Journal of Applied Physics

Reinforcement learning approach to thermal transparency with particles in periodic lattices

Cite as: J. Appl. Phys. **130**, 045103 (2021); https://doi.org/10.1063/5.0054023
Submitted: 13 April 2021 . Accepted: 09 July 2021 . Published Online: 23 July 2021

Bin Liu, Liujun Xu, and Jiping Huang

View Online Export Citation CrossMark

Reinforcement learning approach to thermal transparency with particles in periodic lattices

Cite as: J. Appl. Phys. **130**, 045103 (2021); doi: 10.1063/5.0054023
Submitted: 13 April 2021 · Accepted: 9 July 2021 ·
Published Online: 23 July 2021

Bin Liu,[1,a)] Liujun Xu,[2] and Jiping Huang[2]

AFFILIATIONS

[1]School of Electronics and Materials Engineering, LeShan Normal University, 614099 LeShan, China
[2]Department of Physics, State Key Laboratory of Surface Physics, and Key Laboratory of Micro and Nano Photonic Structures (MOE), Fudan University, 200438 Shanghai, China

[a)]**Author to whom correspondence should be addressed:** liubin@fudan.edu.cn

ABSTRACT

Implementing thermal transparency by using thermal metamaterials, with its potential applications in real-world scenarios, has been a promising field attracting many theoretical and experimental studies. The implementation of thermal transparency, as well as other thermal metamaterial-based applications, often requires solving an inverse design problem to calculate optimal design parameters. In this paper, we propose a periodic interparticle interaction mechanism to realize thermal transparency, in which particles are arranged in periodic lattices with symmetric interactions and anisotropic thermal conductivities. We reframe the inverse design problem of calculating the design parameters of such a periodic interparticle system into a reinforcement learning problem. The essence of our reinforcement learning-based approach is to train an intelligent agent that can vary the design parameters in a series of time steps toward the realization of thermal transparency. Compared to our previous effort to solve the same problem with an autoencoder-based approach, the reinforcement learning-based approach requires significantly less computational resources and thus demonstrates its potential to alleviate the "curse of dimensionality." We also discuss the cause for the superior computational efficiency of the reinforcement learning-based approach over the autoencoder-based approach, and the possibility of extending the use of our reinforcement learning-based approach to solve other inverse design problems.

I. INTRODUCTION

Research and development in the field of thermal metamaterials has contributed to a number of inventions and new applications to manipulate heat transfer behaviors in an exotic way, such as thermal cloaks,[1–10] thermal concentrators,[3,11,12] thermal transparency,[13–16] and thermal camouflage.[17–24] Thermal cloaks feature zero temperature gradients inside them and unaffected thermal profiles outside them, so they can protect any internal object from infrared detection. Thermal concentrators can increase internal temperature gradients but do not distort external temperature profiles, so they have broad applications for improving the efficiency of thermoelectric conversion. As for thermal camouflage, since objects always emit thermal radiation, infrared detection becomes a powerful tool to get information, especially in dark regions. In contrast, thermal camouflage aims to mislead infrared detection. In general, an object has a unique thermal profile. With an artificially designed device tuning the thermal profile, the object can become another one in an infrared camera. Therefore, the actual information cannot be obtained by infrared detection, and thermal camouflage is achieved. Thermal transparency, which attracts our particular interest, aims to give an artificially fabricated composite material or structure effective thermal conductivity equal to that of the surrounding background. In recent years, the rapid adoption of infrared cameras in both academia and industry has made experimental measurements of the effectiveness of thermal transparency readily available, which also facilitates our previous work.[25]

However, the reliance on asymmetric interactions between the background and the fabricated device is a constraint hindering the broader adoption of thermal metamaterials to manipulate heat transfer. Here, the background denotes an area with uniform physical properties or microstructures, excluding the area occupied by the device.

To avoid the issue caused by asymmetric interactions, we carried out a series of works[25,26] that utilize symmetric interactions between periodic particles to realize thermal transparency. First, we propose a mechanism of using a periodic interparticle system (PIS) as the underlying solution to realize thermal transparency.[25] The PIS is illustrated in Fig. 1. In a PIS, two equal numbered types of particles are present. Type A particles have a circular shape and anisotropic thermal conductivities, while type B particles have an elliptical shape and isotropic thermal conductivities. The two types of particles are arranged on a periodic lattice with respect to the background. The thermal conductivities and relative periodic positioning of type A and type B particles offer degrees of freedom one can manipulate to cancel the effect of the presence of these particles on the heat transfer behavior of the background. Initially, for the simplicity of mathematical derivation, we confine the two types of particles on a square lattice.[25] Then, we relax this constraint on the shape of the lattice to a rectangular one.[26] We also remove the constraint that particles must be alternately positioned along both the horizontal and vertical direction.[26] However, with these constraints removed, one can no longer find a closed form solution to the problem. We have to resort to a completely different approach to determine the design parameters required to realize thermal transparency. Here, design parameters refer to the parameters describing the thermal conductivity and positioning of type A and B particles to form the lattice. Then, realizing that thermal transparency can be converted into solving for the inverse design problem, finding a set or multiple sets of design parameters so that the system(s) with this set(s) of design parameters will produce the desired or designated heat transfer behavior.

