# Calibrating the Converted Spiking Reinforcement Learning

Jian Song[1][0009-0002-7528-897X], Xiangfei Yang[1*], Xuetao, Zhang[2], and Donglin Wang[1*]

[1] Westlake University, Hangzhou 310024, China
[2] Xi'an Jiaotong University, Xi'an 710049, China
{songjian76,wangdonglin}@westlake.edu.cn
yxf9011@163.com
xuetaozh@xjtu.edu.cn

**Abstract.** Spiking reinforcement learning (SRL) widely receives attention due to its ultra-low power consumption. Since it is hard to train SRL directly, converting Deep Neural Network into Spiking Neural Network (DNN2SNN) has been a commonly used method to train. However, the conversion error exists with deep reinforcement learning (DRL), resulting in performance degradation of SRL. Inspired by the success of calibrated conversion method in classification tasks, we introduce this method into SRL to further improve the performance of the converted SRL. Specifically, we calibrate the number of spikes fired by converted SNN policy through adjusting the initial membrane potential. Experimental results on MuJoCo robot control tasks demonstrate the effectiveness of *Conversion + Calibration* method in SRL.

**Keywords:** Spiking Reinforcement Learning, Spiking Neural Network, Calibration.

## 1 Introduction

Deep neural networks (DNNs) have laid a solid foundation for many achievements in machine learning. Deep reinforcement learning (DRL) is the one of the beneficiaries, demonstrating the impressive progress in various reinforcement learning (RL) tasks. However, its ultra-high power consumption poses the challenges for real-world applications. Recently, spiking neural networks (SNNs) approaches the performance of DNNs in an energy-efficient manner and is considered as the most promising solution to address the ultra-high power consumption of DNNs [1], which has been verified in image classification [2,3] and natural language processing [4], etc. Therefore, spiking reinforcement learning (SRL), incorporating SNNs into reinforcement learning, may have a broader range of applications than DRL in the real world.

The ultra-low power advantage of SRL comes from event-driven mechanism of SNNs. The spiking neurons only fire spikes when the accumulated membrane potential exceeds the setting threshold. Consequently, the information sequences with binary spikes maintain highly sparse, greatly reducing the cost of the power [5,6]. However, it fails to directly train SNN because of the non-differentiability of the activation function

* Corresponding author

of spiking neuron. And three methods are commonly used to train SNN, including STDP [7], surrogate gradient [8], and DNN2SNN [9].

STDP-type methods originate from the plasticity mechanism of biological neurons but are only applied to training shallow networks so far. The surrogate gradient methods replace the non-differentiable Heaviside function with a differentiable surrogate gradient [10] and can directly train SNN through backpropagation technique, whose effectiveness has been validated in some real classification tasks [11,12]. However, the surrogate gradient methods are extremely hard to train due to the gradient vanishing and gradient explosion and need to set the long simulation time-step to approach the accuracy of DNNs. DNN2SNN methods avoids direct training of SNN by converting the weights trained by DNN with the same architecture into those of SNN. But there exists error gap between the converted SNN and DNN, resulting in the performance degradation of SNN. To reduce the gap, Hao et al. [13] further calibrates the spike sequence by adjusting the initial membrane potential after converting. The experimental results on real datasets show the effectiveness of *conversion+calibration* method in the classification tasks.

Inspired by it, we introduce the *conversion+calibration* method into SRL for the first time, first converting the policy weights trained by deep reinforcement learning (DRL) into those of SRL and then further calibrating the spikes by adjusting the initial membrane potential after converting. Specifically, during conversion phase, the policy weights trained by TD3+BC [14] is converted into those of spiking TD3+BC. During calibration phase, the spikes fired by SNN policy of spiking TD3+BC is calibrated by adjusting the initial membrane potential. Experimental results on MuJoCo robot control tasks [15] demonstrates that calibration method can further improve the performance of the converted SRL.

## 2 Related Work

### 2.1 Offline Reinforcement learning

Offline reinforcement learning (offline RL) [16] is a paradigm that learns exclusively from static datasets of previously collected interactions, making it feasible to extract policies from large and diverse training datasets. The most classical offline RL algorithm is TD3+BC [14], which combines TD3 with behaviour cloning (BC) and achieves competitive performance with less computation time on multiple continuous robot control tasks of D4RL benchmark. In this paper, we use TD3+BC as our RL algorithm.

