박 사 학 위 논 문
Ph.D. Dissertation

딥 러닝을 이용한 비평형 통계물리 연구

Nonequilibrium Statistical Physics Study using Deep Learning

2022

김 동 겸 (金 東 謙 Kim, Dong-Kyum)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

박 사 학 위 논 문

딥 러닝을 이용한 비평형 통계물리 연구

2022

김 동 겸

한 국 과 학 기 술 원

물리학과

딥 러닝을 이용한 비평형 통계물리 연구

김 동 겸

위 논문은 한국과학기술원 박사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2021년 12월 13일

심사위원장 정 하 웅 (인)

심 사 위 원 한 명 준 (인)

심 사 위 원 양 용 수 (인)

심 사 위 원 백 용 주 (인)

심 사 위 원 조 정 효 (인)

Nonequilibrium Statistical Physics Study using Deep Learning

Dong-Kyum Kim

Advisor: Hawoong Jeong

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Physics

Daejeon, Korea
December 13, 2021

Approved by

Hawoong Jeong
Professor of Physics

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

DPH 김동겸. 딥 러닝을 이용한 비평형 통계물리 연구. 물리학과 . 2022년. 50+iv 쪽. 지도교수: 정하웅. (영문 논문)
Dong-Kyum Kim. Nonequilibrium Statistical Physics Study using Deep Learning. Department of Physics . 2022. 50+iv pages. Advisor: Hawoong Jeong. (Text in English)

초 록

자연의 생물학적 시스템 및 활성 물질과 같은 수많은 시스템은 주변 환경의 연료 에너지를 사용하여 비평형 과정을 통해 일을 수행한다. 예를 들어, 모든 생명 현상은 ATP라는 연료를 소비함으로써 발생한다. 이러한 과정을 이해하려면 일과 열 사이의 에너지 변환 과정을 열역학이라는 프레임워크로 기술해야 한다. 본 논문에서는 딥 러닝을 이용하여 자유도가 높은 복잡한 시스템의 비평형 과정에서 발생하는 열을 측정하는 새로운 방법을 개발하고, 이러한 시스템이 일을 최대화하는 최적의 메커니즘을 찾기 위한 연구를 수행했다. 먼저, 우리는 비평형 과정에서 엔트로피 생산량을 추론하는 새로운 방법인 엔트로피 생산량 추정 신경망을 개발하였다. 엔트로피 생산량은 확률적 열역학에서 프로세스의 에너지 역학을 기술하는 핵심 양이지만 시스템의 자유도가 높은 경우 차원의 저주로 인해 기존 방법으로는 측정하기 어렵다. 엔트로피 생산량 추정 신경망은 딥 러닝을 사용하여 이러한 기존 방식의 한계를 극복했다. 특히 이 방법은 프로세스의 작동 메커니즘에 대한 정보 없이 관련 변수의 궤적 데이터만 사용하여 엔트로피 생산량을 추정한다. 다음으로, 우리는 비평형 시스템이 일을 최대로 수행하기 위한 최적의 메커니즘이 무엇일지를 심층 강화 학습을 통해 연구하였다. 구체적으로, 우리는 집단적 점멸 래칫 모델에 의해 설명되는 작은 생물계의 수송 현상을 조사하였다. 이 모델은 비대칭적인 포텐셜을 이용하여 입자를 운반하며, 입자 위치에 따른 피드백 제어를 통해 입자들의 흐름을 증가시킬 수 있다. 흐름을 최대화하기 위한 몇 가지 피드백 전략이 제안되었지만 여러 입자에 대한 최적의 전략은 보고되지 않았다. 연구 결과, 심층 강화 학습으로 학습한 전략이 기존 전략들보다 더 큰 흐름을 만들 수 있었다. 또한 실제 실험에서 발생하는 시간 지연 피드백 상황에서도 이러한 접근 방식을 적용하여 마찬가지로 흐름 향상을 확인하였다. 본 논문에서 제안한 인공지능 접근 방법들이 자연에 존재하는 복잡한 비평형 시스템들을 이해하는 데 큰 도움이 될 것으로 기대한다.

핵 심 낱 말 비평형 통계물리, 확률적 열역학, 딥러닝, 인공지능, 강화학습

Abstract

Numerous systems in nature, such as biological systems and active matter, use the energy of fuels in the surrounding environment to perform work through nonequilibrium processes. For example, all life phenomena occur by consuming a fuel called ATP. To understand such processes, we should describe the transformation process between work and heat with thermodynamics. In this thesis, we developed a new method to measure generated heat in the nonequilibrium process of complex systems with high degrees of freedom using deep learning; we also studied to find out the optimal mechanism for maximizing the work of such systems. First, we developed the neural estimator for entropy production (NEEP), a novel method for inferring entropy production (EP) in a nonequilibrium process. While EP is a key quantity in stochastic thermodynamics to describe the energetics of the process, it is difficult to measure using the conventional methods due to the curse of dimensionality when the system has high degrees of freedom. Our NEEP can efficiently address this problem by using deep learning. In particular, this method estimates EP with only trajectory data of the relevant variables, without information about the underlying mechanism of the process. Next, we studied through deep reinforcement learning what the optimal mechanism would be for the nonequilibrium system to perform work optimally. To be specific, we investigated the transport phenomena in small biological systems that have been described by a

collective flashing ratchet model. This model transports particles using an asymmetric potential, and the net current of the particles can be increased by feedback control based on the particle positions. Several feedback strategies for maximizing the current have been proposed, but optimal policies have not been reported for a moderate number of particles. The results showed that policies discovered by deep reinforcement learning outperform the previous policies. Moreover, we demonstrated this approach by application to a time-delayed feedback situation that occurs in actual experiments. Our AI-based approaches presented in this thesis are expected to be useful for understanding complex nonequilibrium systems in nature.

Keywords Nonequilibrium statistical physics, Stochastic thermodynamics, Deep learning, Artificial intelligence, Reinforcement learning

Contents

List of Tables

List of Figures

Chapter 1. Introduction

Classical thermodynamics (or macroscopic thermodynamics) provides a framework for describing the average behavior of macroscopic systems, consist of a large number N of particles, using thermodynamic variables such as temperature, energy, entropy, and others. Its core result is that a quantity called entropy production can never decrease in any process. This result, called the second law of thermodynamics, leads to fundamental limits on the efficiency of heat engines and refrigerators. In the macroscopic world, the fluctuations in thermodynamic variables are negligible because the relative deviations from the mean value are proportional to $1/\sqrt{N}$, therefore the results from classical thermodynamics are valid in the thermodynamic limit (i.e. $N \to \infty$ with fixed density). But, when we observe the microscopic system with a very small time scale, the fluctuations are not negligible and play an important role in the dynamics. Also, most of the systems in the small world we interest in, such as colloidal particles, biopolymers, and molecular motors, are in a non-equilibrium regime where classical thermodynamics is hard to apply. In such non-equilibrium systems, we require the new framework called stochastic thermodynamics [1] for describing the energetics of fluctuating dynamics.

Fluctuations in small systems can cause rare events that never occur in macroscopic systems, such as entropy in a single trajectory transiently decrease. Here, 'entropy in a single trajectory', called stochastic entropy, is a key idea in stochastic thermodynamics. Once the stochastic entropy is defined, one can derive the fluctuation theorems which are exact relationships that express the properties of the probability distribution of stochastic trajectories of the quantity, such as work and entropy [2]. Based on this, the generalization of the second law of thermodynamics can be established in stochastic thermodynamics. Moreover, another core result of stochastic thermodynamics is the generalization of the first law by defining applied work, internal energy, and heat along individual trajectories [3]. Thereby, we are now able to describe the behavior of the nonequilibrium systems through this framework.

Within this framework, the first step for understanding the energetics of the system is measuring entropy production or dissipation. By measuring the entropy production of the nonequilibrium process, one can tell whether the system is passive (in equilibrium) or active (out of equilibrium). And also, it can be used to infer the used free energy during the process. For instance, biological processes perform work using the energy released from the hydrolysis of ATP into ADP; so the sum of the dissipation and the performed work is equal to the total chemical free energy of consumed ATP. However, measuring entropy production through a calorimetric approach is challenging in small biological systems; because the dissipated energy in the system is too small to make changes in the surrounding medium, such as changes in the water temperature. Therefore, fluctuating trajectories of experimentally accessible degrees of freedom are the only we can have in practice.

One of the key results in stochastic thermodynamics suggests that entropy production can be measured from trajectory data without information about the underlying mechanism of the systems. By estimating the probability that a time-reversed trajectory, which is like playing a recorded video from an experiment backward, will be observed in the ensemble of trajectories can infer the entropy production. Various estimation methods based on it have been developed, but the required trajectory data needs to first be divided into discrete microstates to obtain the empirical probability distributions. This first step suffers from the curse of dimensionality as the degrees of freedom increase. Despite the recent advanced measurement techniques, this problem forced us to coarse-grained the data into a few degrees of freedom

(typically one or two) even the recorded data has high degrees of freedom.

Chapter 2 presents the neural estimator for entropy production (NEEP), a novel type of estimator that evaluates entropy production from the trajectories of relevant variables by deep learning. While the previous methods need the empirical probability distribution for estimating the entropy production, NEEP does not require such information. We will demonstrate that NEEP can efficiently estimate entropy production from data even in high-dimensional space. In addition, we will verify that our approach is even applicable to time-series data with unobservable variables, which is the usual situation in practice.

Chapter 3 proposes a method for estimating the entropy production in a system with odd-parity variables of which the sign changes under time-reversal operation. Most genuine dynamics include velocity as a state variable; but to our best knowledge, previously developed methods for estimating the entropy production are not applicable to systems with odd-parity state variables. In addition to velocity, odd-parity variables are broadly found in many active systems such as flocks of birds, run-and-tumble bacteria, and other biological systems. The inability to handle odd-parity variables originates from the additional contributions to the entropy production, like asymmetry in the steady-state distribution and asymmetry in the waiting-time distribution. In this chapter, we will introduce multiple neural estimators for estimating the entropy production, and it will be demonstrated on nonequilibrium systems with odd-parity variables.

Chapter 2 and Chapter 3 introduced methods for measuring the generated heat in a nonequilibrium process; but, in order to fully understand the nonequilibrium process in nature, we also need to figure out how the work is performed by what mechanism. In Chapter 4, we employ the deep reinforcement learning framework to find what the optimal mechanism would be for a non-equilibrium system to perform work optimally. To be more precise, we focused on the transport phenomena in small biological systems such as ion pumping, molecular transportation, and motor proteins. A collective flashing ratchet, a nonequilibrium model for the transport phenomena, induces a net current of 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. In this chapter, we will use deep reinforcement learning to find optimal policies in the collective flashing ratchet. In addition, we will demonstrate that this method works well for a typical experimental situation, i.e. a time-delayed feedback situation where the on-off switching of the potential is delayed.

Finally, we conclude this thesis with a concluding remark in Chapter 5.

Chapter 2. Deep learning for entropy production estimation

* This chapter is reproduced based on the following published paper with its figures, tables, and contexts - D.-K. Kim, Y. Bae, S. Lee, and H. Jeong, *Phys. Rev. Lett.* **125**, 140604 (2020).

2.1 Introduction

Nonequilibrium states are ubiquitously observed from colloidal particles to biological systems [4, 5, 6, 7, 8, 9]. Injection of energy, lack of relaxation time, or broken detailed balance are ordinary sources of nonequilibrium, and in general, such systems are in contact with a heat bath such as a fluid. Thus, to describe the behavior of a nonequilibrium system, it is necessary to investigate the energetics of the system; however, experimentally, heat flow is difficult to measure directly [10, 11, 12, 13]. In this case, measuring the entropy production (EP) can be one remedy to estimate heat flow in a nonequilibrium system [1, 3, 13].

Many techniques have been developed to accurately measure EP, such as approaches calculating probability currents and density [12, 14]. These methods require detailed information from a governing equation though, so to address this issue, a few methods to estimate the EP rate without such detailed information have been proposed, including the plug-in method [15, 16, 17], the compression-based estimator [18, 16, 17, 19, 20], and the thermodynamic uncertainty relation (TUR) based estimator [21, 22, 14, 23, 24]. The plug-in and compression-based methods estimate the EP rate through the Kullback–Leibler divergence, but they are only applicable for discrete state variables. And while the TUR-based approach has recently been adopted in frameworks for the exact estimation of EP rates and distributions in short time limits [22, 23, 24], estimating stochastic EP remains an unsolved issue for continuous state variables.

Various fields in physics have been employing machine learning (ML) to solve a wide range of nontrivial problems such as identifying relevant variables [25, 26, 27], identifying phase transitions [28, 29, 30, 31, 32, 33], quantum many-body problems [34, 35, 36, 37, 38, 39, 40, 41], and others [42]. Likewise, ML has also been applied to EP rate estimation [43, 24] as well as classification of the direction of time's arrow [44]. Relatedly, in the ML community, a recent work by Rahaman *et al.* [45] proposed a neural network to measure an entropy-like quantity by unsupervised learning; however, the quantity was not physically well defined, i.e. it had no scale. To the best of our knowledge, estimating EP using neural networks has yet to be explored.

2.2 Neural estimator for entropy production

In this work, we propose the neural estimator for entropy production (NEEP), which can estimate stochastic EP from the time-series data of relevant variables without detailed information on the dynamics of the system. For Markov chain trajectory $s_1, s_2, ..., s_L$, we build a function h_θ that takes two states, s_t and s_{t+1}, where θ denotes the trainable neural network parameters. As shown in Fig. 2.1(a), the output of NEEP is defined as

$$\Delta S_\theta(s_t, s_{t+1}) \equiv h_\theta(s_t, s_{t+1}) - h_\theta(s_{t+1}, s_t). \tag{2.1}$$

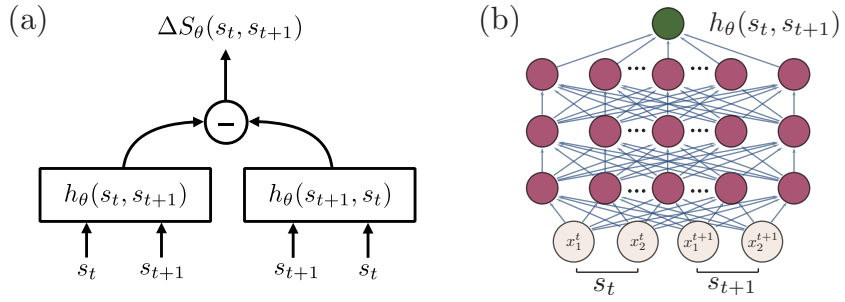

Figure 2.1: (a) Architecture of the neural estimator for entropy production (NEEP). (b) Illustration of a multilayer perceptron (MLP) with three hidden layers for an $N = 2$ bead-spring model where $s_t = (x_1^t, x_2^t)$.

Here, $\Delta S_\theta(s_t, s_{t+1})$ satisfies the antisymmetric relation $\Delta S_\theta(s_t, s_{t+1}) = -\Delta S_\theta(s_{t+1}, s_t)$. We define the objective function to be maximized as

$$J(\theta) = \mathbb{E}_t \mathbb{E}_{s_t \to s_{t+1}}[\Delta S_\theta(s_t, s_{t+1}) - e^{-\Delta S_\theta(s_t, s_{t+1})}], \tag{2.2}$$

where $\mathbb{E}_t$ denotes the expectation over t, which is uniformly sampled from $\{1, ..., L-1\}$, and $\mathbb{E}_{s_t \to s_{t+1}}$ is the expectation over transition $s_t \to s_{t+1}$. If detailed balance is satisfied, then the transition $s \to s'$ and its reverse transition $s' \to s$ equally appear in the ensemble of the trajectories. In this case, the optimized ΔS_θ is zero for all possible transitions, but if detailed balance is broken, then ΔS_θ becomes larger due to more irreversible transitions. In steady state, $J(\theta)$ can be written as

$$J[h] = \sum_{i,j} p_i T_{ij} \left[(h_{ij} - h_{ji}) - e^{-(h_{ij} - h_{ji})} \right] \tag{2.3}$$

where we set $h_{ij} \equiv h(s_i, s_j)$, $p_i \equiv p(s_i)$ is the steady-state probability density, and $T_{ji} \equiv p(s_i, t+1 | s_j, t)$ is a propagator. Because the neural networks tune output $h_{\alpha\beta} \equiv h(s_\alpha, s_\beta)$ by optimizing θ, the maximum condition for Eq. (2.3) becomes

$$\begin{aligned} 0 =& \partial_{h_{\alpha\beta}} J[h] \\ =& \sum_{i,j} \left[p_i T_{ij} (1 + e^{-(h_{ij} - h_{ji})})(\delta_{i\alpha}\delta_{j\beta} - \delta_{i\beta}\delta_{j\alpha}) \right] \\ =& p_\alpha T_{\alpha\beta}(1 + e^{-(h_{\alpha\beta} - h_{\beta\alpha})}) - p_\beta T_{\beta\alpha}(1 + e^{-(h_{\beta\alpha} - h_{\alpha\beta})}). \end{aligned} \tag{2.4}$$

Then the solution for the optimization problem is

$$h_{\alpha\beta} - h_{\beta\alpha} = \ln \frac{p_\alpha T_{\alpha\beta}}{p_\beta T_{\beta\alpha}}, \tag{2.5}$$

which is the definition of stochastic entropy production [1] when $T_{ji} = \widetilde{T}_{ij}$. Here, $\widetilde{T}_{ij}$ is the time-reversal propagator of T_{ij}. This proof supports the ability of our NEEP to learn appropriate EP. We maximize Eq. (2.2) via the stochastic gradient ascent method that is widely used in deep learning literature [46, 47]. See Section 2.5.1 for detailed descriptions on the training setup and algorithm.

To validate our approach, we estimate the EP of two widely studied nonequilibrium systems: the bead-spring model for continuous state variables [8, 48, 14, 43] and the discrete flashing ratchet model for discrete state variables [49, 16, 17]. To attempt more challenging problems, we additionally apply NEEP to high-dimensional continuous models and a hidden Markov model.

2.3 Results

2.3.1 Continuous state variable case: bead-spring model

In the bead-spring model, N beads are coupled to the nearest beads or boundary walls by springs and contacted with thermal heat baths at different temperatures, as described in Fig. 2.1(a). For displacements $x_1, x_2, ..., x_N$, the dynamics of N-beads is governed by an overdamped Langevin equation

$$\dot{x}_i(\tau) = A_{ij}x_j(\tau) + \sqrt{2k_{\rm B}T_i/\gamma}\xi_i(\tau), \tag{2.6}$$

where $A_{ij} = (-2\delta_{i,j} + \delta_{i,j+1} + \delta_{i+1,j})k/\gamma$. Here, k is a spring constant, γ is the Stokes friction coefficient, and the temperature T_i of each heat bath linearly varies from $T_{\rm h}$ to $T_{\rm c}$. ξ_i is an independent Gaussian white noise satisfying $\mathbb{E}[\xi_i(\tau)\xi_j(\tau')] = \delta_{ij}\delta(\tau - \tau')$ where $\mathbb{E}$ denotes the ensemble average. We set all the parameters to be dimensionless and $k_B = k = \gamma = 1$. The linearly varying temperature induces a thermodynamic force that drives the system to a nonequilibrium state.

To attempt EP estimation in a system with continuous variables, we firstly consider $N = 2$ and $N = 5$ bead-spring models. Here, $\dot{\sigma}$ is the analytical value of the ensemble-averaged EP rate (see Section 2.5.3); Fig. 2.2(a) plots $\dot{\sigma}$ for $N = 2$ (5) with a blue (orange) solid line. As illustrated in Fig. 2.1(b), we employ a 3-hidden-layer multilayer perceptron (MLP) for h_θ. See Section 2.5.2 for the configuration and robustness of the architecture. For training and test sets, we numerically sampled 10^3 positional trajectories in steady state for each model. Each trajectory was sampled with time step $\Delta\tau = 10^{-2}$ [1], and the total number of steps L is 10^4. We present the training results at $T_{\rm c}/T_{\rm h} = 0.1$ in Fig. 2.2(b–e). Note that all reported results in Fig. 2.2 are from the test set. We also demonstrate the estimation ability of NEEP with various L (see Fig. 2.8 in Section 2.5.4).

For the $N = 2$ case, as shown in Fig. 2.2(b), it is observed that our estimator provides accurate values not only for the ensemble average but also for a single trajectory over τ. Here, $S(\tau) \equiv \sum_{i=0}^{\tau/\Delta\tau} \Delta S(s_i, s_{i+1})$ and $\sigma(\tau) \equiv \mathbb{E}[S(\tau)]$ where ΔS is the analytic stochastic EP per $\Delta\tau$. Figure 2.2(c) shows that the local EP rate over the displacement space (x_1, x_2) calculated by NEEP (top panel) is the same as the analytical solution (bottom panel). The local EP rate from NEEP at (x_1, x_2) is measured by averaging the EP rate produced when a particle passes through the point (x_1, x_2).

To check the training process, we plot the estimated values of $\dot{\sigma}_\theta$ over training iteration in Fig. 2.2(d). The dashed red line indicates $\dot{\sigma}$. Insets in Fig. 2.2(d) are scatter plots between ΔS_θ and ΔS in a randomly sampled single trajectory. As can be seen in the left inset, there is no correlation between ΔS_θ and ΔS before training. But after training (right inset), ΔS_θ is well-fitted to ΔS (coefficient of determination $R^2 = 0.9931$).

We apply the same process to the $N = 5$ bead-spring model, where estimating ΔS and $\dot{\sigma}$ using the thermodynamic force is difficult due to the curse of dimensionality [14]. The result shows that ΔS_θ is again well-fitted to ΔS with $R^2 = 0.9660$ (see Fig. 2.2(e)). We also train our estimator at $T_{\rm c}$ in the range of 1–10 with $T_{\rm h} = 10$, as indicated in Fig. 2.2(a), and verify that NEEP provides the exact EP rate with small errors. Notably, these results are from the test set, implying that NEEP can be generalized to estimate EP even for unseen data.