To solve for the inverse design problems to accomplish thermal transparency without these two aforementioned constraints, we resort to a machine learning-based approach[26] which has demonstrated its efficacy in tackling inverse design problems.[27–45] The essence of this approach lies in applying autoencoder-based dimensionality reduction to both the design space and the response space.[45] Once the design space is converted into the reduced design space, and the response space is converted into the reduced response space by applying dimensionality reduction, a feed-forward neural network (NN) can be trained to map the reduced response space to the reduced design space, which, in turn, can be restored to the original design space. We show that our approach can successfully solve for the inverse design problem of utilizing a PIS to realize thermal transparency.[26]

However, once we start extending our NN-based approach to solve more challenging problems, such as thermal camouflage, a new issue arises. A training set comprised of systems with randomly generated design parameters must be generated for NN training in our approach, and the random sampling must be dense enough on the dimension of each design parameter to ensure the

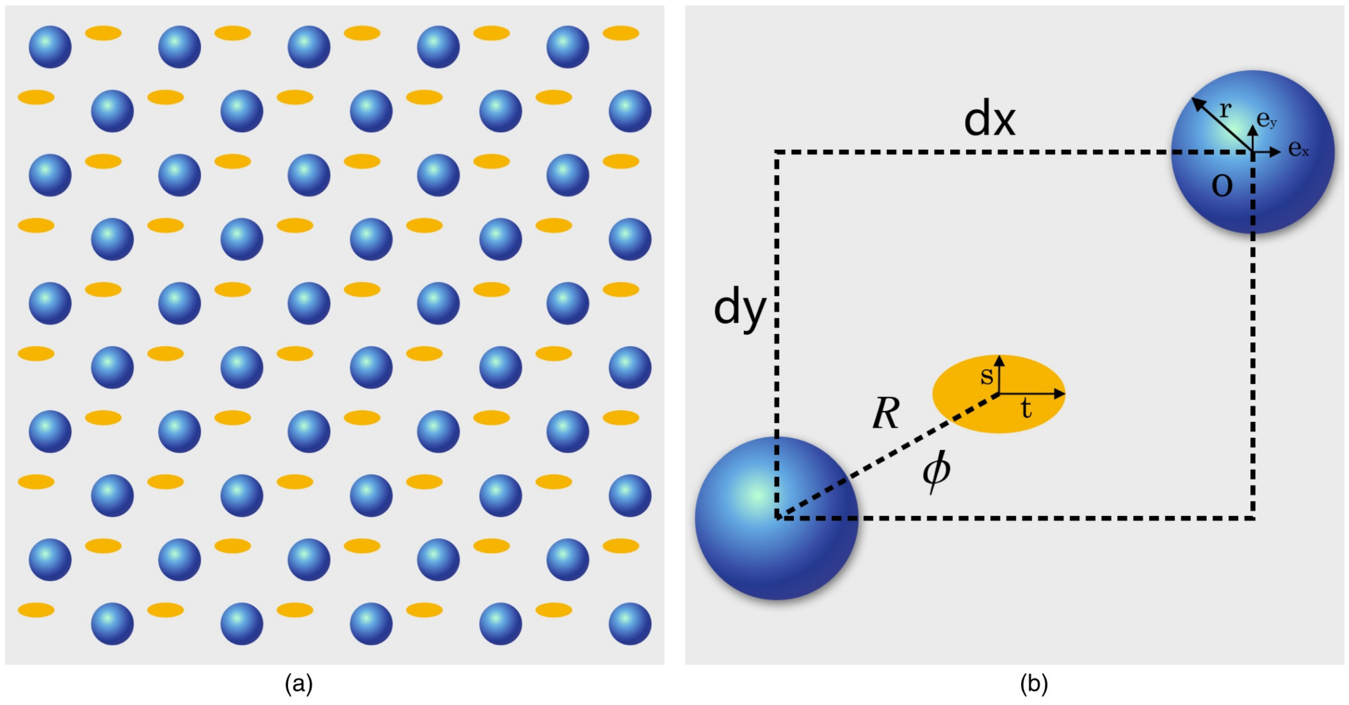

FIG. 1. Periodic composite material and the basic structures. Adapted from Liu *et al.*, J. Appl. Phys. **129**, 065101 (2021). Copyright 2021 AIP Publishing LLC.[26] (a) An example of the periodic structures employed in this work, in which particle A (circle) and particle B (ellipse) are in their respective rectangular lattices. (b) The relative positioning between a pairing particle A and particle B is parameterized by the distance R between their centers and the angular position ϕ relative to the x axis. Particle A has anisotropic thermal conductivities, which is parameterized in a cylindrical coordinate. Particle B has uniform thermal conductivities, with its shape defined by a semi-minor axis s and a semi-major axis t. The predefined ranges for these design parameters used in this work are listed in Table I.

accuracy of NN training. As the number of design parameters increases when we endeavor to solve more challenging problems, the size of the training set required scales exponentially with it. Thus, we begin to face the famous curse of dimensionality.[46,47]

To deal with the curse of dimensionality, we need to find another approach in which the size of the training set does not scale exponentially with the number of design parameters. In this paper, we propose a novel reinforcement learning (RL)-based approach to tackle the inverse design problem. We still realize thermal transparency with the same PIS system and constraints used in our previous work.[26] However, we will show the superiority of our RL-based approach in terms of computational cost. The essence of our RL-based approach is training an intelligent agent that can vary the design parameters in a series of time steps toward the realization of thermal transparency. Our results show that the agent can be trained with a reasonable computational cost and is capable of serving the purpose of solving the inverse design problem.