### 2.2 Spiking Neural Network

The existing SRL traces back to employing neural mechanism observed in brain to reinforcement learning tasks, including synapses [17], temporal difference [18], and STDP [19,20]. However, it is noteworthy that these direct training methods have primarily been explored with shallow networks and remain significantly distant from conventional RL methodologies. Since the presence of surrogate functions and conversion from existing ANN, SRL steps into the realm of complex and deep networks. A simple

Spiking Q-network [21] demonstrated enhanced robustness against occlusion attacks without degraded performance on the Atari Breakout game. To address the discrepancy of conversion, a robust firing rate was introduced to eliminate error spike firing in the final layer [22]. Additionally, parameter normalization was integrated into this spiking DQN, surpassing conventional DQN on 9 Atari games, despite the latency extending to hundreds of simulation steps. [29] established the first directly trained SRL framework based on adaptive spike coder, breaking the barrier between the offline and online SRL.

### 2.3 ANN-SNN Conversion

So far, ANN-SNN may be the most effective method of training SNN, which has been validated in visual tasks. And multiple techniques were introduced to mitigate the error spikes occurring in the converted SNN, aiming to achieve negligible deterioration in classification accuracy in spite of a high simulation steps. These techniques include weight normalization [23] and spiking equivalents of various operations such as max-pooling and softmax, soft-reset and percentile threshold [24].

Several works notice the deviation between actual firing rates and corresponding activation values [25,26]. Calibrating SNN parameters in a layer-wise order during conversion has been introduced as an effective approach to bring substantial improvements in visual tasks. [26] utilized extra training on membrane potential, weight, and bias to mitigate accumulated deviation. Furthermore, [27] conducts a comprehensive analysis of the causes of error spikes. Instead of modifying the SNN, they renovated activation processing of ANN. It is noteworthy that replacing ReLU with quantization clip-floor-shift (QCFS) function in the source ANN, which can achieve the leading accuracy. And [13] designed a specific calibrating algorithm based on QCFS.

## 3 Methodology

In this section, we briefly introduce and formulate the general principle of conversion process, which is subject to neural models. Then we further detail causes of deviation during ANN-SNN conversion.

### 3.1 Conversion Principle

The current DNN-SNN conversion is to map ReLU to Integrate-and-Fire (IF) neuron. ReLU can be formulated as:

$$a^l = f(W^l a^{l-1} + b^l),\ l = 1,2,\dots,L, \tag{1}$$

where $f = max(\cdot)$, the $W^l$ and $b^l$ represent weight parameters and biases in layer $l$, vector $a^l$ denotes activation value of a non-linear function $f(\bullet)$, regarded as the output of all artificial neurons in the $i$-th layer. The accumulation of IF neural potential can be described as follows:

$$M_i^l(t) = V_i^l(t-1) + C_i^l(t) \tag{5}$$

$$V_i^l(t) = M_i^l(t) - x_i^l(t) \tag{6}$$

$$s_i^l(t) = H\left(V_i^l(t)\right) \tag{7}$$

$M_i^l$ and $V_i^l$ indicate respectively membrane potential during accumulation and resetting after triggering a spike. Specifically, we can formulate firing rate of certain layer $l$ in the form of equation 1 by summing equation 6 from $t = 1$ to $t = T$, we can conclude:

$$\sum_{t=1}^{T} V^l(t) = \sum_{t=0}^{T-1} V^l(t) + \sum_{t=1}^{T} C^l(t) - \sum_{t=1}^{T} s^l(t)\theta^l \tag{8}$$

If we eliminate equivalent terms on both sides of equation 8 and substitute $C^l(t)$ with equation 3. We can establish the relationship between the firing rates across layer $l$ and layer $l-1$ by dividing total time $T$ on both sides:

$$\frac{\sum_{t=1}^{T} \theta^l s^l(t)}{T} = W^l \frac{\sum_{t=1}^{T} \theta^{l-1} s^{l-1}(t)}{T} + \frac{V^l(0) - V^l(T)}{T} + b^l \tag{9}$$

We could use $\phi^l(T) = \frac{\sum_{t=1}^{T} \theta^l s^l(t)}{T}$ to indicate average postsynaptic potential over $T$, or $\psi^l(T) = \frac{\sum_{t=1}^{T} s^l(t)}{T}$ to denote spike firing rate. The linear relationship of postsynaptic voltage and firing rate for adjacent layers is described as follows:

$$\phi^l(T) = W^l \phi^{l-1}(T) + \frac{V^l(0) - V^l(T)}{T} + b^l \tag{10}$$

$$\psi^l(T) = W^l \frac{\theta^{l-1}}{\theta^l} \psi^{l-1}(T) + \frac{V^l(0) - V^l(T)}{\theta^l T} + \frac{b^l}{\theta^l} \tag{11}$$

