Toward Energy-Efficient Spike-Based Deep Reinforcement Learning With Temporal Coding

Malu Zhang and *Shuai Wang*
University of Electronic Science and Technology of China, Chengdu, 610054, CHINA

Jibin Wu
The Hong Kong Polytechnic University, Kowloon, HONG KONG

Wenjie Wei, *Dehao Zhang*, *Zijian Zhou*, *Siying Wang*, *Fan Zhang*, and *Yang Yang*
University of Electronic Science and Technology of China, Chengdu, 610054, CHINA

***Abstract*—**Deep reinforcement learning (DRL) facilitates efficient interaction with complex environments by enabling continuous optimization strategies and providing agents with autonomous learning abilities. However, traditional DRL methods often require large-scale neural networks and extensive computational resources, which limits their applicability in power-sensitive and resource-constrained edge environments, such as mobile robots and drones. To overcome these limitations, we leverage the energy-efficient properties of brain-inspired spiking neural networks (SNNs) to develop a novel spike-based DRL framework, referred to as Spike-DRL. Unlike traditional SNN-based reinforcement learning methods, Spike-DRL incorporates the energy-efficient time-to-first-spike (TTFS) encoding scheme, where

Digital Object Identifier 10.1109/MCI.2025.3541572
Date of current version: 7 April 2025

This article was recommended for publication by Associate Editor Yi Mei. Corresponding author: Jibin Wu (e-mail: jibin.wu@polyu.edu.hk).

information is encoded through the precise timing of a single spike. This TTFS-based method allows Spike-DRL to work in a sparse, event-driven manner, significantly reducing energy consumption. In addition, to improve the deployment capability of Spike-DRL in resource-constrained environments, a lightweight strategy for quantizing synaptic weights into low-bit representations is introduced, significantly reducing memory usage and computational complexity. Extensive experiments have been conducted to evaluate the performance of the proposed Spike-DRL, and the results show that our method achieves competitive performance with higher energy efficiency and lower memory requirements. This work presents a biologically inspired model that is well suited for real-time decision-making and autonomous learning in power-sensitive and resource-limited edge environments.

I. Introduction

Deep reinforcement learning (DRL) integrates the advantages of deep learning and reinforcement learning, allowing agents to perceive and make decisions on the basis of unstructured data in complex, high-dimensional environments [1], [2]. In recent years, DRL has achieved significant breakthroughs across various domains [3], [4], [5], [6], [7], [8] and has been widely applied in robotics [5], [6], video games [7], [8], autonomous driving [9], and natural language processing (NLP) [10], [11]. However, DRL requires extensive and frequent interactions with the environment to update agents' network parameters, resulting in significant computational overhead. In addition, DRL models typically contain many parameters, posing challenges for deployment on power-sensitive and resource-limited devices such as mobile robots and drones [12], [13], [14]. Given that agents in DRL often operate on edge devices, achieving the goals of low power consumption and a lightweight design for DRL models is crucial for their practical deployment.

Brain-inspired Spiking Neural Networks (SNNs) [15], [16] simulate the information processing mechanisms of biological neurons by encoding information through discrete temporal spikes and operating in an event-driven manner. Owing to their inherently lower power consumption, SNNs have been regarded as energy-efficient alternatives to traditional Artificial Neural Networks (ANNs) [17], [18], [19]. To fully leverage the advantages of SNNs, researchers have focused on developing efficient encoding methods, learning algorithms, and computational models. Encoding methods are generally classified into population-based, rate-based, and temporal-based coding methods, all of which represent information through spatiotemporal spike patterns [20], [21], [22], [23]. To train SNNs effectively, various learning algorithms have been proposed, including ANN-to-SNN conversion methods [24], [25], [26], surrogate gradient-based learning [27], [28], [29], and spike-driven learning approaches [30], [31], [32]. In terms of computational models, SNNs now incorporate advanced architectures from deep learning, such as attention mechanisms [33], [34], transformers [35], and large language models (LLMs) [36], to significantly enhance their performance in complex tasks. These advancements facilitate the broader application of SNNs across different fields.

Recently, many studies have integrated SNNs with reinforcement learning to create energy-efficient solutions for continuous control tasks [37], [38], [39], [40]. This approach aims to achieve brain-like artificial intelligence by utilizing the event-driven nature of SNNs to reduce power consumption. However, current SNN-based DRL methods still face two main challenges. Most methods rely on population coding, where multiple neurons represent a single value [41], [42]. This approach is inefficient, increases energy use, and adds complexity to the learning process. Additionally, the large size of SNN models complicates deployment on resource-constrained edge devices, such as mobile robots and drones. These limitations restrict their practical use in real-world scenarios where energy efficiency and compact deployment are crucial.

In this paper, an energy-efficient and lightweight spike-based DRL model, named Spike-DRL, is proposed as an effective solution for addressing the above-mentioned problems in power-sensitive, resource-constrained edge environments. Specifically, the proposed Spike-DRL adopts the time-to-first-spike (TTFS) encoding method, where information is represented by the timing of a single spike to reduce energy consumption. In addition, Spike-DRL incorporates a quantization strategy to further enhance memory efficiency and hardware compatibility. Overall, the proposed Spike-DRL achieves low power consumption and a lightweight model design, facilitating the practical deployment of DRL agents on edge devices. The main contributions of this paper can be summarized as follows:

- ❏ This paper introduces a spike-based reinforcement learning model (Spike-DRL) that incorporates the TTFS coding scheme. In Spike-DRL, the information is encoded via the timing of single spikes, and the model operates in a fully event-driven manner. This design enables Spike-DRL to achieve competitive performance and superior efficiency compared to those of RL methods in SNNs.
- ❏ The synaptic weights of Spike-DRL are quantized to low-bit representations to further increase its efficiency. The quantization operation effectively minimizes memory usage and reduces computational complexity, thereby making the Spike-DRL more suitable for deployment in memory-limited edge computing environments.
- ❏ To our knowledge, this is the first TTFS-based DRL method in the SNN domain. Extensive experiments demonstrate that the proposed Spike-DRL achieves competitive performance with higher energy efficiency and lower memory requirements. This work advances the practical deployment of DRL agents in power-sensitive and resource-constrained environments.

The remainder of the paper is organized as follows. Section II reviews related work, including DRL, learning algorithms for SNNs, and reinforcement learning in SNNs. Section III presents the overall framework of our Spike-DRL, which consists of four key components: the TTFS-encoded spiking neuron model, the quantized spiking actor network (Q-SAN), the spike-driven learning algorithm for the Q-SAN, and the soft actor-critic algorithm in Spike-DRL. Section IV outlines our experimental setup using OpenAI Gym environments and presents comparison results between our Spike-DRL and related studies. Ablation studies are presented in different weight bit widths. Section V concludes the paper and discusses the potential directions for future studies.

II. Related Work

A. Deep Reinforcement Learning

DRL has achieved strong performance across various decision-making domains [43], [44], [45], [46]. The goal of DRL is to learn an optimal policy through the interaction between the agent and the environment to maximize a numerical reward signal. Specifically, at each timestamp t, the agent perceives a state s_t and executes an action a_t, following the policy $\pi(a_t|s_t)$, receives a reward r_t, and transitions to the next state s_{t+1}, according to the environment dynamics. By maximizing the future discounted return, we can obtain an agent in the paradigm of automatic learning. Actor-critic algorithms [47], [48] leverage the strengths of both policy-based [49] and value-based [50] approaches, establishing a dominant presence in the domain of continuous control tasks. The critic component adeptly evaluates future discounted returns, providing a robust framework for assessing long-term outcomes. Simultaneously, the actor component, guided by the critic's evaluations, offers reliable directives on optimal actions, thus facilitating improved decision-making in complex environments.

Current studies on actor-critic algorithms focus primarily on how to facilitate efficient learning [51], [52] of the agent. Entropy-regularized DRL improves the sample efficiency by maximizing the sum of the expected return and the policy action entropy [53], achieving the goal of agent learning with fewer interactions. Entropy regularization has been adopted in various domains in RL, such as stochastic optimal control [54] and inverse RL [55], and so on. As the best attempt to extend maximum entropy RL and propose an off-policy actor-critic algorithm, Soft Actor-Critic (SAC) [56] achieves competitive performance for challenging continuous control tasks.

B. Learning Algorithms in SNNs

While discrete spike patterns provide SNNs with significant efficiency advantages, they also present a training challenge related to nondifferentiability. Therefore, conventional training methods for ANNs such as error backpropagation cannot be leveraged directly by SNNs. To overcome this limitation, various algorithms have been proposed. These algorithms can be categorized into three main groups: ANN-to-SNN conversion methods, surrogate gradient learning methods, and spike-driven learning methods.