II. REINFORCEMENT LEARNING AS AN APPROACH TO THE INVERSE DESIGN PROBLEM

Reinforcement learning[48] has been a prominent field with many ground-breaking breakthroughs, especially since 2013 when DeepMind redefined the field by combining reinforcement learning and deep learning and achieved super-human scores in playing Atari video games.[49,50] Other breakthroughs brought by deep reinforcement learning (DRL) includes defeating world champions in Go games,[51,52] controlling robotics,[53–55] autonomous driving,[56] and algorithmic trading.[57,58] Gradually, the application of RL has also penetrated into interdisciplinary fields involving applied physics, such as designing a highly efficient metasurface for holograms.[59]

Reinforcement learning, along with supervised learning and unsupervised learning, constitute a major part in the field of machine learning. However, reinforcement learning represents a vastly different paradigm to obtain machine intelligence, compared to that of supervised learning or unsupervised learning. Unlike supervised learning, in which input and output pairs must be labeled, usually by a human, a reinforcement learning algorithm improves the intelligence of an acting agent by interacting with an environment, aiming to achieve the maximum possible cumulative rewards as feedbacks from the environment.[48] More specifically, during each time step, the agent obtains an observation of the (partial) state of the environment and a reward value and makes the decision to take an action with respect to the environment, which, in turn, will forward the environment to the next time step with the next round of observation and reward value. The behavior of an agent is called a policy, i.e., its way of determining the next action when facing a certain observation of the (partial) state of the environment. The purpose of reinforcement learning is to train the agent with a policy to maximize the cumulative rewards the agent can achieve during one or many episodes of interacting with the environment. The form of reward varies with the fields of reinforcement learning applications; for example, when applied to algorithmic trading, the reward is obviously monetary. Defining the form of reward is crucial to a reinforcement learning training, which will be discussed in more detail later in this paper.

The specific reinforcement learning algorithm we employ in this work is the Double Deep Q-Network (DDQN),[50,60] which is a variant of the Deep Q-Network (DQN) algorithm.[49] DQN belongs to a more broad category of RL algorithms called Q-learning.[61,62] Here, "Q" refers to a function in the algorithm that computes the maximum expectation value for rewards if an action is taken when the environment is in a given state. DQN dramatically improves the power of a Q-learning algorithm by approximating the Q function by a deep neural network, which makes the Q function usable when the number of environment state and action is too large to be stored and computed using traditional Q-learning or, even worse, when the state space is continuous. DDQN further refines DQN by using two different Q-networks, one for evaluating Q values and the other for deciding the next action, which alleviates the training stability issue by removing a weakness in DQN that the Q-network for both evaluating Q-values and deciding the next action tends to overestimate Q-values.

One great invention in the field of Q-learning is the inclusion of a replay buffer, drawing samples from which a technique called experience replay can be used, which drastically improves training stability.[63] Before the use of experience replay, using the Q-learning algorithm in a training process is often problematic and unstable because, unlike supervised learning where samples in the training set are usually uncorrelated, the training samples in RL learning are usually generated by a sequence of interactions with the environment, leading to high correlations between training samples.[63] Correlated training samples also cause the issue that a small change to Q-values could alter the policy and the data distribution fundamentally, which worsens the training stability problem. The experience replay technique with a replay buffer removes the correlations from the training samples and makes the changes in the data distribution far less abruptive. Since its introduction, the experience replay technique has been an integral part of DQN training. More specifically, a DQN training process begins with filling the replay buffer with randomly generated training samples and then proceeds to training the Q-networks and improving the policy of the agent. In our work, we find another use of the training samples initially generated to fill the replay buffer, which essentially accelerates the whole workflow. We will provide more details in Sec. III.

In the remainder of this section, we will describe how to frame the inverse design problem to obtain design parameters to achieve thermal transparency as a RL problem. We present a schematic diagram for our RL approach in Fig. 2. First, we use finite-element method (FEM) simulations to emulate the physical world. Basically, we wrap a FEM engine and use it as an environment. In each time step, a design parameter set is fed to the FEM environment, and a response vector that represents the heat fluxes on the baseline is calculated and resulted as an output. The response vector is also used to compute the reward value, which represents how far the simulated system is away from achieving perfect thermal transparency. More precisely, the nearer the simulated system is to achieving perfect thermal transparency, the higher the reward. Then, we try to train an agent that can gradually change one or more design parameters toward the realization of thermal transparency. The observation the agent obtains is the design parameter set fed to the FEM environment and the (compressed) response vector. During each time step, the agent takes the