Equation 11 differs from the firing rate definition in [22], whose input accumulation initializes potential to 0 at the moment $t = 0$ and defines $C_i^l(\mathrm{t}) = W_i^l s^{l-1}(t) + b_i^l$. [22] believes that converting a spike to an equivalent current diminishes the impact of the threshold. Consequently, the input spike is not multiplied by any threshold in their equation, yet, the absence of $\theta^{l-1}$ will introduce $\frac{1}{\theta^l}$ into $W^l$. On the one hand, equation 10 is exactly the same as ANN's forward propagation (equation 1) if we set initial potential $V^l(0) = 0$ and neglect the term $\frac{V^l(T)}{T}$, on the other hand, it implies that a lossless ANN-SNN conversion is possible if $T$increases to infinity. However, achieving a realistic SNN is more feasible within reasonable latency constraints.

### 3.2 Conversion Error

Before formulating conversion error between ReLU and IF, we assume that layer $l$ in ANN and SNN both receives the same input from layer $l-1$, i.e., $a^{l-1} = \phi^{l-1}(T)$. The conversion error can be defined by following equation:

$$\begin{aligned} E^l &= \phi^l(T) - a^l \\ E^l &= \begin{cases} \frac{V^l(0) - V^l(T)}{T}, & a^l \geq 0 \\ \phi^l(T), & a^l < 0 \end{cases} \end{aligned} \tag{12}$$

It can be inferred that conversion error will never be eliminated if $V^l(0) \neq V^l(T)$. Specifically, conversion error includes clipping error, quantization error (flooring error) and unevenness error (deviation error) between a source ANN and a converted SNN. Based on definition, $\phi^l(T) = \frac{\sum_{t=1}^{T} \theta^l s^l(t)}{T}$, the value range of $\phi_i^l(T)$ is a discrete finite set $\left\{k\frac{\theta^l}{T}\right\}$, $k$ = 0, 1, . . . , $T$, which is a subset of $[0, \theta^l]$. While equation 2 maps output $a_i^l$ of ANN into a continuous range $\left[0, \max\left(a_i^l\right)\right]$. Activation values that beyond $\theta^l$, that is, $a^l \in (\theta^l, a_{\max}^l]$ will be clipped to $\theta^l$ by IF neurons when $\max\left(a_i^l\right) > \theta^l$ and hence clipping error arises due to the mapping equation:

$$\phi^l(T) = \text{clip}\left(\frac{\theta^l}{T}\left\lfloor\frac{a^l T}{\lambda^l}\right\rfloor, 0, \theta^l\right) \tag{13}$$

The term $\lambda^l$ here denotes the ANN counterpart of threshold $\theta^l$ associated with spiking neurons. The maximum value of $a^l$ or a 99th percentile can be used as $\lambda^l$ mapped to $\theta^l$. Besides, the smallest unit for $\phi^l(T)$ to vary is $\frac{\theta^l}{T}$, thus any $a^l \in \left[\frac{n\theta^l}{T}, \frac{(n+1)\theta^l}{T}\right)$ will inevitably be rounded down to $\frac{n\theta^l}{T}$ in SNN when $n$ = 0, 1, . . . , $T-1$, which results in so-called flooring error. Unevenness error refers to a scenario in which the ordering of positive and negative weighted spike influences the variation of $\phi^l(T)$. In comparison to the case where positive and negative spikes are fired alternately, if positive weighted spikes in layer $l-1$ aggregate before negative weighted spikes, there may be one additional spike fired by IF neurons in layer $l$. Conversely, if negative weighted spikes gather earlier than positive ones, there may be one less spike.