First, ANN-to-SNN conversion methods train a conventional ANN and then transfer the pretrained weights to its SNN counterpart [24], [25], [57], [58]. These methods allow for the utilization of well-established ANN training techniques and avoid the training challenges posed by discrete spikes. Second, surrogate gradient learning methods address the nondifferentiability of discrete spikes by employing surrogate gradient functions, enabling end-to-end training of SNNs [27], [59], [60]. These methods fully exploit the temporal information processing capabilities of SNNs, achieving high performance with low inference latency. Third, spike-driven learning algorithms use spike timing as the learning signal instead of the membrane potential, which performs training only when spikes occur [31], [61], [62], [63]. As a temporal encoding scheme of SNNs, TTFS employs the timing of the first spike to encode information, allowing neurons to transmit information with at most one spike, thereby resulting in high storage and computational efficiency. Therefore, the use of spike-driven learning algorithms is highly promising for the applicability of spike-based reinforcement methods in resource-limited edge agents. However, existing spike-based reinforcement studies use the first two categories of learning algorithms and have not yet explored energy-efficient spike-driven algorithms.

C. Reinforcement Learning in SNNs

The integration of RL with SNNs can be broadly categorized into two groups: ANN-based methods and spike-based direct reinforcement methods. Patel et al. [64] pioneered ANN-based methods by applying conversion techniques to deep Q-networks (DQNs). They employed particle swarm optimization (PSO) to reduce conversion errors, leading to improved robustness to adversarial attacks in the Atari Breakout game. Tan et al. [65] proposed a better representation of firing rates to bridge the gap between spiking and artificial neurons. Their converted SNN achieves scores comparable to those of the original DQN in 17 Atari games. Inspired by knowledge distillation, Zhang et al. [66] proposed a spike reinforcement learning method with a spike distillation network (SDN), which does not need spike trains to represent the value function, offering an indirect training method for spike reinforcement learning. Moreover, Tang et al. [67] proposed a hybrid framework, namely, the spiking deep deterministic policy gradient (SDDPG), which combines spiking actor networks (SANs) with deep critic networks, achieving substantial energy savings while maintaining competitive performance in continuous control tasks. Tang et al. [41] also proposed a population-coded spiking actor network (Pop-SAN), which greatly enhances the representation ability of SNNs through a population-coding scheme. On the basis of the Pop-SAN algorithm, Zhang et al. [42] proposed a multiscale dynamic coding improved spiking actor network (MDC-SAN), which

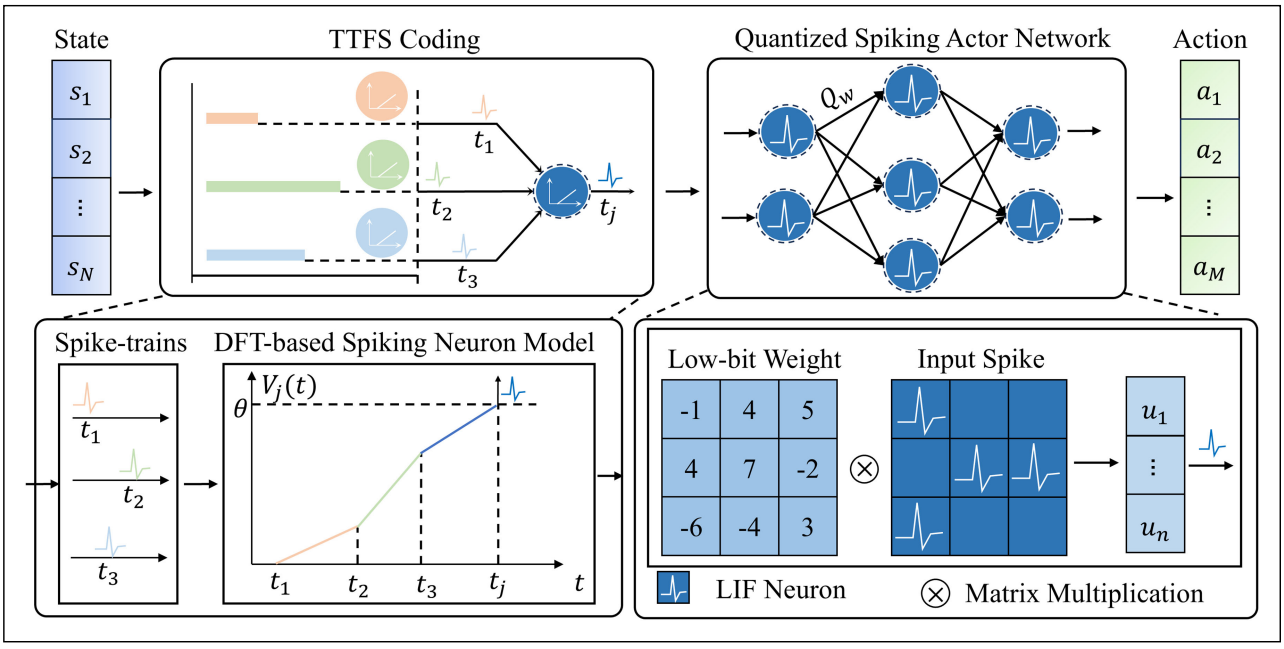

FIGURE 1 Schematic diagram of our proposed Spike-DRL model, which comprises four main components: A TTFS-encoded spiking neuron, the quantized spiking actor network, the spike-driven learning algorithm tailored for the Q-SAN, and the integration of reinforcement learning mechanisms. These components work together to improve the efficiency and performance of SNNs in reinforcement learning tasks.

achieves better performance in four continuous control tasks of OpenAI Gym.

The second category involves spike-based reinforcement learning algorithms, which facilitate the direct training of SNNs without relying on ANNs. Liu et al. [68] addressed the challenging Q-value problem on the basis of frequency encoding and proposed a deep spiking Q-network (DSQN), achieving performance comparable to that of the DQN while surpassing it in terms of stability. Moreover, inspired by intermediate neurons in biological systems that do not emit spikes, Chen et al. [37] designed a decoding mechanism to convert output spikes into continuous values by nonspiking neurons. They also proposed an implementation scheme for a deep spiking Q-network by representing Q-values using output spikes integrated from the membrane potential. Validation in terms of performance, energy consumption, and robustness to adversarial attacks comprehensively demonstrates the effectiveness of their DSQN. In addition to designing effective encoding schemes, the membrane potential can be used to represent continuous values. For example, Akl et al. [69] encoded observations into a two-neuron input scheme first, and used the membrane potential readout as the decoding method. Moreover, Qin et al. [70] utilized learnable matrices to compress spike trains in the temporal domain, providing an alternative approach to perform decoding operations.

III. Method

In this section, the core details of our proposed spike-based reinforcement learning model are presented. As illustrated in Figure 1, Spike-DRL comprises four main components:

- *TTFS-encoded spiking neuron.* The TTFS-encoded spiking neuron model serves as the fundamental unit of the SAN. It represents information in reinforcement learning via at most a single spike, thereby exhibiting substantial energy efficiency advantages over population-encoded and rate-encoded neurons.
- *Quantized spiking actor network (Q-SAN).* To further exploit the efficiency advantage, we implement a quantization-aware training strategy and reduce the weights of the SAN from 32-bit to low-bit representations. This operation can minimize the network's memory storage and computational overhead while maintaining satisfactory performance.
- *Spike-driven learning algorithm for the Q-SAN.* Q-SAN suffers from training challenges due to the nondifferentiability of discrete spikes and quantization operations. To address this issue, we propose a spike-driven learning algorithm for the Q-SAN, where gradient backpropagation is performed only upon spike emission, remaining inactive otherwise.
- *Reinforcement learning integration.* We integrate the soft actor-critic algorithm into our Q-SAN, where the membrane potentials of neurons in the last layer serve as outputs. This integration is a more biologically plausible way to learning representations in reinforcement learning scenarios.

