CDSQN and CDQN in the following Fig. 8, which demonstrate that our method outperforms conventional CDQN on most training frames.

V. CONCLUSION AND FUTURE WORKS

In this article, we proposed a directly trained DSRL architecture called DSQN to address the issue of solving DRL problems with SNNs. To the best of our knowledge, our method was the first one to achieve state-of-the-art performance on multiple Atari games with directly trained SNNs. Our work served as a benchmark for the directly trained SNNs playing Atari games, and paved the way for future research to solving DRL problems with SNNs. This work was designed for the spiking deep Q-Learning-based method. For other DRL algorithms, such as distributional RL, our work needs further study. The theoretical analysis of LIF neuron's nature was not particularly in-depth, we could continue to improve it in the future work. Moreover, based on the present work, and in order to better stimulate the potential of SNNs, we plan to conduct further research on actual neuromorphic hardware in the future.

REFERENCES

[1] X. Ju, B. Fang, R. Yan, X. Xu, and H. Tang, "An FPGA implementation of deep spiking neural networks for low-power and fast classification," *Neural Comput.*, vol. 32, no. 1, pp. 182–204, Jan. 2020.
[2] 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.
[3] 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, 2014.
[4] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan./Feb. 2018.

[5] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, pp. 1311–1318, Jul. 2019.

[6] Y. Wang, Y. Xu, R. Yan, and H. Tang, "Deep spiking neural networks with binary weights for object recognition," *IEEE Trans. Cogn. Devel. Syst.*, vol. 13, no. 3, pp. 514–523, Sep. 2021.

[7] S. Kim, S. Park, B. Na, and S. Yoon, "Spiking-YOLO: Spiking neural network for energy-efficient object detection," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, Apr. 2020, pp. 11270–11277.

[8] J. P. Dominguez-Morales *et al.*, "Deep spiking neural network model for time-variant signals classification: A real-time speech recognition approach," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2018, pp. 1–8.

[9] J. Wu, E. Yılmaz, M. Zhang, H. Li, and K. C. Tan, "Deep spiking neural networks for large vocabulary automatic speech recognition," *Front. Neurosci.*, vol. 14, p. 199, Mar. 2020.

[10] S. Guo, Z. Yu, F. Deng, X. Hu, and F. Chen, "Hierarchical Bayesian inference and learning in spiking neural networks," *IEEE Trans. Cybern.*, vol. 49, no. 1, pp. 133–145, Jan. 2019.

[11] Q. Yu, S. Li, H. Tang, L. Wang, J. Dang, and K. C. Tan, "Toward efficient processing and learning with spikes: New approaches for multispike learning," *IEEE Trans. Cybern.*, vol. 52, no. 3, pp. 1364–1376, Mar. 2022.

[12] G. Aquino *et al.*, "Novel nonlinear hypothesis for the delta parallel robot modeling," *IEEE Access*, vol. 8, pp. 46324–46334, 2020.

[13] G. Hernández, E. Zamora, H. Sossa, G. Téllez, and F. Furlán, "Hybrid neural networks for big data classification," *Neurocomputing*, vol. 390, pp. 327–340, May 2020.

[14] Q. Liu, D. Xing, H. Tang, D. Ma, and G. Pan, "Event-based action recognition using motion information and spiking neural networks," in *Proc. 30th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 1743–1749.

[15] J. Wu *et al.*, "Progressive tandem learning for pattern recognition with deep spiking neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Sep. 21, 2021, doi: 10.1109/TPAMI.2021.3114196.

[16] V. Mnih *et al.*, "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

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

[18] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[19] 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.

[20] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[21] M. Hessel *et al.*, "Rainbow: Combining improvements in deep reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, Apr. 2018, pp. 3215–3222.

[22] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Front. Neurosci.*, vol. 12, p. 774, Oct. 2018.

[23] J. A. Pérez-Carrasco *et al.*, "Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to Feedforward ConvNets," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 11, pp. 2706–2719, Nov. 2013.

[24] 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.

[25] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2015, pp. 1–8.

[26] 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," *Front. Neurosci.*, vol. 11, p. 682, Dec. 2017.

[27] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Front. Neurosci.*, vol. 13, p. 95, Mar. 2019.

[28] W. Tan, D. Patel, and R. Kozma, "Strategy and benchmark for converting deep *Q*-networks to event-driven spiking neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, pp. 9816–9824, May 2021.

[29] X. Xie, H. Qu, Z. Yi, and J. Kurths, "Efficient training of supervised spiking neural network via accurate synaptic-efficiency adjustment method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1411–1424, Jun. 2017.

[30] X. Xie, H. Qu, G. Liu, and M. Zhang, "Efficient training of supervised spiking neural networks via the normalized perceptron based learning rule," *Neurocomputing*, vol. 100, no. 241, pp. 152–163, 2017.

[31] B. Fang, Y. Zhang, R. Yan, and H. Tang, "Spike trains encoding optimization for spiking neural networks implementation in FPGA," in *Proc. 12th Int. Conf. Adv. Comput. Intell. (ICACI)*, Aug. 2020, pp. 412–418.

[32] Y. Xu, H. Tang, J. Xing, and H. Li, "Spike trains encoding and threshold rescaling method for deep spiking neural networks," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2017, pp. 1–6.