### 3.3 Alleviation Of Error

**Quantization Clip-Floor-Shift Activation Function** To better reduce clipping and quantization error, [27] proposed replacing source ANN's ReLU in training with a discrete, bounded activation function, which resembles equation 13:

$$a^l = \frac{\lambda^l}{L} clip\left(\left\lfloor\frac{W^l a^{l-1}{}_L}{\lambda^l} + \varphi\right\rfloor, 0, L\right) \tag{14}$$

where $L$ is the quantization step of ANN, just like total time steps $T$ in SNN. $\lambda^l$ represents the learnable threshold during ANN training stage, rather than a hyper-parameter in equation 13. The trained $\lambda^l$ will be mapped to $\theta^l$ during conversion, that is, $\lambda^l = \theta^l$. While $\varphi$ is a hyperparameter, the shift distance. Clipping and floor errors are effectively mitigated by QCFS when $\varphi = \frac{1}{2}$. However, it has been observed that QCFS sometimes underperforms on certain offline control tasks, achieving slightly lower reward compared to ANN activated by ReLU. Results will be presented in the next section.

**Calibration Spike** Given the absence of valid approaches to address unevenness error, [13] provides a precise adjustment to the number of spikes within the framework of the QCFS function. They utilize $\frac{a^l T}{\theta^l}$ to represent the ideal total spike count in ANN. Consequently, the disparity between the desired spike count and the actual spike count,

$\sum_{t=1}^{T} s^l(t)$, could be regarded as the remaining error. Subsequently, such deviation $E_i^l$ is alleviated by aligning the actual spike count in SNN with the desired spike count in ANN. This study reveals that the sign of $E_i^l$ can be ascertained by $V_i^l(T)$:

**Conclusion 1** *Given* $\phi_i^l = \frac{\sum_{t=1}^{T} \theta_i^l s_i^l(t)}{T} \in [0, \theta_i^l]$,

1. if $V_i^l(T) < 0$ and $\phi_i^l(T) \neq 0$, then $\phi_i^l(T) > a_i^l$ and $E_i^l > 0$.
2. if $V_i^l(T) \geq \theta_i^l$ and $\phi_i^l(T) \neq \theta_i^l$, then $\phi_i^l(T) < a_i^l$ and $E_i^l < 0$.

Once we could determine the sign of error, spike count could be precisely increased or decreased by shifting the initial membrane potential based on following rules. $\hat{s}_i^l(t)$ and $\hat{V}_i^l(0)$ denote calibrated spike output at time step $t$ and shifted starting potential:

**Conclusion 2** *For the i-th neuron in l-th layer at time-step t of SNN, the total spike count* $\sum_{t=1}^{T} s_i^l(t)$ *varies if the preset voltage is shifted by any* $\varepsilon \in (0, \theta_i^l)$

1. if $\hat{V}_i^l(0) = V_i^l(0) - max(\theta_i^l, \varepsilon + min\{V_i^l(t) \mid s_i^l(t) = 1\})$, we can get $\sum_{t=1}^{T} \hat{s}_i^l(t) = \sum_{t=1}^{T} s_i^l(t) - 1$.
2. if $\hat{V}_i^l(0) = V_i^l(0) + max(\theta_i^l, \theta_i^l + \varepsilon - max\{V_i^l(t) \mid s_i^l(t) = 0\})$, we will get $\sum_{t=1}^{T} \hat{s}_i^l(t) = \sum_{t=1}^{T} s_i^l(t) + 1$.

According to the above two conclusions, the calibration of converted SNN introduces a $\rho$ steps simulated spiking before formal $T$ steps processing. Every *IF* neuron will optimize its initial membrane potential based on simulated result. The process is shown in algorithm 1.

**Algorithm 1** Spike Calibration of IF neuron in Spiking Actor Network during Inference

**Input**: Input tensor $x$ of shape (1, 256); Time steps ρ of calibration stage; Time steps $T$ of spiking inference; Converted actor network $f_{IF}$; Number of Calibration I; ε ∈ (0, θ)

IF. $v \leftarrow \frac{\theta}{2}$ ▷ Set $V(0)$ at half of the firing threshold
set empty list $V_f$ and $V_s$
**for** *i=1* to *I* **do**
  **for** *p=1* **to** $\rho$ **do**
    IF($x$) ▷ We will normalize input $x$ based on $\theta$ in actual code
    Recod $v(p)$ of firing neurons in $V_f$ and $v(p)$ of silent neurons in $V_s$. Set silent neurons in $V_f$ at 1000 ; firing neurons in $V_S$ at -1000
  **end for**
  label: label firing neurons with $v(\rho) < 0$ as 1 , and the rest as 0 .
  **for all** $n$ in $IF$ **do**
    $V_{min}(n) \leftarrow min(z(n) \cdot v \text{ for } z \text{ in } V_f)$
    $V_{max}(n) \leftarrow max(z(n).v \text{ for } z \text{ in } V_s)$
  **end for**
  **for all** $n$ in $IF$ **do**
    $D_f \leftarrow max(\theta, V_{min}(n) + \varepsilon)$ ▷ Shifting distance for 1 less spike