Estimating EP in high-dimensional Langevin systems has not been explored because of the curse of dimensionality [23]. While a recent work [24] has made estimations of EP rates up to $N = 15$ using TUR, here, we apply NEEP to bead-spring models with $N = 8$, 16, 32, 64, and 128. For each N, we set $T_{\rm h} = 10$

[1] The time interval between the current and next states can affect the estimation of NEEP. For large enough time intervals, the correlation between the two states weakens and NEEP may provide imprecise estimations.

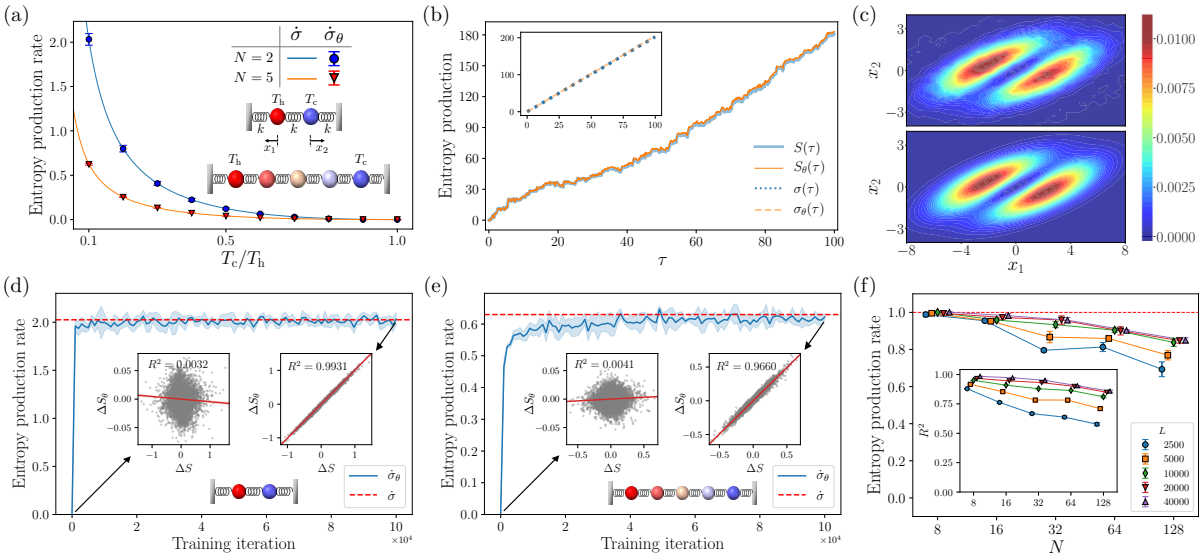

Figure 2.2: (a) Entropy production (EP) rate as a function of T_c/T_h for models with two and five beads. The solid lines (symbols) indicate the analytical EP rate $\dot{\sigma}$ (estimated EP rate $\dot{\sigma}_\theta$). (b) Cumulative EP over time τ along a single trajectory, which is randomly sampled from the test set. The inset shows the ensemble-averaged EP. (c) Local EP rate as a function of x_1 and x_2. The top and bottom panels show the NEEP and the analytical results, respectively. (d,e) $\dot{\sigma}_\theta$ with respect to training iteration for (d) two beads and (e) five beads. The left (right) inset corresponds to results before (after) training, showing scatter plots between ΔS and ΔS_θ with a fitted linear regression line (solid red line). The results in (b–e) are performed at $T_c/T_h = 0.1$. (f) Results of NEEP for high-dimensional bead-spring models. $\dot{\sigma}_\theta$ as a function of N for each number of steps (L) are plotted with five different markers, as indicated in the legend. The R^2 values of the linear regression between ΔS and ΔS_θ are shown in the inset. The red dashed line denotes $\dot{\sigma}$. Error bars and shaded areas represent the standard deviation of estimations from five independently trained estimators.

and T_c to a value where $\dot{\sigma} = 1$ (see Section 2.5.5). By increasing the training data points ($10^3 L$), we can see that $\dot{\sigma}_\theta$ for each N approaches 1 in Fig. 2.2(f). Although EP rate estimation errors of over 10% are seen for $N = 64$ and 128, the R^2 values support that NEEP was able to learn the stochastic EP with appreciable correlations (see the inset in Fig. 2.2(f)). Note that, with an increasing number of beads, the architecture of NEEP does not change except for the number of input nodes ($2N$), which means that our neural estimator's computation time and the number of parameters are linearly proportional to N. Based on these points, we show that NEEP can efficiently mitigate the curse of dimensionality through a neural network.

2.3.2 Discrete state variable case: discrete flashing ratchet

Next, we demonstrate our method on the discrete flashing ratchet model [49], which consists of a particle moving in a one-dimensional periodic lattice. The particle is in contact with a heat bath at temperature T and drifts in a periodic asymmetric sawtooth potential (see Fig. 2.3(b)). For brevity, we set $k_B = T = 1$. In this model, the particle state has two variables, x and η, where $x \in \{0, 1, 2\}$ is the position and $\eta \in \{\mathrm{ON}, \mathrm{OFF}\}$ is the on/off potential; the state is indicated as $i \equiv (i, \mathrm{ON})$ and $i' \equiv (i, \mathrm{OFF})$.

Transition rates between each state $s \in \{0, 1, 2, 0', 1', 2'\}$ are defined as $k_{ij} = e^{(V_j - V_i)/2}$ and $k_{i'j'} = 1$ for $i \neq j$ where V_i is the potential at i that switches on and off at rate $r = 1$, i.e. $k_{ii'} = k_{i'i} = r$. As in a previous work [16], we generate a series of states and remove the information of the times when transitions occur; in this case, the analytic EP per step is given as $\dot{\sigma} = \sum_{\alpha,\beta} p(\alpha, \beta)(V_\alpha - V_\beta)$.

We construct the NEEP as shown in Fig. 2.3(a) using an embedding layer that transforms a discrete state into a trainable continuous vector called an embedding vector. After the transformation, we feed the two embedding vectors of states s_t and s_{t+1} to the MLP (see Section 2.5.2 for the architecture configuration). From a set of different potential values, we sampled two single trajectories with $L = 10^6$ steps for each potential V ($0 \leq V \leq 15$); one trajectory is used for training and the other for testing. For the training data, we build five NEEPs, randomly initialized with five different random seeds for each potential. Figure 2.3(c) shows that $\dot{\sigma}$ is within the error bar of the NEEP estimations of EP per step $\dot{\sigma}_\theta$ where $V \leq 8$. For V in the range 8–14, the overfitting [47] problem occurs due to a lack of transitions from low to high potential, which leads to an underestimation of $\dot{\sigma}_\theta$ (Fig. 2.3(c)). See Section 2.5.4 for a more detailed discussion on how we address the overfitting issue. For $14 \leq V$, the probability to detect the $0 \to 2$ transition is below 0.5 in our simulation with $L = 10^6$. In this case, $\dot{\sigma}_\theta$ diverges because of no observation of the $0 \to 2$ transition (see Fig. 2.9 in Section 2.5.5).

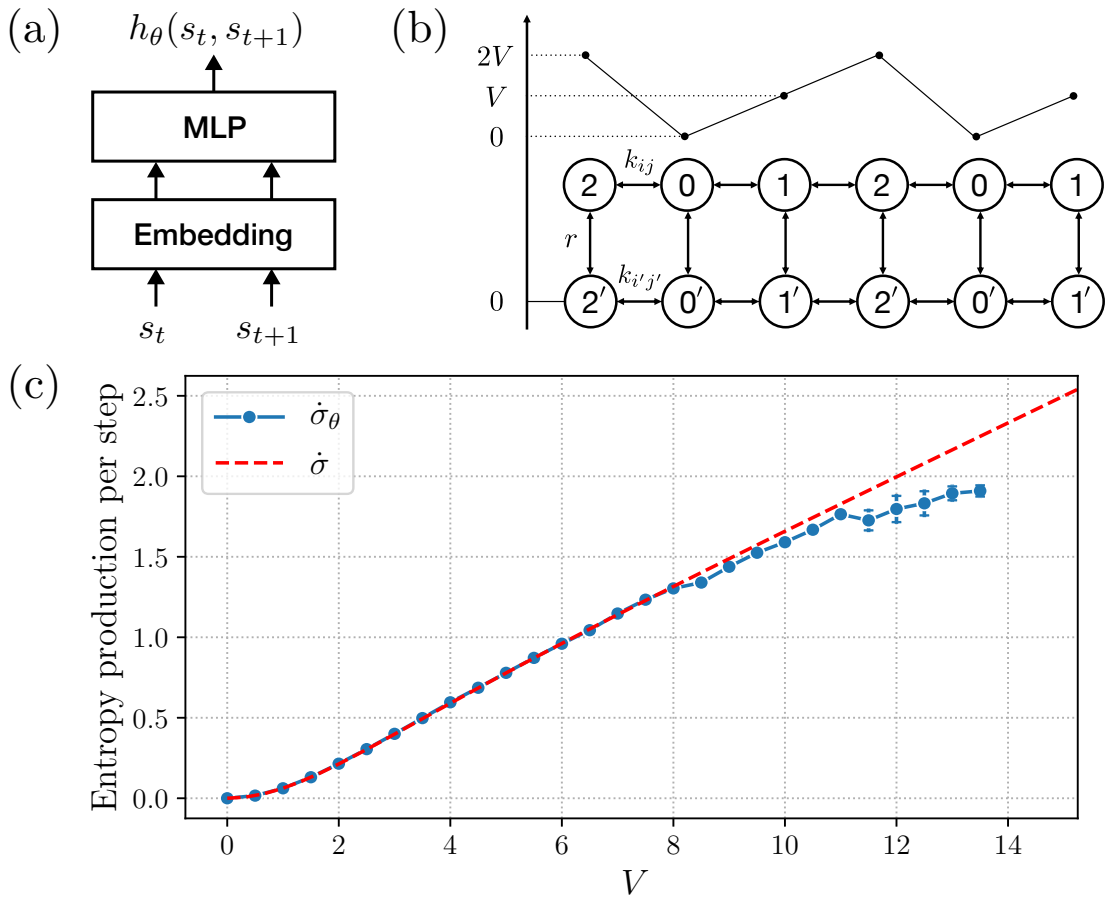

Figure 2.3: (a) NEEP architecture for discrete state Markov chains. (b) Schematic of a discrete flashing ratchet model. (c) Entropy production per step as a function of potential V. Error bars represent the standard deviation of $\dot{\sigma}_\theta$ from five independently trained estimators.

2.3.3 Markov systems with unobservable states

So far, Markovian systems with completely observable states have been tested; however, full state information cannot often be accessed, with only some coarse-grained variables typically available. In such cases, the EP of a coarse-grained trajectory, called coarse-grained EP, is measurable [50, 51, 52, 53]. To test for coarse-grained EP estimation, we assume that the on/off information η is now inaccessible [16, 17]. To address this problem, we build h_θ with a recurrent neural network (RNN), a popular network to consider memory effects in time-series data. We employ a gated recurrent unit (GRU) [54] for the RNN. As shown in Fig. 2.4(a), the RNN version of NEEP (RNEEP) takes input as a series of states with a sequence length of n, and the outputs of the GRU are averaged over the sequence and then fed to a single layer feed-forward neural network, which is the last layer. Now, the RNEEP output is defined as

$$\Delta S_\theta(\boldsymbol{x}_t^n) \equiv h_\theta(\boldsymbol{x}_t^n) - h_\theta(\widetilde{\boldsymbol{x}}_t^n), \tag{2.7}$$

and the objective function is defined as

$$J(\theta) = \mathbb{E}_t \mathbb{E}_{(\boldsymbol{x}_t^n, \boldsymbol{\eta}_t^n)}[\Delta S_\theta(\boldsymbol{x}_t^n) - e^{-\Delta S_\theta(\boldsymbol{x}_t^n)}], \tag{2.8}$$

where $\boldsymbol{x}_t^n = (x_t, x_{t+1}, ..., x_{t+n-1})$ and $\boldsymbol{\eta}_t^n = (\eta_t, \eta_{t+1}, ..., \eta_{t+n-1})$. Here, $\widetilde{\boldsymbol{x}}_t^n$ is the time-reversed trajectory of $\boldsymbol{x}_t^n$. In steady state, the solution for this optimization problem is the stochastic coarse-grained EP along the trajectory $\boldsymbol{x}^n$ (see Section 2.5.6 for the proof):

$$\Delta S_\theta(\boldsymbol{x}^n) = \ln \frac{\sum_{\boldsymbol{\eta}^n} p(\boldsymbol{x}^n, \boldsymbol{\eta}^n)}{\sum_{\widetilde{\boldsymbol{\eta}}^n} p(\widetilde{\boldsymbol{x}}^n, \widetilde{\boldsymbol{\eta}}^n)} = \ln \frac{p(\boldsymbol{x}^n)}{p(\widetilde{\boldsymbol{x}}^n)}. \tag{2.9}$$

Here, the ensemble-averaged coarse-grained EP of trajectory $\boldsymbol{x}^n$ per step is denoted as

$$\dot{\Sigma}^n \equiv \mathbb{E}\left[\frac{1}{n-1} \ln \frac{p(\boldsymbol{x}^n)}{p(\widetilde{\boldsymbol{x}}^n)}\right]. \tag{2.10}$$

In general, $\dot{\Sigma}^n$ provides a lower bound on the actual EP per step $\dot{\sigma}$ [16, 17].

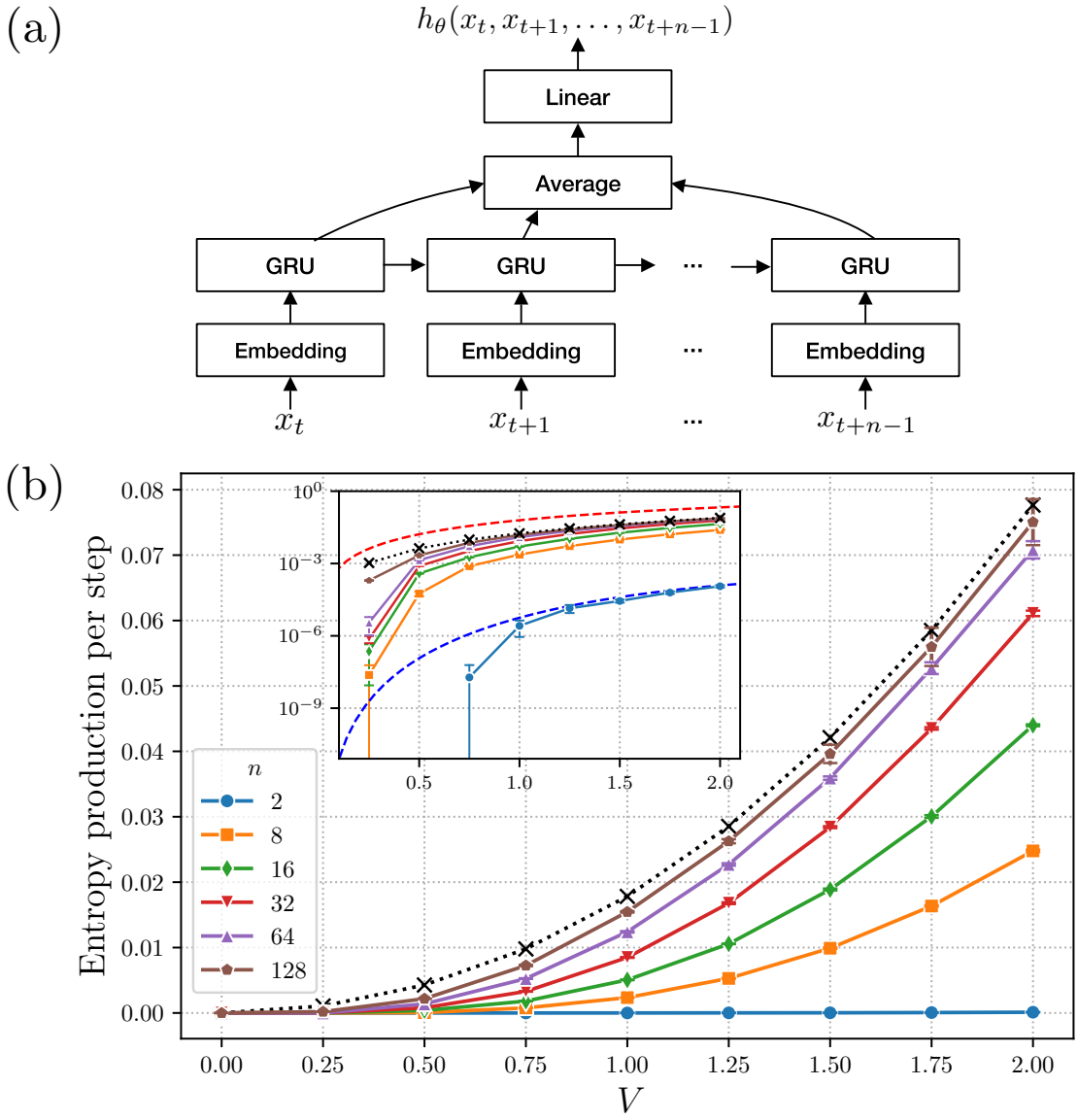

Figure 2.4: (a) An RNN version of NEEP (RNEEP) for a hidden Markov model. (b) Results of RNEEP for a partial information problem. The estimations $\dot{\Sigma}^n_\theta$ as a function of potential V for each sequence length n are plotted with six different markers as shown in the legend. The black x's are the semianalytical values of $\dot{\Sigma}^\infty$. The inset shows a plot of the y-axis in log scale. The red (blue) dashed line denotes the analytic value of $\dot{\sigma}$ ($\dot{\Sigma}^2$). See Fig. 2.10 in Section 2.5.5 for a comparison with $\dot{\sigma}$ in linear scale. Error bars represent the standard deviation of estimations from five independently trained estimators.

For $0 \leq V \leq 2$, we train the RNEEP with six different sequence lengths, $n = 2, 8, 16, 32, 64$, and 128, for maximizing Eq. (2.8) using the position trajectory with $L = 5 \times 10^7$. As can be seen in Fig. 2.4(b), with increasing sequence length n, the estimation of RNEEP ($\dot{\Sigma}^n_\theta$) approaches the semianalytical value of the coarse-grained EP per step for $n \to \infty$ ($\dot{\Sigma}^\infty$) [17]. We can verify that $\dot{\Sigma}^2_\theta$ is well-fitted to the analytic value of $\dot{\Sigma}^2$, but it remains difficult to estimate for $V \leq 1$ (see the inset in Fig. 2.4(b)), because the number of transitions between any two positions, e.g. $x \to y$ or $y \to x$, appears almost equally in the trajectory. While directly estimating Eq. (2.9) by counting the frequency of $\boldsymbol{x}^n$ is not possible for $n \geq 16$ due to the curse of dimensionality, the RNEEP can resolve this issue and enable us to estimate

the coarse-grained EP up to $n = 128$ (see Fig. 2.11 and Fig. 2.12 in Section 2.5.5).

2.4 Discussion

In conclusion, we have developed a novel method, named NEEP, for estimating entropy production via neural networks. Our method does not require detailed information on the system dynamics, but only trajectories of relevant variables. We proved that our method produces appropriate EP when the given system has a Markovian property and is in nonequilibrium steady state. We have demonstrated that NEEP precisely estimates the true EP rate of two nonequilibrium systems, namely bead-spring and discrete flashing ratchet models. We estimated the stochastic EP along a single trajectory and the spatial pattern of the EP rate. In a continuous high-dimensional case, we verified that NEEP efficiently addresses the curse of dimensionality. Moreover, as a system without full description, we considered a hidden Markov model, with results showing that our estimator can provide coarse-grained EP.

In previous approaches [13], estimation of the probability distribution and the probability current was essential to quantify how far the system is out of equilibrium. As NEEP does not require such estimation or detailed information of the system, we expect our estimator to be applicable to various fields such as active matter, biological systems, information machines, electronic devices, and others. This approach will be particularly useful to investigate the stochastic energetics and spatiotemporal patterns of dissipated energy in various systems. We further expect our method to be applicable to the understanding of complex nonequilibrium systems, e.g. soft biological assemblies [48] or molecular motors with hidden internal states [55]. As a future work, modifying our NEEP method to estimate EP in more general nonequilibrium systems like time-dependent states will be intriguing.

The code for NEEP, implemented in `PyTorch` [56], is available in the GitHub repository[2].

[2] https://github.com/kdkyum/neep

2.5 Supporting Information

2.5.1 Training setup and algorithm

We employ ReLU [57] as the activation function for the neural estimator for entropy production (NEEP). We train the estimators with the Adam [58] optimizer using the following hyper-parameters: learning rate 10^{-4}, weight decay 5×10^{-5}, and batch size 4096. We use these hyper-parameters for all processes unless noted. See Algorithm 2.1 for our training procedure. All runs were conducted on a single NVIDIA TITAN V GPU.

Algorithm 2.1 Training the NEEP

Require: Estimator ΔS_θ, optimizer, training set $\mathcal{D}_{\rm train} = \{(s_1^i, s_2^i, ..., s_L^i)\}_{i=1,...,M}$ where L is the trajectory length and M is the number of trajectories

1: **loop**

2: Compute

$$\hat{J}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{(i,t)\in\mathcal{B}} \left[\Delta S_\theta(s_t^i, s_{t+1}^i) - e^{-\Delta S_\theta(s_t^i, s_{t+1}^i)}\right] \tag{2.11}$$

where $\mathcal{B}$ is a randomly sampled subset of $\{1, ..., M\} \times \{1, ..., L-1\}$ where $\times$ is a Cartesian product. The total number of the sampled data points $|\mathcal{B}|$ is equal to the batch size.

3: Compute gradients $\nabla_\theta \hat{J}(\theta)$.

4: Update parameters θ with the optimizer.