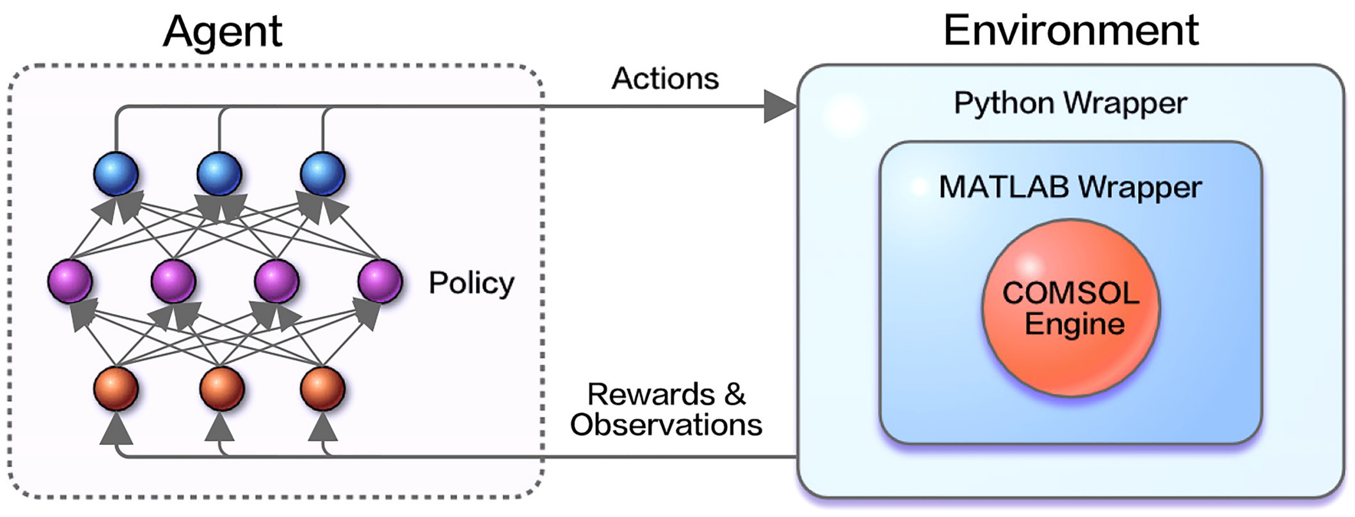

FIG. 2. A schematic description of the agent and the environment in this work.

observation and decides which action to take to maximize the expected reward value. Here, an action is defined as picking one of the design parameters and either increasing or decreasing its value by a predefined stride. After the action is taken, an updated design parameter set is fed into the FEM environment in the next time step, and a reward value and an updated response vector are returned. Once the agent is adequately trained, it will serve as a guide to help search for an optimal design parameter set to achieve thermal transparency. However, if we consider the design parameter space as a high-dimensional manifold and start the search from a randomly generated design parameter set, the agent may not be able to find a viable route to an optimal design parameter set, especially when there is no optimal design parameter set in its neighborhood. Therefore, with an adequately trained agent, one needs to perform the guided search in a parallel fashion, with a number of randomly generated design parameter sets as starting points.

The reason why we choose the same inverse design problem as we did in our previous work[26] with an RL-based approach is largely due to the difficulty and complexity of RL trainings, compared to a typical supervised learning training process. RL training is notoriously difficult to perform, which is often plagued by the correlation in the input data.[64] Therefore, it is preferable to apply our proposed RL-based approach to a known and solved problem in the field of thermal metamaterial design. It is similar to performing a “sanity check” for a software system before applying it in real-world scenarios. Once the algorithms and codes in our RL-based approach are verified, we are ready to apply it to larger and more challenging inverse design problems.

III. METHODS

Since we are addressing the same problem[25,26] by using a different approach, the same type of lattice and the same ranges for design parameters are used. The rectangular lattice is illustrated in Fig. 1. The design parameter dx is fixed to 1 cm. The ellipses (particle B) are fixed on a horizontally and vertically alternating rectangular lattice. The spacing between two horizontally adjacent ellipses is denoted as $2 * dx$, and the spacing between two vertically adjacent ellipses is denoted as $2 * dy$. The circles (type A particles) are also arranged in a rectangular lattice with a displacement with respect to the lattice occupied by type B particles. Cylindrical coordinates R and ϕ [see Fig. 4(f) of Ref. 26] describe the relative positioning between pairing circles and ellipses, with the origin located at the center of the ellipse. More specifically, R denotes the distance between the centers of pairing circle and ellipse, while ϕ denotes the angle between the line connecting the centers of pairing circle and ellipse and the horizontal axis. The circle regions (type A particles) have anisotropic thermal conductivities [a purple graded circle with a radius r, Fig. 4(c) of Ref. 26] $\overleftrightarrow{\kappa}_a = \mathrm{diag}(\kappa_{\rho\rho}, \kappa_{\theta\theta})$. $\overleftrightarrow{\kappa}_a$ is represented in the cylindrical coordinates (ρ, θ), whose origin is at the center of a type A particle. The ellipse regions (type B particles) are defined by uniform thermal conductivities [a red ellipse with a semi-minor axis s and semi-major axis t, and $e = s/t$, Fig. 4(d) of Ref. 26] κ_b. κ_m denotes the thermal conductivity of the background. Finally, the area fraction of type A particles, type B particles, and the background are denoted by $p_a[=\pi r^2/(2d_x * d_y)]$, $p_b[=\pi st/(2d_x * d_y)]$, and $p_m = 1 - p_a - p_b$.

Our entire workflow of solving the inverse design problem to realize thermal transparency with an RL approach starts with the design of the environment with which the agent to be trained will continuously interact. Essentially, the core part of the environment in this work is a COMSOL Multiphysics[65] engine that performs the FEM simulations of the periodic interparticle systems. As our RL training code is written in Python, an interface between Python and a COMSOL Multiphysics engine must be written. However, the COMSOL Multiphysics software package does not have a Python application programming interface (API). Instead, it offers a LiveLink interface to MATLAB.[66] Therefore, we write a MATLAB wrapper around the

COMSOL Multiphysics engine that passes on the inputs from our Python code to the engine, retrieves the outputs from the engine, and feeds them back to our Python code, as illustrated in Fig. 2.