A. TTFS-Encoded Spiking Neuron Model

1) TTFS Encoding Scheme

Unlike ANNs, which can directly process continuous values representing signal intensity, SNNs encode these continuous values into discrete spikes. Common encoding methods in

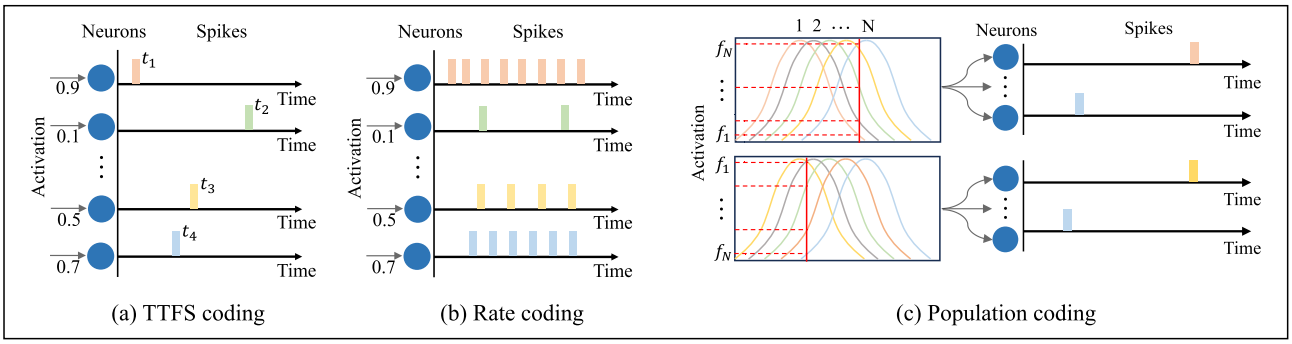

FIGURE 2 Comparison of various encoding schemes: (a) TTFS encoding uses a single neuron that emits at most one spike to represent information. (b) Rate coding involves a single neuron generating multiple spikes for encoding. (c) Population coding relies on multiple neurons for encoding.

SNNs include population coding, rate coding, and TTFS coding schemes [23].

Existing spike-based reinforcement learning studies typically adopt the population coding method, which employs a group of neurons to encode one continuous activation value [40], [41]. Each neuron in the group has its own receptive field, converting continuous values within that field into spikes over a time window. Since population coding uses multiple neurons to encode information, it requires many encoding and decoding neurons, leading to a large network size. In contrast, rate coding encodes one continuous value using only one neuron, which matches the input continuous value with the spike firing rate. The longer the time window of the spiking neuron is, the more accurately the activation value can be encoded [21]. As a result, SNNs using the rate encoding method require long simulation times and multiple spikes to ensure performance in continuous control tasks. In summary, for spike-based reinforcement learning models, neither population coding nor rate coding is the most energy-efficient encoding scheme.

The TTFS encoding scheme is a highly energy-efficient method in SNNs; it allows neurons to fire at most one spike and utilizes the firing timing of this spike to encode information. The relationship between the input activation s_i and the encoded spike time t_i in TTFS encoding scheme is defined as:

$$t_i = \left(1 - \frac{s_i}{s_{max}}\right) \times T_\omega, \tag{1}$$

where s_{max} is the maximum input value and T_ω is the length of the permissible time window. The higher the input value is, the earlier the spike time is encoded. We summarize the above three coding methods in Figure 2, where TTFS coding effectively diminishes the number of spikes and energy requirements. Thus, the TTFS a powerful coding scheme that maximizes the energy efficiency of SNNs. However, since TTFS-encoded SNNs perform gradient backpropagation only when spikes are emitted, the high sparsity offers efficiency advantages but also presents training challenges. Therefore, maintaining a certain number of active neurons in each layer is crucial for TTFS-encoded SNN learning.

2) Dynamic Firing Threshold

To address the training difficulty of TTFS-encoded SNNs caused by high sparsity, we introduce a dynamic firing threshold (DFT) mechanism. The DFT is a layer dependent, time-varying function that constrains the firing time of spike neurons in each layer to a nonoverlapping time window. It is defined as:

$$\varphi_l(t) = \begin{cases} T_\omega(l+1) - t, & \text{if } T_\omega l \le t \le T_\omega(l+1), \\ +\infty, & \text{otherwise}, \end{cases} \tag{2}$$

where $\varphi_l(t)$ denotes the firing threshold of neurons in the l-th layer at time t and T_w is the spike time window at each layer. As described in (2), the DFT causes the threshold of neurons in the l-th layer to decay linearly within the interval $[T_w l, T_w(l+1)]$, whereas the threshold is infinite outside this interval, regulating the firing activity of neurons across different layers within nonoverlapping time windows.

The DFT mechanism effectively mitigates the training challenges associated with TTFS-encoded SNNs, which can be summarized from two key characteristics of the DFT. The first characteristic is the linear decay of the DFT, which gradually reduces the firing threshold from 1 to 0. This allows neuron j to remain active as long as its membrane potential is nonnegative at $T_w(l+1)$, thereby effectively decreasing the number of inactive neurons. The second characteristic is the nonoverlapping time window. The layer-dependent firing property of the DFT ensures nonoverlapping spike time windows for TTFS-encoded SNNs. This means that whenever a presynaptic neuron fires a spike, its postsynaptic neurons will definitely receive it. As a result, neurons can receive more input information, which partially alleviates the issue of excessive sparsity.

3) Neuron Model

In this paper, we employ a DFT-based spiking neuron model that combines the rectified linear postsynaptic potential (ReL-

PSP) neuron with the DFT mechanism [31], [62]. The membrane potential of a DFT-based neuron j in layer l at time t is computed as:

$$v_j^l(t) = \sum_{i=1}^{M} w_{ij}^l \mathcal{K}(t - t_i^{l-1}), \tag{3}$$

where w_{ij}^l is the synaptic weight between neuron j and its presynaptic neuron i. M is the number of neurons in the previous layer $l-1$, t_i^{l-1} is the spike time of input neuron i, and $\mathcal{K}(t - t_i^{l-1})$ is a rectified linear kernel that is defined as follows:

$$\mathcal{K}(t - t_i^{l-1}) = \begin{cases} t - t_i^{l-1}, & \text{if } t > t_i^{l-1}, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

The DFT-based spiking neuron j generates a spike when its membrane potential $v_j^l(t)$ reaches the dynamic firing threshold $\varphi_l(t)$. Mathematically, the spike generation F is formulated as:

$$t_j^l = F\{t | v_j^l(t) \geq \varphi_l(t), t \geq 0\}. \tag{5}$$

Therefore, by combining (2) and (3), the precise firing time of the DFT-based neuron can be obtained, which falls into three categories: spikes at the earliest possible time, spikes within the allowed time window, or no spikes. Overall, the spike time t_j^l of a DFT-based neuron j in layer l is summarized as:

$$t_j^l = \begin{cases} T_\omega l, & \text{if } v_j^l(T_\omega l) \geq \varphi^l(T_\omega l), \\ \frac{T_\omega(l+1) + \sum_{i \in S_j} w_{ij}^l t_i^{l-1}}{1 + \sum_{i \in S_j} w_{ij}^l}, & \text{if } T_\omega l < t_j^l \leq T_\omega(l+1), \\ \text{nonspike}, & \text{otherwise}, \end{cases} \tag{6}$$

where S_j is the set of presynaptic neurons that emit spikes, and these neurons will contribute to neuron j. The DFT-based neuron offers significant efficiency advantages, so it is adopted as the basic unit of our Spike-DRL model.

B. Quantized Spiking Actor Network (Q-SAN)

1) TTFS-Encoded SNN Backbone

The TTFS-encoded SNN backbone is a key aspect of Q-SAN implementation. The backbone comprises synaptic layers and neuronal layers. Typically, synaptic layers can be convolutional or fully connected operations, followed by a neuronal layer. Consistent with previous studies that integrate SNNs with RL [40], we employ only fully connected layers as synaptic layers. However, our approach differs from these studies due to the utilization of the TTFS encoding strategy. Unlike the population coding scheme, which requires a group of neurons to represent the state s_t, our TTFS-encoded SNN backbone utilizes only one spiking neuron to represent s_t. This encoding method significantly reduces the overall number of neurons and spikes needed to represent complex states, thereby achieving substantial energy efficiency advantages. Moreover, for tasks with S-dimensional states, we use a backbone SNN with three layers, structured as S-256-256, where each layer is composed of DFT-based neurons. The SNN backbone receives an input representing the current state of the environment, which is then encoded by the DFT-based spiking neuron model into the firing time of a single spike. The network's output provides the probability distribution statistics of actions, i.e., the mean and variance. Notably, (2) restricts neurons in each layer to fire within a specified time window. To enhance performance, we use the membrane potential at the beginning of the time window as the output in the last layer. As described above, in the context of reinforcement learning, where an agent interacts with an environment to learn optimal behavior, this TTFS-encoded SNN backbone offers a promising approach for balancing computational efficiency and performance in complex decision-making processes.

2) Quantization Strategy

Weight quantization is another key aspect of Q-SAN implementation. This operation can further reduce memory and computational requirements, making it especially important for resource-constrained edge environments. Despite its potential benefits, weight quantization has not yet been explored in the domain of spike-based reinforcement learning. The Spike-DRL framework is designed to bridge this gap.