5: **end loop**

2.5.2 Architecture robustness and Evaluation metric

The purpose of training is to get the maximum value of J in general, not only for training set $\mathcal{D}_{\rm train}$, and therefore we evaluate $J_{\rm test}(\theta)$ with test set $\mathcal{D}_{\rm test}$ after training where

$$J_{\rm test}(\theta) \equiv \frac{1}{M(L-1)} \sum_{(s_t^i, s_{t+1}^i)\in\mathcal{D}_{\rm test}} \left[\Delta S_\theta(s_t^i, s_{t+1}^i) - e^{-\Delta S_\theta(s_t^i, s_{t+1}^i)}\right]. \tag{2.12}$$

For the $N = 5$ bead-spring model, we train NEEP built with various MLP configurations over 10^5 training iterations: numbers of hidden layers of 1, 2, 3, and 4 and numbers of hidden units (H) of 32, 64, 128, 256, 512, and 1024. See Table 2.1 for the architecture details. After training, we evaluate the EP rate estimation $\dot{\sigma}_\theta$ and $J_{\rm test}(\theta)$ (Eq. (2.12)) for each MLP configuration. Figure 2.5(a) shows that EP rate estimations are robust to different model configurations and within a 10% error analytic EP rate (red dashed line). The inset in Fig. 2.5(a) shows the $J_{\rm test}(\theta)$ for each MLP configuration. As can be seen, the MLP configurations with $J_{\rm test}(\theta) < -0.9940$ either underestimate (single-layer MLPs) or overestimate (4-hidden-layer MLP with 1024 hidden units) as compared to other configurations. This supports that $J_{\rm test}(\theta)$ can evaluate whether the NEEP has learned EP well. Based on these observations, we choose the 3-hidden-layer MLP architecture with $H = 256$, which shows the highest $J_{\rm test}(\theta)$ value, for h_θ in the paper.

For the discrete flashing ratchet model, we train NEEP built with various configurations (see Table 2.2) over 5×10^4 training iterations: numbers of hidden layers of 1, 2, 3, and 4 and dimensions of the embedding vector (H) of 4, 8, 16, 32, 64, and 128. As with the previous results for the bead-spring model, the estimations of EP per step are robust to the configurations except for overparameterized

models (see Fig. 2.5(b)). As shown in Fig 2.5(b) inset, we choose the single-hidden-layer MLP with $H = 128$ for h_θ in the paper.

The large variances in $J_{\text{test}}(\theta)$ and $\dot{\sigma}_\theta$ for overparameterized models (see the inset in Fig. 2.5(b)) are addressed in Section 2.5.4.

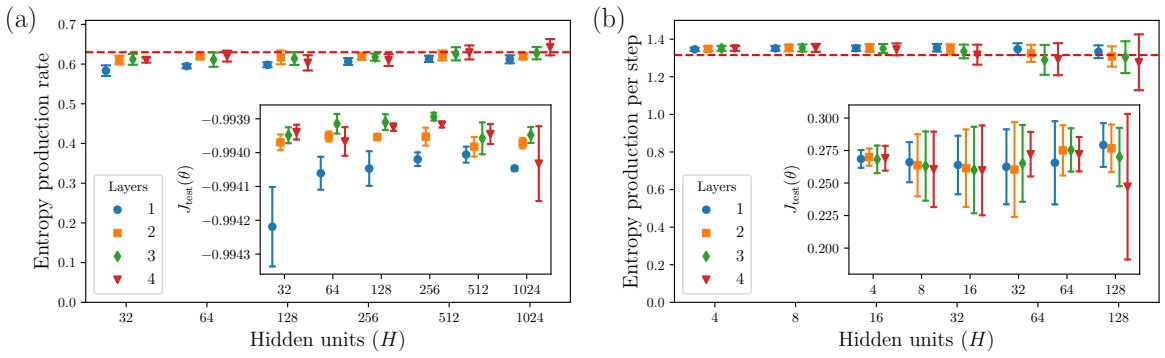

Figure 2.5: Estimations on EP rate (per step) $\dot{\sigma}_\theta$ with multiple architecture configurations. Training and test set data are generated at (a) $T_c/T_h = 0.1$ with $M = 10^3$ and $L = 10^4$ for five-bead models and at (b) $V = 8$ with $M = 1$ and $L = 10^6$ for discrete flashing ratchet models. The insets shows the $J_{\text{test}}(\theta)$ values after training. The red dashed line denotes the analytic EP rate (EP per step) $\dot{\sigma}$. Error bars represent the standard deviation of the evaluated values from five independently trained estimators.

3-hidden-layer MLP NEEP		
Layer	Output dim	Activation function
Input (s_t, s_{t+1})	$2N$	
Fully-connected	H	ReLU
Fully-connected	H	ReLU
Fully-connected	H	ReLU
Output layer	1	None

Table 2.1: Three-hidden-layer MLP configuration for N bead-spring model. H is the number of hidden units.

Embedding-MLP NEEP		
Layer	Output dim	Activation function
Input s	$\{0, 1, 2, 0', 1', 2'\}$	
Embedding	H	
Concatenate (s_t, s_{t+1})	$2H$	
Fully-connected	$2H$	ReLU
Output layer	1	None

Table 2.2: MLP with embedding layer for discrete flashing ratchet model. H is the size of the embedding dimension.

2.5.3 Analytic description of bead-spring model

The dynamics of the beads can be described by an overdamped Langevin equation given by

$$\dot{x}_i(\tau) = A_{ij}x_j(\tau) + \sqrt{2k_{\mathrm{B}}T_i/\gamma}\xi_i(\tau), \tag{2.13}$$

where $A_{ij} = -2k/\gamma\ \delta_{i,j} + k/\gamma\ (\delta_{i,j+1} + \delta_{i+1,j})$, $\boldsymbol{x} = (x_1(\tau), x_2(\tau), ..., x_N(\tau))$, $\boldsymbol{\xi} = (\xi_1(\tau), \xi_2(\tau), ..., \xi_N(\tau))$ and the temperature T_i of each heat bath linearly varies from T_{h} to T_{c}. These different temperatures induce a thermodynamic force which drives the system out of equilibrium. Here, γ is the Stokes friction coefficient, and $\boldsymbol{\xi}$ is an independent Gaussian white noise vector satisfying $\mathbb{E}[\xi_i(\tau)\xi_j(\tau')] = \delta_{ij}\delta(\tau - \tau')$. For brevity, we set $k_B = 1$. To calculate the EP of the bead-spring model, we have to consider the Fokker–Planck equation given by

$$\frac{\partial p(\boldsymbol{x}, \tau)}{\partial \tau} = -\nabla \cdot \boldsymbol{j}(\boldsymbol{x}, \tau), \tag{2.14}$$

where the probability current $\boldsymbol{j}(\boldsymbol{x}, \tau)$ is defined by

$$\boldsymbol{j}(\boldsymbol{x}, \tau) = \mathsf{A}\boldsymbol{x}p(\boldsymbol{x}, \tau) - \mathsf{D}\nabla p(\boldsymbol{x}, \tau). \tag{2.15}$$

Here, D is the diffusion matrix defined as $\mathrm{diag}\{\mathrm{T}_1/\gamma, \cdots, \mathrm{T}_\mathrm{N}/\gamma\}$. Since our system is an Ornstein–Uhlenbeck process, the steady-state probability density function is Gaussian as $p(\boldsymbol{x}) \propto \exp[-(1/2)\boldsymbol{x}^T\mathsf{C}^{-1}\boldsymbol{x}]$ with a covariance matrix C. Using the Lyapunov equation $\mathsf{AC} + \mathsf{CA}^{\mathsf{T}} = -2\mathsf{D}$, the probability density function and the probability current in steady state, $p(\boldsymbol{x})$ and $j_{ss}(\boldsymbol{x})$, can be obtained.

The EP rate along a trajectory is given by [2]

$$\dot{S}(\tau) = -\frac{\partial_\tau p(\boldsymbol{x}, \tau)}{p(\boldsymbol{x}, \tau)}\Big|_{\boldsymbol{x}(\tau)} + \frac{\boldsymbol{j}(\boldsymbol{x}, \tau)^T\mathsf{D}^{-1}}{p(\boldsymbol{x}, \tau)}\Big|_{\boldsymbol{x}(\tau)}\dot{\boldsymbol{x}}. \tag{2.16}$$

Because the first term on the right-hand side vanishes in steady state, the ensemble averaged entropy production rate is obtained by

$$\dot{\sigma} = \int d\boldsymbol{x}\ \frac{\boldsymbol{j}_{ss}(\boldsymbol{x})^T\mathsf{D}^{-1}}{p(\boldsymbol{x})}\boldsymbol{j}_{ss}(\boldsymbol{x}) = \mathsf{Tr}\left[\mathsf{D}^{-1}\mathsf{ACA}^{\mathsf{T}} - \mathsf{C}^{-1}\mathsf{D}\right], \tag{2.17}$$

where $\mathsf{Tr}[\cdot]$ is the trace operator. We used the Lyapunov equation to derive the last expression in Eq. (2.17).

For $N = 2$, the deterministic term $\mathsf{A} \equiv \frac{k}{\gamma}\left(\begin{smallmatrix} -2 & 1 \\ 1 & -2 \end{smallmatrix}\right)$ and the diffusion matrix $\mathsf{D} \equiv \frac{1}{\gamma}\left(\begin{smallmatrix} T_1 & 0 \\ 0 & T_2 \end{smallmatrix}\right)$. The covariance matrix can be derived by the Lyapunov equation as

$$\mathsf{C} = \frac{1}{12k}\begin{pmatrix} 7T_{\mathrm{h}} + T_{\mathrm{c}} & 2(T_{\mathrm{h}} + T_{\mathrm{c}}) \\ 2(T_{\mathrm{h}} + T_{\mathrm{c}}) & T_{\mathrm{h}} + 7T_{\mathrm{c}} \end{pmatrix}. \tag{2.18}$$

Using Eq. (2.18), $\dot{\sigma}$ for $N = 2$ is obtained as

$$\dot{\sigma} = \frac{k(T_{\mathrm{h}} - T_{\mathrm{c}})^2}{4\gamma T_{\mathrm{h}}T_{\mathrm{c}}}. \tag{2.19}$$

In the same way, $\dot{\sigma}$ for the $N = 5$ bead-spring model can be obtained as

$$\dot{\sigma} = \frac{k(T_{\mathrm{h}} - T_{\mathrm{c}})^2(111T_{\mathrm{h}}^2 + 430T_{\mathrm{h}}T_{\mathrm{c}} + 111T_{\mathrm{c}}^2)}{495\gamma T_{\mathrm{h}}T_{\mathrm{c}}(3T_{\mathrm{h}} + T_{\mathrm{c}})(T_{\mathrm{h}} + 3T_{\mathrm{c}})}. \tag{2.20}$$

For $N = 8$, 16, 32, 64, and 128 bead-spring models, we can also calcuate $\dot{\sigma}$ using Eq. (2.17).

We analytically calculated the local EP rate in Fig. 2.2(c) (bottom) using the integrand of Eq. (2.17). Because NEEP estimates along the time evolution of a particle, the local EP rate from NEEP $\dot{\sigma}_\theta(\boldsymbol{x})$ is measured by the following equation:

$$\dot{\sigma}_\theta(\boldsymbol{x}) = \frac{1}{L\Delta\tau}\sum_{t=0}^{L} \delta_{\boldsymbol{x},\boldsymbol{x}_t^m}\, \Delta S_\theta(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}), \tag{2.21}$$

where $\Delta\tau$ is the time step of a trajectory $\boldsymbol{x}_1, ..., \boldsymbol{x}_L$ and $\boldsymbol{x}_i^m = (\boldsymbol{x}_t + \boldsymbol{x}_{t+1})/2$.

2.5.4 Overfitting

Algorithm 2.2 Monitoring $J_{\mathrm{test}}(\theta)$ during training

Require: Estimator ΔS_θ, optimizer, training set $\mathcal{D}_{\mathrm{train}}$, test set $\mathcal{D}_{\mathrm{test}}$.

Require: Best model parameter $\theta^\star$, best value J_{best}.

1: Initialize $J_{\mathrm{best}} \leftarrow J_{\mathrm{test}}(\theta)$, $\theta^\star \leftarrow \theta$.
2: **loop**
3: Compute $\hat{J}(\theta)$ (Eq. (2.11)) from $\mathcal{D}_{\mathrm{train}}$.
4: Compute gradients $\nabla_\theta \hat{J}(\theta)$.
5: Update parameters θ with the optimizer.
6: Compute $J_{\mathrm{test}}(\theta)$ (Eq. (2.12)) from $\mathcal{D}_{\mathrm{test}}$.
7: **if** $J_{\mathrm{test}}(\theta) > J_{\mathrm{best}}$ **then**
8: $J_{\mathrm{best}} \leftarrow J_{\mathrm{test}}(\theta)$
9: $\theta^\star \leftarrow \theta$
10: **end if**
11: **end loop**

In this section, we show an example of the overfitting phenomenon that occurs when we train the NEEP for the $N = 5$ bead-spring trajectory with $M = 10^3$ and $L = 200$ at $T_{\mathrm{c}}/T_{\mathrm{h}} = 0.5$. Figure 2.6(a) shows that the gap between $J_{\mathrm{train}}(\theta)$ and $J_{\mathrm{test}}(\theta)$ keeps increasing. This phenomenon is called overfitting [47]. To address this problem, we monitor the $J_{\mathrm{test}}(\theta)$ value during the training process. See Algorithm 2.2 for details. Steps 6–10 in Algorithm 2.2 are computationally impractical, so we only run these steps every 100 iterations. As can be seen in Fig. 2.6(b), $J_{\mathrm{test}}(\theta)$ has a maximum value at a training iteration of 400, which is marked with a yellow star. The EP rate estimation at the maximum value $\dot{\sigma}_{\theta^\star}|_{\mathrm{test}}$ is represented with a red dotted line in Fig. 2.6(b). Figure 2.6(c) shows that EP estimation using training set $\dot{\sigma}_\theta|_{\mathrm{train}}$ (orange line) keeps increasing, and EP estimation using test set $\dot{\sigma}_\theta|_{\mathrm{test}}$ (blue line) does not converge to a certain value; however, the $\dot{\sigma}_{\theta^\star}|_{\mathrm{test}}$ (green line) remains unchanged after the training iteration of 400 and is close to the analytic EP rate $\dot{\sigma}$ (red dashed line).

In Section 2.5.2, there are large variances in $J_{\mathrm{test}}(\theta)$ and $\dot{\sigma}_\theta|_{\mathrm{test}}$ for overparameterized models. Here, we employ Algorithm 2.2 for the same runs as in Fig. 2.5. Figure 2.7 shows that the variances of $\dot{\sigma}_{\theta^\star}|_{\mathrm{test}}$ and $J_{\mathrm{test}}(\theta^\star)$ are significantly smaller than before (see Fig. 2.5). This result supports that Algorithm 2.2 can ensure robustness even for overparameterized neural network architectures.

Next, we train NEEP for various L with two- and five-bead models at $T_{\mathrm{c}}/T_{\mathrm{h}} = 0.1$, 0.5, and 1. Figure 2.8 shows that $\dot{\sigma}_{\theta^\star}|_{\mathrm{test}}$ can estimate $\dot{\sigma}$ with small error even when the number of training data points is small, while estimation of the five-bead model at $T_{\mathrm{c}}/T_{\mathrm{h}} = 0.1$ approaches $\dot{\sigma}$ with increasing L.

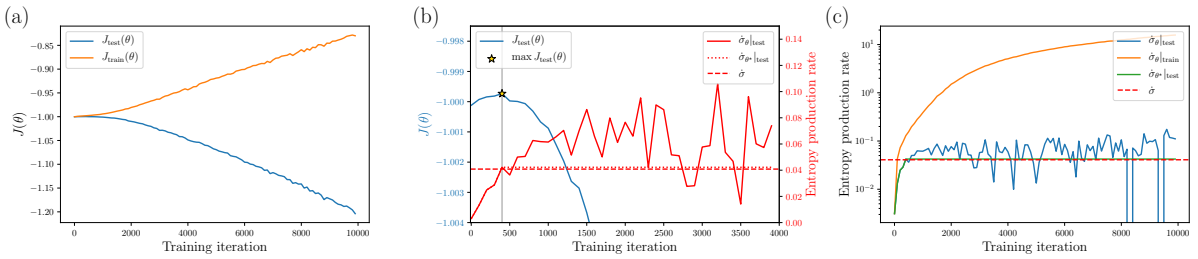

Figure 2.6: (a) $J_{\mathrm{test}}(\theta)$ (blue) and $J_{\mathrm{train}}(\theta)$ (orange) with respect to training iteration. (b) $J_{\mathrm{test}}(\theta)$ and $\dot{\sigma}_\theta|_{\mathrm{test}}$. The yellow star shows the maximum value of $J_{\mathrm{test}}(\theta)$ during the whole training process. The red solid line denotes the EP rate estimation $\dot{\sigma}_\theta|_{\mathrm{test}}$. (c) The EP rate estimation with respect to training iteration. The red dashed line denotes the analytic EP rate $\dot{\sigma}$.

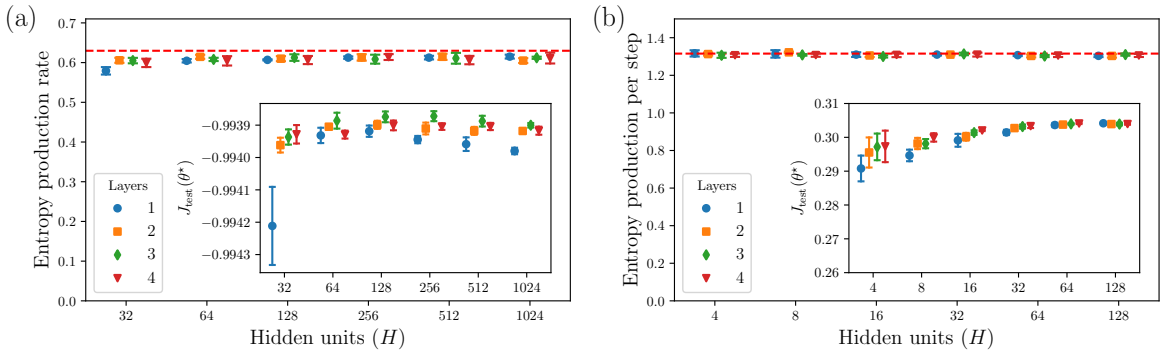

Figure 2.7: Estimations on EP rate (per step) $\dot{\sigma}_{\theta^\star}|_{\mathrm{test}}$ with multiple architecture configurations. Training and test set data are generated at (a) $T_{\mathrm{c}}/T_{\mathrm{h}} = 0.1$ with $M = 10^3$ and $L = 10^4$ for five-bead models and at (b) $V = 8$ with $M = 1$ and $L = 10^6$ for discrete flashing ratchet models. The insets show the $J_{\mathrm{test}}(\theta^\star)$ values. The red dashed lines denote the analytic EP rate (EP per step) $\dot{\sigma}$. Error bars represent the standard deviation of the evaluated values from five independently trained estimators.

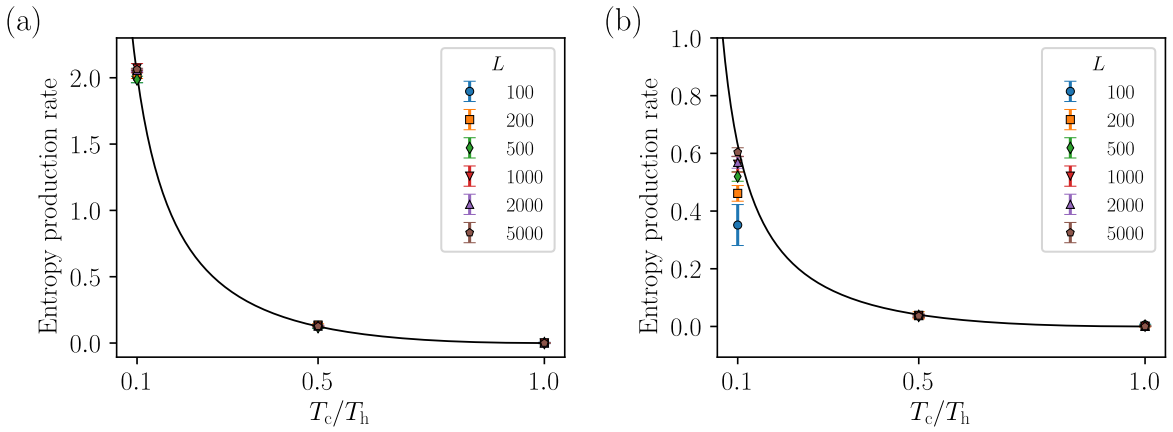

Figure 2.8: Test set prediction of NEEP with $L = 100$, 200, 500, 1000, 2000, and 5000 for the (a) two-bead and (b) five-bead models. The solid black line denotes the analytic EP rate. Error bars represent the standard deviation of estimations from five independently trained estimators.

2.5.5 Training details

Bead-spring model

For $N = 2$ and $N = 5$ bead-spring models, we report the EP rate estimation and stochastic EP using the parameter θ at the last training iteration in Fig. 2.2(a–e). The results of EP estimation with $\theta^\star$ are almost the same as the results with θ because there is no overfitting issue. The number of training iterations is 10^5, and we evaluate $J_{\rm test}$ every 10^3 iterations.

For the high-dimensional bead-spring model, we report $\dot{\sigma}_{\theta^\star}|_{\rm test}$ in Fig. 2.2(f). Due to the variance of each bead's position being inhomogeneous, we normalize each position in a data preprocessing step, e.g. $\bar{x}_i = (x_i - \mathrm{mean}(x_i))/\mathrm{std}(x_i)$, where the mean and standard deviation are estimated from $\mathcal{D}_{\rm train}$. The number of training iterations is 10^6, the weight decay is 10^{-5}, and we evaluate $J_{\rm test}$ every 10^4 iterations. At $T_{\rm h} = 10$, the respective $T_{\rm c}$ values for $N = 8$, 16, 32, 64, and 128 are 0.416997, 0.20768, 0.10358, 0.05171, and 0.02583, where the EP rate is one.

