Letter

Deep reinforcement learning for feedback control in a collective flashing ratchet

Dong-Kyum Kim [1] and Hawoong Jeong [1,2,*]

[1]Department of Physics, Korea Advanced Institute of Science and Technology, Daejeon 34141, Korea

[2]Center for Complex Systems, Korea Advanced Institute of Science and Technology, Daejeon 34141, Korea

(Received 20 November 2020; accepted 4 March 2021; published 2 April 2021)

A collective flashing ratchet transports Brownian particles using a spatially periodic, asymmetric, and time-dependent on-off switchable potential. The net current of the particles in this system can be substantially increased by feedback control based on the particle positions. Several feedback policies for maximizing the current have been proposed, but optimal policies have not been found for a moderate number of particles. Here, we use deep reinforcement learning (RL) to find optimal policies, with results showing that policies built with a suitable neural network architecture outperform the previous policies. Moreover, even in a time-delayed feedback situation where the on-off switching of the potential is delayed, we demonstrate that the policies provided by deep RL provide higher currents than the previous strategies.

DOI: 10.1103/PhysRevResearch.3.L022002

Introduction. A flashing ratchet is a nonequilibrium model that induces a net current of Brownian particles in a spatially periodic asymmetric potential that can be temporally switched on and off [1–4]. If one can access the position information of the particles, the current can be greatly improved by feedback control that switches the potential on-off based on the position information [5]. Feedback strategies for maximizing the current in flashing ratchets have been extensively studied [4–13] due to the model's applicability in various disciplines [14]; for instance, flashing ratchets have been used for explaining transport phenomena in biological processes such as ion pumping [15], molecular transportation [16], and by motor proteins [17–20]. However, the proposed feedback strategies [4–11] are not optimal policies for a moderate number of particles and require prior information of the system as well.

Owing to the recent advances in deep learning [21], physicists in diverse fields have been applying it to complex problems that are analytically intractable, e.g., glassy systems [22], quantum matter [23], and others [24]. In particular, reinforcement learning (RL) [25] has shown unprecedented success in previously unsolvable problems through combination with deep neural networks [26–29]. This framework, so-called deep RL, has become a highly efficient tool for quantum feedback control, showing similar or better performance than previous handcrafted policies [30–34]. In this Letter, we employ deep RL to obtain optimal policies in the collective flashing ratchet model, and validate our approach by application to a time-delayed feedback situation that occurs in actual experiments [12].

Collective flashing ratchet. We consider the collective flashing ratchet model [5], which consists of an ensemble of N noninteracting Brownian particles in contact with a heat bath at temperature T and that drift in a spatially periodic asymmetric potential U. The dynamics of the N particles is governed by the following overdamped Langevin equation,

$$\begin{aligned} \eta\dot{x}_i(t) &= \alpha(s_t)F(x_i(t)) + \xi_i(t), \\ s_t &\equiv \{x_1(t), \ldots, x_N(t)\}, \\ i &= 1, \ldots, N, \end{aligned} \tag{1}$$

where $x_i(t)$ is the position of particle i, η is the friction coefficient, and ξ_i is a Gaussian noise with zero mean and correlation $\mathbb{E}[\xi_i(t)\xi_j(t')] = 2\eta k_B T\delta_{ij}\delta(t-t')$ where $\mathbb{E}$ denotes the ensemble average. Here, α is a deterministic control policy that depends on a set of positions s_t with an output of 0 (off) or 1 (on). The force is given by $F(x) = -\partial_x U(x)$ with the potential [see Fig. 1(a)]

$$U(x) = U_0\left[\sin\left(\frac{2\pi x}{L}\right) + \frac{1}{4}\sin\left(\frac{4\pi x}{L}\right)\right]. \tag{2}$$

In all simulations, we set $L = 1$, $k_B T = 1$, diffusion coefficient $D = k_B T/\eta = 1$, $U_0 = 5k_B T$, and time step size $\Delta t = 10^{-3}L^2/D$. The current of the particles in steady state under policy α is denoted as

$$\mathbb{E}_\alpha[\dot{x}] \equiv \mathbb{E}_\alpha\left[\frac{1}{N}\sum_{i=1}^{N}\dot{x}_i\right] \quad \text{(unit: } D/L\text{)}. \tag{3}$$

Various policies for maximizing the current (3) have been proposed as follows: the periodic switching policy [4], maximizing instantaneous current (greedy policy) [5], threshold policy [6–8], and Bellman's criterion [13].