Quantization methods are typically categorized into two primary approaches: posttraining quantization (PTQ) and quantization-aware training (QAT) [71], [72], [73]. In PTQ, a pretrained model is calibrated using a small subset of training data to determine the clipping ranges and scaling factors. As PTQ quantizes the model and adjusts the weights without any fine-tuning, its computational overhead is minimal and often negligible. However, this simplicity often comes at the cost of lower accuracy, particularly in the case of low-precision quantization. In contrast, QAT incorporates quantization operations during the model training process, effectively mitigating quantization errors throughout the training process. As a result, QAT typically allows models to use low-bit weights to achieve satisfactory performance [74]. This motivates us to adopt the QAT in our proposed Q-SAN backbone. We employ uniform quantization for the synaptic weights because of its simplicity, low computational cost, and suitability for efficient hardware implementation. For a 32-bit weight w, the uniform quantization is defined as follows:

$$Q(w) = \lfloor s(n) \cdot \text{clamp}\left(\frac{w}{\alpha}; -1, 1\right) \rceil, \tag{7}$$

$$\text{clamp}\left(\frac{w}{\alpha}; -1, 1\right) = \min\left(\max\left(\frac{w}{\alpha}, -1\right), 1\right), \tag{8}$$

where n denotes the number of bits used for quantization (e.g., 4, 8, and 16 bits), and the scalar $s(n) = 2^{n-1} - 1$. $\text{clip}(\cdot)$ is a clipping operator used to constrain the weights within the interval $[-1, 1]$, and $\lfloor \cdot \rceil$ maps the continuous floating-point weights to their nearest discrete integer

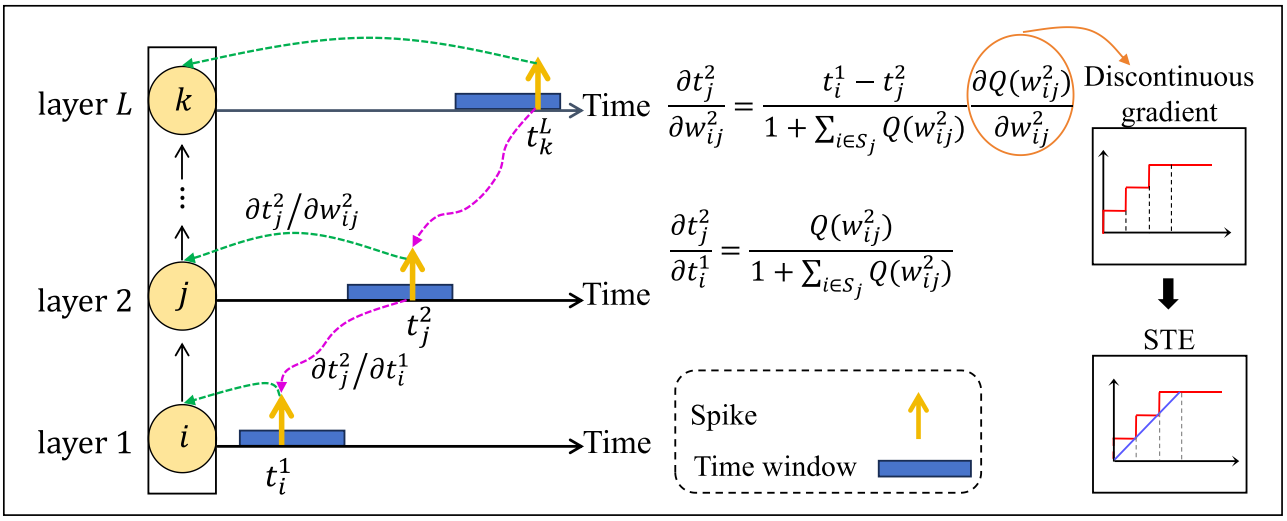

FIGURE 3 The proposed spike-driven learning algorithm for the Q-SAN model. This algorithm propagates gradients only when spikes are emitted and remains inactive otherwise, leading to significant efficiency advantages during the training process.

representations. α is a channelwise scaling factor for mitigating quantization errors, it is calculated as the average of the absolute value of weights in each output channel [75], [76]. This quantization operation maps 32- bits w onto the signed integer grid $\{-2^{n-1}+1, \ldots, 2^{n-1}-1\}$. To reconstruct w through their quantized counterparts $Q(w)$, the dequantization is defined as follows:

$$\hat{w} = \frac{Q(w)}{s(n)} \times \alpha. \tag{9}$$

By integrating the above two formulas, the general definition of quantization is as follows:

$$w \approx \hat{w} = \frac{\lfloor s(n) \cdot \text{clamp}\left(\frac{w}{\alpha}; -1, 1\right) \rceil}{s(n)} \times \alpha. \tag{10}$$

By applying this uniform quantization formula during training, the efficiency advantages of the Q-SAN are further exploited. This approach allows our Spike-DRL framework to offer an effective and energy-efficient brain-inspired solution for continuous control tasks. Although quantization offers substantial efficiency benefits for Spike-DRL, it introduces training challenges due to nondifferentiability. To leverage the widely used backpropagation algorithm, it is required to address this issue during the backpropagation process.

C. Spike-Driven Learning Algorithm for the Q-SAN

We propose a spike-driven learning algorithm for the Q-SAN, that updates the synaptic weights only when spikes are emitted, optimizing both energy efficiency and memory usage in the training process. The spike-driven learning algorithm minimizes the loss function by adjusting the spike times during gradient backpropagation. The key to this process is to calculate the derivatives of the output spike time t_j^l with respect to the synaptic weight w_{ij}^l and the input spike time t_i^{l-1}. Based on the chain rule in [31], these two derivatives can be described as:

$$\frac{\partial t_j^l}{\partial w_{ij}^l} = \begin{cases} \frac{t_i^{l-1} - t_j^l}{1 + \sum_{i \in S_j} Q(w_{ij}^l)} \frac{\partial Q(w_{ij}^l)}{\partial w_{ij}^l}, & \text{if } T_w l < t_j^l < T_w(l+1), \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

$$\frac{\partial t_j^l}{\partial t_i^{l-1}} = \begin{cases} \frac{Q(w_{ij}^l)}{1 + \sum_{i \in S_j} Q(w_{ij}^l)}, & \text{if } T_w l < t_j^l < T_w(l+1), \\ 0, & \text{otherwise}, \end{cases} \tag{12}$$

where $Q(w_{ij}^l)$ represents the low-bit quantization of the synaptic weight w_{ij}^l. These equations form the basis for gradient-based learning in TTFS-encoded SNNs and result in effective backpropagation of error signals, allowing precise adjustment of synaptic weights. This spike-driven approach maintains the benefits of temporal coding while potentially improving both accuracy and energy efficiency.

The term $\frac{\partial Q(w_{ij}^l)}{\partial w_{ij}^l}$ in the derivative of the output spike to the synaptic weight is the gradient of the quantization function. For low-bit quantization, this operation is nondifferentiable because of its discrete nature. To address this issue, the straight through estimator (STE) [77] is employed:

$$\frac{\partial Q(w_{ij}^l)}{\partial w_{ij}^l} \approx \begin{cases} 1, & \text{if } |w_{ij}^l| \le 1, \\ 0, & \text{otherwise}. \end{cases} \tag{13}$$

This surrogate gradient allows for the backpropagation of error signals through the quantization function, leading to effective training of the network with low-bit quantized weights. The overall process of this algorithm is shown in Figure 3.

In TTFS coding, earlier spikes represent stronger signals. By negating the spike times before inputting them into loss functions, we transform this relationship so that larger (more positive) values correspond to earlier spikes. This conversion

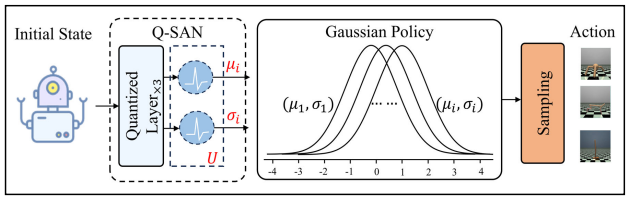

FIGURE 4 Structure of the spike-based DRL: The initial state is input into the Q-SAN, which outputs the membrane potential, denoted as (μ, σ). A sampling method is subsequently employed to determine the next action.

makes the optimization process more intuitive and aligns with common loss function designs, where larger values typically indicate better performance.

D. Maximum-Entropy RL Paradigm With SNNs

As illustrated in Figure 4, we present the specific workflow of the proposed Spike-DRL. The initial state is encoded using the TTFS method and input into the Q-SAN. The TTFS encoding paradigm facilitates the processing of different inputs in an event-driven manner, thereby ensuring the model's low-power characteristics. Moreover, the Q-SAN enables the model's parameters to be stored in a low-bit format, facilitating deployment on resource-constrained devices. Importantly, the Q-SAN output serves as the Gaussian policy's μ and σ. To further enhance the representational capacity of the Gaussian policy, the membrane potential information from the final layer is utilized as the output, which offers superior representational capabilities compared with those of discrete pulse information. Ultimately, we sample from the generated Gaussian distribution to determine the action state next. More details can be found in Algorithm 1.