Discrete flashing ratchet

For the discrete flashing ratchet model with full information, we report $\dot{\sigma}_{\theta^\star}|_{\rm test}$ in Fig. 2.3(c). The number of training iterations is 5×10^4, and we evaluate $J_{\rm test}$ every 100 iterations. Note that these runs use a single trajectory, i.e. $M = 1$ in Algorithm 2.1. As can be seen in Fig. 2.9, $\dot{\sigma}_{\theta^\star}|_{\rm test}$ and $\dot{\sigma}_{\theta}|_{\rm test}$ have no difference when V is less than 8 and approach the true value (red dashed line). At $V = 12$, the overfitting issue is clearly shown as $\dot{\sigma}_{\theta}|_{\rm test}$ diverges, but $\dot{\sigma}_{\theta^\star}|_{\rm test}$ converges. $\dot{\sigma}_{\theta^\star}|_{\rm test}$ also diverges for $V \geq 14$ because there is no $0 \to 2$ transition in the trajectory.

For the discrete flashing ratchet model with partial information, we report $\dot{\Sigma}^n_{\theta^\star}|_{\rm test}$ in Fig. 2.4(b). See Fig. 2.10 for a comparison between $\dot{\Sigma}^n_{\theta^\star}|_{\rm test}$ and the actual EP per step $\dot{\sigma}$ in linear scale. We set the dimension of the embedding vector and the number of hidden units (H) in the GRU to 128. See Table 2.3 for a detailed configuration of the RNEEP. We train the RNEEP with a trajectory length of $L = 5 \times 10^7$, and the states $0', 1'$, and $2'$ are converted to $0, 1$, and 2 to remove the ON/OFF information. The number of training iterations is 10^5, and we evaluate $J_{\rm test}$ every 10^3 iterations. Figure 2.11 shows the training process of RNEEP with sequence lengths $n = 32$, 64, and 128 at potential $V = 2$. For $n = 32$, there is no overfitting issue: $J_{\rm train}$, $J_{\rm test}$, and $J_{\rm best}$ ($J_{\rm test}(\theta^\star)$) are almost identical, as can be seen in Fig. 2.11(a). The overfitting issue occurs for $n = 64$ (see Fig. 2.11(b)). For $n = 128$, Fig. 2.11(c) shows that $J_{\rm train}$ and $J_{\rm test}$ values highly fluctuate, while $J_{\rm best}$ is robust over the training iterations; the variance of $\dot{\Sigma}^n_{\theta^\star}|_{\rm test}$ is also lower than $\dot{\Sigma}^n_{\theta}|_{\rm test}$.

We now compare the RNEEP ($\dot{\Sigma}^n_{\theta^\star}|_{\rm test}$) with a plug-in estimator ($\dot{\Sigma}^n_{\rm plug}$) that directly estimates Eq. (2.24) by counting the frequency of $\boldsymbol{x}^n$ [17]. Figure 2.12 shows that $\dot{\Sigma}^n_{\rm plug}$ agrees with $\dot{\Sigma}^n_{\theta^\star}|_{\rm test}$ for $n = 2$ and $n = 8$; however, at $n = 16$, $\dot{\Sigma}^n_{\rm plug}$ cannot estimate well. We note that the plug-in estimator uses both training and test set data for a fair comparison with RNEEP.

RNEEP with sequence length n		
Layer	Output dim	Activation function
Input x	$\{0, 1, 2\}$	
Embedding	H	
Concatenate $\boldsymbol{x}^n$	$n \times H$	
GRU($\boldsymbol{x}^n$)	$n \times H$	None
Average	H	None
Output layer	1	None

Table 2.3: RNEEP configuration for the partial information problem.

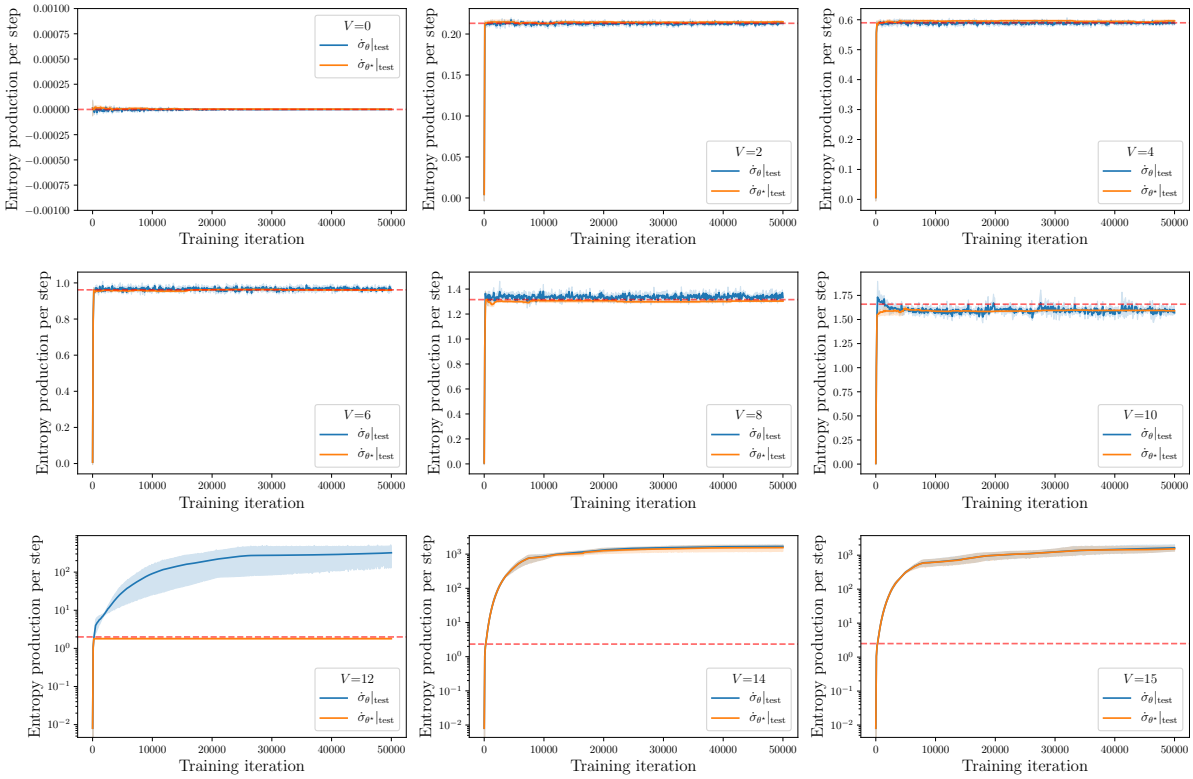

Figure 2.9: Estimations of EP per step over training iteration with respect to nine different potential V. Red dashed lines represent the analytic EP per step $\dot{\sigma}$. The shaded areas represent the standard deviation of estimations from five independently trained estimators.

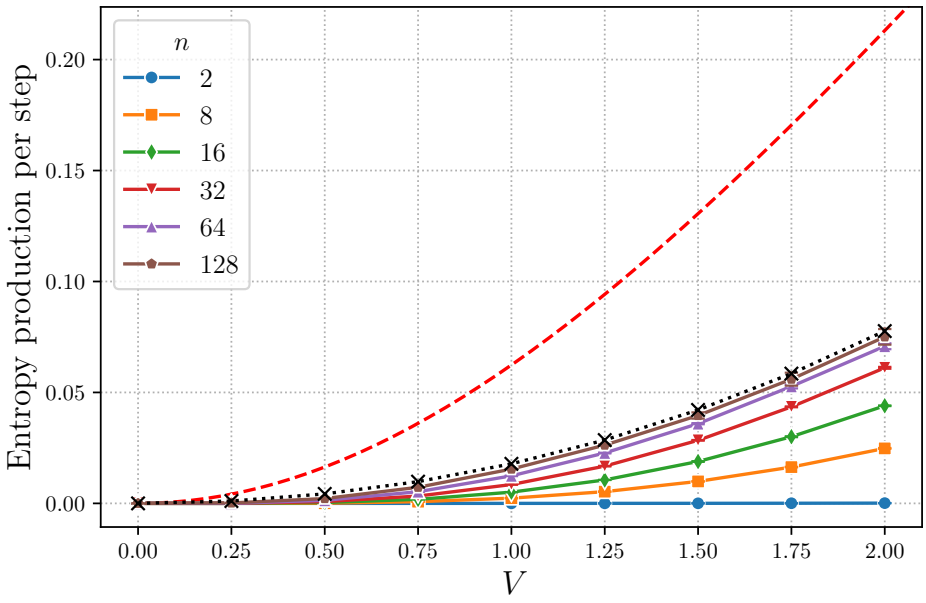

Figure 2.10: Estimated coarse-grained EP per step by RNEEP ($\dot{\Sigma}^n_{\theta^\star}|_{\text{test}}$) as a function of potential V for each sequence length n plotted with six different markers as shown in the legend. The black x's are the semianalytical values of $\dot{\Sigma}^\infty$, and the red dashed line denotes the analytic value of the actual EP per step $\dot{\sigma}$. The error bars represent the standard deviation of $\dot{\Sigma}^n_{\theta^\star}|_{\text{test}}$ from five independently trained estimators.

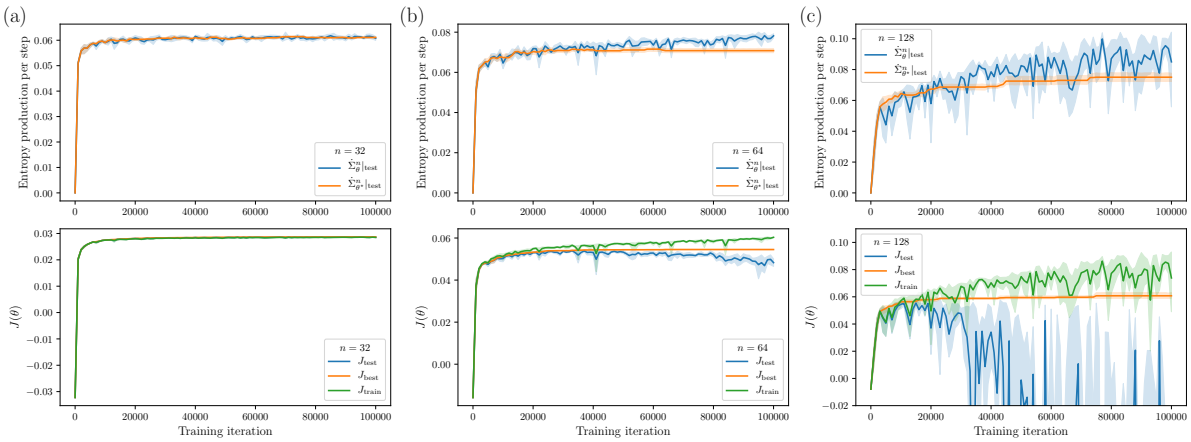

Figure 2.11: Estimations of the coarse-grained EP per step $\dot{\Sigma}^n_\theta$ (top) and $J(\theta)$ (bottom) over training iteration by the RNEEP with (a) $n = 32$, (b) $n = 64$, and (c) $n = 128$. The shaded areas represent the standard deviation of $\dot{\Sigma}^n_\theta$ and J values from five independently trained estimators.

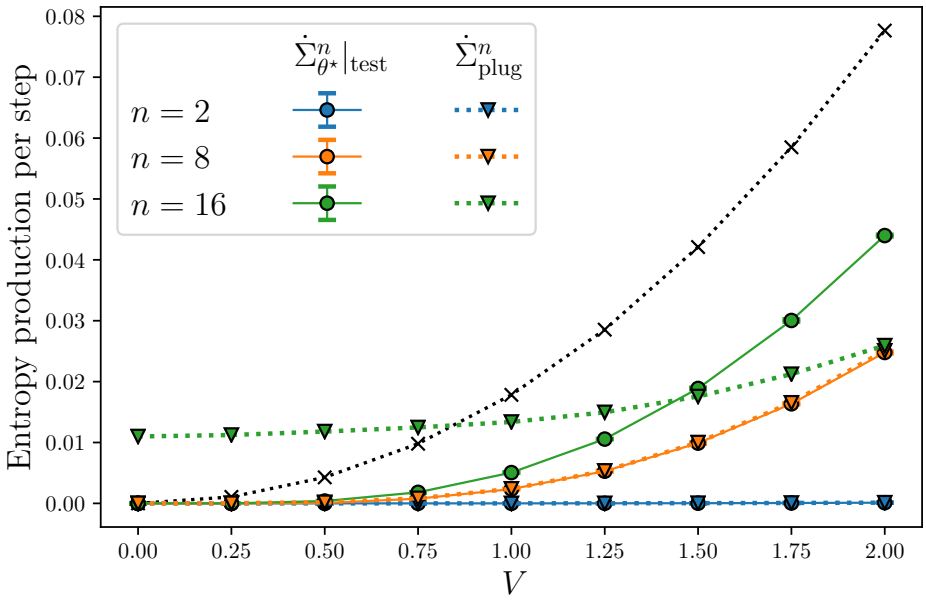

Figure 2.12: Estimation of the coarse-grained EP per step by the plug-in method $\dot{\Sigma}^n_{\text{plug}}$ (triangles) and RNEEP $\dot{\Sigma}^n_{\theta^\star}|_{\text{test}}$ (circles) as a function of potential V for $n = 2$ (blue), 8 (orange), and 16 (green) in the partial information problem. The black x's are the semianalytical values of $\dot{\Sigma}^\infty$. The error bars represent the standard deviation of $\dot{\Sigma}^n_{\theta^\star}|_{\text{test}}$ from five independently trained estimators.

2.5.6 Proof for coarse-grained entropy production

In this section, we prove that our objective function $J(\theta)$ has a maximum value when ΔS_θ is the stochastic coarse-grained EP. In steady state, the objective function $J(\theta)$ can be written as

$$J[h] = \sum_{\boldsymbol{x}^n} \sum_{\boldsymbol{\eta}^n} p(\boldsymbol{x}^n, \boldsymbol{\eta}^n) \left[h(\boldsymbol{x}^n) - h(\widetilde{\boldsymbol{x}}^n) - e^{-(h(\boldsymbol{x}^n) - h(\widetilde{\boldsymbol{x}}^n))} \right], \tag{2.22}$$

where n is the length of sequence $\boldsymbol{x}^n$ and

$$\begin{aligned} \boldsymbol{x}^n &= (x_1, x_2, ..., x_n), \quad \widetilde{\boldsymbol{x}}^n = (x_n, ..., x_2, x_1), \\ \boldsymbol{\eta}^n &= (\eta_1, \eta_2, ..., \eta_n), \quad \widetilde{\boldsymbol{\eta}}^n = (\eta_n, ..., \eta_2, \eta_1). \end{aligned}$$

The maximum condition for Eq. (2.22) can be obtained as follows:

$$\begin{aligned} 0 =& \partial_{h(\boldsymbol{x}'^n)} J[h] \\ =& \sum_{\boldsymbol{x}^n} \sum_{\boldsymbol{\eta}^n} p(\boldsymbol{x}^n, \boldsymbol{\eta}^n) \left(\delta_{\boldsymbol{x}^n, \boldsymbol{x}'^n} - \delta_{\widetilde{\boldsymbol{x}}^n, \boldsymbol{x}'^n} \right) \left[1 + e^{-(h(\boldsymbol{x}^n) - h(\widetilde{\boldsymbol{x}}^n))} \right] \\ =& \sum_{\boldsymbol{\eta}^n} p(\boldsymbol{x}'^n, \boldsymbol{\eta}^n) \left[1 + e^{-(h(\boldsymbol{x}'^n) - h(\widetilde{\boldsymbol{x}}'^n))} \right] - \sum_{\boldsymbol{\eta}^n} p(\widetilde{\boldsymbol{x}}'^n, \boldsymbol{\eta}^n) \left[1 + e^{-(h(\widetilde{\boldsymbol{x}}'^n) - h(\boldsymbol{x}'^n))} \right] \\ =& \left[1 + e^{-(h(\widetilde{\boldsymbol{x}}'^n) - h(\boldsymbol{x}'^n))} \right] \left[e^{-(h(\boldsymbol{x}'^n) - h(\widetilde{\boldsymbol{x}}'^n))} \sum_{\boldsymbol{\eta}^n} p(\boldsymbol{x}'^n, \boldsymbol{\eta}^n) - \sum_{\boldsymbol{\eta}^n} p(\widetilde{\boldsymbol{x}}'^n, \boldsymbol{\eta}^n) \right]. \end{aligned} \tag{2.23}$$

Then the solution for the optimization problem is

$$h(\boldsymbol{x}'^n) - h(\widetilde{\boldsymbol{x}}'^n) = \ln \frac{\sum_{\boldsymbol{\eta}^n} p(\boldsymbol{x}'^n, \boldsymbol{\eta}^n)}{\sum_{\widetilde{\boldsymbol{\eta}}^n} p(\widetilde{\boldsymbol{x}}'^n, \widetilde{\boldsymbol{\eta}}^n)} = \ln \frac{p(\boldsymbol{x}'^n)}{p(\widetilde{\boldsymbol{x}}'^n)}. \tag{2.24}$$

Chapter 3. Estimating entropy production in a stochastic system with odd-parity variables

* This chapter is reproduced based on the following paper with its figures and contexts - D.-K. Kim, S. Lee, and H. Jeong, *arXiv preprint arXiv:2112.04681*.

3.1 Introduction

Entropy production (EP) plays an important role in the field of stochastic thermodynamics because of its diverse relations to the dissipation, free energy, and fundamental bounds on the fluctuation of an observable. Through the fluctuation relation [2], it has been found that EP has a symmetry that enables us to measure the free energy difference from nonequilibrium experimental data [59]. Also, trade-off relations between EP and the relative error of physical quantities have been discovered recently [21, 60, 61, 62]. Such relations can provide a tighter version of the thermodynamic second law and a universal bound for fluctuations of diverse currents.

Following from the importance, various estimation methods for EP have been developed, such as the plug-in method [16] and compression algorithm [19]. Also, the development of thermodynamic uncertainty relations (TURs) open the possibility of measuring the EP in overdamped Langevin processes [63]. To estimate the exact EP with TURs, we have to measure the relative error of EP, which presents us with circular reasoning. To resolve this point, a few optimization strategies have been established [14, 23, 24, 64]. However, measuring EP remains challenging because of the curse of dimensionality.

Recently, machine learning based algorithms for EP estimation have been developed to efficiently mitigate such issues by using a neural network [65, 66, 67]. These methods employ neural networks as an EP estimator [65], where the estimator is trained for optimizing the variational representation of the EP [68]. Owing to the great power of neural networks, these approaches work well even in high-dimensional systems. Moreover, unlike the TURs methods, the neural estimator can provide stochastic EP directly from trajectory data without any correction [66].

Despite such progress, the developed methods have only considered systems with absent odd-parity variables, of which the sign changes under time-reversal operation [69, 70, 71]. For instance, underdamped Langevin dynamics includes an odd-parity variable called momentum [72]; in many cases, active matter or other biological systems are often modeled with underdamped dynamics [73, 53]. To estimate the EP of such systems, existing methodologies require additional information because the definition of EP in these systems differs from that in systems without odd-parity variables. Indeed, TURs for underdamped Langevin systems require detailed information about the systems [74, 75, 71], and therefore we cannot exactly measure EP from only trajectory data using TURs. Due to these difficulties, to our knowledge, methods for estimating EP in a stochastic system with odd-parity state variables have yet to be developed.

In this work, we propose a machine learning based method for estimating EP by using trajectory data obtained from a stochastic system with odd-parity variables. The proposed method can be applied to Langevin dynamics and Markov jump processes. For a system with continuous state variables, we demonstrate our estimation method with an underdamped bead-spring model in Sec. 3.2, and for the Markov jump case, we apply our approach in Sec. 3.3 to a one-particle odd-parity Markov jump process in which the state variables are discrete and the time interval between jumps is continuous.

3.2 Underdamped bead-spring model

In the underdamped bead spring model, N beads are coupled to the next nearest beads. Let $\boldsymbol{q} = (x_1, \ldots, x_N, v_1, \ldots, v_N)$ be a state vector of this model, and its time reversal counterpart is denoted as $\widetilde{\boldsymbol{q}} = (x_1, \ldots, x_N, -v_1, \ldots, -v_N)$ where x_i and v_i are the position and velocity of the i-th bead, respectively. Then, the dynamics of the state $\boldsymbol{q}$ with time τ is governed by the following underdamped Langevin equation:

$$\dot{\boldsymbol{q}}(\tau) = -\mathsf{M} \cdot \boldsymbol{q}(\tau) + \sqrt{2\mathsf{D}} \cdot \boldsymbol{\xi}(\tau), \tag{3.1}$$

with

$$\mathsf{M} = \begin{pmatrix} 0 & -\mathsf{I} \\ \mathsf{A} & \Gamma \end{pmatrix}, \quad \mathsf{D} = \begin{pmatrix} 0 & 0 \\ 0 & \mathsf{T} \end{pmatrix}, \tag{3.2}$$

where $[\mathsf{A}]_{ij} = (2\delta_{i,j} - \delta_{i,j+1} - \delta_{i+1,j})k/m$, $[\Gamma]_{ij} = \gamma\delta_{i,j}$, and $[\mathsf{T}]_{ij} = \delta_{i,j}\gamma k_{\mathrm{B}} T_i/m$. Otherwise, $\boldsymbol{\xi}$, is a Gaussian noise vector with zero mean and variance given by $\langle \xi_i(\tau)\xi_j(\tau')\rangle = \delta_{ij}\delta(\tau - \tau')$, and $\langle \ldots \rangle$ denotes the ensemble average. $\mathsf{0}$ and I are the $N \times N$ null and identity matrices, γ is the friction coefficient, and k is a spring constant. We set $k = k_{\mathrm{B}} = \gamma = 1$ [1].