$D_s \leftarrow max(\theta, \theta + \varepsilon - V_{max}(n))$ ▷ Distance for 1 more spike
**end for**
**for all** $n$ in $IF$ **do**
**if** $n.v \geq \theta$ **then**
$n.v \leftarrow \frac{\theta}{2} + D_S$
**else**
$n \cdot v \leftarrow \frac{\theta}{2} - label \cdot D_f$
**end if**
**end for**
**end for**
**for** $t=1$ **to** $T$ **do**
Output[$t$] ←$IF(x)$
**end for**
**Output**: Output spike of shape ($T$, 256)

Furthermore, utilizing an adaptive calibration process with a dynamic $I$, instead of a fixed value, presents a more rational approach for implementing optimal membrane potential adjustments. We introduce Algorithm 2, which dynamically adjusts the number of iterations. This algorithm initiates from the minimum allowed value $I_{min}$ and progressively raises $I$ by 1 through step-wise interaction until the upper bound $I_{max}$ is reached. This adjustment of $I$ renders the calibration process flexible, preventing unnecessary calibration once error spikes have been eliminated and under circumstance that some error spikes are acceptable.

Algorithm 2 Dynamically set iteration $I$ of calibration

**Input**: The initial state of agent, $s_0$; Environment of simulation, env; Converted SNN $f_{IF}$; Source ANN $f_{QCFS}$; Minimum number of iteration, $I_{min}$; Maximum number of iterations, $I_{max}$

$s \leftarrow s_0$
$f_{IF} \cdot I \leftarrow I_{min}$
**while** Interaction is not done **do**
Action ← $f_{IF}(s)$
**for all** IF layer in $f_{IF}$ do
$\psi(T) = \frac{\sum_{t=1}^{T} s(t)}{T}$ ▷ Record firing rate of IF neuron
**end for**
$f_{QCFS}$
**for all** QCFS layer in $f_{QCFS}$ **do**
$a \leftarrow$ Output of QCFS function
end for
Count error spike: count ← $\left|\left(\frac{a}{\lambda} - \psi(T)\right) \cdot T\right|_{abs}$

13: **if** count $> 0$ and $f_{IF} \cdot I < I_{max}$ **then**
14: $f_{IF} \cdot I \leftarrow f_{IF} \cdot I + 1$
15: **else**
16: $f_{IF}.I \leftarrow I_{min}$
17: **end if**
18: $s \leftarrow env(\text{Action})$
19: **end while**

# 4 Experiment

## 4.1 Environment

We evaluate performance of the calibrated spiking actor network across four control tasks: hopper, Halfcheetah, walker2d, and Ant. For each task, the artificial actor network is trained using samples collected by four policies, each exhibiting distinct performance level. D4RL[28] is a convenient tool to collect trajectories for offline reinforcement learning on multiple tasks such as continuous robot control, navigation and autonomous driving. The entire training and testing are implemented on D4RL environment. All actor networks undergo training for 1 million steps over 4 quantization steps $L = 4$. We not only compare the performance of calibrated conversion actor network with four baselines, including an actor network with ReLU, an actor network activated by QCFS and a directly trained spiking actor network, as well as a converted network without calibration, but also investigate this adjustment through iterations of 2 and 3 times. All tests are conducted based on simulating steps $\rho = 4$ during calibration and actual steps $T = 4$ during inference.

## 4.2 Results

As illustrated in Table 1, the calibrated networks ($I = 1$, 2 or 3) consistently exhibit superior performance compared to the converted policy without any adjustment ($I = 0$) across all tasks. Notably, the transition from ReLU activation function to QCFS results in a performance decline in specific tasks, particularly evident in the medium-expert level dataset for Halfcheetah control. Compared to the QCFS network and the directly trained spiking policy, the converted network without calibration mainly underperforms on Halfcheetah control, lagging behind on expert-level dataset. Although a performance gap persists between ANN (ReLU) and converted SNN, the
calibrated spiking policy already outperforms the source ANN(QCFS) in some tests. Interestingly, both the native training spiking policy ($T = 1$) and converted network ($T = 4$) surpass each other on specific tasks. However, the involvement of calibration ($I > 0$) enhances the effectiveness of the conversion method more effective on most tasks.