The DRL problem is commonly expressed as a Markov decision process (MDP), represented by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, r, \gamma)$. In this context, an RL agent interacts with the environment by observing a state $s_t \in \mathcal{S}$ within the state space $\mathcal{S}$ and choosing an action $a_t \in \mathcal{A}$ from the action space $\mathcal{A}$ on the basis of the policy $\pi(a_t \mid s_t)$. The agent then receives a reward $r(s_t, a_t)$, and the environment transitions to a new state $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$ according to the transition probability.

The primary purpose of the standard RL agent is to maximize the discounted expected total reward received from the environment. In addition, to enhance the exploratory quality of an agent's decision-making in an environment, a more general objective known as the maximum entropy objective is often included in the optimization target within an actor-critic framework to construct the agent's policy [78]. This augmented reinforcement learning optimization objective requires not only the maximization of the cumulative reward function but also the maximization of the expected entropy of the policy over $\rho_\pi(s_t)$:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} [r(\mathbf{s}_t, \mathbf{a}_t) + \alpha \mathcal{H}(\pi(\cdot \mid \mathbf{s}_t))], \tag{14}$$

Algorithm 1. Soft Actor-Critic Algorithm in Spike-DRL.

Initialize actor network $\pi_\phi(a|s)$, critic networks $Q_{\theta_1}(s,a)$, $Q_{\theta_2}(s,a)$
Initialize target critic networks $Q_{\theta'_1}(s,a) \leftarrow Q_{\theta_1}(s,a)$, $Q_{\theta'_2}(s,a) \leftarrow Q_{\theta_2}(s,a)$
Initialize replay buffer $\mathcal{D}$
for each episode **do**
  Select action $a_t \sim \pi_\phi(v, \sigma|s_t)$, where v, σ are calculated by the membrane potential of spiking neurons
  Execute action a_t in the environment
  Observe next state s_{t+1}, reward r, and done signal d
  Store transition tuple $(s_t, a_t, r, s_{t+1}, d)$ in $\mathcal{D}$
  if $\mathcal{D}$ is sufficiently large **then**
    for each gradient step **do**
      Sample minibatch $\{(s_t, a_t, r, s_{t+1}, d)\}$ from $\mathcal{D}$
      $a_{t+1} \sim \pi_\phi(v', \sigma'|s_{t+1})$, where v', σ' are calculated by the membrane potential of spiking neurons
      Compute targets for the Q functions:
      $y_i(r, s_{t+1}, d) = r + \gamma(1-d)Q_{\theta'_i}(s_t, a_t) - \alpha log\pi(a_{t+1}|s_{t+1})$ for i=1 and 2
      $y = min\{y_1, y_2\}$
      Update critics θ_i by minimizing:
      $\frac{1}{|B|}\sum_{(s_t, a_t, r, s_{t+1}, d) \in B}(Q_{\theta_i}(s_t, a_t) - y(r, s_{t+1}, d))^2$
      Update actor ϕ by the policy gradient:
      $\frac{1}{|B|}\sum_{s \in B, a \sim \pi} \nabla_\phi [\alpha log\pi(a|s_t) - \log \pi_\phi(a|s_t) Q_{\theta_1}(s_t, a)]$
      Update target networks:
      $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$
    end for
  end if
end for

where γ is the discount factor $\rho_\pi(s_t, a_t)$ indicates the state and state-action marginals of the trajectory distribution induced by the policy $\pi(a_t \mid s_t)$. The entropy term $\mathcal{H}(\pi(\cdot \mid \mathbf{s}_t))$ regulates the degree of uncertainty in the agent's policy with the temperature parameter α, which determines the relative importance of the entropy term against the reward and thus controls the stochasticity of the optimal policy.

In the aforementioned setting, a single-agent off-policy algorithm, the SAC algorithm [56], was introduced and quickly became the state-of-the-art baseline for addressing continuous action space problems in RL. The SAC uses an actor-critic architecture with separate policy and Q value networks $Q_\theta(s, a)$ to iterate the soft policy $\pi_\phi(a|s)$ as follows:

$$\mathcal{T}^\pi Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}}[V(s_{t+1})], \tag{15}$$

where $\mathcal{T}^\pi$ indicates the soft Bellman backup update operator applied to the function $Q(s_t, a_t)$. The soft state value function is depicted as:

$$V(s_{t+1}) = Q(s_{t+1}, a_{t+1}) - \alpha log\pi(a_{t+1}|s_{t+1}). \tag{16}$$

The optimization process employs bootstrapping, sampling from the replay buffer $\mathcal{D}$, and updating the Q value networks

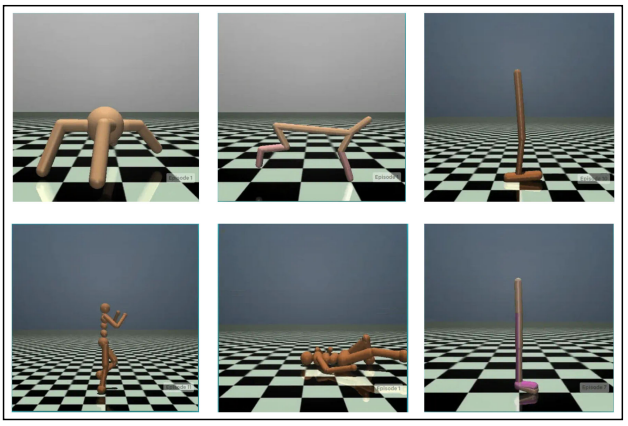

FIGURE 5 This experiment is conducted in the MuJoCo (Multijoint Dynamics with Contact) environment, a physics simulator widely used for research in robotics control optimization. The figure shows simulations for Ant-v3, HalfCheetah-v3, Hopper-v3, Humanoid-v3, HumanoidStandup-v2, and Walker2d-v3. These tasks are commonly used to evaluate the efficacy of various control algorithms in complex, dynamic environments.

using the calculated Bellman residual as follows:

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \mathcal{T}^\pi Q_\theta(s_t,a_t)\right)^2\right]. \tag{17}$$

The policy network π_ϕ can be learned by minimizing the discrepancy between the distributions of old and new policy. The final optimization objective can be formulated as:

$$\underset{\pi}{argmin}\, \mathbb{E}_{s\sim\mathcal{D},a\sim\pi(a|s)}\alpha log\pi(a|s) - Q_\theta(s,a). \tag{18}$$

In the development of our Spike-DRL model, we employ the Q-SAN architecture to serve as the actor network, leveraging a spike-driven learning algorithm for the mapping of state–action pairs. This integration of spiking neural mechanisms within the actor framework leads to a more biologically plausible approach to learning representations in reinforcement learning scenarios. The remainder of the SAC reward strategy remains unchanged, preserving the integrity and robustness of the original framework.

IV. Experiment

As shown in Figure 5, we present comprehensive experimental results demonstrating the superior performance of our Spike-DRL framework. Initially, we detail the experimental setup, including the specific hyperparameters used in our study. We subsequently perform a comparative analysis of our model against state-of-the-art (SOTA) methods across various continuous state space tasks. Following this, through a series of extensive ablation studies, we validate the effectiveness of our proposed Q-SAN, emphasizing its crucial role in enhancing the overall system performance. Finally, we thoroughly compare the performance, memory usage, and average energy consumption of different methods, highlighting the optimal balance among them.

A. Experimental setting

In this study, we utilized OpenAI Gym environments, including Ant, HalfCheetah, Hopper, Walker2d, Humanoid, and HumanoidStandup to thoroughly evaluate the effectiveness of our proposed methodology. These tasks are well-established benchmarks in the field of continuous control, providing a robust framework for assessing performance across diverse and complex scenarios. To ensure the reproducibility and statistical significance of our findings, each model was trained several times, each with a different random seed, allowing us to account for variability and confirming the robustness of our approach.

- **Ant-v3:** involves a four-legged robot that needs to coordinate its limbs for efficient movement. It has a state space of 111 dimensions and an action space of 8 dimensions.
- **HalfCheetah-v3:** simulates a two-dimensional cheetah robot, aiming to maximize forward velocity. The environment has a state space of 17 dimensions and an action space of 6 dimensions.
- **Hopper-v3:** involves a one-legged robot that must learn to hop or balance. It features a state space of 11 dimensions and an action space of 3 dimensions.
- **Walker2d-v3:** focuses on a bipedal robot that is tasked with walking forward as efficiently as possible. This environment includes a state space of 17 dimensions and an action space of 6 dimensions.
- **Humanoid-v3:** simulates a three-dimensional humanoid robot designed to achieve bipedal locomotion. The environment features a state space of 376 dimensions and an action space of 17 dimensions.
- **HumanoidStandup-v2:** focuses on a three-dimensional humanoid robot that starts from a prone position and aims to stand up. This environment features a state space of 376 dimensions and an action space of 17 dimensions.