The periodic switching policy [4] is $\alpha(t) = 1$ for $t \in [0, \mathcal{T}_{\text{on}})$, $\alpha(t) = 0$ for $t \in [\mathcal{T}_{\text{on}}, \mathcal{T}_{\text{on}} + \mathcal{T}_{\text{off}})$, and periodic

*hjeong@kaist.edu

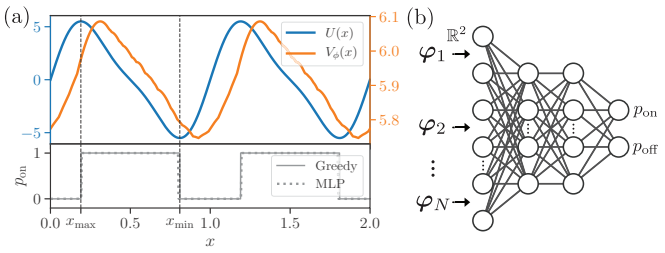

FIG. 1. (a) $N = 1$ case. Top: Potential U and trained value network V_ϕ as a function of position x are denoted by blue and orange lines, respectively. Bottom: The solid line denotes the probability of switching on the potential (p_{on}) as a function of x for the greedy policy. The dotted line represents p_{on} of the trained MLP policy. (b) Illustration of a MLP with two hidden layers for the policy network π_θ.

$\alpha(t + \mathcal{T}_{on} + \mathcal{T}_{off}) = \alpha(t)$ with optimal periods $\mathcal{T}_{on} \approx 0.03L^2/D$ and $\mathcal{T}_{off} \approx 0.04L^2/D$. For any N, this policy gives the current $\mathbb{E}_\alpha[\dot{x}] \approx 0.862D/L$ because it does not depend on the position but only time.

The greedy policy [5] is defined as $\alpha(s_t) = \Theta(f(s_t))$, where $f(s_t) = \sum_{i=1}^N F(x_i(t))/N$ is the mean force and Θ is the Heaviside function given by $\Theta(z) = 1$ if $z > 0$ or else 0. While the greedy policy is the optimal one for $N = 1$, this policy is outperformed by the periodic switching policy for large N.

The threshold policy [6–8] is $\alpha(s_t) = 0$ if $f(s_t) \leqslant u_{on}$ when $f(t)$ is decreasing, and $\alpha(s_t) = 1$ if $f(s_t) \geqslant u_{off}$ when $f(t)$ is increasing, with thresholds $u_{on} \geqslant 0$ and $u_{off} \leqslant 0$. The threshold policy with optimal thresholds gives a mostly similar performance to the greedy policy for $N < 10^2$–10^3 and is better than the greedy policy for larger N. It is also optimal for $N = \infty$, which is equivalent to the periodic switching policy.

Neither greedy nor threshold policy is optimal for finite $N > 1$. Roca *et al.* [13] proposed a general framework for finding the optimal policy via Bellman's principle, and found it for $N = 2$ using numerical integration. However, this numerical method requires prior information of the model and is computationally infeasible for large N due to the curse of dimensionality.

Methods. We employ the actor-critic algorithm, which is one of the policy gradient methods in RL [25], together with deep neural networks to find the optimal policies in the collective flashing ratchet for any N.

