Multiscale Dynamic Coding improved Spiking Actor Network for Reinforcement Learning

Duzhen Zhang[1,2*], **Tielin Zhang**[1,2*†], **Shuncheng Jia**[1,2], **Bo Xu**[1,2,3†]

[1]Research Center for Brain-inspired Intelligence, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3]Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai, China
{xubo, tielin.zhang, zhangduzhen2019}@ia.ac.cn

Abstract

With the help of deep neural networks (DNNs), deep reinforcement learning (DRL) has achieved great success on many complex tasks, from games to robotic control. Compared to DNNs with partial brain-inspired structures and functions, spiking neural networks (SNNs) consider more biological features, including spiking neurons with complex dynamics and learning paradigms with biologically plausible plasticity principles. Inspired by the efficient computation of cell assembly in the biological brain, whereby memory-based coding is much more complex than readout, we propose a multiscale dynamic coding improved spiking actor network (MDC-SAN) for reinforcement learning to achieve effective decision-making. The population coding at the network scale is integrated with the dynamic neurons coding (containing 2nd-order neuronal dynamics) at the neuron scale towards a powerful spatial-temporal state representation. Extensive experimental results show that our MDC-SAN performs better than its counterpart deep actor network (based on DNNs) on four continuous control tasks from OpenAI gym. We think this is a significant attempt to improve SNNs from the perspective of efficient coding towards effective decision-making, just like that in biological networks.

Introduction

Reinforcement learning (RL) is staying at an increasingly important position in the research area of machine learning (Kaelbling, Littman, and Moore 1996), where agents interact with the environment in a trial-and-error manner and learn an optimal policy by maximizing accumulated rewards to reach excellent decision-making performance (Sutton and Barto 2018). However, it is a general but challenging problem for all conventional RL algorithms to extract features from complex state space efficiently. With deep neural networks (DNNs) as powerful function approximators, deep reinforcement learning (DRL) has resolved this problem to some extent by directly learning a mapping from raw state space to action space and has been well applied on various applications, including recommendation systems (Zou et al. 2019; Warlop, Lazaric, and Mary 2018), games (Mnih et al. 2015; Vinyals et al. 2019), and robotic control (Duan et al. 2016; Lillicrap et al. 2016), etc.

However, the powerful DRL is still far from efficient reward-based learning in the biological brain, where spiking neurons with more complex dynamics and learning paradigms with biologically plausible plasticity principles are integrated to generate complex cognitive functions. The biological brain makes efficient computation possible by cell assembly (Harris et al. 2003) which focuses more on spatial-temporal coding for memory than readout for decision-making. Compared to DNNs, SNNs have more significant potential in simulating brain-inspired topology and functions due to their complex dynamics. For instance, SNNs can be seamlessly compatible with multiscale dynamic coding, including network and neuron scales, towards a powerful temporal information representation. The SNNs inherently transmit and compute information with dynamic spikes distributed over time (Maass 1997). Further research on them might help us open the black box of efficient information coding of the brain (Painkras et al. 2013). Hence, we think RL using SNNs may be better than using DNNs.

To this end, we propose a multiscale dynamic coding improved spiking actor network (MDC-SAN) to simulate the cell assembly in the biological brain, which contains a complex spiking coding module for state representation but a simple readout module for action inference. The coding module combines population coding and dynamic neurons (DNs) coding, making it more potent on state representation at both network and neuron scales. Specifically, for a given input state, we encode each dimension in individual neuron populations with learnable receptive fields. Then the coded analog information is directly delivered to the network as input. Inside the network, we propose novel DNs to improve SNNs for a better information representation during spatial-temporal learning. The DNs contain 2nd-order dynamics of membrane potentials supported by key dynamic parameters. These parameters are self-learned from one of OpenAI gym (Brockman et al. 2016) tasks (e.g., Ant-v3) first, and then extend to other similar tasks (e.g., HalfCheetah-v3, Walker2d-v3, and Hopper-v3). After dynamic coding, we average the accumulated spikes in a predefined time window to obtain the average firing rate, further used to infer the output action by a simple readout module.

For effective learning, the proposed MDC-SAN is trained

*These authors contributed equally.
†Corresponding author.

in conjunction with deep critic networks using the Twin Delayed Deep Deterministic policy gradient (TD3) algorithm (Fujimoto, Hoof, and Meger 2018; Tang, Kumar, and Michmizos 2020). We evaluate the trained MDC-SAN on four standard OpenAI gym tasks (Brockman et al. 2016), including Ant-v3, HalfCheetah-v3, Walker2d-v3, and Hopper-v3. Experimental results demonstrate that multiscale dynamic codings, including population coding and complex spatial-temporal coding of DNs, are consistently beneficial to the performance of the MDC-SAN. In addition, the proposed MDC-SAN significantly outperforms its counterpart deep actor network (DAN) on the above four tasks under the same experimental configurations.

The main contributions of this paper can be summarized as follows:

- We propose a MDC-SAN to simulate the cell assembly in the biological brain for effective decision-making, which contains a complicated coding module for state representation from network scale and neuron scale, and a simple readout module for action inference.
- For the network scale, we apply population coding to increase the representation capacity of the network, which encodes each dimension of the input state in individual neuron populations with learnable receptive fields. We also have verified the advantages of population coding and comprehensively compared the impact of various input coding methods on performance.
- For the neuron scale, we construct novel DNs, which contain 2nd-order neuronal dynamics for complex spatial-temporal information coding. We also analyze the membrane-potential dynamics of DNs and demonstrate the performance advantage of DNs against standard leaky-integrate-and-fire (LIF) neurons.
- Under the same experimental configurations, our MDC-SAN, which integrates population coding and DNs coding, achieves better performance than its counterpart DAN in each task. To the best of our knowledge, our work is the first to achieve state-of-the-art performance on multiple continuous control tasks with SNNs.