The essence of designing an environment is determining how to calculate rewards and return an observation to reflect the state of the environment. The reward value is measured by how far the simulated system is away from achieving perfect thermal transparency. More specifically, as in our previous work,[26] heat fluxes are evaluated on 400 equally distance points on the baseline located at $x = -6$ cm relative to the origin, which is located at the center of the simulation box. The two ends of the 400 equally distance points have coordinates $(-6, -10)$ cm and $(-6, 10)$ cm relative to the origin, respectively. If perfect thermal transparency were achieved, one would expect the same value of heat fluxes on all 400 points on the baseline as that in the background, as if no periodic interparticle system exists. For example, if the same thermal transparency with heat fluxes $J_S \approx 82\,\mathrm{W m^{-2}}$ were achieved, one would observe $J_S \approx 82\,\mathrm{W m^{-2}}$ on all 400 points on the baseline. To measure the deviation from a perfect thermal transparency, we measure the heat flux values on all 400 points on the baseline and calculate the mean squared error (*MSE*) between the measured heat fluxes on the 400 points on the baseline and the value indicating a perfect thermal transparency,

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\phi_i - \phi_{ref})^2. \tag{1}$$

Here, n denotes the number of points selected on the baseline for the measurement of heat fluxes, ϕ_i denotes the measured heat flux value on a specific point, and ϕ_{ref} denotes the desired heat flux value in a perfect thermal transparency case.

Then, we define the unadjusted reward to be the negative of the MSE between the measured heat fluxes on the 400 points on the baseline and the value indicating a perfect thermal transparency. We also call the measured heat fluxes on the 400 points on the baseline response vector. However, this unadjusted reward could have a very wide range of magnitude, which is unfavorable for neural network (i.e., the Q-network) training. Then, we define the adjusted logarithmic reward r to be

$$r = \begin{cases} 9 - \log_{10} MSE & \text{if } MSE \leq 10^9 \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

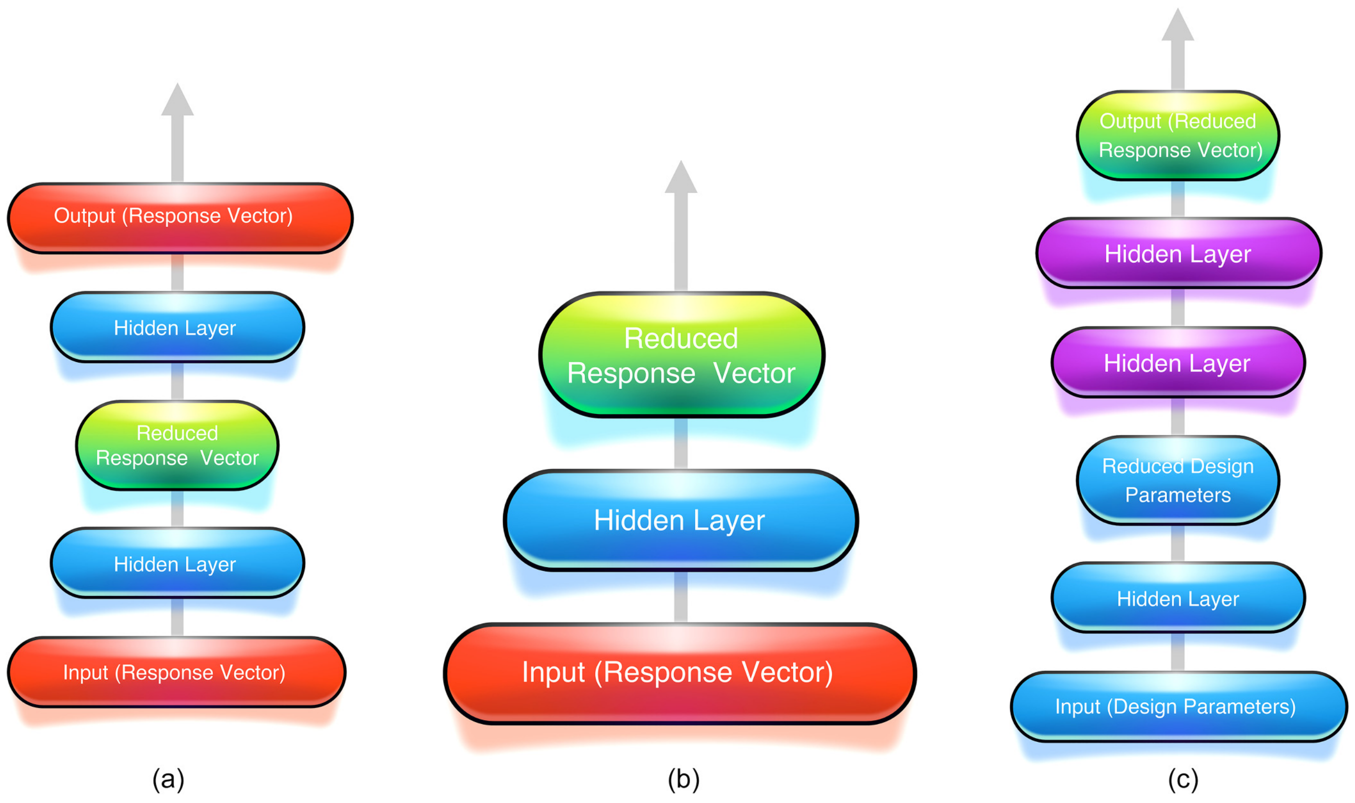

FIG. 3. Schematic diagrams of the NN architectures. (a) An autoencoder to map the response vector to the reduced response vector by performing dimensionality reduction. (b) The encoder part of (a). (c) An NN to map the design parameters to the reduced design parameters by performing dimensionality reduction.

TABLE I. Ranges of the design parameters, in which the design parameter sets for the initial filling of the replay buffer are randomly generated. The range for R is selected to avoid overlapping between type A and type B particles.