To formulate this problem in RL language, we define the reward as the total mean displacement of the particles:

$$r_t = \frac{1}{N}\sum_{i=1}^{N}(x_i(t + \Delta t) - x_i(t)). \tag{4}$$

The total discounted reward from time t, called the return, is $G_t = \sum_{k=0}^{\infty}\gamma^k r_{t+(k+1)\Delta t}$ where $\gamma \in [0, 1)$ is the discounting factor and we set $\gamma = 0.999$. We build a policy network π_θ, called the actor, where θ denotes the trainable neural network parameters, that takes system state s as an input. The outputs $\pi_\theta(s) = (p_{on}, p_{off})$ are the probabilities for switching the potential on or off [see Fig. 1(b)]. We sample the on-off probability from $\pi_\theta(s_t)$ every t in the training process.

The goal in RL is to obtain the optimal policy π^* that maximizes the expected total future reward, i.e., $\pi^* = \arg\max_\pi \mathbb{E}_\pi[G_t]$. If the equation of motion is known, $\mathbb{E}_\pi[G_t]$ can be numerically calculated using Bellman's equation [13]. However, in this Letter, we assume that we can only access the system state s_t and reward r_t. In such a case, called a model-free RL, we need an estimator V_ϕ for a *value function*,

$$V^\pi(s_t) = \mathbb{E}_\pi[G_t|s_t], \tag{5}$$

which is the expected return a given state s_t under a policy π. The estimator V_ϕ, called the value network or critic, where ϕ denotes the trainable parameters, is also built with another neural network.

There are various optimization methods for the actor-critic algorithm [35]. Among them, we employ proximal policy optimization [36], which is widely used in RL because of its scalability, data efficiency, and robustness for hyperparameters (see Supplemental Material [37] for training details). After the training process is complete, we test the policy deterministically, i.e.,

$$\alpha(s_t) = \begin{cases} 1 & \text{if } p_{on} > 0.5, \\ 0 & \text{if } p_{off} > 0.5, \end{cases} \quad \text{where } (p_{on}, p_{off}) = \pi_\theta(s_t).$$

Neural network architecture. First, we employ multilayer perceptron (MLP) architecture for the policy network π_θ and value network V_ϕ [see Fig. 1(b)]. The configuration details of the neural network architectures are given in the Supplemental Material [37]. Using the periodicity of the potential $U(x)$, we transform the state s_t into the input feature $\psi_t = [\boldsymbol{\varphi}_1(t), \boldsymbol{\varphi}_2(t), \ldots, \boldsymbol{\varphi}_N(t)]$ for the neural network input where

$$\boldsymbol{\varphi}_i(t) = \left[\cos\left(\frac{2\pi x_i(t)}{L}\right), \sin\left(\frac{2\pi x_i(t)}{L}\right)\right]. \tag{6}$$

Therefore, the input dimension of the MLP is $2N$ and the output dimension is two for π_θ. The value network V_ϕ has the same configuration except for having an output dimension of one rather than two. We note that the discounting factor $\gamma = 0.999$, which indicates the return G_t, can effectively be considered as the total mean displacement between t and $t + \Delta t/(1 - \gamma)$. Accordingly, $V_\phi(\psi_t)$ can be interpreted as the expected current given ψ_t because the time step size is $\Delta t = 10^{-3}L^2/D$.

For the $N = 1$ case, Fig. 1(a) shows that the trained π_θ agrees with the greedy policy (bottom panel), while V_ϕ is slightly shifted to the right from potential U (top panel). This is because, at the top of the potential valley (x_{max}), the particle can slide to the right or left with a 50/50 chance, and therefore the expected current is maximum slightly right of x_{max}.