[33] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, "A tandem learning rule for effective training and rapid inference of deep spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jul. 21, 2021, doi: 10.1109/TNNLS.2021.3095724.

[34] M. Zhang *et al.*, "Rectified linear postsynaptic potential function for backpropagation in deep spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 5, pp. 1947–1958, May 2022.

[35] H. Zheng, Y. Wu, L. Deng, Y. Hu, and G. Li, "Going deeper with directly-trained larger spiking neural networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, May 2021, pp. 11062–11070.

[36] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks," *IEEE Signal Process. Mag.*, vol. 36, no. 6, pp. 51–63, Nov. 2019.

[37] M. Yuan, X. Wu, R. Yan, and H. Tang, "Reinforcement learning in spiking neural networks with stochastic and deterministic synapses," *Neural Comput.*, vol. 31, no. 12, pp. 2368–2389, Dec. 2019.

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

[39] X. Shen, G. Dong, Y. Zheng, L. Lan, I. W. Tsang, and Q.-S. Sun, "Deep co-image-label hashing for multi-label image retrieval," *IEEE Trans. Multimedia*, vol. 24, pp. 1116–1126, 2021. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9582767

[40] X. Shen, F. Shen, Q.-S. Sun, Y. Yang, Y.-H. Yuan, and H. T. Shen, "Semi-paired discrete hashing: Learning latent hash codes for semi-paired cross-view retrieval," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4275–4288, Dec. 2017.

[41] X. Shen, W. Liu, I. W. Tsang, Q.-S. Sun, and Y.-S. Ong, "Multilabel prediction via cross-view search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4324–4338, Sep. 2018.

[42] W. Fang *et al.* "SpikingJelly." Github.com. 2020. [Online]. Available: https://github.com/fangwei123456/spikingjelly

[43] L. Deng *et al.*, "Rethinking the performance comparison between SNNS and ANNS," *Neural Netw.*, vol. 121, pp. 294–307, Jan. 2020.

[44] J. Weng *et al.*, "Tianshou: A highly modularized deep reinforcement learning library," 2021, *arXiv:2107.14171*.

[45] Z. T. Wang and M. Ueda, "Convergent and efficient deep Q learning algorithm," in *Proc. Int. Conf. Learn. Represent.*, Mar. 2022, pp. 1–27.

Guisong Liu received the B.S. degree in mechanics from Xi'an Jiaotong University, Xi'an, China, in 1995, and the M.S. degree in automatics and the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2000 and 2007, respectively.

He is currently a Professor and the Dean of the School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu. He has filed over 20 patents and published over 70 scientific conference and journal papers, and was a Visiting Scholar with Humbolt University of Berlin, Berlin, Germany, in 2015. Before 2021, he was a Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interests include pattern recognition, neural networks, and machine learning.

Wenjie Deng received the B.S. degree in computer science and technology from the Southwest University of Science and Technology, Mianyang, China, in 2020. He is currently pursuing the M.S. degree with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China.

His research interests include neural networks and reinforcement learning.

Xiurui Xie received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2016.

She worked as a Research Fellow with Nanyang Technological University, Singapore, from 2017 to 2018, and a Research Scientist with the Agency for Science, Technology and Research, Singapore, from 2018 to 2020. She is currently an Associate Professor with the University of Electronic Science and Technology of China. She has authored over ten technical papers in prominent journals and conferences. Her primary research interests are neural networks, neuromorphic chips, transfer learning, and pattern recognition.

Li Huang received the Ph.D. degree from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2021, supervised by Prof. W. Chen.

She is a Lecturer with the School of Computing and Artificial Intelligence, Southwestern University of Finance and Economics, Chengdu. Her research interests include aspect sentiment analysis, machine translation, text summarization, and continual learning in natural language processing.

Huajin Tang (Senior Member, IEEE) received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 1998, the M.Eng. degree from Shanghai Jiao Tong University, Shanghai, China, in 2001, and the Ph.D. degree from the National University of Singapore, Singapore, in 2005.

He was a System Engineer with STMicroelectronics, Singapore, from 2004 to 2006. He was a Postdoctoral Fellow with the Queensland Brain Institute, The University of Queensland, Brisbane, QLD, Australia, from 2006 to 2008. He has been the Head of the Robotic Cognition Lab, Institute for Infocomm Research, A*STAR, Singapore, since 2008. He has been a Professor with the College of Computer Science, Sichuan University, Chengdu, China, since 2014. He is currently a Professor with the College of Computer Science and Technology, Zhejiang University, and also with the Institute of Artificial Intelligence, Zhejiang Laboratory. His current research interests include neuromorphic computing, neuromorphic hardware and cognitive systems, and robotic cognition.

Dr. Tang received the 2016 IEEE Outstanding TNNLS Paper Award and the 2019 *IEEE Computational Intelligence Magazine* Outstanding Paper Award. He was the Program Chair of IEEE CIS-RAM in 2015 and 2017, and ISNN in 2019, and the Co-Chair of the IEEE Symposium on Neuromorphic Cognitive Computing from 2016 to 2019. He is a Board of Governor Member of the International Neural Networks Society. He has served as an Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS, *Frontiers in Neuromorphic Engineering*, and *Neural Networks*.