Design parameter	Range	Design parameter	Range
dy	(0.8, 1.2) cm	e	(0.05, 0.2)
R	(0.67, 1.36) cm	$\kappa_{\rho\rho}$	(0, 1) $\mathrm{W m^{-1} K^{-1}}$
ϕ	$\left(-\frac{\pi}{4}, \frac{\pi}{4}\right)$	$\kappa_{\theta\theta}$	(0, 1) $\mathrm{W m^{-1} K^{-1}}$
p_a	(0.05, 0.4)	κ_b	(0, 1) $\mathrm{W m^{-1} K^{-1}}$
p_b	(0.05, 0.4)	κ_m	(0, 1) $\mathrm{W m^{-1} K^{-1}}$

In this definition, the nearer the simulated system is to achieving perfect thermal transparency, the higher the reward.

To enable the agent to know the full state of the environment, both the input (i.e., design parameter set) and output (i.e., the response vector) are concatenated to form the observation vector. However, a direct use of the 400 dimensional response vector would make the dimensionality of the observation vector too large, which would impair the Q-network training and introduce a large amount of redundant information (note that the observation vector feeds into the input layer of the Q-network). To alleviate this issue, we take the liberty of reusing a part from our previous works:[26] using an autoencoder[67] to compress the response vector into a reduced response vector. We use a 400–50–10–50–400 architecture for the autoencoder, which is illustrated in Fig. 3(a). Note that only the architecture of the autoencoder has been reused, not the model for the autoencoder itself. The autoencoder presented in this work is trained with the training samples in the replay buffer, which will be discussed extensively later in this paper. Our previous results demonstrate that the 10-dimensional reduced response vector can conserve the vast majority of information for the 400-dimensional response vector. Therefore, the 9-dimensional design parameters

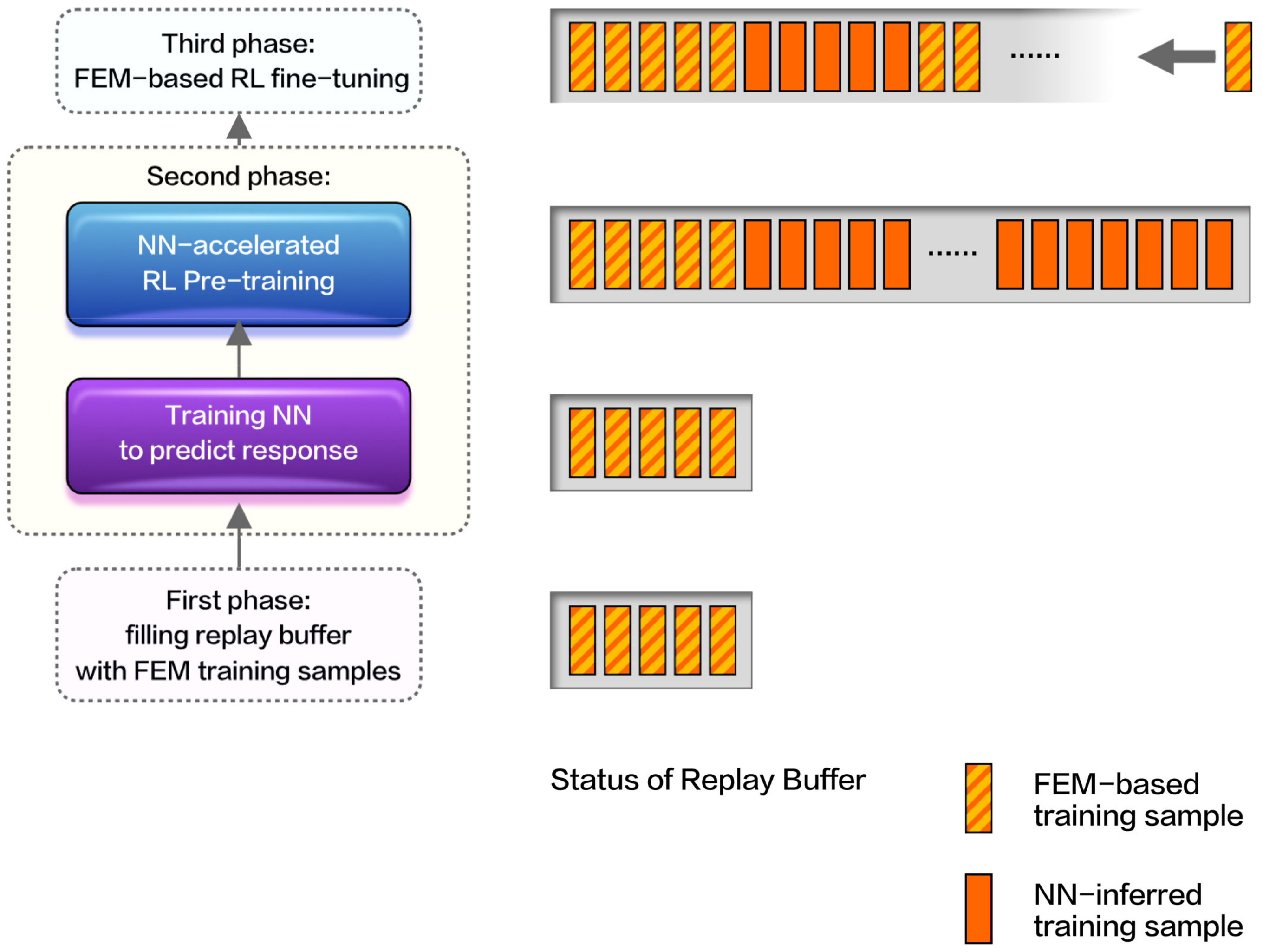

FIG. 4. The workflow of RL training. Left: The workflow is divided into three phases. Right: The corresponding status of the replay buffer.