Related work

Integrating SNNs with RL Recently, the literature has grown up around introducing SNNs into RL algorithms (Florian 2007; O'Brien and Srinivasa 2013; Yuan et al. 2019; Doya 2000; Frémaux, Sprekeler, and Gerstner 2013). These approaches are typically based on reward-modulated local plasticity rules that perform well in simple control tasks but commonly fail in complex robotic control tasks due to limited optimization capability.

To address the limitation, some approaches integrate SNNs with DRL optimization. One of the approaches directly converts Deep Q-Networks (DQNs) (Mnih et al. 2015) to SNNs and achieves competitive scores on Atari games with discrete action space (Patel et al. 2019; Tan, Patel, and Kozma 2021). However, these converted SNNs usually exhibit inferior performance to DNNs with the same structure (Rathi et al. 2019). Another approach is based on a hybrid learning framework. It achieves success in the mapless navigation task of a mobile robot, where the SAN is trained in conjunction with deep critic networks using a DRL algorithm (Tang, Kumar, and Michmizos 2020). We also want to highlight this hybrid viewpoint during training and further extend it with multiscale dynamic codings that have played vital roles in the efficient information representation of SNNs.

Information coding methods in SNNs There are two categories of the input coding scheme in SNNs, rate coding, and temporal coding. Rate coding uses the firing rate of spike trains in a time window to encode information, where input real numbers are converted into spike trains with a frequency proportional to the input value (Cheng et al. 2020a). Temporal coding encodes information with the relative timing of individual spikes, where input values are usually converted into spike trains with the precise time (Comsa et al. 2020; Sboev et al. 2018). Besides that, population coding is special in integrating these two types. For example, each neuron in a population can generate spike trains with precise time and also contain a relation with other neurons (e.g., Gaussian receptive field) for better information encoding at a global scale (Georgopoulos, Schwartz, and Kettner 1986).

For the neuron coding scheme in SNNs, there are various types of spiking neurons (Tuckwell 1988). The integrate-and-fire (IF) neuron is the simplest neuron type. It fires when the membrane potential exceeds the firing threshold, and the potential is then reset as a predefined resting membrane potential (Rathi and Roy 2020). Another leaky integrate-and-fire (LIF) neuron allows the membrane potential to keep shrinking over time by introducing a leak factor (Gerstner and Kistler 2002). They are commonly used as standard 1st-order neurons. Moreover, the Izhikevich neuron with 2nd-order equations of membrane potential is proposed, which can better represent the complex neuron dynamics, but requires some predefined hyper-parameters (Izhikevich 2003).

Methods

The overview of our MDC-SAN is presented in Figure 1, which contains an efficient coding module for spatial-temporal state representation and a relatively simple readout module for action inference. The MDC-SAN simulates the cell assembly in the biological brain by incorporating both population coding at the network scale and dynamic neurons coding at the neuron scale. For the population coding, each dimension of the input state is encoded with a group of dynamic receptive fields first and then fed into the SAN. For the dynamic neurons coding, the DNs inside of the SAN contain 2nd-order dynamic membrane potentials with up to two equilibrium points to describe complex neuronal dynamics. Finally, the average firing rate of the accumulated spikes in a predefined time window is decoded into output action with an additional group decoder.

Population coding

For a input state $\boldsymbol{s} \in \mathbb{R}^N$, it is encoded as an input $\boldsymbol{I}_t, t = \{1, 2, ..., T_1\}$ for each time step, where T_1 is the time window of the SNNs.

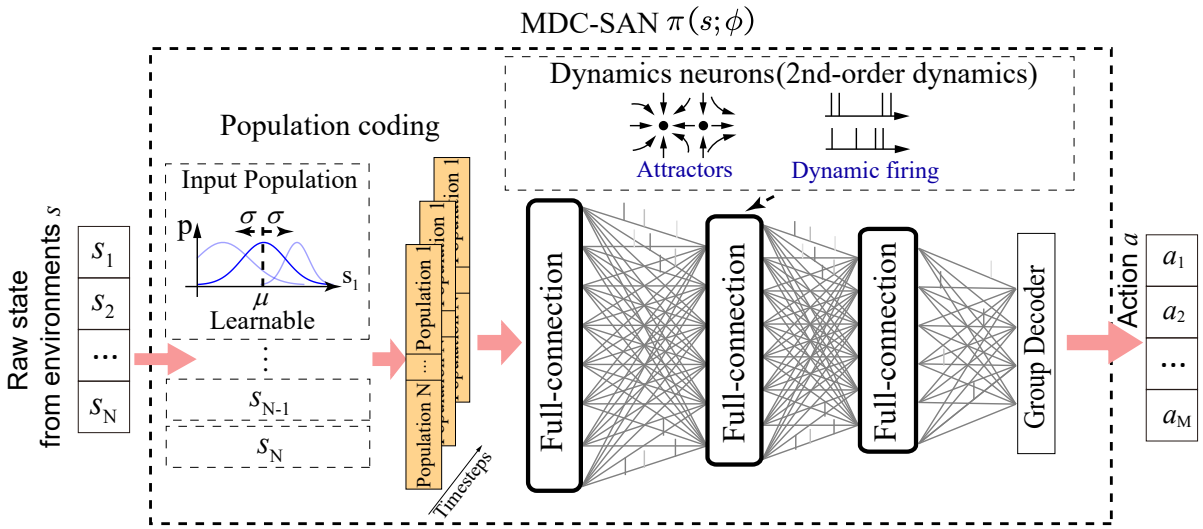

Figure 1: The overall architecture of proposed MDC-SAN.