To ensure reproducibility, each model is trained for ten rounds with distinct random seeds. Each round consists of 1 million training steps, with evaluations conducted every 10,000 steps. Each evaluation reports the average reward over ten episodes using a deterministic policy. Each episode is capped at a maximum of 1,000 steps, allowing for a controlled and comprehensive assessment of the model performance. Throughout all the experiments, we apply the SAC algorithm to train the actor networks. The hyperparameter configuration for our models is as follows: the actor network has a learning rate set at 1×10^{-3}; the reward discount factor is set at 0.99; Gaussian exploration noise with a standard deviation of 0.1 is used; Gaussian smoothing noise for the target policy is set at 0.2, with noise constrained to a maximum of 0.5; and the policy delay factor is set at 2.

B. Comparison With SOTA models

We use the average rewards from the ANN-based SAC and TD3 methods [48], [56], [79] across various tasks as a

TABLE I Max average rewards over 10 random seeds for different methods (32bits, 16bits, 8bits, and 4 bits).

	Ant-v3	HalfCheetah-v3	Hopper-v3	Walker2d-v3	Humanoid-v3	HumanoidStandup-v2
SAC [56], [79]	5714 ± 368	12122 ± 1006	3532 ± 42	4961 ± 327	5416 ± 214	129271 ± 14321
TD3 [48], [79]	5116 ± 235	10201 ± 1246	3564 ± 114	5007 ± 539	5488 ± 457	132798 ± 24864
Pop-SAN [41]	4848 ± 1023	10523 ± 1142	517 ± 1057	4199 ± 1426	5821 ± 440	155387 ± 21632
DNN-SNN [41]	3026± 1718	10722 ± 806	2523 ± 1346	4546 ± 1033	5104 ± 396	132415 ± 21451
DSRL [69]	3820 ± 942	3220 ± 1857	2713 ± 1424	3208 ± 1514	135 ± 123	73620 ± 33564
AC-BCQ [70]	4070 ± 34	10610 ± 63	2527 ± 455	2551 ± 932	2276 ± 334	145311 ± 16932
Spike-DRL W_{32Bit}	5354 ± 276	11394 ± 1784	3351 ± 282	4817 ± 868	5272 ± 484	127631 ± 27328
Spike-DRL W_{16Bit}	4827 ± 534	10322 ± 1409	3164 ± 193	4721 ± 573	4978 ± 972	121835 ± 22402
Spike-DRL W_{8Bit}	4153 ± 963	9457 ± 1021	2874 ± 384	4378 ± 628	4367 ± 652	111973 ± 25112
Spike-DRL W_{4Bit}	3173 ± 928	8274 ± 1773	2164 ±683	3672 ± 612	3842 ± 843	105879 ± 24132

benchmark and compare our spike-DRL model against the latest SNN-based methods [41], [69], [70]. Notably, owing to the unavailability of the official version and source code for the AC-BCQ method [70], we replicate it on the basis of the original publication, ensuring that it shares the same actor network architecture as our Spike-DRL. As presented in Table I, Spike-DRL demonstrates strong competitiveness with full-precision weights. Although its performance is slightly lower than that of SAC [56] and TD3 [48] and Pop-SAN [41] in some cases, SAC and TD3 require significantly higher computational overhead due to full-precision MAC operations, whereas Pop-SAN imposes an additional system burden because of its 10-fold increase in the number of neuron encoding states. Notably, compared with other SNN-based tasks, our Spike-DRL exhibits strong competitiveness, even under 8-bit quantization conditions. These results indicate that our Spike-DRL model not only significantly reduces storage requirements and power consumption but also demonstrates commendable performance, underscoring its potential for practical deployment in resource-constrained environments.

C. Ablation Study

To assess the effectiveness of the proposed approaches, we perform ablation studies, examining the impact of the Q-SAN schemes. We conduct experiments with various precision levels (16-bit, 8-bit, and 4-bit) and compare them with their full-precision counterparts, as illustrated in Figures 6(a) and (b). For the Ant-v3, the 16-bit and 8-bit methods maintain strong performance and stability, whereas the 4-bit precision results in lower average rewards and reduced stability. For HalfCheetah-v3, the trends observed align with those of Ant-v3. Overall, the 8-bit precision shows performance comparable to the full-precision model, offering a favorable trade-off between performance and resource consumption.

D. Energy Efficiency Analysis

To thoroughly evaluate the energy efficiency of the proposed Spike-DRL, we assess energy consumption under different architectures and encoding schemes. Specifically, we compare the total number of Synops executed by the actor-network in the HalfCheetah-v3 environment. For ANN-based models,

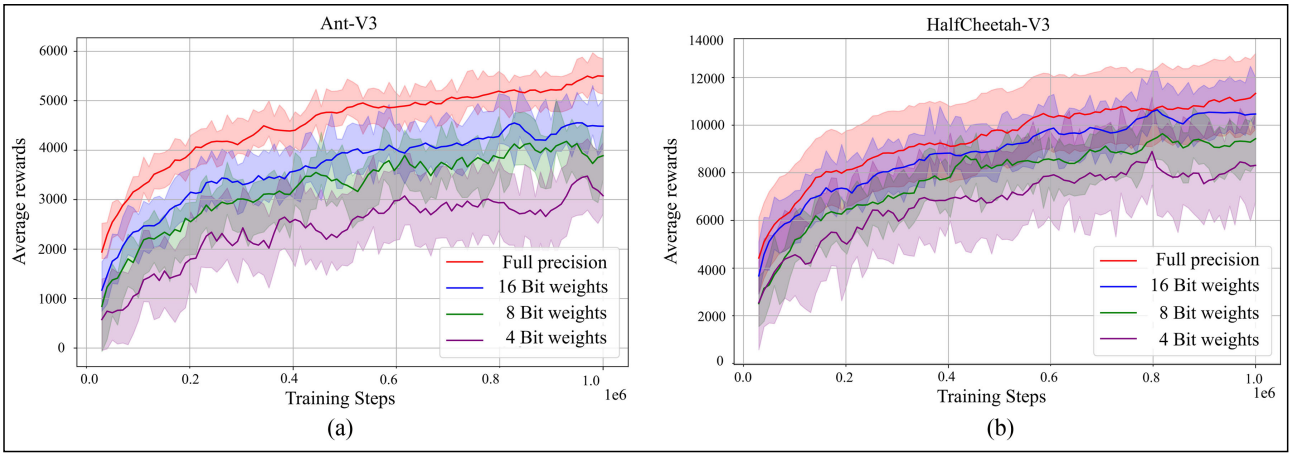

FIGURE 6 Ablation experiments for our proposed Spike-DRL model with weight quantization methods. (a), (b) show the results of the quantized models with different precisions and their full-precision counterparts on Ant-V3 and HalfCheetah-V3, respectively.

TABLE II Comparison of hardware energy consumption for different models in HalfCheetah-v3.

METHOD	TYPE	ENCODING	PARAMETER	APR	AVERAGE ENERGY CONSUMPTION (AEC)			ENERGY RATIO
					MAC	AC	COST	
SAC [56]	ANN	-	72.96 K	100.00%	72.96 K	-	335.6 nJ	0.30
TD3 [48]	ANN	-	72.96 K	84.15%	72.96 K	-	335.6 nJ	0.25
Pop-SAN [41]	SNN	Population Encoding	177.83 k	86.81%	0.45 K	95.66 K	88.6 nJ	0.98
AC-BCQ [70]	SNN	Direct Encoding	73.25 K	87.53%	19.6 K	13.73 K	102.5 nJ	0.85
DSRL [69]	SNN	Population Encoding	77.35 K	26.56%	0.14 k	72.85 K	66.21 nJ	0.40
Spike-DRL	SNN	TTFS Encoding	71 K	78.02%	-	18.56 k	16.71 nJ	4.70

including SAC [56] and TD3 [48], the total number of Synops of the actor-network can be computed as:

$$E_{\text{ANN}} = \text{MAC} \cdot \sum_{l=1}^{L} \sum_{j=1}^{N_l} f_{\text{in, j}}^{l}, \tag{19}$$

where $f_{\text{in},j}^{l}$ denotes the fan-in count for the j-th neuron in the l-th layer, N_l is the number of neurons in the l-th layer, and L is the total number of layers in the network. Since SAC and TD3 share the same actor-network, their energy consumption is identical. As shown in Table II, the energy demand of these models is 335.6 nJ.