For the $N = 2$ case, as shown in the left panel of Fig. 2(b), the greedy policy switches on (off) the potential when the particles are inside (outside) the white contour. On the other hand, the decision boundary of the trained MLP policy π_θ (red contour) agrees with the policy discovered by Roca *et al.* [13] and shows better performance than the greedy policy by considering the future expected current. For instance, in the orange dashed area, the instantaneous net current will be negative because the mean force $f(x_1, x_2)$ is negative when the potential is on. But considering each particle with a long-term view, particles 1 and 2 are located on the downhill of

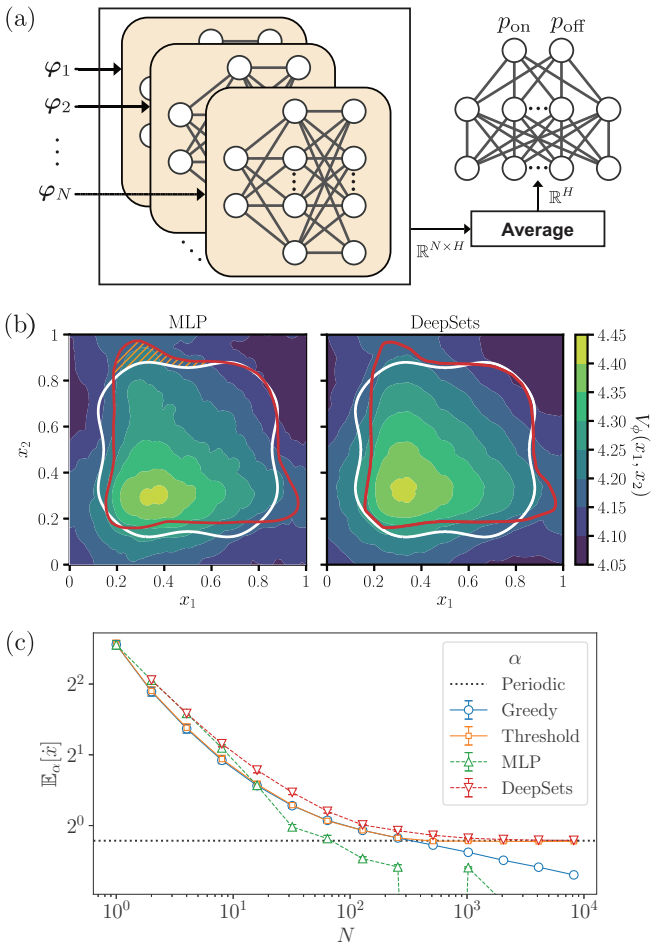

FIG. 2. (a) DeepSets architecture for the policy network π_θ. H is the number of hidden units for each layer. (b) Decision boundaries from a trained MLP (left) and trained DeepSets (right) for $N = 2$. The white contour denotes where the mean force $f(x_1, x_2)$ is zero. The red contour is $p_{\rm on} = 0.5$ from the trained policy network π_θ. The color gradient represents the trained value network V_ϕ. (c) Current $\mathbb{E}_\alpha[\dot{x}]$ as a function of N for each policy α. Throughout this Letter, error bars represent the standard deviation of the current measured from the realized trajectory ensemble over the period $t = 50L^2/D$.

the potential ($x_{\rm max} < x < x_{\rm min}$) and near the minimum ($x_{\rm min}$), respectively; while particle 2 will soon reach $x_{\rm min}$ and become trapped in the potential well, particle 1 can keep moving down along the potential [13].

However, the decision boundary (red contour) and V_ϕ (color gradient) are not symmetric over the line $x_1 = x_2$ [see Fig. 2(b), left] because MLP outputs are not permutation invariant to the order of the elements in the input feature ψ_t. To address this issue, we employ a permutation invariant architecture, called DeepSets [38], for the policy and value networks. In this architecture [see Fig. 2(a)], each element φ_i in the input feature ψ_t is independently fed into a single MLP (beige), and the outputs of the MLP are averaged over the elements and then fed to an another MLP. By using DeepSets for training, the decision boundary and V_ϕ show perfect symmetry over the $x_1 = x_2$ line [see Fig. 2(b), right].