For data preparation, we sample an ensemble of trajectories from Eq. (3.1). $\boldsymbol{q}_1$ is sampled from steady-state distribution (SSD) $p(\boldsymbol{q})$, and we use the second-order integrator to sample a trajectory $(\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_L)$, where $\boldsymbol{q}_t \equiv \boldsymbol{q}(t\Delta\tau)$ with fixed sampling interval $\Delta\tau$ [76]. In this case, the stochastic EP per step is defined as

$$\Delta S_{\mathrm{tot}}(\boldsymbol{q}_t, \boldsymbol{q}_{t+1}) = \ln \frac{p(\boldsymbol{q}_{t+1}|\boldsymbol{q}_t)}{p(\widetilde{\boldsymbol{q}}_t|\widetilde{\boldsymbol{q}}_{t+1})} + \ln \frac{p(\boldsymbol{q}_t)}{p(\boldsymbol{q}_{t+1})}, \tag{3.3}$$

where the first (second) term on the right-hand side is the environment EP (system entropy change) [1].

In order to estimate Eq. (3.3), we employ the objective function proposed in Ref. [65]:

$$J_{\theta_1} = \langle \Delta S_{\theta_1}(\boldsymbol{q}_t, \boldsymbol{q}_{t+1}) - e^{-\Delta S_{\theta_1}(\boldsymbol{q}_t, \boldsymbol{q}_{t+1})} + 1 \rangle_t, \tag{3.4}$$

where

$$\Delta S_{\theta_1}(\boldsymbol{q}_t, \boldsymbol{q}_{t+1}) = h_{\theta_1}(\boldsymbol{q}_t, \boldsymbol{q}_{t+1}) - h_{\theta_1}(\widetilde{\boldsymbol{q}}_{t+1}, \widetilde{\boldsymbol{q}}_t),$$

and $\langle \ldots \rangle_t$ denotes the expectation over t and the ensemble. Here, we use a multi-layer perceptron (MLP) for h_{θ_1}, where θ_1 represents the trainable neural network parameters. Using the stochastic gradient ascent method, we train the neural network h_{θ_1} to find the optimal parameter $\theta_1^\star$ which gives the maximum value of Eq. (3.4). The optimal parameter $\theta_1^\star$ gives

$$\Delta S_{\theta_1^\star}(\boldsymbol{q}_t, \boldsymbol{q}_{t+1}) = \ln \frac{p(\boldsymbol{q}_t, \boldsymbol{q}_{t+1})}{p(\widetilde{\boldsymbol{q}}_{t+1}, \widetilde{\boldsymbol{q}}_t)}, \tag{3.5}$$

and this can be split into

$$\Delta S_{\theta_1^\star}(\boldsymbol{q}_t, \boldsymbol{q}_{t+1}) = \ln \frac{p(\boldsymbol{q}_{t+1}|\boldsymbol{q}_t)}{p(\widetilde{\boldsymbol{q}}_t|\widetilde{\boldsymbol{q}}_{t+1})} + \ln \frac{p(\boldsymbol{q}_t)}{p(\boldsymbol{q}_{t+1})} + \ln \frac{p(\boldsymbol{q}_{t+1})}{p(\widetilde{\boldsymbol{q}}_{t+1})}.$$

[1] Here, we note that $[\Gamma]_{ij}$ is not $\delta_{i,j}\gamma/m$ but $\delta_{i,j}\gamma$. This means that the governing equation is not reduced to an overdamped Langevin equation when mass m approaches zero because the velocity autocorrelation is proportional to $e^{-\tau\gamma}$ rather than $e^{-\tau\gamma/m}$.

Therefore, the maximum value of J_{θ_1} is

$$J_{\theta_1^\star} = \left\langle \ln \frac{p(\boldsymbol{q}_{t+1}|\boldsymbol{q}_t)}{p(\widetilde{\boldsymbol{q}}_t|\widetilde{\boldsymbol{q}}_{t+1})} + \ln \frac{p(\boldsymbol{q}_t)}{p(\boldsymbol{q}_{t+1})} + \ln \frac{p(\boldsymbol{q}_{t+1})}{p(\widetilde{\boldsymbol{q}}_{t+1})} \right\rangle_t \tag{3.6}$$

$$= \langle \Delta S_{\rm tot}(\boldsymbol{q}_t, \boldsymbol{q}_{t+1}) \rangle_t + \left\langle \ln \frac{p(\boldsymbol{q})}{p(\widetilde{\boldsymbol{q}})} \right\rangle . \tag{3.7}$$

The last term of Eq. (3.7) is the Kullback–Leibler divergence between the SSD $p(\boldsymbol{q})$ and its mirror state distribution $p(\widetilde{\boldsymbol{q}})$. Equation (3.7) can be written as

$$J_{\theta_1^\star} = \dot{\sigma}\Delta\tau + \langle \Delta S_{\rm as}(\boldsymbol{q}) \rangle \,, \tag{3.8}$$

where $\dot{\sigma}$ is the EP rate and $\Delta S_{\rm as}(\boldsymbol{q})$ is the asymmetry in SSD-given state $\boldsymbol{q}$:

$$\Delta S_{\rm as}(\boldsymbol{q}) = \ln \frac{p(\boldsymbol{q})}{p(\widetilde{\boldsymbol{q}})}. \tag{3.9}$$

To measure the EP rate $\dot{\sigma}$, we need to estimate $\Delta S_{\rm as}(\boldsymbol{q})$. To do so, we define another objective function:

$$J_{\theta_0} = \langle \Delta S_{\theta_0}(\boldsymbol{q}_t) - e^{-\Delta S_{\theta_0}(\boldsymbol{q}_t)} + 1 \rangle_t, \tag{3.10}$$

where $\Delta S_{\theta_0}(\boldsymbol{q}_t) = h_{\theta_0}(\boldsymbol{q}_t) - h_{\theta_0}(\widetilde{\boldsymbol{q}}_t)$ and h_{θ_0} is another MLP. Using the same method as before, we maximize Eq. (3.10). The optimal parameter $\theta_0^\star$ gives

$$\Delta S_{\theta_0^\star}(\boldsymbol{q}_t) = \ln \frac{p(\boldsymbol{q}_t)}{p(\widetilde{\boldsymbol{q}}_t)}, \tag{3.11}$$

$$J_{\theta_0^\star} = \langle \Delta S_{\rm as}(\boldsymbol{q}) \rangle. \tag{3.12}$$

Using the two trained neural estimators, the estimations of the stochastic EP $\Delta S_{\rm tot}$ and EP rate $\dot{\sigma}$ are, respectively,

$$\Delta \hat{S}_{\rm tot}(\boldsymbol{q}_t, \boldsymbol{q}_{t+1}) = \Delta S_{\theta_1}(\boldsymbol{q}_t, \boldsymbol{q}_{t+1}) - \Delta S_{\theta_0}(\boldsymbol{q}_{t+1}), \tag{3.13}$$

and

$$\hat{\dot{\sigma}} = \frac{J_{\theta_1} - J_{\theta_0}}{\Delta\tau}. \tag{3.14}$$

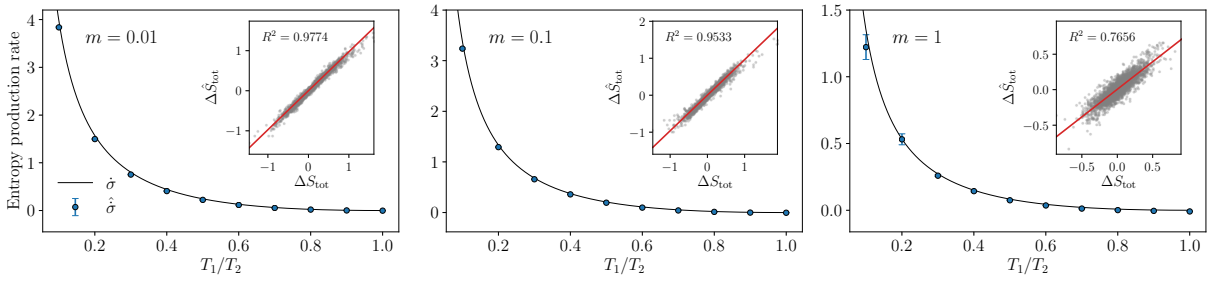

Figure 3.1: Entropy production rate as a function of T_1/T_2 in the underdamped bead-spring model for $m = 0.01$ (left), 0.1 (middle), and 1 (right). The solid lines (blue dots) represent the analytical EP rate $\dot{\sigma}$ (estimated EP rate $\hat{\dot{\sigma}}$). The insets are scatter plots of the analytical $\Delta S_{\rm tot}$ and the estimation $\Delta\hat{S}_{\rm tot}$ at $T_1/T_2 = 0.1$, where the red lines denote linear regression. Error bars represent the standard deviation of $\hat{\dot{\sigma}}$ from five independently trained estimators.

To verify our approach, we sample the training, validation, and test set data with sampling interval $\Delta\tau = 0.01$, trajectory length $L = 4000$, ensemble size (number of trajectories) $M = 10^4$, and number of beads $N = 2$. We fix the temperature of the second bead to $T_2 = 10$, and the datasets are sampled from $T_1 \in \{1, 2, \ldots, 10\}$, each with three different values of mass $m \in \{0.01, 0.1, 1.0\}$.

We train two MLPs, h_{θ_0} and h_{θ_1}, to maximize Eq. (3.4) and Eq. (3.10), respectively, by using the Adam optimizer [58] with learning rate 10^{-5} and batch size 4096. Each MLP has a single output unit and two hidden layers of 256 units with rectified linear units (ReLU) nonlinearity. The parameters of these MLPs are initialized with the orthogonal initialization scheme [77], which is good for stable and efficient convergence. We use this setup for all training processes unless otherwise noted. As a data preprocessing step, we normalize each position and velocity in state $\mathbf{q}$ before feeding to the MLPs. The total number of training iterations is 10^5, and we evaluate J_{θ_0} and J_{θ_1} values from the validation set every 400 training iterations. The best trained parameters $\{\theta_0^\star, \theta_1^\star\}$ are obtained by monitoring the J_{θ_0} and J_{θ_1} values during the training processes. Throughout this work, all the results in the figures are evaluated from the test set, where the error bars represent the standard deviation of the estimations from five independently trained estimators.

In Fig. 3.1, we plot the training results for estimating EP in the underdamped bead-spring model. For $m = 0.01$ and $m = 0.1$ (left and middle plot), the estimated EP rates $\hat{\dot{\sigma}}$ (blue dots) are well matched with the analytic EP rates $\dot{\sigma}$ (solid lines) with small variance. The insets in Fig. 3.1 are scatter plots between the estimated stochastic EP $\Delta\hat{S}_{\mathrm{tot}}$ and analytic stochastic EP ΔS_{tot} with a fitted linear regression line (red line), showing that the stochastic EP is also well estimated with high R^2 values. However, the training result for $m = 1$ (right plot) shows high variance in the EP rate estimation at $T_1/T_2 = 0.1$, and the stochastic EP is inaccurately estimated with $R^2 = 0.7656$.

In order to identify these inaccuracies, in Fig. 3.2 we compare the analytic value of $\langle \Delta S_{\mathrm{as}} \rangle$ and EP per step $\dot{\sigma}\Delta\tau$ while changing the mass m at $T_1/T_2 = 0.1$. As can be seen in the figure, the total EP per step is much larger than $\langle \Delta S_{\mathrm{as}} \rangle$ in the small m regime, but with increasing m, the total EP per step becomes smaller than $\langle \Delta S_{\mathrm{as}} \rangle$. At $m = 1$, the ratio $\dot{\sigma}\Delta\tau / \langle \Delta S_{\mathrm{as}} \rangle$ is almost 0.01; such a small ratio leads to a very slight difference between Eq. (3.8) and Eq. (3.12), which may contribute to the difficulty in EP estimation from Eq. (3.14) that optimizes J_{θ_1} and J_{θ_0} separately.

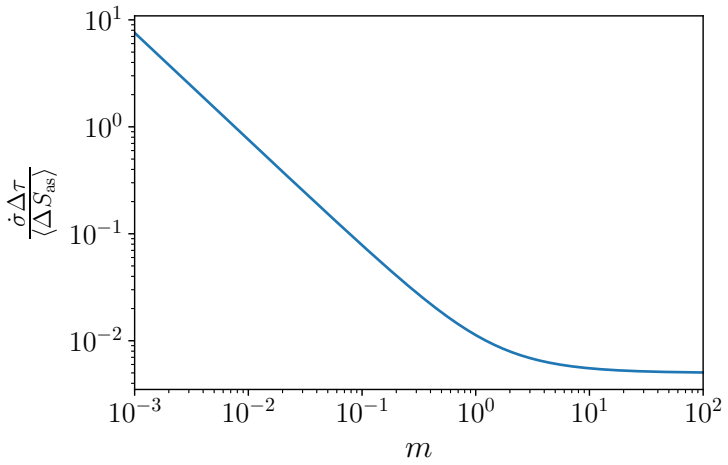

Figure 3.2: Entropy production per step over $\langle \Delta S_{\mathrm{as}} \rangle$ as a function of mass m at $T_1/T_2 = 0.1$.

3.3 One-particle odd-parity Markov jump process

Next, we consider a one-particle Markov jump process on a ring with N sites [70]. In this model, the particle state is defined as $\boldsymbol{q} = (n, v)$, and its time reversal state is denoted as $\widetilde{\boldsymbol{q}} = (n, -v)$ where $n \in \{1, \dots, N\}$ is the position variable and $v \in \{-1, +1\}$ is velocity variable. The master equation is given as

$$\partial_t p(\boldsymbol{q}, t) = \sum_{\boldsymbol{q}' \neq \boldsymbol{q}} [\omega_{\boldsymbol{q},\boldsymbol{q}'} p(\boldsymbol{q}', t) - \omega_{\boldsymbol{q}',\boldsymbol{q}} p(\boldsymbol{q}, t)], \tag{3.15}$$

where $\omega_{\boldsymbol{q}',\boldsymbol{q}}$ denotes the transition rate for the jump from $\boldsymbol{q}$ to $\boldsymbol{q}'$, and the transition rates are given as $\omega_{(n\pm v,\pm v),(n,\pm v)} = f_{\pm v}$ and $\omega_{(n,\mp v),(n,\pm v)} = r_{\pm v}$. For a jump of waiting time τ ($\boldsymbol{q} \to \boldsymbol{q}'$, τ), the entropy production [70, 55] is written as

$$\Delta S_{\mathrm{tot}}(\boldsymbol{q}, \boldsymbol{q}', \tau) = \ln \frac{\omega_{\boldsymbol{q}',\boldsymbol{q}} e^{-\tau/\tau_{\boldsymbol{q}}}}{\omega_{\widetilde{\boldsymbol{q}},\widetilde{\boldsymbol{q}}'} e^{-\tau/\tau_{\widetilde{\boldsymbol{q}}}}} + \ln \frac{p(\boldsymbol{q})}{p(\boldsymbol{q}')}, \tag{3.16}$$

where $p(\boldsymbol{q}) \equiv p(\boldsymbol{q}, \infty)$ is the steady state of Eq. (3.15). The rightmost term in Eq. (3.16) is the system EP of the steady state. The first term on the right-hand side denotes the EP due to the jump and the wait. The numerator $\omega_{\boldsymbol{q}',\boldsymbol{q}} \psi_{\boldsymbol{q}}(\tau)$ is the transition rate $\boldsymbol{q} \to \boldsymbol{q}'$ after waiting for time τ in the state $\boldsymbol{q}$. The denominator $\omega_{\widetilde{\boldsymbol{q}},\widetilde{\boldsymbol{q}}'} e^{-\tau/\tau_{\widetilde{\boldsymbol{q}}}}$ is the corresponding reversed transition rate with τ waiting time in the state $\widetilde{\boldsymbol{q}}$ after a jump $\widetilde{\boldsymbol{q}}' \to \widetilde{\boldsymbol{q}}$. The mean waiting time of the state $\boldsymbol{q}$ is $\tau_{\boldsymbol{q}} = 1/\sum_{\boldsymbol{q}' \neq \boldsymbol{q}} \omega_{\boldsymbol{q}',\boldsymbol{q}}$.

For a Markov jump process, we represent a trajectory Γ as a sequence of jumps:

$$\Gamma = \{(\boldsymbol{q}_1 \to \boldsymbol{q}_2, \tau_1), \dots, (\boldsymbol{q}_i \to \boldsymbol{q}_{i+1}, \tau_i), \dots\}, \tag{3.17}$$

where q_i denotes the i-th state. To train neural networks with the sequence of jumps, we rearrange Eq. (3.17) into

$$\begin{aligned} \Delta S_{\mathrm{tot}}(\boldsymbol{q}, \boldsymbol{q}', \tau) =& \ln \frac{R(\widetilde{\boldsymbol{q}}')}{R(\boldsymbol{q}')} + \ln \frac{R(\widetilde{\boldsymbol{q}})}{R(\boldsymbol{q})} \\ &+ \ln \frac{\rho_{\boldsymbol{q}',\boldsymbol{q}} R(\boldsymbol{q})}{\rho_{\widetilde{\boldsymbol{q}},\widetilde{\boldsymbol{q}}'} R(\widetilde{\boldsymbol{q}}')} + \ln \frac{\psi_{\boldsymbol{q}}(\tau) R(\boldsymbol{q})}{\psi_{\widetilde{\boldsymbol{q}}}(\tau) R(\widetilde{\boldsymbol{q}})} \\ &+ \ln \frac{\tau_{\boldsymbol{q}} \tau_{\widetilde{\boldsymbol{q}}'}}{\tau_{\widetilde{\boldsymbol{q}}}, \tau_{\boldsymbol{q}'}} \end{aligned} \tag{3.18}$$

$$\equiv \Delta S_{\mathrm{alt}}(\boldsymbol{q}, \boldsymbol{q}', \tau) + \ln \frac{\tau_{\boldsymbol{q}} \tau_{\widetilde{\boldsymbol{q}}'}}{\tau_{\widetilde{\boldsymbol{q}}}, \tau_{\boldsymbol{q}'}}, \tag{3.19}$$

where $\rho_{\boldsymbol{q}',\boldsymbol{q}} \equiv \tau_{\boldsymbol{q}} \omega_{\boldsymbol{q}',\boldsymbol{q}}$ denotes the jump probability from $\boldsymbol{q}$ to $\boldsymbol{q}'$ and $R_{\boldsymbol{q}} \equiv \tau_{\boldsymbol{q}}^{-1} p(\boldsymbol{q}) / \sum_{\boldsymbol{q}'} \tau_{\boldsymbol{q}'}^{-1} p(\boldsymbol{q}')$ is the stationary probability distribution of jump probabilities $\rho_{\boldsymbol{q}',\boldsymbol{q}}$ [55]. The waiting time distribution (WTD) in the state $\boldsymbol{q}$ is denoted by $\psi_{\boldsymbol{q}}(\tau) \equiv \frac{e^{-\tau/\tau_{\boldsymbol{q}}}}{\tau_{\boldsymbol{q}}}$. The last term of Eq. (3.18), which consists of mean waiting times, can be directly measured from without any estimator hence we focus on estimating $\Delta S_{\mathrm{alt}}(\boldsymbol{q}, \boldsymbol{q}', \tau)$. Because the average of the last term is zero, an averaged total entropy production and an averaged alternative entropy production are the same $\langle \Delta S_{\mathrm{tot}} \rangle = \langle \Delta S_{\mathrm{alt}} \rangle$. To estimate the stochastic EP [Eq. (3.18)], we require three kinds of estimators: ΔS_{θ_0}, ΔS_{θ_1}, and ΔS_{θ_2}. The first and second terms on the right-hand side of Eq. (3.18) can be estimated with $\Delta S_{\theta_0}(\boldsymbol{q})$ and $\Delta S_{\theta_0}(\boldsymbol{q}')$, respectively. The third term can be estimated with $\Delta S_{\theta_1}(\boldsymbol{q}, \boldsymbol{q}')$. We use the same objective functions as the previous section, Eq. (3.10) and Eq. (3.4), for training ΔS_{θ_0} and ΔS_{θ_1}, but we develop a different kind of estimator with a new objective function for estimating the fourth term on the right-hand side of Eq. (3.18):

$$J_{\theta_2} = \langle \Delta S_{\theta_2}(\boldsymbol{q}, \tau) - e^{-\Delta S_{\theta_2}(\boldsymbol{q}, \tau)} + 1 \rangle_\tau, \tag{3.20}$$

where

$$\Delta S_{\theta_2}(\boldsymbol{q},\tau) = h_{\theta_2}(\boldsymbol{q},\tau) - h_{\theta_2}(\widetilde{\boldsymbol{q}},\tau). \tag{3.21}$$

The optimal parameter $\theta_2^\star$ gives

$$\Delta S_{\theta_2^\star}(\boldsymbol{q},\tau) = \ln\frac{\psi_{\boldsymbol{q}}(\tau)}{\psi_{\widetilde{\boldsymbol{q}}}(\tau)} + \ln\frac{R_{\boldsymbol{q}}}{R_{\widetilde{\boldsymbol{q}}}}, \tag{3.22}$$

$$J_{\theta_2^\star} = \langle \Delta S_{\theta_2}(\boldsymbol{q},\tau)\rangle_\tau + \langle \Delta S_{\mathrm{as}}(\boldsymbol{q})\rangle. \tag{3.23}$$

The first term of Eq. (3.22) is the contribution from the waiting time τ to the EP. We denote this contribution as

$$\Delta S_{\mathrm{WTD}}(\boldsymbol{q},\tau) \equiv \ln\frac{\psi_{\boldsymbol{q}}(\tau)}{\psi_{\widetilde{\boldsymbol{q}}}(\tau)}. \tag{3.24}$$

Then, using these three trained neural estimators, we can estimate the EP ΔS_{alt} and EP rate $\dot\sigma$ with the following equations:

$$\begin{aligned}\Delta S_{\mathrm{alt}}(\boldsymbol{q},\boldsymbol{q}',\tau) = {}& \Delta S_{\theta_1}(\boldsymbol{q},\boldsymbol{q}') + \Delta S_{\theta_2}(\boldsymbol{q},\tau) \\ & - \Delta S_{\theta_0}(\boldsymbol{q}) - \Delta S_{\theta_0}(\boldsymbol{q}'),\end{aligned} \tag{3.25}$$

and

$$\hat{\dot\sigma} = \frac{J_{\theta_1} + J_{\theta_2} - 2J_{\theta_0}}{\tau_{\mathrm{avg}}}, \tag{3.26}$$

where τ_{avg} denotes the average of the waiting time τ from the trajectory data. To handle the discrete state variables, we replace the input layers of h_{θ_0}, h_{θ_1}, and h_{θ_2} with an embedding layer that converts discrete values into continuous space vectors. The dimension of the embedding layer is 128; other neural network configurations and training setup are the same as in the previous section.