Pure population coding C_{pop} We create a population of neurons P_i to encode each dimension of state s_i, where each neuron $P_{i,j}$ in the population has a Gaussian receptive field $(\mu_{i,j}, \sigma_{i,j})$ with two learnable parameters of mean and standard deviation. The population coding C_{pop} is formulated as:

$$\begin{cases} A_{P_{i,j}} = \exp^{-\frac{(s_i-\mu_{i,j})^2}{2\sigma_{i,j}^2}} \\ \boldsymbol{A}_P = \left[A_{P_{1,1}}, \ldots, A_{P_{i,j}}, \ldots, A_{P_{N,J}}\right] \\ \boldsymbol{I}_t = \boldsymbol{A}_P \end{cases}, \tag{1}$$

where i is index of the input state ($i = 1, ..., N$), j is index of neurons in a population ($j = 1, ..., J$), $\boldsymbol{A}_P$ is the stimulation strength after population coding, used as network input $\boldsymbol{I}_t$ directly.

There are other candidate input coding methods that combine population coding and rate coding (including uniform coding, Poisson coding, and deterministic coding). They contain two phases (Tang et al. 2020): first the state $\boldsymbol{s}$ is transformed into the stimulation strength $\boldsymbol{A}_P$ by population coding and then the computed $\boldsymbol{A}_P$ is used to generate the input $\boldsymbol{I}_t$ by rate coding. We formalize these methods as follows.

The population and uniform coding (C_{pop}+C_{uni}) We generate random numbers $Rand_k(t)$ from 0 to 1 evenly distributed, which has the same size as the input stimulation strength $\boldsymbol{A}_P$ at every time step. Then we compare every generated random number with its corresponding input data. If the generated random number is less than its input data, $I_{k,t}$ is set to 1. Otherwise, it is set to 0, formulated as:

$$I_{k,t} = \begin{cases} 1, & A_{P_k} > Rand_k(t); \\ 0, & \text{Otherwise.} \end{cases}, \tag{2}$$

where k is the index of input stimulation strength ($k = 1, ..., N * J$).

The population and Poisson coding (C_{pop}+C_{poi}) Considering that the Poisson process can be considered as the limit of a Bernoulli process, the input stimulation strength $\boldsymbol{A}_P$ containing probability can be used for drawing the binary random number. The $I_{k,t}$ will draw a value 1 according to the k^{th} probability value A_{P_k} given in $\boldsymbol{A}_P$, formulated as:

$$P(I_{k,t} = 1) = C_R^r A_{P_k}^r \left(1 - A_{P_k}\right)^{R-r}, \tag{3}$$

The population and deterministic coding (C_{pop}+C_{det}) The input stimulation strength $\boldsymbol{A}_P$ acts as the presynaptic inputs to the postsynaptic neurons (Tang et al. 2020), formulated as:

$$V_{k,t} = V_{k,t-1} + A_{P_k} \tag{4}$$

$$I_{k,t} = \begin{cases} 1 & If(V_{k,t} > 1) \\ 0 & Else \end{cases}, \tag{5}$$

where $V_{k,t}$ is reset as $V_{k,t} - 1$ when $I_{k,t} = 1$, $V_{k,t}$ is pseudo membrane voltage.

Dynamic Neurons

This section introduces the traditional 1st-order neurons (e.g., LIF neurons) with a maximum of one equilibrium point and then defines the improved 2nd-order DNs with up to two equilibrium points. The procedure for constructing DNs is also introduced in the following sections.

The traditional 1st-order neurons The traditional 1st-order neurons in SNNs are the LIF neurons, which are the simplest abstraction of the Hodgkin–Huxley model. To show the essential equilibrium point characteristics of LIF neurons, here we give a simple definition of LIF neurons as the following description:

$$\tau \frac{dV_{i,t}}{dt} = -V_{i,t} + In_t, \tag{6}$$

where $V_{i,t}$ is dynamic membrane potential for neuron i at time t, In_t is input represented as integrated post-synaptic

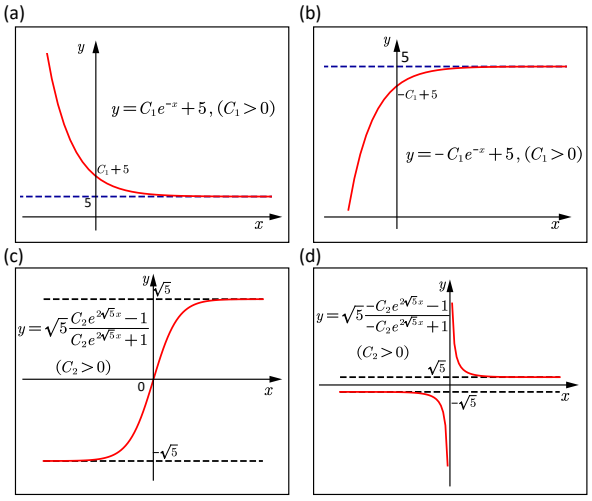

Figure 2: Dynamics of membrane potentials with different equilibrium points.

.

potential. The single equilibrium point can be calculated as $V^*_{i,t}$ with input In_t within period time of τ.

The number of equilibrium points will be the key to distinguishing different neuronal dynamics levels. For example, the dynamic field of $V_{i,t}$ for LIF model is reaching the single attractor $V^*_{i,t} = 5$. When the firing threshold is bigger than $V^*_{i,t}$, the neuron will be mostly leaky (Fig. 2(a)), or else it will be continuously firing (Fig. 2(b)).