In addition to SAC, other SNN-based reinforcement learning frameworks employ various combinations of encoding layers and actor networks. Specifically, AC-BCQ [70] employs learnable matrix encoding, which can be expressed as follows:

$$E_{\text{DC}} = \text{MAC} \cdot \sum_{t=1}^{T} \sum_{j=1}^{N_1} f_{\text{in},j}^{1} + \text{AC} \cdot \sum_{t=1}^{T} \sum_{l=2}^{L} \sum_{j=1}^{N_l} f_{\text{in},j}^{l} \cdot o_j^l[t]. \tag{20}$$

$o_j^l[t]$ represents the output of the j-th neuron in the l-th layer at time step t, T is the total number of time steps. Additionally, both Pop-SAN [41] and DSRL [69] employ the population encoding method. This approach does not introduce floating-point operations within the network. Instead, it encodes state information S using approximately 10–20 encoding neurons P. Consequently, its energy consumption can be expressed as follows:

$$E_{\text{Pop}} = \text{MAC} \times S \times P + \text{AC} \cdot \sum_{t=1}^{T} \sum_{l=1}^{L} \sum_{j=1}^{N_l} f_{\text{in},j}^{l} \cdot o_j^l[t]. \tag{21}$$

Finally, our Spike-DRL promotes energy conservation by exclusively utilizing full spike-based transmissions and computations:

$$E_{\text{Spike-DRL}} = \text{AC} \cdot \sum_{t=1}^{T} \sum_{l=1}^{L} \sum_{j=1}^{N_l} f_{\text{in},j}^{l} \cdot o_j^l[t]. \tag{22}$$

To ensure fairness as much as possible, we use a uniform three-layer linear architecture as the baseline for all methods. For the SNN-based RL methods, the timesteps are all set to 4. Moreover, $P = 25$ in Pop-SAN and $P = 2$ in DSRL. Each MAC operation consumes 4.6 pJ, and each AC operation consumes 0.9 pJ (for 64-bit precision) [80], [81], [82], [83]. For lower bit-widths, the values are smaller. Additionally, the spike firing rates were 13.48% in Pop-SAN, 16.71% in AC-BCQ, and 23.56% in DSRL. The average energy consumption (AEC) is shown in Table II. Our findings indicate that Spike-DRL offers substantial energy savings, as it approximately 20× and 5× more efficient than the traditional ANN models and Pop-SAN, respectively. The significant improvement in energy efficiency highlights the advantages of our Spike-DRL approach.

E. Performance–Energy Ratio

Spike-DRL is designed to achieve the best trade-off between low energy consumption and high performance. Therefore, the energy-performance ratio serves as the primary metric for evaluating the performance of our Spike-DRL. To facilitate the comparison of performance across different methods, we initially propose the average performance ratio (APR), which can be described as:

$$\text{APR}_{\mathcal{T}} = \frac{\text{M}_{\mathcal{T}}}{\text{SAC}_{\mathcal{T}}}. \tag{23}$$

For the task $\mathcal{T}$, the maximum average reward for it is represented by $\text{M}_{\mathcal{T}}$. M can refer to any network, including TD3 [48], Pop-SAN [41], AC-BCQ [70], DSRL [84], and Spike-DRL(Ours). This value is determined by evaluating the performance of a given actor network over 10 different random seeds and selecting the highest average reward. $\text{SAC}_{\mathcal{T}}$ [56] serves as the baseline for comparison. Building upon the ARP and energy consumption, we introduce a straightforward calculation paradigm for the performance–energy ratio (PER), defined as:

$$\text{PER} = \frac{\text{APR (\%)}}{\text{AEC (nJ)}}, \tag{24}$$

where APR measures the average performance ratios, and AEC indicates the average energy consumption. A higher PER signifies better relative performance with fixed energy consumption. On the basis of ablation studies showing that 8-bit encoding optimally balances memory usage and performance, we choose 8-bit Spike-DRL for our comparative analysis. As demonstrated in Table II, our model achieves an impressive PER of 4.70. This significant metric demonstrates that our model achieves an optimal trade-off between energy consumption and performance. Therefore, Spike-DRL is well-suited for applications on edge devices.

V. Conclusion

In this work, we developed Spike-DRL that leverages the energy-efficient properties of SNNs for control tasks in resource-constrained environments. By encoding information through precise spike timing and employing a lightweight strategy to quantize synaptic weights, Spike-DRL operates within a low-power, event-driven paradigm, reducing memory requirements and computational complexity. Extensive experiments demonstrate that Spike-DRL achieves competitive performance compared with existing SNN-based reinforcement learning models, advancing energy-efficient, biologically plausible systems suitable for real-time decision-making and autonomous learning.

In future work, we plan to enhance Spike-DRL by developing more efficient encoding methods, effective learning algorithms, and powerful computational models. This work includes exploring advanced spike-based encoding techniques to improve information processing and refining learning algorithms to better integrate with reinforcement learning frameworks, such as using reward-modulated STDP and surrogate gradient methods. We also aim to optimize the architecture of spiking networks for scalability and efficiency, particularly for neuromorphic hardware. These improvements will enable Spike-DRL to handle more complex tasks and extend its applicability to real-world scenarios, such as autonomous robotics, smart prosthetics, and edge computing systems, where energy efficiency and real-time decision-making are crucial.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant U20B2063, Grant 62220106008, and Grant 62106038, in part by Sichuan Science and Technology Program under Grant 2024NSFTD0034 and Grant 2023YFG0259, and in part by the Open Research Fund of the State Key Laboratory of Brain-Machine Intelligence, Zhejiang University under Grant BMI2400020.

References

[1] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*.
[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.
[3] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming and reinforcement learning on learning heuristics for dynamic scheduling: A preliminary comparison," *IEEE Comput. Intell. Mag.*, vol. 19, no. 2, pp. 18–33, May 2024.
[4] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Niching genetic programming to learn actions for deep reinforcement learning in dynamic flexible scheduling," *IEEE Trans. Evol. Comput.*, early access, May 01, 2024, doi: 10.1109/TEVC.2024.3395699.
[5] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: A survey," in *Proc. 2020 IEEE Symp. Ser. Comput. Intell.*, 2020, pp. 737–744.
[6] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. 2017 IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3389–3396.
[7] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A survey of deep reinforcement learning in video games," 2019, *arXiv:1912.10944*.
[8] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. Perez-Liebana, "Deep reinforcement learning for general video game AI," in *Proc. 2018 IEEE Conf. Comput. Intell. Games*, 2018, pp. 1–8.
[9] Y. Wu, S. Liao, X. Liu, Z. Li, and R. Lu, "Deep reinforcement learning on autonomous driving policy with auxiliary critic network," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 7, pp. 3680–3690, Jul. 2023.
[10] A. R. Sharma and P. Kaushik, "Literature survey of statistical, deep and reinforcement learning in natural language processing," in *Proc. 2017 Int. Conf. Comput., Commun. Automat.*, 2017, pp. 350–354.
[11] J. He et al., "Deep reinforcement learning with a natural language action space," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, 2016, pp. 1621–1630.
[12] M. Xi et al., "A lightweight reinforcement learning-based real-time path planning method for unmanned aerial vehicles," *IEEE Internet Things J.*, vol. 11, no. 12, pp. 21061–21071, Jun. 2024.
[13] H. Li, R. Cai, N. Liu, X. Lin, and Y. Wang, "Deep reinforcement learning: Algorithm, applications, and ultra-low-power implementation," *Nano Commun. Netw.*, vol. 16, pp. 81–90, 2018.
[14] K. Nemoto and H. Matsutani, "A lightweight reinforcement learning based packet routing method using online sequential learning," *IEICE Trans. Inf. Syst.*, vol. 106, no. 11, pp. 1796–1807, 2023.
[15] E. M. Izhikevich, "Simple model of spiking neurons," *IEEE Trans. Neural Netw.*, vol. 14, no. 6, pp. 1569–1572, 2003.
[16] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge, U.K.: Cambridge Univ. Press, 2002.
[17] W. Maass and H. Markram, "On the computational power of circuits of spiking neurons," *J. Comput. Syst. Sci.*, vol. 69, no. 4, pp. 593–616, 2004.
[18] C. Hong et al., "SPAIC: A spike-based artificial intelligence computing framework," *IEEE Comput. Intell. Mag.*, vol. 19, no. 1, pp. 51–65, Feb. 2024.
[19] J. Hu, H. Tang, K. C. Tan, and H. Li, "How the brain formulates memory: A spatio-temporal model research frontier," *IEEE Comput. Intell. Mag.*, vol. 11, no. 2, pp. 56–68, May 2016.
[20] C. Eliasmith and C. H. Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. Cambridge, MA, USA: MIT Press, 2003.
[21] A. K. Seth, "Neural coding: Rate and time codes work together," *Curr. Biol.*, vol. 25, no. 3, pp. R110–R113, 2015.
[22] T. Gollisch and M. Meister, "Rapid neural coding in the retina with relative spike latencies," *Science*, vol. 319, no. 5866, pp. 1108–1111, 2008.
[23] D. Auge, J. Hille, E. Mueller, and A. Knoll, "A survey of encoding techniques for signal processing in spiking neural networks," *Neural Process. Lett.*, vol. 53, no. 6, pp. 4693–4710, 2021.
[24] 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.*, vol. 34, no. 1, pp. 446–460, Jan. 2023.
[25] J. Wu et al., "Progressive tandem learning for pattern recognition with deep spiking neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 11, pp. 7824–7840, Nov. 2022.
[26] P. U. Diehl, G. Zarrella, A. Cassidy, B. U. Pedroni, and E. Neftci, "Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware," in *Proc. 2016 IEEE Int. Conf. Rebooting Comput.*, 2016, pp. 1–8.
[27] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Front. Neurosci.*, vol. 12, 2018, Art. no. 331.
[28] 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.*, 2019, vol. 33, no. 01, pp. 1311–1318.
[29] S. Deng, Y. Li, S. Zhang, and S. Gu, "Temporal efficient training of spiking neural network via gradient re-weighting," 2022, *arXiv:2202.11946*.
[30] W. Wei et al., "Event-driven learning for spiking neural networks," 2024, *arXiv:2403.00270*.
[31] W. Wei, M. Zhang, H. Qu, A. Belatreche, J. Zhang, and H. Chen, "Temporal-coded spiking neural networks with dynamic firing threshold: Learning with event-driven backpropagation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 10518–10528.
[32] I.-M. Comşa, K. Potempa, L. Versari, T. Fischbacher, A. Gesmundo, and J. Alakuijala, "Temporal coding in spiking neural networks with alpha synaptic