Now we apply these methods for $N = 2^2, 2^3, \ldots, 2^{13}$, and compare the training results with the greedy (blue circles), threshold (orange squares), and periodic switching (black dotted line) policies in Fig. 2(c). Results show that the trained MLP policies (green triangles) outperform the greedy and threshold policies for $N < 10$, but perform poorly for $N > 10$ due to the lack of permutation invariance. On the other hand, the trained DeepSets policies (red triangles) outperform the other policies for any $N > 1$ while converging to the periodic policy as N increases (see Fig. S1, Supplemental Material [37]). We have also verified that deep RL works well for the sawtooth potential (Fig. S2, Supplemental Material [37]).

Time-delayed feedback. In an actual experiment, there is an inevitable time delay between the measurement and the feedback due to the calculation time in the feedback algorithm [9–12]. To verify that deep RL is applicable to such a realistic situation, we consider a feedback time delay τ in Eq. (1), i.e., $\alpha(s_t)$ is replaced by $\alpha(s_{t-\tau})$. In this case, the maximal net displacement (MND) policy [11], defined by

$$\alpha(s_{t-\tau}) = \Theta\left(\sum_{i=1}^{N} d(x_i(t-\tau))\right), \tag{7}$$

where the displacement function is $d(x) = x_{\rm min} + x_0 - x$ for $x_{\rm max} < x \leqslant x_{\rm max} + L$ and periodic $d(x) = d(x + L)$, can perform better than the greedy policy for $\tau > 0$ with optimal $x_0 < 0$ [12]. This can be considered as a τ-delayed greedy policy because it predicts the arrival of the particles at $x_{\rm min}$ after τ from $x_0 + x_{\rm min}$. We train the neural networks for $N = 1, 2^1, 2^2, \ldots, 2^5$ with time delay τ in the range of 0.00–$0.05L^2/D$, and compare them with the greedy policy and the MND policy with optimal x_0.

For the time-delayed $N = 1$ case [see Fig. 3(b), first row], the results show that the trained MLP policies (gray diamonds) agree with the MND policy (orange triangles) and perform better than the greedy policy (blue circles). For $N = 2$, the trained DeepSets policies (green triangles) outperform the greedy policy and are slightly better than the MND policy.

While the actor-critic algorithm assumes that the feedback-controlled system is a Markov decision process (MDP), the delayed-feedback process is not a MDP because the next state $s_{t+\Delta t}$ not only depends on the previous state s_t but also the history of the on-off information. This problem can be reformulated as a MDP by augmenting the input feature ψ_t with the on-off history [39]. Here, the d-step augmented state at time t is defined as

$$I_t = (\alpha_{t-\tau}, \alpha_{t-\tau+\Delta t}, \ldots, \alpha_{t-\tau+(d-1)\Delta t}, \psi_t), \quad d = \tau/\Delta t.$$

In order to efficiently handle the augmented state, we build the policy network with a recurrent neural network (RNN). We employ an embedding layer to transform the discrete variable α into a continuous variable, and we use a gated recurrent unit (GRU) [40], a widely used gating mechanism in RNNs due to its parameter efficiency and good performance on the sequential data sets, for the RNN. As shown in Fig. 3(a), we concatenate the output vectors from DeepSets (orange nodes) and the RNN (blue nodes), where DeepSets and the RNN encode the position information ψ_t and potential on-off history, respectively. We then feed the concatenated vector to a MLP. See the Supplemental Material [37] for the configuration details. As can be seen in Fig. 3(b), the trained RNN policies (red stars) show slightly better performance than the other policies