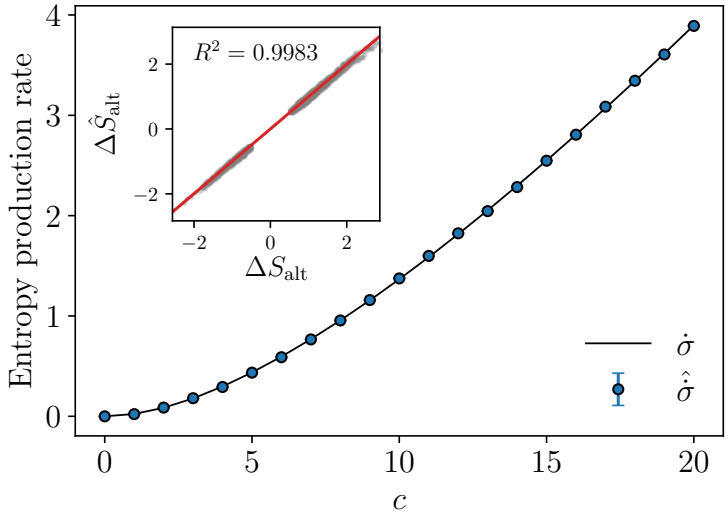

Figure 3.3: Entropy production rate as a function of c in the one-particle odd-parity Markov jump process. The inset is a scatter plot of ΔS_{tot} and $\Delta\hat{S}_{\mathrm{tot}}$ at $c = 10$ where the red line denotes the linear regression. The standard deviation (error bar) of the EP rate estimation $\hat{\dot\sigma}$ is much smaller than the symbol size.

To validate this method, we sample an ensemble of trajectories with total number of jumps $L = 10^4$, ensemble size $M = 50$, and number of sites $N = 10$. The Gillespie algorithm [78] is used for sampling a sequence of $L - 1$ jumps,

$$\Gamma = \{(\boldsymbol{q}_1 \to \boldsymbol{q}_2, \tau_1), \ldots, (\boldsymbol{q}_{L-1} \to \boldsymbol{q}_L, \tau_{L-1})\}.$$

We set the transition rates as $f_v = r_{-v} = 1$, $f_{-v} = 1 + 0.1c$, and $r_v = 1 + 0.2c$, where c is a positive real number. Each of the training, validation, and test trajectory datasets are sampled with c in the 0–20 range. When $c = 0$, the steady state of this system is in equilibrium. Non-zero c induces thermodynamic currents and increases EP in the steady state.

We train three neural networks h_{θ_0}, h_{θ_1}, h_{θ_2} to maximize J_{θ_0}, J_{θ_1}, and J_{θ_2}, respectively. The total training iterations are 10^4, and J_{θ_0}, J_{θ_1}, and J_{θ_2} are evaluated every 500 iterations from the validation dataset. As shown in Fig. 3.3, the EP rate estimation $\hat{\dot{\sigma}}$ [Eq. (3.26)] agrees with the analytic EP rate $\dot{\sigma}$. A scatter plot between ΔS_{tot} and $\Delta \hat{S}_{\mathrm{tot}}$ [Eq. (3.25)] at $c = 10$ with a fitted linear regression line (red line) is shown in the Fig. 3.3 inset. Overall R^2 values of the linear regression between ΔS_{alt} and $\Delta \hat{S}_{\mathrm{alt}}$ are above 0.99.

3.4 Discussion

Unlike systems with only even-parity variables, in which EP can be estimated with a single neural estimator [65], we require additional estimators to treat the mirror state asymmetry ΔS_{as} [Eq. (3.9)] and waiting-time contribution ΔS_{WTD} [Eq. (3.24)] in the estimation of EP in systems with odd-parity variables.

The first example in this work demonstrated that EP in the underdamped bead-spring model can be estimated by two neural estimators ΔS_{θ_0} and ΔS_{θ_1} which are trained to maximize J_{θ_0} and J_{θ_1}, respectively. However, this method has a limitation when J_{θ_0} and J_{θ_1} values are very close to each other; a small difference between these two values may cause a high variance in the EP estimations. This problem comes the training of two independent objective functions J_{θ_0} and J_{θ_1}. As shown in Eq. (3.12) and Eq. (3.8), J_{θ_1} includes contribution from $\langle \Delta S_{\mathrm{as}} \rangle$ [Eq. (3.9)], so if the $\langle S_{\mathrm{as}} \rangle$ contribution in J_{θ_1} is dominant, then estimating the EP from the difference between J_{θ_0} and J_{θ_1} might be hard. Also, $\hat{\dot{\sigma}}$ [Eq. (3.14)] is not a lower bound on EP rate $\dot{\sigma}$. While J_{θ_0} and J_{θ_1} are the lower bounds on Eq. (3.12) and Eq. (3.8), respectively, this does not guarantee that $\hat{\dot{\sigma}} = (J_{\theta_1} - J_{\theta_1})/\Delta\tau$ is a lower bound on the EP rate. Therefore, designing a single objective function that can bound the EP will be an important future work.

The second example demonstrated the estimation of the EP in an odd-parity Markov jump process with three neural estimators ΔS_{θ_0}, ΔS_{θ_1}, and ΔS_{θ_2} trained to maximize J_{θ_0}, J_{θ_1}, and J_{θ_2}, respectively. We stress that our method is the first tool to estimate EP in a Markov jump process with odd-parity variables. As shown in Eq. (3.24), the WTD does not contribute to the EP when the system has no odd-parity variables. However, if the accessible states are coarse-grained variables, then the WTD plays an important role despite the absence of odd-parity variables. In a related work, Martínez *et al.* [55] developed a method using WTD statistics to infer the EP from trajectories of coarse-grained states whose dynamics are described as semi-Markov processes; this method can infer EP even in the absence of observable currents. Also, a recent work by Skinner and Dunkel [79] uses the fluctuation in WTD to infer the EP rate in the hidden Markov processes of two coarse-grained states. As a future work, developing an estimator for EP when odd-parity variables are coarse-grained will be an intriguing task,

because the inferred EP from the trajectory of coarse-grained states can be larger than the actual EP when odd-parity variables are coarse-grained [80, 81].

In summary, we have developed a machine learning based method that can estimate the stochastic EP and EP rate in a fluctuating system with odd-parity variables via neural networks. Our method was demonstrated with both an underdamped Langevin process and Markov jump process with odd-parity variables. It is expected to provide a practical tool for studying nonequilibrium systems in general experimental situations.

The results of all runs and the code implemented in `PyTorch` [56] are available online [2].

[2] https://github.com/kdkyum/odd_neep

Chapter 4. Deep reinforcement learning for finding optimal feedback control in a collective flashing ratchet

* This chapter is reproduced based on the following published paper with its figures, tables, and contexts - D.-K. Kim and H. Jeong, *Phys. Rev. Research* **3**, L022002 (2021).

4.1 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 [82, 83, 84, 85]. 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 [86]. Feedback strategies for maximizing the current in flashing ratchets have been extensively studied [85, 86, 87, 88, 89, 90, 91, 92, 93, 94] due to the model's applicability in various disciplines [95]; for instance, flashing ratchets have been used for explaining transport phenomena in biological processes such as ion pumping [96], molecular transportation [97], and by motor proteins [98, 99, 100, 101]. However, the proposed feedback strategies [85, 86, 87, 88, 89, 90, 91, 92] are not optimal policies for a moderate number of particles and require prior information of the system as well.

Thanks to the recent advances in deep learning [47], physicists in diverse fields have been applying it to complex problems that are analytically intractable, e.g. glassy systems [102], quantum matter [103], and others [42]. In particular, reinforcement learning (RL) [104] has shown unprecedented success in previously unsolvable problems through combination with deep neural networks [105, 106, 107, 108]. 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 [109, 110, 111, 112, 113]. In this work, 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 [93].

4.2 Collective flashing ratchet

We consider the collective flashing ratchet model [86], which consists of an ensemble of N non-interacting 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 over-damped Langevin equation:

$$\begin{aligned} \eta\dot{x}_i(t) &= \alpha(s_t)F(x_i(t)) + \xi_i(t); \\ s_t &\equiv \{x_1(t), \ldots, x_N(t)\}, \quad i = 1, \ldots, N, \end{aligned} \tag{4.1}$$

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 $\mathbb{E}[\xi_i(t)\xi_j(t')] = 2\eta k_{\mathrm{B}}T\delta_{ij}\delta(t-t')$ 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. 4.1(a)]

$$U(x) = U_0\left[\sin\left(\frac{2\pi x}{L}\right) + \frac{1}{4}\sin\left(\frac{4\pi x}{L}\right)\right]. \tag{4.2}$$

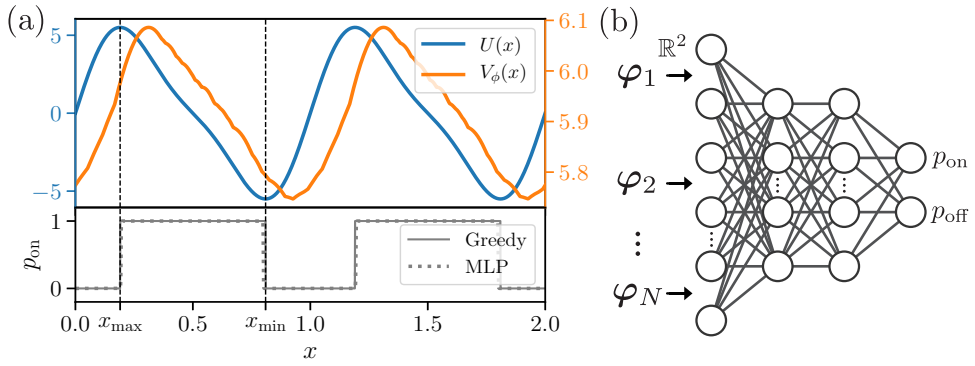

Figure 4.1: (a) $N = 1$ case. Top: Potential U and trained value network V_ϕ as a function of position x are denoted by blue and orange lines, respectively. Bottom: The solid line denotes the probability of switching on the potential ($p_{\rm on}$) as a function of x for the greedy policy. The dotted line represents $p_{\rm on}$ of the trained MLP policy. (b) Illustration of a MLP with two hidden layers for the policy network π_θ.

In all simulations, we set $L = 1$, $k_{\rm B}T = 1$, diffusion coefficient $D = k_{\rm B}T/\eta = 1$, $U_0 = 5k_{\rm B}T$, 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

$$\mathbb{E}_\alpha[\dot{x}] \equiv \mathbb{E}_\alpha\left[\frac{1}{N}\sum_{i=1}^{N}\dot{x}_i\right] \quad (\text{Unit} : D/L). \tag{4.3}$$

Various policies for maximizing the current (4.3) have been proposed as follows: the periodic switching policy [85], maximizing instantaneous current (greedy policy) [86], threshold policy [87, 88, 89], and Bellman's criterion [94].

The periodic switching policy [85] is $\alpha(t) = 1$ for $t \in [0, \mathcal{T}_{\rm on})$, $\alpha(t) = 0$ for $t \in [\mathcal{T}_{\rm on}, \mathcal{T}_{\rm on} + \mathcal{T}_{\rm off})$, and periodic $\alpha(t + \mathcal{T}_{\rm on} + \mathcal{T}_{\rm off}) = \alpha(t)$ with optimal periods $\mathcal{T}_{\rm on} \approx 0.03L^2/D$ and $\mathcal{T}_{\rm off} \approx 0.04L^2/D$. 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 [86] 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 [87, 88, 89] is $\alpha(s_t) = 0$ if $f(s_t) \le u_{\rm on}$ when $f(t)$ is decreasing, and $\alpha(s_t) = 1$ if $f(s_t) \ge u_{\rm off}$ when $f(t)$ is increasing, with thresholds $u_{\rm on} \ge 0$ and $u_{\rm off} \le 0$. The threshold policy with optimal thresholds gives 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.* [94] 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.

4.3 Actor-critic algorithm

We employ the actor-critic algorithm, which is one of the policy gradient methods in RL [104], 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:

$$r_t = \frac{1}{N}\sum_{i=1}^{N}\left(x_i(t+\Delta t) - x_i(t)\right). \tag{4.4}$$

The total discounted reward from time t, called 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 actor, where θ denotes the trainable neural network parameters, that takes system state s as an input. The outputs $\pi_\theta(s) = (p_{\rm on}, p_{\rm off})$ are the probabilities for switching the potential on or off [see Fig. 4.1(b)]. We sample the on-off probability from $\pi_\theta(s_t)$ every t in the training process.

The goal in RL is obtaining 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 [94]. However, in this work, we assume that we can only access the system state s_t and reward r_t. In such case, called model-free RL, we need an estimator V_ϕ for a *value function*:

$$V^\pi(s_t) = \mathbb{E}_\pi[G_t|s_t], \tag{4.5}$$

which is the expected return given state s_t under a policy π. The estimator V_ϕ, called 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 [114]. Among them, we employ proximal policy optimization [115], which is widely used in RL because of its scalability, data efficiency, and robustness for hyperparameters (see Section 4.6.1 for training details). After the training process is complete, we test the policy deterministically, i.e.

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

4.4 Results

First, we employ multilayer perceptron (MLP) architecture for the policy network π_θ and value network V_ϕ [see Fig. 4.1(b)]. The configuration details of the neural network architectures are given in Section 4.6.2. 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 neural network input where

$$\boldsymbol{\varphi}_i(t) = \left[\cos\left(\frac{2\pi x_i(t)}{L}\right), \sin\left(\frac{2\pi x_i(t)}{L}\right)\right]. \tag{4.6}$$

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. 4.1(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 ($x_{\rm max}$), the particle can slide to the right or left with a 50/50 chance, and therefore the expected current is maximum slightly right of $x_{\rm max}$.

For the $N = 2$ case, as shown in the left panel of Fig. 4.2(b), the greedy policy switches on (off) the potential when the particles are inside (outside) the white contour. On the other hand, the decision

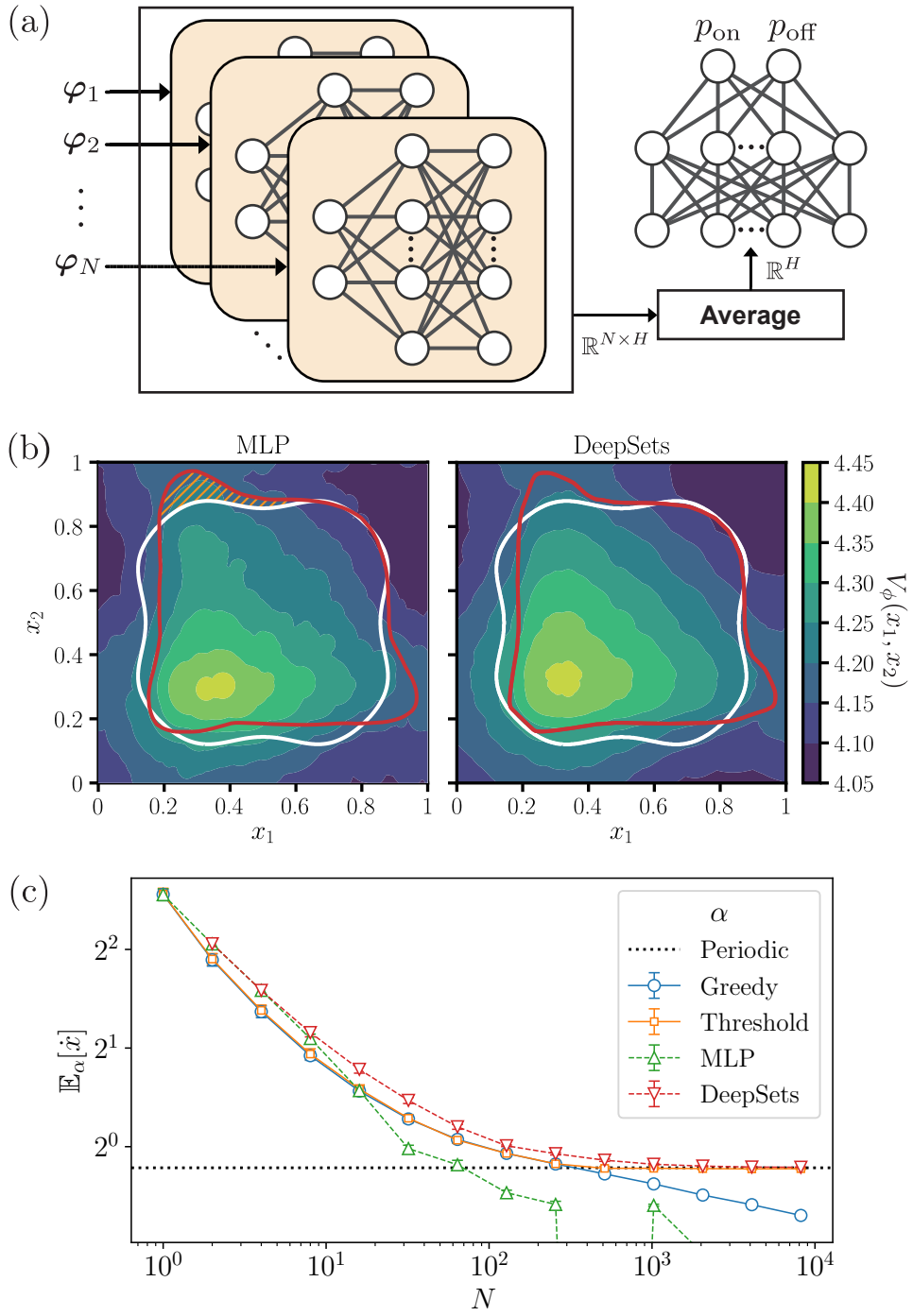

Figure 4.2: (a) DeepSets architecture for the policy network π_θ. H is the number of hidden units for each layer. (b) Decision boundaries from a trained MLP (left) and trained DeepSets (right) for $N = 2$. The white contour denotes where the mean force $f(x_1, x_2)$ is zero. The red contour is $p_{\text{on}} = 0.5$ 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 work, error bars represent the standard deviation of the current measured from the realized trajectory ensemble over the period $t = 50L^2/D$.

boundary of the trained MLP policy π_θ (red contour) agrees with the policy discovered by Roca *et al.* [94] 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, particle 1 and particle 2 are located on the downhill of the potential ($x_{\max} < x < x_{\min}$) and near the minimum ($x_{\min}$), respectively; while particle 2 will soon reach $x_{\min}$ and become trapped in the potential well, particle 1 can keep moving down along the potential [94].

However, the decision boundary (red contour) and V_ϕ (color gradient) are not symmetric over the line $x_1 = x_2$ [see Fig. 4.2(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 [116], for the policy and value networks. In this architecture [see Fig. 4.2(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. 4.2(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. 4.2(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. Figure 4.3 shows that with increasing N, the deterministic control $\alpha(t)$ of the trained DeepSets policy as a function of time t converges with that of the periodic switching policy.

We have also applied this methods to the sawtooth potential:

$$U(x) = \begin{cases} \dfrac{3U_0}{L} x & \text{for} \quad 0 \le x \le \frac{L}{3} \\ \\ U_0 - \dfrac{3U_0}{2L}\left(x - \dfrac{L}{3}\right) & \text{for} \quad \frac{L}{3} < x < L, \end{cases} \qquad \text{and } U(x+L) = U(x). \tag{4.7}$$

Figure 4.4(a) shows the sawtooth potential Eq. (4.7) with $L = 1$ and $U_0 = 5$. And the training results are shown in Fig. 4.4(b–c).

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 [90, 91, 92, 93]. To verify that deep RL is applicable to such a realistic situation, we consider a feedback time-delay τ in Eq. (4.1), i.e. $\alpha(s_t)$ is replaced by

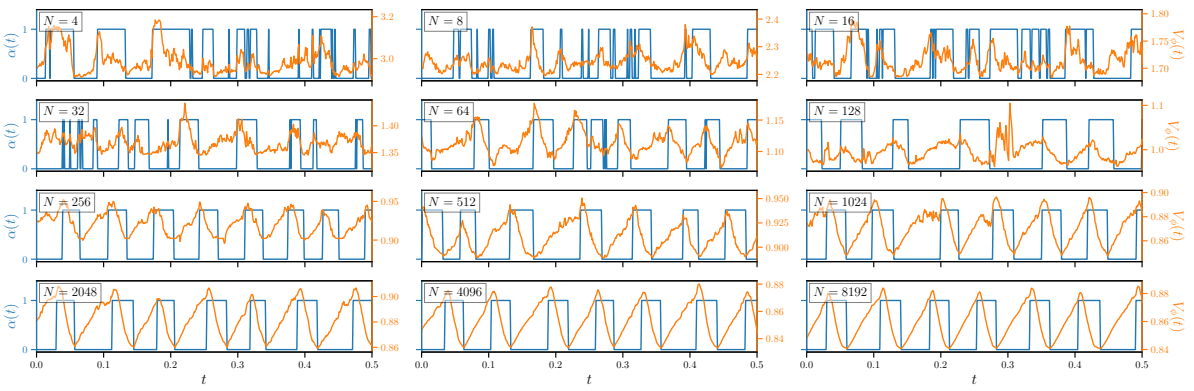

Figure 4.3: Policy and value networks over time. The blue and orange lines denote $\alpha(t)$ and $V_\phi(t)$, respectively, as a function of time t for $N = 2^2, 2^3, \ldots, 2^{13}$.