The designed 2nd-order DNs The neurons with higher-order dynamics mean that these neurons' number of equilibrium points will be more than one. Here we set it as 2 for simplicity. The 2nd-order neuronal dynamics is shown as follows:

$$\frac{dV_{j,t}}{dt} = V^2_{j,t} - V_{j,t} - U_{j,t} + In_{i,t}, \tag{7}$$

where $V^2_{j,t}$ and $V_{j,t}$ are membrane potentials with different degrees of dynamics, $U_{j,t}$ is a resistance item for simulating hyper-polarization. The dynamic membrane potential $V_{j,t}$ will be attracted or non-stable at some points when we set the ordinary differential equation as 0. Fig. 2 (c) and Fig. 2 (d) show a diagram depicting dynamic fields of membrane potential with $N = 2$, where the period for reaching stable points take around time τ. For simplicity, we formulate traditional mV or ms units as 1. Besides membrane potential, some other implicit variables are also used for the description of 2nd-order dynamics, shown as follows:

$$\begin{cases} \frac{dU_{j,t}}{dt} = \theta_v V_{j,t} - \theta_u U_{j,t} & \\ V_{j,t} = \theta_r & if(V_{j,t} > V_{th}), \\ U_{j,t} = U_{j,t} + \theta_s & if(spike) \end{cases} \tag{8}$$

where equilibrium point of $V_{j,t}$ is decided by both $U_{j,t}$ and input currents In_t. The number and value of equilibrium points will be dynamically affected by four parameters of θ_v (conductivity of V), θ_u (conductivity of U), θ_r (reset membrane potential), and θ_s (spike improvement of U), which is different from Izhikevich neurons (Izhikevich 2003) by only focusing on higher-order dynamics.

Algorithm 1: Forward propagation in MDC-SAN

Initialize coding means $\boldsymbol{\mu}$ and standard deviations $\boldsymbol{\sigma}$ for all population encoders;
Randomly initialize synaptic weight $\boldsymbol{W}$ and biases $\boldsymbol{b}$ for each SNN layer;
Load the best dynamic parameters of DNs $\boldsymbol{\theta}^* = (\theta^*_v, \theta^*_u, \theta^*_r, \theta^*_s)$ (pre-learning from a task);
Randomly initialize decoding weight vectors $\boldsymbol{W}^d$ and bias b^d for each action dimension;
Initialize the current decay factor d_c and firing threshold V_{th};
$N-$dimensional input state, $\boldsymbol{s}$;
Inputs from populations generated by the input coding module: $\boldsymbol{A}_P$;
for $t = 1, ..., T_1$ **do**
  Inputs at timestep t: $\boldsymbol{O}^{(0)}_t = \boldsymbol{I}_t$=$\boldsymbol{A}_P$;
  for *l=1,...,L* **do**
    Update DNs in layer l at timestep t based on spikes from layer $l-1$:
    $\boldsymbol{C}^{(l)}_t = d_c \cdot \boldsymbol{C}^{(l)}_{t-1} + \boldsymbol{W}^{(l)} \boldsymbol{O}^{(l-1)}_t + \boldsymbol{b}^{(l)}$;
    $\boldsymbol{V}^{(l)}_t = \boldsymbol{V}^{(l)}_{t-1} \cdot (1 - \boldsymbol{O}^{(l)}_{t-1}) + \boldsymbol{O}^{(l)}_{t-1} \cdot \theta^*_r$;
    $\boldsymbol{U}^{(l)}_t = \boldsymbol{U}^{(l)}_{t-1} + \boldsymbol{O}^{(l)}_{t-1} \cdot \theta^*_s$;
    $\boldsymbol{V}_{delta} = {\boldsymbol{V}^{(l)}_t}^2 - \boldsymbol{V}^{(l)}_t - \boldsymbol{U}^{(l)}_t + \boldsymbol{C}^{(l)}_t$;
    $\boldsymbol{U}_{delta} = \theta^*_v \cdot \boldsymbol{V}^{(l)}_t - \theta^*_u \cdot \boldsymbol{U}^{(l)}_t$;
    $\boldsymbol{V}^{(l)}_t = \boldsymbol{V}^{(l)}_t + V_{delta}$;
    $\boldsymbol{U}^{(l)}_t = \boldsymbol{U}^{(l)}_t + U_{delta}$;
    $\boldsymbol{O}^{(l)}_t = (\boldsymbol{V}^{(l)}_t > V_{th})$;
  end
end
Sum up the output spikes: $\boldsymbol{sc} = \sum_{t=1}^{T_1} \boldsymbol{O}^{(L)}_t$;
Compute the average firing rate: $\boldsymbol{fr} = \boldsymbol{sc}/T_1$;
Divide $\boldsymbol{fr}$ into M output groups:$\{\boldsymbol{fr}_m\}, m = 1, ..., M$;
Generate M-dimensional action $\boldsymbol{a}$ by a grouped decoder: $a_m = \boldsymbol{W}^d_m \cdot \boldsymbol{fr}_m + b^d_m, m = 1, ..., M$;

The procedure for constructing the DNs The construction of DNs is mainly based on identifying some key parameters in them. As $\theta_{v,u,r,s}$, for example, each set of these four dynamic parameters describes a dynamic state of a spiking neuron. Hence, we want to obtain a set of optimal dynamic parameters.

We randomly initialize the dynamic parameters ($\theta_{v,u,r,s}$) of each neuron in the network. Together with synaptic weights, these dynamic parameters will be tuned on one of the tasks with the TD3 algorithm. After learning, where most of the learnable variables have reached stable points, these parameters will be plotted and clustered with the k-means method to get the best center of $\theta_{v,u,r,s}$ parameters. The four key parameters corresponding to the best center will be further used as the fixed configuration of all dynamic neurons for all tasks.