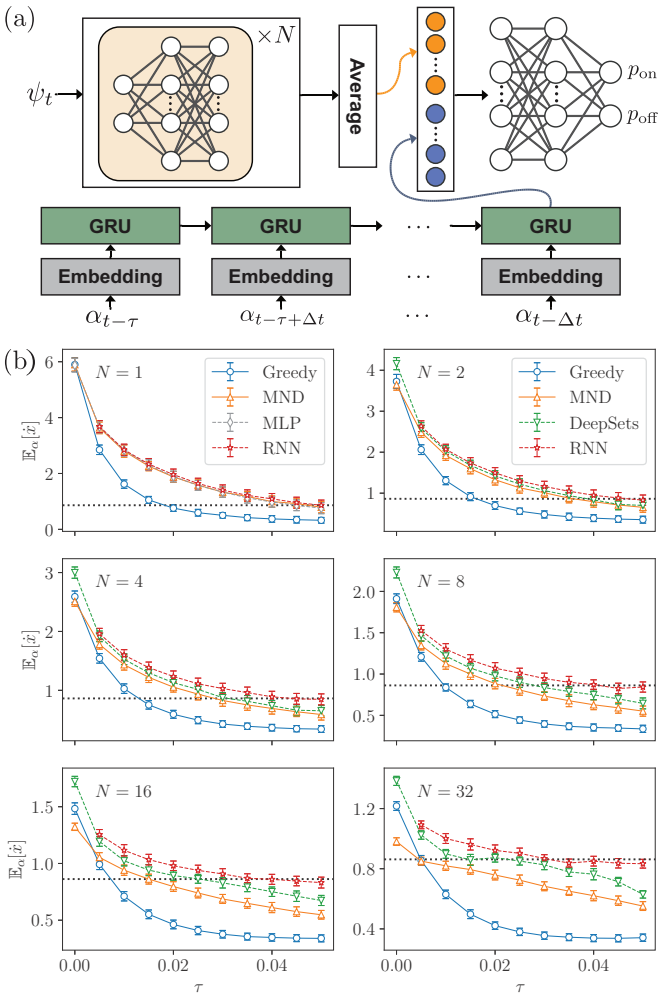

FIG. 3. (a) Architecture of policy network π_θ augmented with an RNN. (b) Time-delayed feedback results for the greedy, MND, MLP (only for $N = 1$), DeepSets (for $N > 1$), and RNN policies at increasing N. The black dotted lines denote the current of the periodic switching policy.

for $N = 1$ and noticeably better performance than the others for $N = 2$. And also, the RNN policies outperform the greedy, MND, and DeepSets policies for the $N = 4, 8, 16, 32$ cases.

Conclusions and outlook. We have tackled the problem of finding an improved policy for maximizing the current in the collective flashing ratchet model through deep RL. Unlike the previous model-based method [13], the model-free RL approach used in this study does not require information on the parameters of the system (e.g., potential, diffusion coefficient, and others). The deep RL approach makes it is possible to find state-of-the-art feedback strategies using suitable neural network architectures through training only in the process of interacting with the environment. Also, we have demonstrated that deep RL outperforms the previous strategies in a time-delayed feedback situation; therefore, we expect that this study can be effectively applied experimentally.

Although feedback control in the collective flashing ratchet can induce an effective coupling between noninteracting particles, molecular motors such as kinesin, for example, explicitly interact with each other via hard-core repulsion. According to previous studies on interacting molecular motors [17–19], their cooperative behavior can enhance transportation ability several times or more compared to individual motors. Further research applying deep RL on interacting molecular motors will be intriguing.

Another interesting future task would be the application of deep RL to a collective flashing ratchet in which a time-periodic external driving force acts on the particles [41]. A ratchetlike mechanism for transportation in the cell membrane (such as ion pumping [15] or glycerol transportation [16]) can improve the current via the periodic driving force. Therefore, investigating whether a deep RL agent can exploit not only fluctuations in the environment but also time-dependent environmental dynamics is expected to aid the understanding of such biological processes.

In real-world scenarios, there may be measurement or feedback errors due to instrument noise [42–44]. Such cases are not only important in physics, e.g., information thermodynamics [45], but also in RL for real-world applications [46]. Therefore, it will also be an interesting future work to study RL from a thermodynamics perspective; we expect that the collective flashing ratchet model can be utilized as a useful environment to benchmark RL algorithms in such situations.