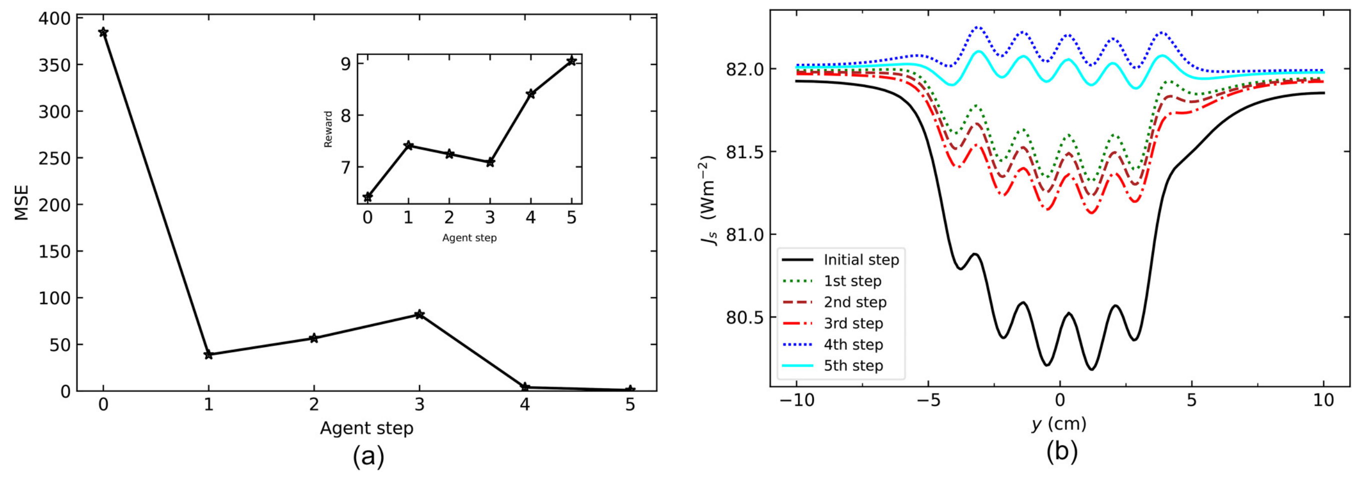

FIG. 5. An episode that ends with the terminating step finding a design parameter set that achieves thermal transparency at the baseline located at x = −6 cm with heat fluxes $J_S \approx 82\,\mathrm{Wm^{-2}}$. The episode consists of an initial step and five subsequent steps. (a) The MSE between the response vector in each step and the response vector in the perfect thermal transparency case. Inset: The corresponding reward value in each step calculated using Eq. (2). (b) The response vector for each step converging near the perfect response vector. The FEM simulation of the configuration in the initial step is presented in Figs. 6(a) and 6(b), and the FEM simulation of the configuration in the fifth step is presented in Figs. 6(c) and 6(d).

and 10-dimensional reduced response vector are concatenated to form the 19-dimensional observation vector.

During the first time step of each episode, the input to the environment (i.e., design parameter set) is randomly generated within the predefined range shown in Table I. Then, the response vector is calculated, and the observation vector is formed. The agent decides which action to take or evaluates a probability distribution of actions to be taken (for a ϵ-greedy exploration–exploitation strategy). The action will trigger the transition of the environment to the next time step. We define the action space for the agent as a two-dimensional one. In the first dimension of the action space, the agent decides which design parameter to vary. In the second dimension, the agent decides whether to increase or decrease the design parameter by a predefined stride. We select the predefined stride for each design parameter to be 1/10 of its respective range, as listed in Table I. The agent may take an action that could make the design parameter out of the range. In this case, the episode is terminated, and a new episode is initiated. To ensure that an episode does not last forever, we define a hyperparameter called the maximum per episode steps, which is set to 10 000 for the RL training. Once the step count exceeds this limit, the episode is terminated, even when the design parameters are still in the valid range. However, in real training, this limit is never reached. Another situation is that the agent finds the optimal design parameter set to achieve thermal transparency. In this case, the episode is also terminated.

The workflow of our RL training can be summarized into two phases, as depicted in Fig. 4. During the first phase, training samples are initially generated with randomly selected design parameter sets and the corresponding response vector and rewards obtained by FEM simulations to fill the replay buffer. A training sample consists of five parts: the observation of the environment's state, the action the agent decides to take, the reward from taking the said action, the types of the current time step, and the next time step (whether the time step is the initial step, a running step, or the terminating step of an episode). Once the replay buffer is filled with a certain number of training samples, the first phase is completed and the response vectors corresponding to the training samples are used to train the autoencoder, which performs dimensionality reduction for the response space. During the second phase, the RL algorithm continues to explore the environment by using a collect policy to feed actions to the environment and obtain observations as new training samples. These new training samples are placed into the replay buffer, and the old training samples are evicted while the RL

TABLE II. The design parameters of two systems (A and B) to achieve thermal transparency at the baseline located at x = −6 cm with heat fluxes $J_S \approx 82\,\mathrm{Wm^{-2}}$. System A corresponds to Figs. 6(a) and 6(b), and system B corresponds to Figs. 6(c) and 6(d). dy and R are in cm; $\kappa_{\rho\rho}$, $\kappa_{\theta\theta}$, κ_b, and κ_m are in $\mathrm{Wm^{-1}\,K^{-1}}$; and the other design parameters are unitless.