Table 1 The performance of calibrated networks and baseline methods

| | Tasks | ReLU | QCFS | Direct training | I=0 | I=1 | I=2 | I=3 |
|---|---|---|---|---|---|---|---|---|
| Expert | Hopper | 112.2±0.2 | 112.1±1.3 | 112.1±0.1 | 110.7±1.4 | 111.8±0.4 | 110.2±3.0 | 112.3±0.2 |
| | Halfcheetah | 105.7±1.9 | 94.0±3.2 | 71.0±4.7 | 59.5±11.6 | 89.9±3.9 | 87.2 ±9.4 | 89.1±5.4 |
| | Walker2d | 105.7±2.7 | 91.5±15.8 | 105.6±1.1 | 98.5±9.4 | 99.2±7.7 | 102.7±4.7 | 103.7±3.8 |
| | Ant | - | 72.1 ±21.1 | 59.9±8.8 | 63.1±14.6 | 72.2±17.4 | 81.7±22.8 | 75.4±23.2 |
| Medium Expert | Hopper | 112.2±0.2 | 95.4±26.6 | 112.0±0.3 | 94.6±16.1 | 101.5± | 100±12.3 | 103.3± |
| | Halfcheetah | 97.9 ±4.4 | 56.8 ±7.9 | 39.5±3.2 | 45.8±9.2 | 64.6±9.0 | 58.4±4.1 | 65.3±3.2 |
| | Walker2d | 101.1±9.3 | 94.1±11.5 | 83.7±19.3 | 96.7±9.2 | 105.9±2.0 | 101.9±2.3 | 102.7±4.4 |
| | Ant | - | 111.3±10.7 | 107.4 ±12.8 | 128.7±6.6 | 117.5± | 112.3±12.7 | 112.2± |
| Medium | Hopper | 99.5±1.0 | 99.6±0.1 | 96.8±3.4 | 100.0±0.2 | 99.8±0.2 | 99.9±0.3 | 99.7±0.9 |
| | Halfcheetah | 42.8± | 42.3 ±0.4 | 42.1±0.3 | 40.8±0.2 | 42.5 ±0.4 | 42.4±0.5 | 42.4±0.4 |
| | Wal ker2d | 79.7±1.8 | 76.5±3.2 | 76.7±6.6 | 72.6±6.9 | 76.2±2.11 | 79.8±0.8 | 77.7±2.5 |
| | Ant | - | 119.6 ±3.6 | 108.8±6.9 | 120.3±5.2 | 120.5± | 122.4±0.9 | 122.4±1.1 |
| Medium Replay | Hopper | 31.4±3.0 | 31.1±1.9 | 30.6±2.8 | 32.5±3.9 | 31.9±3.1 | 34.1±6.0 | 31.5±1.7 |
| | Halfcheetah | 43.3 ±0.5 | 42.1±0.4 | 40.4 ±0.1 | 33.7±2.3 | 41.3±0.3 | 41.1±0.3 | 41.3±0.3 |
| | Walker2d | 25.2±5.1 | 23.8±6.7 | 27.4±4.0 | 21.0±5.5 | 22.4±5.9 | 24.9±7.4 | 26.2±7.4 |
| | Ant | - | 75.9 ±4.1 | 77.5±2.4 | 75.2 ±6.3 | 74.5±1.9 | 76.9±2.9 | 76.1±3.5 |

For dynamic *I*, converted spiking network with adaptive *I* successfully avoids dramatic performance decline caused by successive calibration which can be seen from Table 2. Furthermore, among a set of 16 continuous control tasks, it attains favorable outcomes in 6 tasks, showing competitive performance in another 5 tasks. The flexible *I* yields moderate performance in 3 out of the remaining 5 tasks.

Table 2 The performance of converted spiking network with adaptive *I*

| | Tasks | I = 0 | I = 1 | I = 2 | I = 3 | **Adaptive** |
|---|---|---|---|---|---|---|
| Expert | Hopper | 114.49±0.70 | 110.8±1.90 | 109.81±4.86 | 110.75±2.83 | 112.19±0.33(*) |
| | Halfcheetah | 64.93±16.46 | 91.88±3.00(*) | 84.00±8.70 | 88.95±7.99 | 86.40±11.07 |