The results of all runs and the code implemented in PYTORCH [47] are available in the Supplemental Material [37].

Acknowledgments. This study was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) (Grant No. NRF-2017R1A2B3006930).

[1] J. Prost, J.-F. Chauwin, L. Peliti, and A. Ajdari, Asymmetric Pumping of Particles, Phys. Rev. Lett. **72**, 2652 (1994).
[2] R. D. Astumian and M. Bier, Fluctuation Driven Ratchets: Molecular Motors, Phys. Rev. Lett. **72**, 1766 (1994).
[3] R. D. Astumian, Thermodynamics and kinetics of a Brownian motor, Science **276**, 917 (1997).
[4] M. B. Tarlie and R. D. Astumian, Optimal modulation of a Brownian ratchet and enhanced sensitivity to a weak external force, Proc. Natl. Acad. Sci. USA **95**, 2039 (1998).
[5] F. J. Cao, L. Dinis, and J. M. R. Parrondo, Feedback Control in a Collective Flashing Ratchet, Phys. Rev. Lett. **93**, 040603 (2004).
[6] L. Dinis, J. M. R. Parrondo, and F. J. Cao, Closed-loop control strategy with improved current for a flashing ratchet, Europhys. Lett. **71**, 536 (2005).
[7] M. Feito and F. J. Cao, Threshold feedback control for a collective flashing ratchet: Threshold dependence, Phys. Rev. E **74**, 041109 (2006).
[8] M. Feito and F. J. Cao, Optimal operation of feedback flashing ratchets, J. Stat. Mech. (2009) P01031.
[9] M. Feito and F. J. Cao, Time-delayed feedback control of a flashing ratchet, Phys. Rev. E **76**, 061113 (2007).
[10] E. M. Craig, B. R. Long, J. M. R. Parrondo, and H. Linke, Effect of time delay on feedback control of a flashing ratchet, Europhys. Lett. **81**, 10002 (2007).

[11] E. M. Craig, N. J. Kuwada, B. J. Lopez, and H. Linke, Feedback control in flashing ratchets, Ann. Phys. **17**, 115 (2008).

[12] B. J. Lopez, N. J. Kuwada, E. M. Craig, B. R. Long, and H. Linke, Realization of a Feedback Controlled Flashing Ratchet, Phys. Rev. Lett. **101**, 220601 (2008).

[13] F. Roca, J. P. G. Villaluenga, and L. Dinis, Optimal protocol for a collective flashing ratchet, Europhys. Lett. **107**, 10006 (2014).

[14] P. Reimann, Brownian motors: Noisy transport far from equilibrium, Phys. Rep. **361**, 57 (2002).

[15] Z. Siwy and A. Fuliński, Fabrication of a Synthetic Nanopore Ion Pump, Phys. Rev. Lett. **89**, 198103 (2002).

[16] I. Kosztin and K. Schulten, Fluctuation-Driven Molecular Transport Through an Asymmetric Membrane Channel, Phys. Rev. Lett. **93**, 238102 (2004).

[17] O. Campàs, Y. Kafri, K. B. Zeldovich, J. Casademunt, and J.-F. Joanny, Collective Dynamics of Interacting Molecular Motors, Phys. Rev. Lett. **97**, 038101 (2006).

[18] J. Brugués and J. Casademunt, Self-Organization and Cooperativity of Weakly Coupled Molecular Motors under Unequal Loading, Phys. Rev. Lett. **102**, 118104 (2009).

[19] D. Oriola and J. Casademunt, Cooperative Force Generation of KIF1A Brownian Motors, Phys. Rev. Lett. **111**, 048103 (2013).

[20] W. Hwang and M. Karplus, Structural basis for power Stroke vs. Brownian ratchet mechanisms of motor proteins, Proc. Natl. Acad. Sci. USA **116**, 19777 (2019).

[21] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, MA, 2016).