The simple readout module

The spikes at the output layer are summed up in a predefined time window to compute the average firing rate first. Then, the output action $\boldsymbol{a}$ is returned as the weighted sum of the computed average firing rate by a grouped decoder. More details can be found in the forward propagation of MDC-SAN (Algorithm 1).

The learning of MDC-SAN with TD3

The MDC-SAN is trained in conjunction with deep critic networks (i.e., a multi-layer fully-connected network) using the TD3 algorithm (Fujimoto, Hoof, and Meger 2018; Tang, Kumar, and Michmizos 2020). During training, the MDC-SAN infers an action $\boldsymbol{a}$ from a given state $\boldsymbol{s}$ to represent the policy, and a deep critic network estimates the associated action-value $Q(s, a)$ to guide the MDC-SAN to learn a better policy. We evaluate the trained MDC-SAN on a suit of continuous control benchmarks and compare its performance with its counterpart DAN (i.e., a multi-layer fully-connected network) under the same settings. The specific training procedure of the TD3 algorithm can be found in (Fujimoto, Hoof, and Meger 2018).

Tuning MDC-SAN with approximate BP

It is a challenge to tune parameters well in a network at multi scales, e.g., synaptic weights at different layers, parameters in population encoder, and group decoder. Many candidate methods for tuning multi-layer SNNs have been proposed, including approximate BP (Cheng et al. 2020b; Zenke and Ganguli 2018), equilibrium balancing (Shi, Zhang, and Zeng 2020; Zhang, Tielin and Zeng, Yi and Shi, Mengting and Zhao, Dongcheng 2018), Hopfield-like tuning (Zhang et al. 2016), and biologically plausible plasticity rules (Zeng, Zhang, and Xu 2017).

Here, we select the approximate BP for its efficiency and flexibility. The key feature of the approximate BP is replacing the non-differential parts of spiking neurons during BP to a predefined gradient number, shown as equation (9), where we use the rectangular function equation to approximate the gradient of a spike.

$$z(V) = \begin{cases} 1 & \text{if } |V - V_{th}| < w \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

where z is the pseudo-gradient, V is membrane voltage, V_{th} is the firing threshold and w is the threshold window for passing the gradient.

Experiments

To evaluate our model, we measured its performance on four continuous control tasks from the OpenAI gym (Fig. 3) (Brockman et al. 2016).

Implement details

Due to recent concerns in reproducibility (Henderson et al. 2018), all our experiments were reported over 10 random seeds of the network initialization and gym simulator. Each task was run for 1 million steps and evaluated every 10k steps, where each evaluation reported the average reward over 10 episodes without exploration noise, and each episode lasted for a maximum of 1000 execution steps.

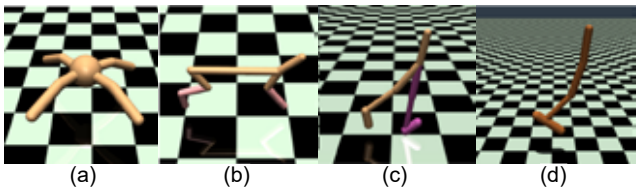

Figure 3: Four OpenAI gym tasks: (a) Ant-v3: make a four-legged creature walk forward as fast as possible, (b) HalfCheetah-v3: make a 2D cheetah robot run as fast as possible, (c) Walker2d-v3: make a 2D bipedal robot walk forward as fast as possible, and (d) Hopper-v3: make a 2D one-legged robot hop forward as fast as possible.

We compared our MDC-SAN against DAN and Pop-DAN (integrated population coding with DAN; it had the same amount of parameters as MDC-SAN for a fair comparison). DAN, Pop-DAN, and our MDC-SAN were all trained in conjunction with deep critic networks of the same structure using TD3 algorithm (Fujimoto, Hoof, and Meger 2018). We evaluated the trained DAN, Pop-DAN, and MDC-SAN on four continuous control tasks (Fig. 3) under the same settings and compared their performance (rewards gained). Pop-DAN and MDC-SAN used the same hyper-parameters as the DAN unless explicitly stated. The hyperparameter configurations of these models were as follows:

Integrate DAN with TD3 Actor network was (256, relu, 256, relu, action dim M, tanh); critic network was (256, relu, 256, relu, 1, linear); actor learning rate was 10^{-3}; critic learning rate was 10^{-3}; reward discount factor was $\gamma = 0.99$; soft target update factor was $\eta = 0.005$; maximum length of replay buffer was $T = 10^6$; Gaussian exploration noise was $\sigma = 0.1, \tilde{\sigma} = 0.2$; noise clip was $c = 0.5$; mini-batch size was $n = 100$; and policy delay factor was $d = 2$;

Integrate Pop-DAN with TD3 Actor network was (Population Encoder, 256, relu, 256, relu, Group Decoder, action dim M, tanh); input population size for single state dimension was $J = 10$; input coding used population coding (C_{pop} for all tasks);

Integrate MDC-SAN with TD3 MDC-SAN used (Population Encoder, 256, DNs, 256, DNs, Group Decoder, action dim M, tanh), where the current decay factor and firing threshold of DNs were both 0.5; input population size for single state dimension was $J = 10$; MDC-SAN learning rate was 10^{-4}; input coding used population coding (C_{pop} for all tasks).