| | | | | | | |
|---|---|---|---|---|---|---|
| | Walker2d | 97.53±13.58 | 101.25±4.34 | 101.11±3.93 | 101.71±4.18 | 99.41±3.90 |
| | Ant | 77.56±15.63 | 74.46±16.26 | 82.23±9.63 | 87.79±9.79 | 89.44±11.48(*) |
| Medium Expert | Hopper | 83.18±17.72 | 88.22±28.29 | 91.19±24.00 | 89.10±27.90 | 95.07±21.34(*) |
| | Halfcheetah | 46.08±9.49 | 59.62±5.06 | 60.53±5.17 | 60.79±6.12 | 60.59±4.84 |
| | Walker2d | 95.22±12.31 | 99.23±4.81(*) | 95.90±7.85 | 91.68±10.79 | 92.96±12.67 |
| | Ant | 117.64±13.88 | 122.66±14.70 | 104.80±8.05 | 105.22±10.51 | 116.11±7.89 |
| Medium | Hopper | 98.57±2.76 | 99.70±0.18 | 99.79±0.20 | 99.68±0.22 | 99.08±1.48 |
| | Halfcheetah | 40.65±0.22 | 42.40±0.47 | 42.36±0.44 | 42.31±0.53 | 42.12±0.53 |
| | Walker2d | 72.85±6.61 | 78.95±2.63(*) | 75.85±3.34 | 77.13±2.91 | 74.00±3.50 |
| | Ant | 121.46±1.98 | 122.22±0.86(*) | 120.50±3.01 | 120.48±2.85 | 122.17±1.21(*) |
| Medium Replay | Hopper | 23.82±4.29 | 29.78±3.15 | 229.68±3.44 | 29.25±2.60 | 29.81±2.46 |
| | Halfcheetah | 35.85±1.61 | 41.23±0.41(*) | 41.28±0.32(*) | 41.29±0.50(*) | 40.66±0.34 |
| | Walker2d | 17.86±4.09 | 26.61±8,98(*) | 24.47±9.13 | 26.21±12.38(*) | 26.03±8.04(*) |
| | Ant | 77.32±4.26 | 77.69±2.18 | 76.48±3.78 | 76.48±3.89 | 79.74±1.95(*) |

### 4.3 Analysis

The hyperparameter $I$ determines the number of times calibration will be implemented. [13] suggests that calibrating once is sufficient for improvement. This is particularly relevant in the context of deep convolution networks, where the majority of disparity in deeper layers is attributed to a ±1 error spike. It is important to note that the optimal number of calibrations can vary, especially for tasks that involve long-horizon interactions. Achieving the highest scores does not necessarily correspond to using a fixed number $I$ of iterations. Ideally, the successive occurrences of potential-shifting should ensure a steady improvement or yield nearly identical results. This phenomenon is exemplified in the results of medium-Ant task and expert-Walker2d for $I$ = 1, 2, 3. Conversely, a sudden drop in score accompanies the increment of $I$. This observation is evident in the results of medium-expert Ant task.

## 5 Conclusions

To the best of our knowledge, we are the first to utilize calibration of membrane potential to boost converted spiking neural network's performance on offline reinforcement learning tasks. We find that iterative calibration sometimes results in performance collapse, and we use the changeable number of iterations to alleviate the problem brought by consecutive adjustment.

**Acknowledgments.** This work is supported by the National Science and Technology Innovation 2030 – Major Project (Grant No.2022ZD0208800, Grant No. 2022ZD0208801) and the National Natural Science Foundation of China (No. 62176215).