[22] V. Bapst, T. Keck, A. Grabska-Barwińska, C. Donner, E. D. Cubuk, S. S. Schoenholz, A. Obika, A. W. R. Nelson, T. Back, D. Hassabis, and P. Kohli, Unveiling the predictive power of static structure in glassy systems, Nat. Phys. **16**, 448 (2020).

[23] J. Carrasquilla, Machine learning for quantum matter, Adv. Phys.: X **5**, 1797528 (2020).

[24] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, Machine learning and the physical sciences, Rev. Mod. Phys. **91**, 045002 (2019).

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

[26] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, Human-level control through deep reinforcement learning, Nature (London) **518**, 529 (2015).

[27] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, Mastering the game of Go with deep neural networks and tree search, Nature (London) **529**, 484 (2016).

[28] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play, Science **362**, 1140 (2018).

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

[30] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt, Reinforcement Learning with Neural Networks for Quantum Feedback, Phys. Rev. X **8**, 031084 (2018).

[31] R. Porotti, D. Tamascelli, M. Restelli, and E. Prati, Coherent transport of quantum states by deep reinforcement learning, Commun. Phys. **2**, 61 (2019).

[32] M. Y. Niu, S. Boixo, V. N. Smelyanskiy, and H. Neven, Universal quantum control through deep reinforcement learning, npj Quantum Inf. **5**, 33 (2019).

[33] Z. An and D. L. Zhou, Deep reinforcement learning for quantum gate control, Europhys. Lett. **126**, 60002 (2019).

[34] Z. T. Wang, Y. Ashida, and M. Ueda, Deep Reinforcement Learning Control of Quantum Cartpoles, Phys. Rev. Lett. **125**, 100401 (2020).

[35] J. Achiam, Spinning Up in Deep Reinforcement Learning, 2018, https://spinningup.openai.com.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, Proximal policy optimization algorithms, arXiv:1707.06347.

[37] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevResearch.3.L022002 for the training details, hyperparameters, neural network architecture configurations, policy and value networks over time, the results on the sawtooth potential, and the source code for the runs and results, which includes Refs. [35,36,47–50].

[38] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, Deep sets, in *Advances in Neural Information Processing Systems 30* (Curran Associates, Long Beach, CA, 2017), pp. 3391–3401.

[39] K. V. Katsikopoulos and S. E. Engelbrecht, Markov decision processes with delays and asynchronous cost collection, IEEE Trans. Autom. Control **48**, 568 (2003).

[40] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (Association for Computational Linguistics, Doha, Qatar, 2014), pp. 1724–1734.

[41] M. Feito, J. P. Baltanás, and F. J. Cao, Rocking feedback-controlled ratchets, Phys. Rev. E **80**, 031128 (2009).

[42] F. J. Cao, M. Feito, and H. Touchette, Information and flux in a feedback controlled Brownian ratchet, Physica A **388**, 113 (2009).

[43] F. J. Cao and M. Feito, Thermodynamics of feedback controlled systems, Phys. Rev. E **79**, 041118 (2009).

[44] T. Sagawa and M. Ueda, Nonequilibrium thermodynamics of feedback control, Phys. Rev. E **85**, 021104 (2012).

[45] J. M. R. Parrondo, J. M. Horowitz, and T. Sagawa, Thermodynamics of information, Nat. Phys. **11**, 131 (2015).

[46] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, An empirical investigation of the challenges of real-world reinforcement learning, arXiv:2003.11881.

[47] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, PyTorch:

An imperative style, high-performance deep learning library, in *Advances in Neural Information Processing Systems 32* (Curran Associates, Vancouver, 2019), pp. 8024–8035.

[48] V. Nair and G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel* (Omnipress, Madison, WI, 2010), pp. 807–814.

[49] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization, in *International Conference on Learning Representations* (2015), arXiv:1412.6980.

[50] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, High-dimensional continuous control using generalized advantage estimation, in *International Conference on Learning Representations* (2016), arXiv:1506.02438.