function: Learning with backpropagation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5939–5952, Oct. 2022.
[33] M. Yao et al., "Attention spiking neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 8, pp. 9393–9410, Aug. 2023.
[34] R.-J. Zhu, M. Zhang, Q. Zhao, H. Deng, Y. Duan, and L.-J. Deng, "TCJA-SNN: Temporal-channel joint attention for spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 3, pp. 5112–5125, Mar. 2025.
[35] Z. Zhou et al., "SpikFormer: When spiking neural network meets transformer," 2022, *arXiv:2209.15425*.
[36] X. Xing et al., "SpikeLLM: Scaling up spiking neural network to large language models via saliency-based spiking," 2024, *arXiv:2407.04752*.
[37] D. Chen, P. Peng, T. Huang, and Y. Tian, "Deep reinforcement learning with spiking Q-learning," 2022, *arXiv:2201.09754*.
[38] M. Yuan, X. Wu, R. Yan, and H. Tang, "Reinforcement learning in spiking neural networks with stochastic and deterministic synapses," *Neural Computation*, vol. 31, no. 12, pp. 2368–2389, 2019.
[39] D. Vlasov et al., "Memristor-based spiking neural network with online reinforcement learning," *Neural Netw.*, vol. 166, pp. 512–523, 2023.
[40] D. Chen, P. Peng, T. Huang, and Y. Tian, "Fully spiking actor network with intralayer connections for reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 2, pp. 2881–2893, Feb. 2025.
[41] 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.
[42] D. Zhang, T. Zhang, S. Jia, and B. Xu, "Multi-sacle dynamic coding improved spiking actor network for reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 1, pp. 59–67.
[43] X. Yang et al., "Hierarchical reinforcement learning with universal policies for multistep robotic manipulation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 4727–4741, Sep. 2022.
[44] G. Xiang and J. Su, "Task-oriented deep reinforcement learning for robotic skill acquisition and control," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 1056–1069, Feb. 2021.
[45] R. Akrour, D. Tateo, and J. Peters, "Continuous action reinforcement learning from a mixture of interpretable experts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 10, pp. 6795–6806, Oct. 2022.
[46] Z. Zhang, Y.-S. Ong, D. Wang, and B. Xue, "A collaborative multiagent reinforcement learning method based on policy gradient potential," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 1015–1027, Feb. 2021.
[47] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[48] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*. PMLR, 2018, pp. 1587–1596.
[49] J. Zhang, J. Kim, B. O'Donoghue, and S. Boyd, "Sample efficient reinforcement learning with reinforce," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 12, pp. 10887–10895.
[50] V. Mnih et al., "Playing Atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.
[51] L. Espeholt, R. Marinier, P. Stanczyk, K. Wang, and M. Michalski, "SEED RL: Scalable and efficient deep-RL with accelerated central inference," 2019, *arXiv:1910.06591*.
[52] C. Jin, Z. Allen-Zhu, S. Bubeck, and M. I. Jordan, "Is Q-learning provably efficient?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 4868–4878.
[53] S. Han and Y. Sung, "Diversity actor-critic: Sample-aware entropy regularization for sample-efficient exploration," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 4018–4029.
[54] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," in *Proc. Robot.: Sci. Syst.*, 2012, pp. 1–8.
[55] Z. Zheng, J. Oh, and S. Singh, "On learning intrinsic rewards for policy gradient methods," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, vol. 31, pp. 1861–1870.
[56] 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.
[57] Y. Wang, M. Zhang, Y. Chen, and H. Qu, "Signed neuron with memory: Towards simple, accurate and high-efficient ANN-SNN conversion," in *Proc. Int. Joint Conf. Artif. Intell.*, 2022, pp. 2501–2508.
[58] Z. Hao, J. Ding, T. Bu, T. Huang, and Z. Yu, "Bridging the gap between ANNs and SNNs by calibrating offset spikes," 2023, *arXiv:2302.10685*.
[59] 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.
[60] Q. Yang, J. Wu, M. Zhang, Y. Chua, X. Wang, and H. Li, "Training spiking neural networks with local tandem learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, vol. 35, pp. 12662–12676.
[61] H. Mostafa, "Supervised learning based on temporal coding in spiking neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 7, pp. 3227–3235, Jul. 2018.
[62] 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.
[63] Y. Zhu, Z. Yu, W. Fang, X. Xie, T. Huang, and T. Masquelier, "Training spiking neural networks with event-driven backpropagation," in *Proc. 36th Conf. Neural Inf. Process. Syst.*, 2022, pp. 30528–30541.
[64] 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, 2019.
[65] 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.*, 2021, vol. 35, no. 11, pp. 9816–9824.
[66] L. Zhang, J. Cao, Y. Zhang, B. Zhou, and S. Feng, "Distilling neuron spike with high temperature in reinforcement learning agents," 2021, *arXiv:2108.10078*.
[67] 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. 2020 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 6090–6097.
[68] 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, Nov. 2022.
[69] M. Akl, D. Ergene, F. Walter, and A. Knoll, "Toward robust and scalable deep spiking reinforcement learning," *Front. Neurorobot.*, vol. 16, 2023, Art. no. 1075647.
[70] L. Qin, R. Yan, and H. Tang, "A low latency adaptive coding spiking framework for deep reinforcement learning," in *Proc. 32nd Int. Joint Conf. Artif. Intell.*, 2023, pp. 3049–3057.
[71] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021.
[72] Y. Guo, "A survey on methods and theories of quantized neural networks," 2018, *arXiv:1808.04752*.
[73] B. Rokh, A. Azarpeyvand, and A. Khanteymoori, "A comprehensive survey on model quantization for deep neural networks in image classification," *ACM Trans. Intell. Syst. Technol.*, vol. 14, no. 6, pp. 1–50, 2023.
[74] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. London, U.K.: Chapman and Hall/CRC, 2022, pp. 291–326.
[75] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*. Springer, 2016, pp. 525–542.
[76] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, "Binary neural networks: A survey," *Pattern Recognit.*, vol. 105, 2020, Art. no. 107281.
[77] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.
[78] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 2010.
[79] J. Weng et al., "Tianshou: A highly modularized deep reinforcement learning library," *J. Mach. Learn. Res.*, vol. 23, no. 267, pp. 1–6, 2022. [Online]. Available: http://jmlr.org/papers/v23/21-1127.html
[80] Y. Guo et al., "Ternary spike: Learning ternary spikes for spiking neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 12244–12252.
[81] Y. Hu, H. Tang, and G. Pan, "Spiking deep residual networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 8, pp. 5200–5205, Aug. 2023.
[82] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *Proc. 2014 IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, 2014, pp. 10–14, doi: 10.1109/isscc.2014.6757323.
[83] 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, 2017, Art. no. 682.
[84] Y. Zhang et al., "Adaptive safe reinforcement learning with full-state constraints and constrained adaptation for autonomous vehicles," *IEEE Trans. Cybern.*, vol. 54, no. 3, pp. 1907–1920, Mar. 2024.