$\alpha(s_{t-\tau})$. In this case, the maximal net displacement (MND) policy [92], defined by

$$\alpha(s_{t-\tau}) = \Theta\left(\sum_{i=1}^{N} d(x_i(t-\tau))\right), \tag{4.8}$$

where the displacement function is $d(x) = x_{\min} + x_0 - x$ for $x_{\max} < x \leq x_{\max} + L$ and periodic $d(x) = d(x+L)$, can perform better than the greedy policy for $\tau > 0$ with optimal $x_0 < 0$ [93]. This can be considered as a τ-delayed greedy policy because it predicts the arrival of the particles at $x_{\min}$ after τ from $x_0 + x_{\min}$. We train the neural networks for $N = 1, 2^1, 2^2, ..., 2^5$ 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. 4.5(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 [117]. 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) [54], a widely used gating mechanism in RNNs due to its parameter efficiency and good performance on the sequential datasets, for the RNN. As shown in Fig. 4.5(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 Section 4.6.2 for the configuration details. As can be seen in Fig. 4.5(b), the trained RNN policies (red stars) show slightly better performance than the other policies 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.

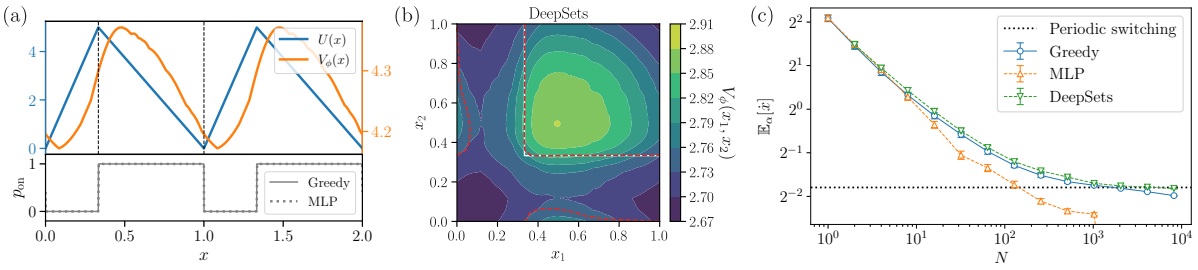

Figure 4.4: Training results for the sawtooth potential. (a) $N = 1$ case. Top: Sawtooth potential U (4.7) and trained value network V_ϕ as a function of position x are denoted by blue and orange lines, respectively. Bottom: The solid line denotes the probability of switching on the potential ($p_{\rm on}$) as a function of x for the greedy policy. The dotted line represents $p_{\rm on}$ of the trained MLP policy. (b) $N = 2$ case. The decision boundaries of the trained DeepSets policy and the greedy policy are shown with the red dashed and solid white contours, respectively. (c) The current $\mathbb{E}_\alpha[\dot{x}]$ as function of N for each policy.

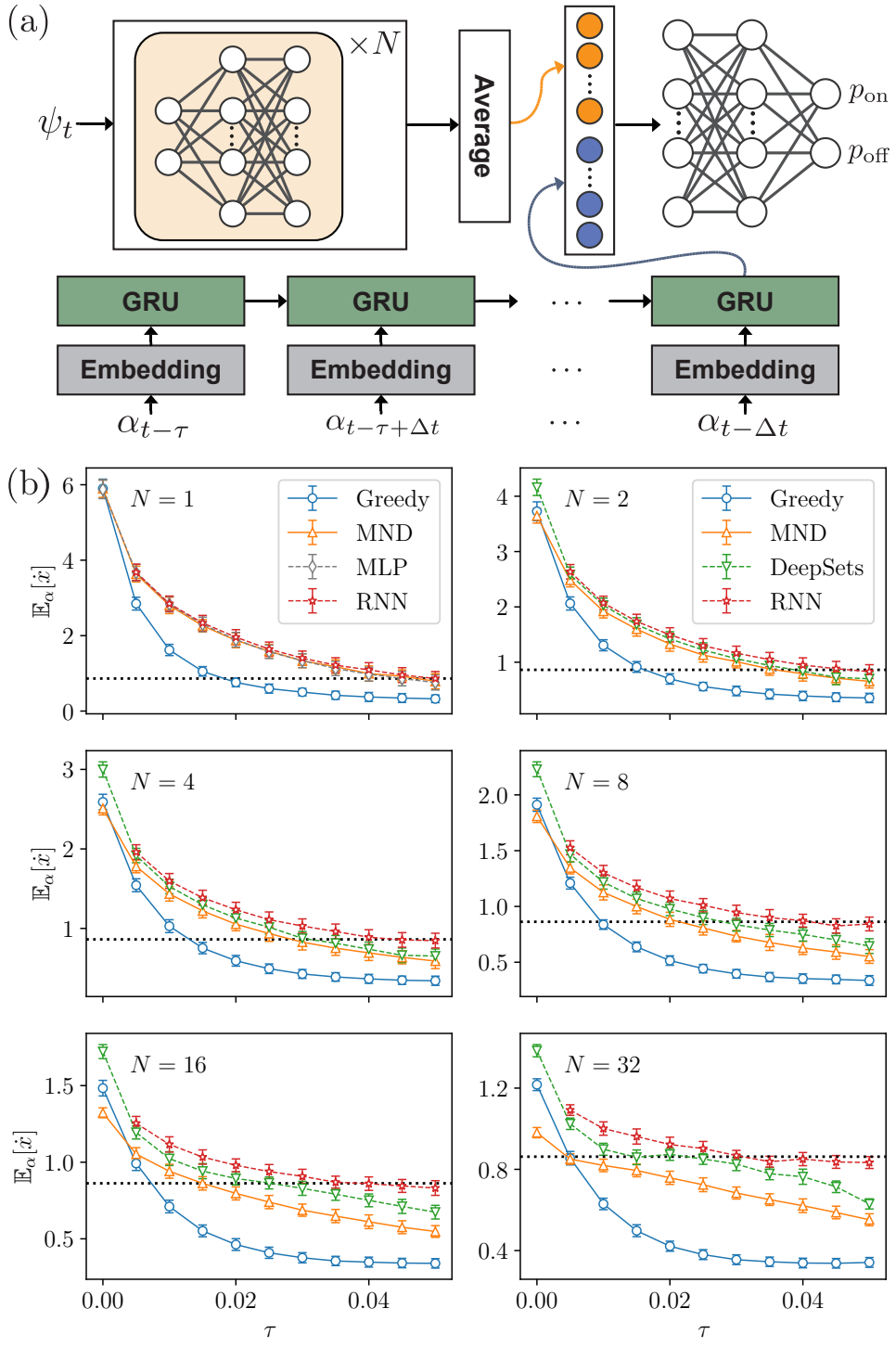

Figure 4.5: (a) Architecture of policy network π_θ augmented with an RNN. (b) Time-delayed feedback results for the greedy, MND, MLP (only for $N = 1$), DeepSets (for $N > 1$), and RNN policies at increasing N. The black dotted lines denote the current of the periodic switching policy.

4.5 Discussion

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 [94], 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 non-interacting particles, molecular motors like kinesin, for example, explicitly interact with each other via hard-core repulsion. According to previous studies on interacting molecular motors [98, 99, 100], 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 [118]. A ratchet-like mechanism for transportation in the cell membrane (such as ion pumping [96] or glycerol transportation [97]) 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 [119, 120, 121]. Such cases are not only important in physics, e.g. information thermodynamics [122], but also in RL for real-world applications [123]. 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` [56] are available in the GitHub repository[1].

[1] https://github.com/kdkyum/RatchetDRL

4.6 Supporting Information

4.6.1 Training details

We use the proximal policy optimization (PPO) [115] algorithm implemented in Ref. [114]. The PPO algorithm updates the parameter θ_i of policy network π_θ at epoch i by the following equations:

$$\theta_{i+1} = \arg\max_\theta L(\theta), \tag{4.9}$$

$$L(\theta) = \mathop{\mathbb{E}}_{(s,a)\sim\pi_{\theta_i}} \left[\min \left(\frac{\pi_\theta(s,a)}{\pi_{\theta_i}(s,a)} A^{\pi_{\theta_i}}(s,a),\ g(\epsilon, A^{\pi_{\theta_i}}(s,a)) \right) \right], \tag{4.10}$$

where $\pi_\theta(s,a)$ is the probability of choosing action a given the state s, and ϵ is a hyperparameter that restricts how much the new policy can be changed from the old policy. Function g is defined as

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & \text{if } A \geq 0 \\ (1-\epsilon)A & \text{if } A < 0. \end{cases} \tag{4.11}$$

Here, $A^\pi(s,a)$ is the advantage function that indicates how much better or worse the action a is than the other actions on average for the present policy π and given state s. In this section, t represents the time step, rather than time, for convenience. The definition of $A^\pi(s_t, a_t)$ is

$$A^\pi(s_t, a_t) = Q^\pi(s_t, a_t) - V^\pi(s_t), \tag{4.12}$$

where $Q^\pi(s_t, a_t) = \mathbb{E}_\pi[G_t | s_t, a_t]$ and $V^\pi(s_t) = \mathbb{E}_\pi[G_t | s_t]$ are the action-value function and the value function, respectively. The return G_t is defined as $G_t = \sum_{l=0}^\infty \gamma^l r_{t+l+1}$ where γ is the discounting factor. PPO uses the generalized advantage estimator (GAE) [124] $\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}$ for an accurate evaluation of the advantage function. $\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}$ is defined as

$$\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} = \sum_{l=0}^\infty (\gamma\lambda)^l \delta_{t+l}^V, \tag{4.13}$$

$$\delta_t^V = \underbrace{r_t + \gamma V^\pi(s_{t+1})}_{=Q^\pi(s_t,a_t)} - V^\pi(s_t), \tag{4.14}$$

where $\lambda \in (0, 1]$ is a hyperparameter for adjusting the bias-variance tradeoff in $\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}$. See Algorithm 4.1 for the detailed training procedure. The policy and value networks are initialized with the default random initialization setup in `PyTorch` [56]. Two Adam [58] optimizers are used respectively for the policy and value networks training. During the stochastic gradient ascent stage for the policy network (step 4 in Algorithm 4.1), the early stopping method is applied to prevent the updated policy from going too far from the old policy [114]. Early stopping means that if the Kullback–Leibler divergence (KLD) from the updated policy π_θ to the old policy π_{θ_i}

$$D_{\mathrm{KL}}(\pi_{\theta_i} \parallel \pi_\theta) = \mathop{\mathbb{E}}_{(s,a)\sim\pi_{\theta_i}} \left[\log \frac{\pi_{\theta_i}(s,a)}{\pi_\theta(s,a)} \right] \tag{4.15}$$

exceeds the threshold $1.5 \times d_{\mathrm{targ}}$, then the gradient steps stop. See Table 4.1 for the values of all the hyperparameters used in the training. For each N, we run Algorithm 4.1 five times independently with five different random seeds, and we pick the policy and value network that show the best performance among the five. All reported results in this paper are from the best-performing neural networks. Each run was conducted on a single NVIDIA TITAN V GPU.

Algorithm 4.1 Proximal policy optimization algorithm

Require: Environment, policy network π_θ, value network V_ϕ, optimizer for θ, optimizer for ϕ

1: **for** $i = 1, 2, \ldots, \mathcal{E}$ **do**

2: Run simulation of multiple trajectories simultaneously under the present policy network π_{θ_i} in the environment and collect the dataset $\mathcal{D}_i = \{\Gamma^{(m)}\}_{m=1}^{m=M}$ where M is the number of trajectories. Here, Γ denotes the trajectory of the state, action, and reward, and can be represented as $\Gamma = [s_1, a_1, r_1, s_2, a_2, r_2, s_3, \ldots, r_\mathrm{T}, s_{\mathrm{T}+1}]$ where T is the length of a single trajectory.

3: Compute estimated return

$$\hat{G}_t = \sum_{l=0}^{\mathrm{T}-t-1} \left(\gamma^l r_{t+l+1}\right) + \gamma^{\mathrm{T}-t} V^\pi(s_{\mathrm{T}+1}),$$

δ_t^V (4.14), and $\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}$ (4.13) for all time step t and trajectory Γ in $\mathcal{D}_i$ using the present value network $V^\pi = V_{\phi_i}$.

4: Update the policy network by maximizing the estimated $L(\theta)$ (4.10):

$$\theta_{i+1} = \arg\max_\theta \frac{1}{M\mathrm{T}} \sum_{\Gamma\in\mathcal{D}_i} \sum_{t=1}^{\mathrm{T}} \min\left(\frac{\pi_\theta(s_t,a_t)}{\pi_{\theta_i}(s_t,a_t)} \hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)},\ g(\epsilon, \hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)})\right),$$

via the optimizer for θ with a mini-batch size of $\mathcal{B}$.

5: Update the value network by minimizing the mean-squared error:

$$\phi_{i+1} = \arg\min_\phi \frac{1}{M\mathrm{T}} \sum_{\Gamma\in\mathcal{D}_i} \sum_{t=1}^{\mathrm{T}} \left(\hat{G}_t - V_\phi(s_t)\right)^2,$$

via the optimizer for ϕ with a mini-batch size of $\mathcal{B}$.

6: **end for**

Hyperparameter	Value
Trajectory length T	2000
Number of epochs $\mathcal{E}$	400
Discounting factor γ	0.999
GAE parameter λ	0.95
Clipping parameter ϵ	0.2
Target KLD for early stopping d_targ	0.01
Training iterations for π_θ per epoch	625
Training iterations for V_ϕ per epoch	625
Learning rate for π_θ	3×10^{-4}
Learning rate for V_ϕ	10^{-3}

N	M	$\mathcal{B}$
1	1024	4096
2	512	4096
4	256	4096
8	128	4096
16	64	2048
32	32	1024
64	16	512

N	M	$\mathcal{B}$
128	8	256
256	8	256
512	8	256
1024	8	256
2048	8	256
4096	8	256
8192	8	256

Table 4.1: Left: Hyperparameters. The hyperparameters not listed in this table are set as defaults in `PyTorch`. Right: The number of trajectories M and mini-batch size $\mathcal{B}$ for each number of particles N.

4.6.2 Architecture configurations

We use ReLU [57] as the activation function for the policy network π_θ and value network V_ϕ. See Tables 4.2, 4.3, and 4.4 for configuration details of the MLP, DeepSets, and RNN policy networks,

respectively. The policy network π_θ computes the on-off probabilities using the softmax function in the output layer. The value network V_ϕ has the same configuration except for having an output dimension of one rather than two. We set the number of hidden units to $H = 64$ and the embedding dimension to $E = 16$. Here, $\boldsymbol{\alpha}_t^d$ is the potential on-off history:

$$\boldsymbol{\alpha}_t^d = (\alpha_{t-\tau}, \alpha_{t-\tau+\Delta t}, \ldots, \alpha_{t-\tau+(d-1)\Delta t}), \quad d = \tau/\Delta t. \tag{4.16}$$

MLP policy network		
Layer	Output dim	Activation function
Input ψ_t	$2N$	
Fully-connected	H	ReLU
Fully-connected	H	ReLU
Output layer	2	None

Table 4.2: Two-hidden-layer MLP configuration.

DeepSets policy network		
Layer	Output dim	Activation function
Input ψ_t	$N \times 2$	
Fully-connected	$N \times H$	ReLU
Fully-connected	$N \times H$	None
Average	H	
Fully-connected	H	ReLU
Output layer	2	None

Table 4.3: DeepSets configuration.

RNN policy network			
Module	Layer	Output dim	Activation function
DeepSets(ψ_t)	Input ψ_t	$N \times 2$	
	Fully-connected	$N \times H$	ReLU
	Fully-connected	$N \times H$	None
	Average	H	
RNN($\boldsymbol{\alpha}_t^d$)	Embedding $\boldsymbol{\alpha}_t^d$	$d \times E$	
	GRU($\boldsymbol{\alpha}_t^d$) last output	$2E$	
MLP	Concatenate [DeepSets(ψ_t), RNN($\boldsymbol{\alpha}_t^d$)]	$H + 2E$	
	Fully-connected	H	ReLU
	Output layer	2	None

Table 4.4: RNN configuration.

Chapter 5. Conclusions

In this thesis, we have discussed how to study nonequilibrium processes through deep learning. Here I present a short summary of the results and conclusions of each chapter. Note that this thesis focused on two thermodynamic quantities, heat (Chapter 2 and 3) and work (Chapter 4). In addition, we are able to efficiently solve high-dimensional problems by using a deep learning framework that has recently made remarkable progress.

In Chapter 2, we developed the neural estimator for entropy production (NEEP), which estimates entropy production (EP) from trajectories of relevant variables without information on the underlying mechanism of systems. We rigorously proved that the estimator, which can be built up from different choices of deep neural networks, provides stochastic EP by optimizing the objective function we proposed. We also verified our approach via simulation on two widely studied nonequilibrium models, namely bead-spring and discrete flashing ratchet. Thanks to the unprecedented advances of deep learning, we showed that NEEP is applicable to high-dimensional state variables. In addition, we demonstrated that NEEP can provide coarse-grained EP for Markov systems with unobservable states, even when we can only access partial information, which is the usual situation in practice. It is expected to be beneficial in applications to intricately organized systems where numerous variables are entangled and hidden, such as biological systems, active matter, and others, for a deeper look into their high-dimensional fluctuating dynamics.

In Chapter 3, we resolved the difficulty of measuring the EP of a system with odd-parity variables via multiple neural networks, which represent a generalized version of NEEP. With only the state variable's parity information, our method can learn the exact EP from trajectory data. Owing to our method, the EP of a system with mass can be estimated. We demonstrated our method by using a system with inertia and an active system, namely an underdamped bead-spring model and a one-particle odd-parity Markov jump process. We expect that this method will be a practical tool for studying nonequilibrium systems in general experimental situations.

In Chapter 4, we employed deep reinforcement learning (RL) to find the optimal policy for maximizing the current in the collective flashing ratchet. Our results show that the policies discovered by deep RL not only agree with the previously known optimal solutions for particle number $N = 1$ and $N = 2$ case, but also outperform the previous policies at $N > 2$. As a realistic situation, we considered time-delayed feedback, where the on-off switching of the potential is delayed due to the calculation of the feedback algorithm. Even in such cases with time-delays, the deep RL shows better performance than the previous policies. Therefore, we expect this study to be effectively applied to experiments involving the inevitable time-delays.

In short, we showed that deep learning can address complex problems in nonequilibrium physics. Using deep learning, we can measure heat from high-dimensional trajectory data and figure out what mechanism can maximize work in the collective flashing ratchet. I hope the presented studies in this thesis will be practical tools for understanding complex nonequilibrium systems, which prevail in nature.

Bibliography

[1] Seifert, U. Stochastic thermodynamics, fluctuation theorems and molecular machines. *Rep. Prog. Phys.* **75**, 126001 (2012).

[2] Seifert, U. Entropy Production along a Stochastic Trajectory and an Integral Fluctuation Theorem. *Phys. Rev. Lett.* **95**, 040602 (2005).

[3] Sekimoto, K. Stochastic energetics. (Springer-Verlag, 2010).

[4] Martin, P., Hudspeth, A. & Jülicher, F. Comparison of a hair bundle's spontaneous oscillations with its response to mechanical stimulation reveals the underlying active process. *Proc. Natl. Acad. Sci. U.S.A.* **98**, 14380-14385 (2001).

[5] Ben-Isaac, E., Park, Y., Popescu, G., Brown, F., Gov, N. & Shokef, Y. Effective Temperature of Red-Blood-Cell Membrane Fluctuations. *Phys. Rev. Lett.* **106**, 238103 (2011).

[6] Weber, S., Spakowitz, A. & Theriot, J. Nonthermal ATP-dependent fluctuations contribute to the in vivo motion of chromosomal loci. *Proc. Natl. Acad. Sci. U.S.A.* **109**, 7338-7343 (2012).

[7] Battle, C., Ott, C., Burnette, D., Lippincott-Schwartz, J. & Schmidt, C. Intracellular and extracellular forces drive primary cilia movement. *Proc. Natl. Acad. Sci. U.S.A.* **112**, 1410-1415 (2015).

[8] Battle, C., Broedersz, C., Fakhri, N., Geyer, V., Howard, J., Schmidt, C. & MacKintosh, F. Broken detailed balance at mesoscopic scales in active biological systems. *Science* **352**, 604-607 (2016).

[9] Gladrow, J., Fakhri, N., MacKintosh, F., Schmidt, C. & Broedersz, C. Broken Detailed Balance of Filament Dynamics in Active Networks. *Phys. Rev. Lett.* **116**, 248301 (2016).

[10] Harada, T. & Sasa, S. Equality Connecting Energy Dissipation with a Violation of the Fluctuation-Response Relation. *Phys. Rev. Lett.* **95**, 130602 (2005).

[11] Toyabe, S., Jiang, H., Nakamura, T., Murayama, Y. & Sano, M. Experimental test of a new equality: Measuring heat dissipation in an optically driven colloidal system. *Phys. Rev. E* **75**, 011122 (2007).

[12] Lander, B., Mehl, J., Blickle, V., Bechinger, C. & Seifert, U. Noninvasive measurement of dissipation in colloidal systems. *Phys. Rev. E* **86**, 030401(R) (2012).

[13] Gnesotto, F., Mura, F., Gladrow, J. & Broedersz, C. Broken detailed balance and non-equilibrium dynamics in living systems: a review. *Rep. Prog. Phys.* **81**, 066601 (2018).

[14] Li, J., Horowitz, J., Gingrich, T. & Fakhri, N. Quantifying dissipation using fluctuating currents. *Nat. Commun.* **10**, 1666 (2019).

[15] Wang, Q., Kulkarni, S. & Verdú, S. Divergence estimation of continuous distributions based on data-dependent partitions. *IEEE Trans. Inf. Theory* **51**, 3064-3074 (2005).

[16] Roldán, É. & Parrondo, J. Estimating Dissipation from Single Stationary Trajectories. *Phys. Rev. Lett.* **105**, 150607 (2010).

[17] Roldán, É. & Parrondo, J. Entropy production and Kullback-Leibler divergence between stationary trajectories of discrete systems. *Phys. Rev. E* **85**, 031129 (2012).

[18] Ziv, J. & Merhav, N. A measure of relative entropy between individual sequences with application to universal classification. *IEEE Trans. Inf. Theory* **39**, 1270-1279 (1993).

[19] Avinery, R., Kornreich, M. & Beck, R. Universal and Accessible Entropy Estimation Using a Compression Algorithm. *Phys. Rev. Lett.* **123**, 178102 (2019).