**Disclosure of Interests.** The authors declare that they have no competing interests.

## References

1. Roy, K., Jaiswal, A., Panda, P.: Towards spike-based machine intelligence with neuromorphic computing. Nature **575**(7784), 607–617 (2019)
2. Hu, Y., Tang H., Pan G.: Spiking deep residual networks. IEEE Transactions on Neural Networks and Learning Systems **34**(8), 5200-5205 (2021).
3. Fang, W., Yu, Z., Chen, Y. et al.: Deep residual learning in spiking neural networks. Advances in Neural Information Processing Systems, 3**4**, 21056–21069(2021)
4. Lv, C., Li, T., Xu, J., et al.: Spikebert: A language spikformer trained with two-stage knowledge distillation from bert. arXiv, 2023
5. Roy, D., Chakraborty, I., Roy, K.: Scaling deep spiking neural networks with binary stochastic activations. In: 2019 IEEE International Conference on Cognitive Computing (ICCC), pp. 50–58. IEEE, 2019
6. Deng, L., Wu, Y., Hu, X., et al.: Rethinking the performance comparison between snns and anns. Neural networks, **121**, 294–307(2020)
7. Caporale, N., Dan, Y.: Spike timing–dependent plasticity: a hebbian learning rule. Annu. Rev. Neurosci., **31**, 25–46(2008)
8. Neftci, E.O., Mostafa H., Zenke, F.: Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. IEEE Signal Processing Magazine, **36**(6), 51–63(2019)
9. Cao, Y., Chen Y., Khosla, D.: Spiking deep convolutional neural networks for energy-efficient object recognition. International Journal of Computer Vision, **113**, 54–66(2015)
10. Wu, Y., Deng, L., Li, G., Zhu, J., Shi, L.: Spatio-temporal backpropagation for training high-performance spiking neural networks. Frontiers in neuroscience, **12**(331), 2018.
11. Zhang, W., Li, P.: Temporal spike sequence learning via backpropagation for deep spiking neural networks. In: Advances in Neural Information Processing Systems, pp. 12022–12033, 2020.
12. Wu, Y., Deng, L., Li, G., Zhu, J., Xie, Y., Shi, L.: Direct training for spiking neural networks: Faster, larger, better. In Proceedings of the AAAI conference on artificial intelligence, pp. 1311–1318, 2019
13. Hao, Z., Ding, J., Bu, T., Huang, T., Yu, Z.: Bridging the gap between anns and snns by calibrating offset spikes. arXiv, 2023
14. Fujimoto, S., Gu, S. S.: A minimalist approach to offline reinforcement learning. In: Advances in neural information processing systems, pp. 20132–20145, 2021
15. Todorov, E., Erez, T., Mujoco, Y. T.: A physics engine for model-based control. In: 2012 IEEE/RSJ international conference on intelligent robots and systems, pp. 5026–5033. IEEE, 2012
16. Prudencio, R. F., Maximo, M, Colombini, E. L.: A survey on offline reinforcement learning: Taxonomy, review, and open problems. IEEE Transactions on Neural Networks and Learning Systems, 2023
17. Seung, H.: Learning in spiking neural networks by reinforcement of stochastic synaptic transmission. Neuron, **40**(6), 1063–1073(2003)

18. Potjans, W., Morrison, A., Diesmann, M.: A spiking neural network model of an actor-critic learning agent. Neural computation, **21**(2), 301–339(2009)
19. Florian, R. V.: Reinforcement learning through modulation of spike-timingdependent synaptic plasticity. Neural computation, **19**(6), 1468–1502(2007)
20. Bing, Z., Claus , Meschede, K., Chen, G., et al.: End to end learning of spiking neural network based on r-stdp for a lane keeping vehicle. In 2018 IEEE international conference on robotics and automation (ICRA), pp. 4725–4732. IEEE, 2018
21. Patel, D., Hazan, H., Saunders, D. J., Hava T , Siegelmann, R. K.: Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to atari breakout game. Neural Networks, **120**, 108–115(2019)
22. Tan, W., Patel, D., Kozma, R.: Strategy and benchmark for converting deep q-networks to event-driven spiking neural networks. In: Proceedings of the AAAI conference on artificial intelligence, pp. 9816–9824, 2021
23. Diehl, P. U., Neil, D., Binas, J., Cook, M., Liu, S. C., Pfeiffer, Michael.: Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing. In: 2015 International joint conference on neural networks (IJCNN), pp. 1–8. IEEE, 2015
24. Rueckauer, B., Lungu, I. A. , Hu, Y., Pfeiffer, M., Liu, S.C.: Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. Frontiers in neuroscience, **11**, 2017
25. Deng, S., Gu, S.: Optimal conversion of conventional artificial neural networks to spiking neural networks. arXiv, 2021
26. Li, Y., Deng, S., Dong, X., Gong, R., Gu, Shi.: A free lunch from ann: Towards efficient, accurate spiking neural networks calibration. In: International conference on machine learning, pp. 6316–6325. PMLR, 2021.
27. Bu, T., Fang, W., Ding, J. et al.: Optimal ann-snn conversion for high-accuracy and ultra-low-latency spiking neural networks. arXiv, 2023
28. Fu, J., Kumar, A., Nachum, O., Tucker, G., Levine, S.: D4RL: Datasets for deep data-driven reinforcement learning. arXiv, 2020
29. Qin, L., Yan, R., Tang, H.: A low latency adaptive coding spike framework for deep reinforcement learning. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, pp. 3049–3057, 2023