Pre-learning of the dynamic parameters in DNs

We selected Ant-v3 as the basic source task to pre-learn the dynamic parameters of DNs inside the MDC-SAN. All MDC-SAN parameters, including synaptic weights and dynamic parameters, were tuned with approximate BP. The

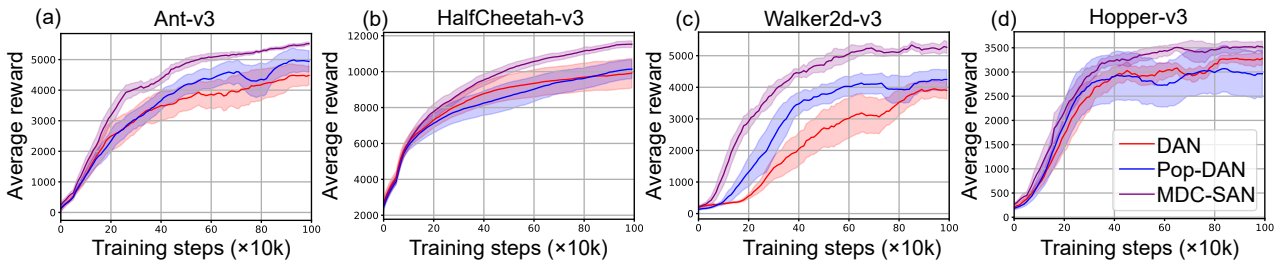

Figure 4: Comparison of average rewards for different models. (a) Performance of DAN, Pop-DAN, and MDC-SAN during training on the Ant-V3 gym task. (b, c, d) Performances of these three models on HalfCheetah-v3, Walker2d-v3, and Hopper-v3, respectively, where our proposed MDC-SAN performed best. The shaded region represents half a standard deviation of the average evaluation over 10 seeds, and the curves are smoothed for clarity.

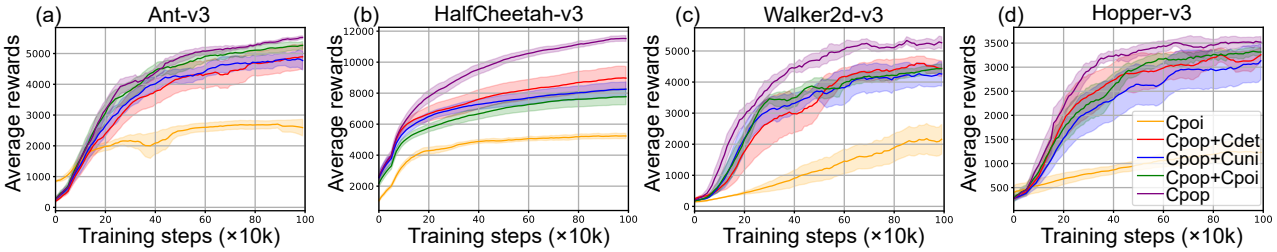

Figure 5: Comprehensive comparison of the impact of various input coding methods, where the SAN using C_{pop} achieved the best performance.

learning curve (not shown) was continuously increased and converged after around 1 million training steps.

After learning, all dynamic parameters related to DNs were clustered to ensure the optimal parameter configuration by selecting their clustering center. As shown in Fig. 6(a, b), we obtained a clustering center of parameters $\theta_{v,u,r,s}$ (the red stars). Here we set $k = 1$ in k-means for simplicity. The best dynamic parameters of DNs were θ_v^*=-0.172 (conductivity of membrane potential), θ_u^*=0.529 (conductivity of hidden state U), θ_r^*=0.021 (reset membrane potential), and θ_s^*=0.132 (spike effect to U). These parameters were denoted as $\boldsymbol{\theta}^*$ and further used as the fixed configuration of all dynamic neurons for all tasks in the following experiments.

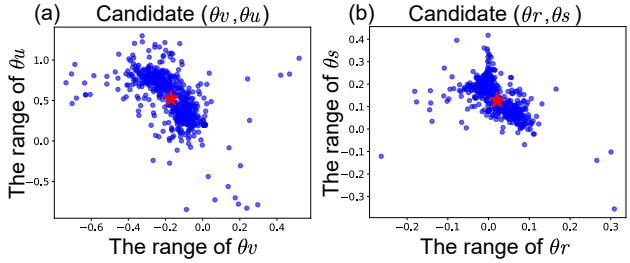

Figure 6: The clustering of the candidate dynamic parameters $\theta_{v,u,r,s}$ after learning on Ant-v3. (a) The parameters of θ_v and θ_u learned from Ant-v3. The single center of the cluster was measured by the standard k-means algorithm with k=1 (labeled as a red star). (b) Similar to that of (a) but for candidate parameters of θ_r and θ_s.

Table 1: Max average rewards over 10 random seeds for DAN, Pop-DAN and MDC-SAN. Bold numbers are maximal values.

Actor Network	Ant	HalfCheetah	Walker2d	Hopper
DAN	4671±777	10020±1696	4139±429	3396±77
Pop-DAN	5146±669	10206±1089	4361±621	3201±957
MDC-SAN	**5628±155**	**11657±400**	**5566±332**	**3636±146**

Benchmarking MDC-SAN against DAN

We compared the performance of our MDC-SAN with DAN and Pop-DAN. As Fig. 4 and Table 1 show, our model achieved the best performance across all four tasks, showing the effectiveness of our proposed MDC-SAN for continuous control tasks. In addition, Pop-DAN had no obvious advantage over DAN on the four tasks. Further analysis in Fig. 5 showed that SAN with population coding achieved significant performance improvements compared to that without population coding. SAN seems to be more compatible with population coding than DAN.