[20] Martiniani, S., Chaikin, P. & Levine, D. Quantifying Hidden Order out of Equilibrium. *Phys. Rev. X* **9**, 011031 (2019).

[21] Barato, A. & Seifert, U. Thermodynamic Uncertainty Relation for Biomolecular Processes. *Phys. Rev. Lett.* **114**, 158101 (2015).

[22] Manikandan, S., Gupta, D. & Krishnamurthy, S. Inferring Entropy Production from Short Experiments. *Phys. Rev. Lett.* **124**, 120603 (2020).

[23] Van Vu, T., Vo, V. & Hasegawa, Y. Entropy production estimation with optimal current. *Phys. Rev. E* **101**, 042138 (2020).

[24] Otsubo, S., Ito, S., Dechant, A. & Sagawa, T. Estimating entropy production by machine learning of short-time fluctuating currents. *Phys. Rev. E* **101**, 062106 (2020).

[25] Mehta, P. & Schwab, D. An exact mapping between the variational renormalization group and deep learning. *arXiv preprint arXiv:1410.3831*.

[26] Koch-Janusz, M. & Ringel, Z. Mutual information, neural networks and the renormalization group. *Nat. Phys.* **14**, 578-582 (2018).

[27] Li, S. & Wang, L. Neural Network Renormalization Group. *Phys. Rev. Lett.* **121**, 260601 (2018).

[28] Carrasquilla, J. & Melko, R. Machine learning phases of matter. *Nat. Phys.* **13**, 431-434 (2017).

[29] Nieuwenburg, E., Liu, Y. & Huber, S. Learning phase transitions by confusion. *Nat. Phys.* **13**, 435-439 (2017).

[30] Venderley, J., Khemani, V. & Kim, E. Machine Learning Out-of-Equilibrium Phases of Matter. *Phys. Rev. Lett.* **120**, 257204 (2018).

[31] Liu, Y. & Nieuwenburg, E. Discriminative Cooperative Networks for Detecting Phase Transitions. *Phys. Rev. Lett.* **120**, 176401 (2018).

[32] Beach, M., Golubeva, A. & Melko, R. Machine learning vortices at the Kosterlitz-Thouless transition. *Phys. Rev. B* **97**, 045207 (2018).

[33] Zhang, P., Shen, H. & Zhai, H. Machine Learning Topological Invariants with Neural Networks. *Phys. Rev. Lett.* **120**, 066401 (2018).

[34] Carleo, G. & Troyer, M. Solving the quantum many-body problem with artificial neural networks. *Science* **355**, 602-606 (2017).

[35] Deng, D., Li, X. & Das Sarma, S. Quantum Entanglement in Neural Network States. *Phys. Rev. X* **7**, 021021 (2017).

[36] Ch'ng, K., Carrasquilla, J., Melko, R. & Khatami, E. Machine Learning Phases of Strongly Correlated Fermions. *Phys. Rev. X* **7**, 031038 (2017).

[37] Choo, K., Carleo, G., Regnault, N. & Neupert, T. Symmetries and Many-Body Excitations with Neural-Network Quantum States. *Phys. Rev. Lett.* **121**, 167204 (2018).

[38] Torlai, G., Mazzola, G., Carrasquilla, J., Troyer, M., Melko, R. & Carleo, G. Neural-network quantum state tomography. *Nat. Phys.* **14**, 447-450 (2018).

[39] Hartmann, M. & Carleo, G. Neural-Network Approach to Dissipative Quantum Many-Body Dynamics. *Phys. Rev. Lett.* **122**, 250502 (2019).

[40] Nagy, A. & Savona, V. Variational Quantum Monte Carlo Method with a Neural-Network Ansatz for Open Quantum Systems. *Phys. Rev. Lett.* **122**, 250501 (2019).

[41] Vicentini, F., Biella, A., Regnault, N. & Ciuti, C. Variational Neural-Network Ansatz for Steady States in Open Quantum Systems. *Phys. Rev. Lett.* **122**, 250503 (2019).

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

[43] Gnesotto, F., Gradziuk, G., Ronceray, P. & Broedersz, C. Learning the Non-Equilibrium Dynamics of Brownian Movies. *Nat. Commun.* **11**, 5378 (2020).

[44] Seif, A., Hafezi, M. & Jarzynski, C. Machine learning the thermodynamic arrow of time. *Nat. Phys.* **17**, 105–113 (2021).

[45] Rahaman, N., Wolf, S., Goyal, A., Remme, R. & Bengio, Y. Learning the Arrow of Time for Problems in Reinforcement Learning. *International Conference on Learning Representations* (2020).

[46] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature (London)* **521**, 436-444 (2015).

[47] Goodfellow, I., Bengio, Y. & Courville, A. Deep Learning. (MIT press, 2016).

[48] Mura, F., Gradziuk, G. & Broedersz, C. Nonequilibrium Scaling Behavior in Driven Soft Biological Assemblies. *Phys. Rev. Lett.* **121**, 038002 (2018).

[49] Ajdari, A. & Prost, J. Mouvement induit par un potentiel périodique de basse symétrie: diélectrophorese pulsée. *Comptes Rendus De L'Académie Des Sciences. Série 2, Mécanique, Physique, Chimie, Sciences De L'univers, Sciences De La Terre* **315**, 1635-1639 (1992).

[50] Gomez-Marin, A., Parrondo, J. & Broeck, C. Lower bounds on dissipation upon coarse graining. *Phys. Rev. E* **78**, 011107 (2008).

[51] Esposito, M. Stochastic thermodynamics under coarse graining. *Phys. Rev. E* **85**, 041125 (2012).

[52] Polettini, M. & Esposito, M. Effective Thermodynamics for a Marginal Observer. *Phys. Rev. Lett.* **119**, 240601 (2017).

[53] Dabelow, L., Bo, S. & Eichhorn, R. Irreversibility in Active Matter Systems: Fluctuation Theorem and Mutual Information. *Phys. Rev. X* **9**, 021009 (2019).

[54] Cho, K., Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings Of The 2014 Conference On Empirical Methods In Natural Language Processing (EMNLP)* pp. 1724-1734 (2014).

[55] Martınez, I., Bisker, G., Horowitz, J. & Parrondo, J. Inferring broken detailed balance in the absence of observable currents. *Nat. Commun.* **10**, 3542 (2019).

[56] Paszke, A. *et al.* PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances In Neural Information Processing Systems 32* pp. 8024-8035 (2019).

[57] Nair, V. & Hinton, G. Rectified linear units improve restricted boltzmann machines. *Proceedings Of The 27th International Conference On Machine Learning* pp. 807-814 (2010).

[58] Kingma, D. & Ba, J. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*.

[59] Collin, D., Ritort, F., Jarzynski, C., Smith, S., Tinoco, I. & Bustamante, C. Verification of the Crooks fluctuation theorem and recovery of RNA folding free energies. *Nature* **437**, 231-234 (2005).

[60] Proesmans, K. & Broeck, C. Discrete-time thermodynamic uncertainty relation. *EPL (Europhysics Letters)* **119**, 20001 (2017).

[61] Hasegawa, Y. & Van Vu, T. Fluctuation Theorem Uncertainty Relation. *Phys. Rev. Lett.* **123**, 110602 (2019).

[62] Koyuk, T. & Seifert, U. Thermodynamic Uncertainty Relation for Time-Dependent Driving. *Phys. Rev. Lett.* **125**, 260604 (2020).

[63] Gingrich, T., Horowitz, J., Perunov, N. & England, J. Dissipation Bounds All Steady-State Current Fluctuations. *Phys. Rev. Lett.* **116**, 120601 (2016).

[64] Dechant, A. & Sasa, S. Improving thermodynamic bounds using correlations. *arXiv preprint arXiv:2104.04169*.

[65] Kim, D.-K., Bae, Y., Lee, S. & Jeong, H. Learning Entropy Production via Neural Networks. *Phys. Rev. Lett.* **125**, 140604 (2020).

[66] Otsubo, S., Manikandan, S., Sagawa, T. & Krishnamurthy, S. Estimating time-dependent entropy production from non-equilibrium trajectories. *arXiv preprint arXiv:2010.03852*.

[67] Bae, Y., Kim, D.-K. & Jeong, H. Attaining entropy production and dissipation maps from Brownian movies via neural networks. *arXiv preprint arXiv:2106.15108*.

[68] Donsker, M. & Varadhan, S. Asymptotic evaluation of certain markov process expectations for large time. IV. *Commun. Pure Appl. Math.* **36**, 183 (2020).

[69] Spinney, R. & Ford, I. Nonequilibrium Thermodynamics of Stochastic Systems with Odd and Even Variables. *Phys. Rev. Lett.* **108**, 170603 (2012).

[70] Lee, H., Kwon, C. & Park, H. Fluctuation Theorems and Entropy Production with Odd-Parity Variables. *Phys. Rev. Lett.* **110**, 050602 (2013).

[71] Lee, J., Park, J. & Park, H. Universal form of thermodynamic uncertainty relation for Langevin dynamics. *Phys. Rev. E* **104**, L052102 (2021).

[72] Risken, H. The Fokker-Planck Equation: Methods of Solutions and Applications. (Springer-Verlag, Berlin, 1991).

[73] Shankar, S. & Marchetti, M. Hidden entropy production and work fluctuations in an ideal active gas. *Phys. Rev. E* **98**, 020604 (2018).

[74] Van Vu, T. & Hasegawa, Y. Uncertainty relations for underdamped Langevin dynamics. *Phys. Rev. E* **100**, 032130 (2019).

[75] Lee, J., Park, J. & Park, H. Thermodynamic uncertainty relation for underdamped Langevin systems driven by a velocity-dependent force. *Phys. Rev. E* **100**, 062132 (2019).

[76] Vanden-Eijnden, E. & Ciccotti, G. Second-order integrators for Langevin equations with holonomic constraints. *Chemical Physics Letters* **429**, 310-316 (2006).

[77] Saxe, A., McClelland, J. & Ganguli, S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations* (2014).

[78] Gillespie, D. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem.* **81**, 2340-2361 (1977).

[79] Skinner, D. & Dunkel, J. Estimating Entropy Production from Waiting Time Distributions. *Phys. Rev. Lett.* **127**, 198101 (2021).

[80] Kawaguchi, K. & Nakayama, Y. Fluctuation theorem for hidden entropy production. *Phys. Rev. E* **88**, 022147 (2013).

[81] Esposito, M. & Parrondo, J. Stochastic thermodynamics of hidden pumps. *Phys. Rev. E* **91**, 052114 (2015).

[82] Prost, J., Chauwin, J., Peliti, L. & Ajdari, A. Asymmetric pumping of particles. *Phys. Rev. Lett.* **72**, 2652-2655 (1994).

[83] Astumian, R. & Bier, M. Fluctuation driven ratchets: Molecular motors. *Phys. Rev. Lett.* **72**, 1766-1769 (1994).

[84] Astumian, R. Thermodynamics and Kinetics of a Brownian Motor. *Science* **276**, 917-922 (1997).

[85] Tarlie, M. & Astumian, R. Optimal modulation of a Brownian ratchet and enhanced sensitivity to a weak external force. *Proc. Natl. Acad. Sci. U.S.A.* **95**, 2039-2043 (1998).

[86] Cao, F., Dinis, L. & Parrondo, J. Feedback Control in a Collective Flashing Ratchet. *Phys. Rev. Lett.* **93**, 040603 (2004).

[87] Dinis, L., Parrondo, J. & Cao, F. Closed-loop control strategy with improved current for a flashing ratchet. *Europhys. Lett.* **71**, 536-541 (2005).

[88] Feito, M. & Cao, F. Threshold feedback control for a collective flashing ratchet: Threshold dependence. *Phys. Rev. E* **74**, 041109 (2006).

[89] Feito, M. & Cao, F. Optimal operation of feedback flashing ratchets. *J. Stat. Mech.* **2009**, P01031 (2009).

[90] Feito, M. & Cao, F. Time-delayed feedback control of a flashing ratchet. *Phys. Rev. E* **76**, 061113 (2007).

[91] Craig, E., Long, B., Parrondo, J. & Linke, H. Effect of time delay on feedback control of a flashing ratchet. *Europhys. Lett.* **81**, 10002 (2007).

[92] Craig, E., Kuwada, N., Lopez, B. & Linke, H. Feedback control in flashing ratchets. *Ann. Phys.* **17**, 115-129 (2008).

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

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

[95] Reimann, P. Brownian motors: noisy transport far from equilibrium. *Physics Reports* **361**, 57-265 (2002).

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

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

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

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

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

[101] Hwang, W. & Karplus, M. Structural basis for power stroke vs. Brownian ratchet mechanisms of motor proteins. *Proc. Natl. Acad. Sci. U.S.A.* **116**, 19777-19785 (2019).

[102] Bapst, V. *et al.* Unveiling the predictive power of static structure in glassy systems. *Nat. Phys.* **16**, 448-454 (2020).

[103] Carrasquilla, J. Machine learning for quantum matter. *Advances In Physics: X* **5**, 1797528 (2020).

[104] Sutton, R. & Barto, A. Reinforcement learning: An introduction. (MIT Press, 2018).

[105] Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature (London)* **518**, 529-533 (2015).

[106] Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature (London)* **529**, 484-489 (2016).

[107] Silver, D., *et al.* A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **362**, 1140-1144 (2018).

[108] Vinyals, O. *et al.* Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature (London)* **575**, 350-354 (2019).

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

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

[111] Niu, M., Boixo, S., Smelyanskiy, V. & Neven, H. Universal quantum control through deep reinforcement learning. *Npj Quantum Inf.* **5**, 33 (2019).

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

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

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

[115] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[116] Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R. & Smola, A. Deep Sets. *Advances In Neural Information Processing Systems 30* pp. 3391-3401 (2017).

[117] Katsikopoulos, K. & Engelbrecht, S. Markov decision processes with delays and asynchronous cost collection. *IEEE Trans. Automat. Contr.* **48**, 568-574 (2003).

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

[119] Cao, F., Feito, M. & Touchette, H. Information and flux in a feedback controlled Brownian ratchet. *Physica A* **388**, 113-119 (2009).

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

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

[122] Parrondo, J., Horowitz, J. & Sagawa, T. Thermodynamics of information. *Nat. Phys.* **11**, 131-139 (2015).

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

[124] Schulman, J., Moritz, P., Levine, S., Jordan, M. & Abbeel, P. High-Dimensional Continuous Control Using Generalized Advantage Estimation. *International Conference on Learning Representations* (2016).

Acknowledgments in Korean

본 학위논문의 연구는 저의 긴 학위 과정이라는 여정 동안 저와 함께해주신 많은 분의 도움으로 완성되었기에 이 자리를 빌려 감사의 말씀을 드립니다.

아마도 제 학위 과정의 시작은 제가 KAIST 대학원 물리학과에 입학한 2016년도에 있었던 딥마인드의 알파고와 이세돌의 대결이었을 것입니다. 이 대결이 있기 전까지 제가 박사과정 동안 무엇을 연구하게 될지 전혀 감도 없었습니다. 이 대결에서 알파고가 이기고 난 후, 저의 지도교수님께서 인공지능과 이를 이용한 여러 복잡계를 연구해보자는 제안을 해주셨고, 이때부터 저의 여정이 시작되었습니다. 따라서 이렇게 연구를 시작할 수 있게 기회를 주시고, 온갖 최신 인공지능 지식을 배울 수 있는 환경을 제공해 주신 저의 지도교수님 정하웅 교수님께 깊은 감사를 전합니다.

석사·박사통합과정 2년 차가 될 무렵 저에게 데이터 과학과 머신러닝을 삼성전자 반도체 공장 현장에서 배울 수 있는 인턴 경험을 제공해준 연구실 선배 다니엘 김 박사님께도 감사드립니다. 같이 삼성전자에서 일할 때 다니엘 박사님이 보여준 일에 대한 열정과 집중력은 제가 연구자로서 가져야 할 마음가짐의 본이 되어주었습니다. 또한 박사과정 내내 언제나 연구의 기술적 문제 및 커리어에 대한 문제도 상담해주셔서 감사합니다.

대학원 시절 가족처럼 오랜 시간 함께한 복잡계 및 통계물리 연구실의 선후배님들에게도 감사드립니다. 이병휘 박사님과 함께 연구하면서 글쓰기, 발표, 연구 방법 등 여러 지식을 배울 수 있었고, 덕분에 미국 벌링턴에서 열린 NetSci 학회에도 참가할 기회를 얻을 수 있었습니다. 또한 배영경 학생과 이상윤 박사님과 함께 작업하면서 비평형 통계물리에 대해 정말로 많은 것들을 배울 수 있었습니다. 서로 연구 주제가 다름에도 매번 미팅 때마다 열정과 호기심으로 질문과 토의를 함께 나눈 정기홍 박사님, 소형준 박사님, 김혜원 박사님, 하미순 교수님, 김판준 교수님, 김승현 박사님, 하승웅, 김광수, 손강민, 허재석 후배님, 그리고 송영우 인턴님께도 감사를 드립니다. 돌이켜보면 저널 미팅 때마다 서로 다른 분야의 논문들을 소개하는 환경이었기에 정말 다양한 지식을 접할 수 있었고, 이를 통해 참신하고 재미있는 연구를 할 수 있었습니다. 그리고 짧게나마 연구실에 방문하신 윤혜진 교수님, 엄영호 교수님, 손승우 교수님, 안용열 교수님께서 주신 아낌없는 조언에 감사를 드립니다. 연구실 가족분들과 함께한 미팅, 학회, 스쿨, 홈커밍 등등 매 순간 즐거웠고 행복한 시간이었습니다.

바쁘신 와중에 이 학위 논문을 심사해준 KAIST 한명준 교수님, 양용수 교수님, 서울대 조정효 교수님, 백용주 교수님께도 감사드립니다. 심사에서 보여주신 깊이 있는 관심과 질문 덕에 기존에 했던 연구와 앞으로 할 연구에 대해 다양한 관점을 가지게 되었습니다.

마지막으로 제가 대전에서 박사과정을 하는 동안 저 멀리 일산에서 한결같은 사랑으로 묵묵히 격려해주시고 응원해주신 저의 부모님 그리고 누나한테도 진심으로 감사의 마음을 전합니다.

Curriculum Vitae in Korean

이　　름: 김 동 겸

학　　력

2009. 3. – 2011. 2.　대전과학고등학교

2011. 3. – 2015. 2.　서울대학교 물리천문학부 (학사)

2016. 3. –　한국과학기술원 물리학과 (석박통합과정)

경　　력

2016. 3. – 2017. 2.　한국과학기술원 물리학과 일반물리학 조교

2017. 9. – 2017. 12.　삼성전자 반도체 딥러닝 기반 분석 인프라 구축

2021. 8.　한국과학기술원 물리학과 박사전 연구원생

연 구 업 적

1. **Dong-Kyum Kim**, Byunghwee Lee, Daniel Kim, and Hawoong Jeong, *Multi-Label Classification of Historical Documents by Using Hierarchical Attention Networks*, J. Korean Phys. Soc. (2020).
2. **Dong-Kyum Kim**, Youngkyoung Bae, Sangyun Lee, and Hawoong Jeong, *Learning Entropy Production via Neural Networks*, Phys. Rev. Lett. (2020).
3. **Dong-Kyum Kim** and Hawoong Jeong, *Deep reinforcement learning for feedback control in a collective flashing ratchet*, Phys. Rev. Research (2021).
4. Youngkyoung Bae, **Dong-Kyum Kim**, and Hawoong Jeong, *Attaining entropy production and dissipation maps from Brownian movies via neural networks*, under review.
5. Gwangsu Kim, **Dong-Kyum Kim**, and Hawoong Jeong, *Spontaneous emergence of music-selectivity in a deep neural network trained for natural sound detection*, under review.
6. **Dong-Kyum Kim**, Sangyun Lee, and Hawoong Jeong, *Estimating entropy production in a stochastic system with odd-parity variables*, under review.

학 회 활 동

1. (국내, 구두) **Dong-Kyum Kim** and Hawoong Jeong, *Exploring optimal mechanisms in active Brownian particles via deep reinforcement learning*, APCTP Workshop for Physics and Machine Learning, Jeju, Korea, Nov. 26, 2021.

2. (국내, 구두) **Dong-Kyum Kim**, Sangyun Lee, and Hawoong Jeong, *Neural estimator for entropy production in stochastic systems with odd-parity variables*, The 21st Statistical Physics Workshop, Korea (Online), Aug. 25-27, 2021.
3. (국내, 구두) **Dong-Kyum Kim** and Hawoong Jeong, *Deep reinforcement learning for feedback-controlled flashing ratchets*, Korean Physical Society Fall Meeting, Korea (Online), Nov 6, 2020.
4. (국제, 구두) **Dong-Kyum Kim** and Hawoong Jeong, *Discovering wiring patterns of neural networks via backboning*, NetSci2019, University of Vermont, Rome, Italy (Online), Sep. 22, 2020.
5. (국내, 구두) **Dong-Kyum Kim**, Youngkyoung Bae, Sangyun Lee, and Hawoong Jeong, *Neural estimator for entropy production*, Korean Physical Society Spring Meeting, Korea (Online), Jul. 13-15, 2020.
6. (국내, 포스터) **Dong-Kyum Kim**, Byunghwee Lee, Donghyeok Choi, Juyong Park, and Hawoong Jeong, *Quantitative Analysis on the Success of Individual Career in the Joseon Dynasty and Modelling the Effect of Social Class on Career Dynamics*, Korean Physical Society Fall Meeting, Gwangju, Korea, Oct. 23-25, 2019.
7. (국내, 구두) **Dong-Kyum Kim** and Hawoong Jeong, *On the connectivity pattern of wired neural networks discovered by pruning methods*, The 20st Statistical Physics Workshop, Byeonsan, Korea, Aug. 21-23, 2019.
8. (국제, 구두) **Dong-Kyum Kim**, Byunghwee Lee, and Hawoong Jeong, *Quantifying Individual Reputation in Large-scale Historical Documents*, NetSci2019, University of Vermont, Burlington, USA, May. 27-31, 2019.