Further discussion of different input codings

We comprehensively compared the impact of various input coding methods on performance while keeping the neuronal coding method fixed to DNs. As Fig. 5 shows, performance of the rate coding method alone (C_{poi}, encode the input state with Poisson coding directly) was far inferior to population coding-based methods ($C_{pop} + C_{uni}$, $C_{pop} + C_{poi}$, $C_{pop} + C_{det}$, and C_{pop}) on all four tasks. This might be be-

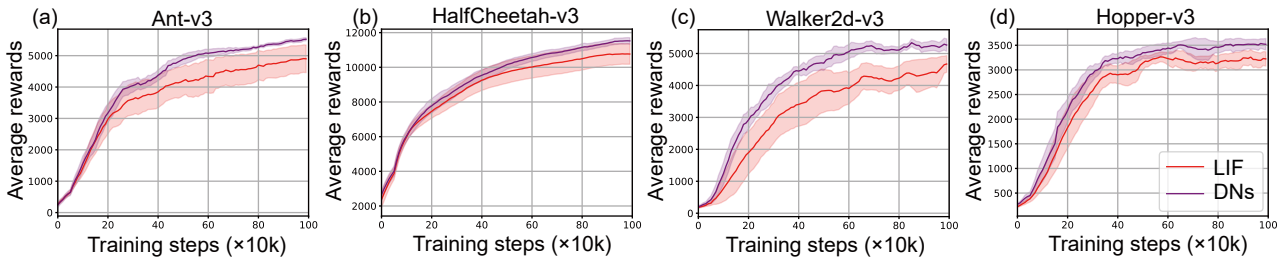

Figure 7: The DNs consistently performed better than LIF neurons on all four tasks.

cause the rate coding methods have an inherent limitation on the representation capacity of individual neurons. And population coding can better separate different states in a higher-dimension manner to produce better input representation. As for the population coding-based methods, C_{pop} achieved the best performance on all tasks. And the other three population coding-based methods had little difference in performance. Hence, it seemed to be more efficient to directly use the analog value of the state after population coding as the network input, without further using rate coding to encode analog value into spike trains.

The neuronal analysis in MDC-SAN

We further tested the constructed DNs on all four tasks and compared them with the LIF neurons while keeping the input coding method fixed to pure population coding (C_{pop}). As shown in Fig. 7, the DNs achieved better performance than LIF neurons on all tested tasks, including the source task (where the dynamic parameters of DNs were learned, i.e., Ant-v3) and other similar tasks (i.e., HalfCheetah-v3, Walker2d-v3, and Hopper-v3). This result verified the generalization capabilities of DNs, i.e., the dynamic parameters of DNs learned from a task could be generalized to other similar tasks.

One possible reason for the performance difference between LIF neurons and DNs was that the DNs contained a higher-order dynamics of membrane potentials, a more complex configuration of conductivity (both θ_v and θ_u) and reset potential (θ_r). Hence, the model using DNs showed a higher complexity at spatial-temporal information processing, which might contribute to the higher performance. An additional simulation of these two types of neurons was given, shown in Fig. 8, including the neuronal dynamics for different explicit (e.g., membrane potential V and stimulated input I) and implicit variables (e.g., resistance item U).

For the standard LIF neuron in Fig. 8(a), the membrane potential was positively proportional to the neuron input. For example, for the sin-like input with the value range from -1 to 1, the dynamic V was dynamically integrated until reaching the firing threshold V_{th} only with a strong positive stimulus, or else decay accordingly with weak-positive or negative stimulus. Unlike LIF neurons, the DNs showed a higher complexity with an additional implicit U, making the dynamical changing of equilibrium points different. The slight differences of U would cause a significant update of V according to the definition of DNs, especially when the parameter θ_u is small in Equation (8). Hence, the DNs would show similar firing patterns with the strong positive stimulus and exhibit a sparse firing with the weak-positive or negative stimulus, instead of stopping firing like LIF neurons. This result showed a better dynamic representation of DNs than LIF neurons.

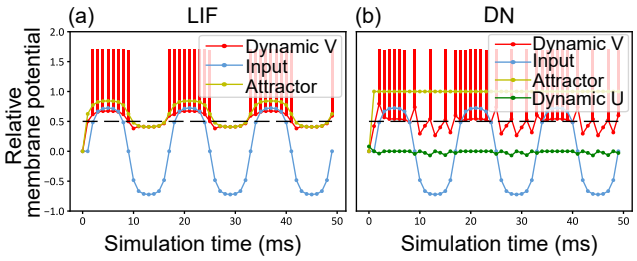

Figure 8: Membrane potential of DNs and LIF. The red lines represent dynamic membrane potential V, green lines are dynamic resistance item U, blue lines are simulated input, and yellow lines are the values of equilibrium points for membrane potentials.

Conclusion

Efficient coding at multi scales is essential for the next-step decision-making in biological neural networks. This paper incorporates spatial population-coding at the input layer and temporal DN-coding at hidden layers as an integrative spiking actor network (MDC-SAN), reaching a better performance on the four benchmark Open-AI gym tasks than its counterpart DAN. Unlike generally used LIF neurons in SAN, the more complex DNs have shown a more vital ability on temporally non-linear information processing and achieved higher performance. In addition, population coding has also demonstrated an advantage in spatial information coding.

In the future, more biologically plausible principles can be borrowed from biological networks and applied to SAN towards lower energy cost, more vital adaptability, and higher robust computation. These interactions between neuroscience and artificial intelligence have much in store for the future.

Acknowledgment

This work was supported by the National Key R&D Program of China (2020AAA0104305), the National Natural Science Foundation of China (61806195), the Strategic Priority Research Program of the Chinese Academy of Sciences (XDA27010404, XDB32070000).

References

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Cheng, X.; Hao, Y.; Xu, J.; and Xu, B. 2020a. LISNN: Improving Spiking Neural Networks with Lateral Interactions for Robust Object Recognition. In *IJCAI*, 1519–1525.

Cheng, X.; Zhang, T.; Jia, S.; and Xu, B. 2020b. Finite Meta-Dynamic Neurons in Spiking Neural Networks for Spatio-temporal Learning. *arXiv preprint arXiv:2010.03140*.

Comsa, I. M.; Potempa, K.; Versari, L.; Fischbacher, T.; Gesmundo, A.; and Alakuijala, J. 2020. Temporal coding in spiking neural networks with alpha synaptic function. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8529–8533. IEEE.

Doya, K. 2000. Reinforcement learning in continuous time and space. *Neural computation*, 12(1): 219–245.

Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; and Abbeel, P. 2016. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, 1329–1338. PMLR.

Florian, R. V. 2007. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural computation*, 19(6): 1468–1502.

Frémaux, N.; Sprekeler, H.; and Gerstner, W. 2013. Reinforcement learning using a continuous time actor-critic framework with spiking neurons. *PLoS computational biology*, 9(4): e1003024.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 1587–1596. PMLR.

Georgopoulos, A. P.; Schwartz, A. B.; and Kettner, R. E. 1986. Neuronal population coding of movement direction. *Science*, 233(4771): 1416–1419.

Gerstner, W.; and Kistler, W. M. 2002. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press.

Harris, K. D.; Csicsvari, J.; Hirase, H.; Dragoi, G.; and Buzsaki, G. 2003. Organization of cell assemblies in the hippocampus. *Nature*, 424(6948): 552–6.

Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; and Meger, D. 2018. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Izhikevich, E. M. 2003. Simple model of spiking neurons. *IEEE Transactions on neural networks*, 14(6): 1569–1572.

Kaelbling, L. P.; Littman, M. L.; and Moore, A. W. 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4: 237–285.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *ICLR (Poster)*.

Maass, W. 1997. Networks of spiking neurons: the third generation of neural network models. *Neural networks*, 10(9): 1659–1671.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533.

O'Brien, M. J.; and Srinivasa, N. 2013. A spiking neural model for stable reinforcement of synapses based on multiple distal rewards. *Neural Computation*, 25(1): 123–156.

Painkras, E.; Plana, L. A.; Garside, J.; Temple, S.; Galluppi, F.; Patterson, C.; Lester, D. R.; Brown, A. D.; and Furber, S. B. 2013. SpiNNaker: A 1-W 18-core system-on-chip for massively-parallel neural network simulation. *IEEE Journal of Solid-State Circuits*, 48(8): 1943–1953.

Patel, D.; Hazan, H.; Saunders, D. J.; Siegelmann, H. T.; and Kozma, R. 2019. Improved robustness of reinforcement learning policies upon conversion to spiking neuronal network platforms applied to Atari Breakout game. *Neural Networks*, 120: 108–115.

Rathi, N.; and Roy, K. 2020. DIET-SNN: Direct input encoding with leakage and threshold optimization in deep spiking neural networks. *arXiv preprint arXiv:2008.03658*.

Rathi, N.; Srinivasan, G.; Panda, P.; and Roy, K. 2019. Enabling Deep Spiking Neural Networks with Hybrid Conversion and Spike Timing Dependent Backpropagation. In *International Conference on Learning Representations*.

Sboev, A.; Vlasov, D.; Rybka, R.; and Serenko, A. 2018. Spiking neural network reinforcement learning method based on temporal coding and STDP. *Procedia computer science*, 145: 458–463.

Shi, M.; Zhang, T.; and Zeng, Y. 2020. A Curiosity-Based Learning Method for Spiking Neural Networks. *Frontiers in Computational Neuroscience*, 14: 7.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Tan, W.; Patel, D.; and Kozma, R. 2021. Strategy and Benchmark for Converting Deep Q-Networks to Event-Driven Spiking Neural Networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 9816–9824.

Tang, G.; Kumar, N.; and Michmizos, K. P. 2020. Reinforcement co-learning of deep and spiking neural networks for energy-efficient mapless navigation with neuromorphic hardware. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 6090–6097. IEEE.

Tang, G.; Kumar, N.; Yoo, R.; and Michmizos, K. P. 2020. Deep Reinforcement Learning with Population-Coded Spiking Neural Network for Continuous Control. *arXiv preprint arXiv:2010.09635*.

Tuckwell, H. C. 1988. *Introduction to theoretical neurobiology: volume 2, nonlinear and stochastic theories*. 8. Cambridge University Press.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Warlop, R.; Lazaric, A.; and Mary, J. 2018. Fighting boredom in recommender systems with linear reinforcement learning. In *Neural Information Processing Systems*.

Yuan, M.; Wu, X.; Yan, R.; and Tang, H. 2019. Reinforcement learning in spiking neural networks with stochastic and deterministic synapses. *Neural computation*, 31(12): 2368–2389.

Zeng, Y.; Zhang, T.; and Xu, B. 2017. Improving multi-layer spiking neural networks by incorporating brain-inspired rules. *Science China Information Sciences*, 60(5): 052201.

Zenke, F.; and Ganguli, S. 2018. Superspike: Supervised learning in multilayer spiking neural networks. *Neural computation*, 30(6): 1514–1541.

Zhang, T.; Zeng, Y.; Zhao, D.; Wang, L.; Zhao, Y.; and Xu, B. 2016. HMSNN: Hippocampus inspired Memory Spiking Neural Network. In *IEEE International Conference on Systems, Man and Cybernetics*, 002301–002306.

Zhang, Tielin and Zeng, Yi and Shi, Mengting and Zhao, Dongcheng. 2018. A Plasticity-centric Approach to Train the Non-differential Spiking Neural Networks. In *Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence*, 620–628.

Zou, L.; Xia, L.; Ding, Z.; Song, J.; Liu, W.; and Yin, D. 2019. Reinforcement learning to optimize long-term user engagement in recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2810–2818.