Design parameter	dy	R	ϕ	p_a	p_b	e	$\kappa_{\rho\rho}$	$\kappa_{\theta\theta}$	κ_b	κ_m
A	0.849	0.957	−0.209	0.0603	0.0647	1.60	0.960	0.656	0.0523	0.410
B	0.849	0.957	−0.209	0.0603	0.0647	1.60	0.660	0.656	0.252	0.410

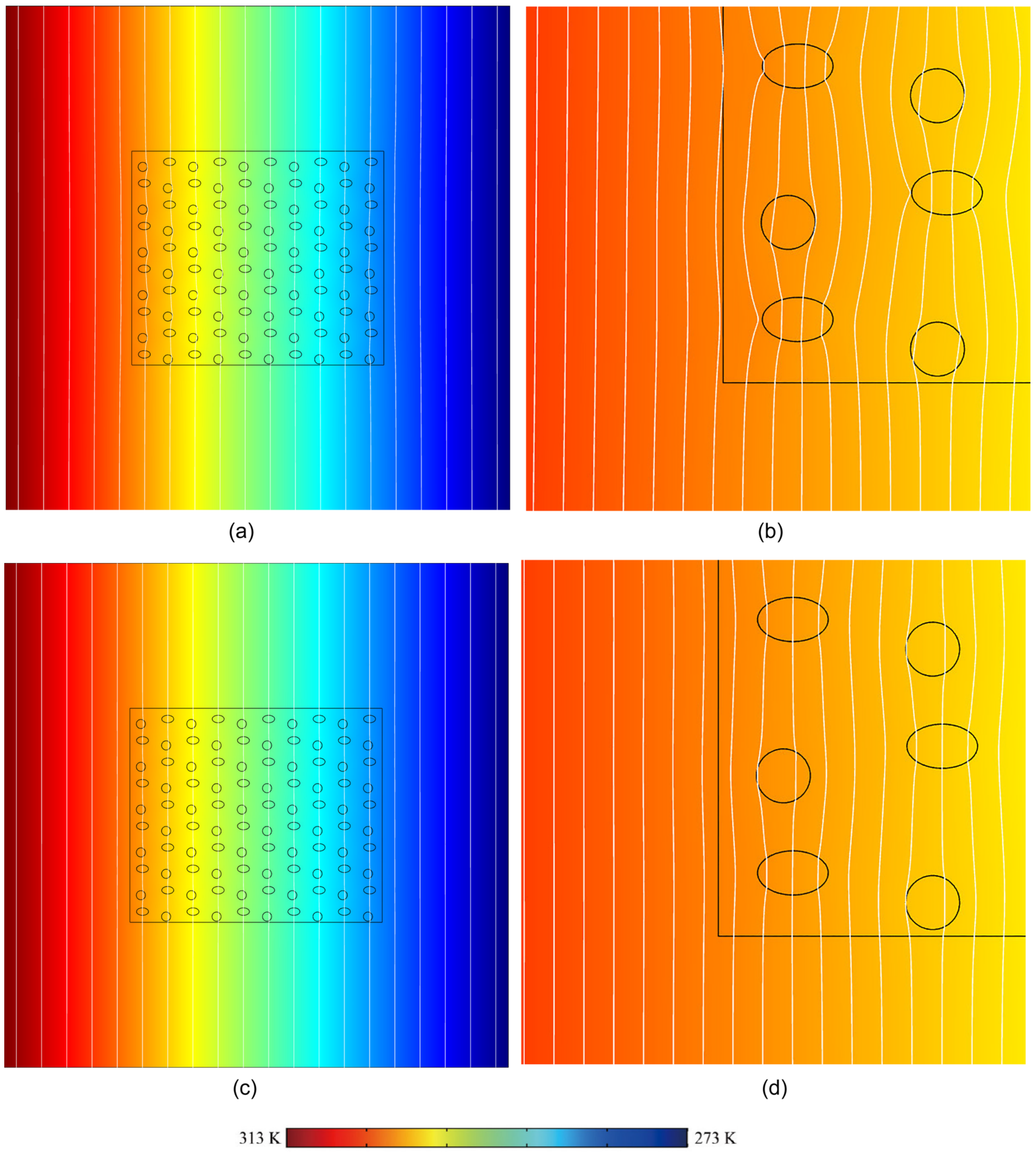

FIG. 6. Finite-element simulations of thermal transparency with heat fluxes $J_S \approx 82\,\mathrm{W m^{-2}}$. The size of the simulation box is $20 \times 20\,\mathrm{cm}^2$, and the periodic composite material is around $10 \times 10\,\mathrm{cm}^2$. The baseline for measuring heat fluxes is located at $x = -6$ cm (not shown on the figures), and the origin is located at the center of the simulation box. White lines represent isotherms. (a) The design parameter corresponds to the initial step in the episode shown in Fig. 5. The calculated MSE [defined in Eq. (1)] is 384.56. (b) The zoom-in of the left-bottom corner of the periodic composite material box in (a). (c) The design parameter corresponds to the terminating step in the episode shown in Fig. 5. The calculated MSE [defined in Eq. (1)] is 0.90. (d) The zoom-in of the left-bottom corner of the periodic composite material box in (c). The temperature isotherms in (c) and (d) demonstrate that in the vicinity of the periodic composite material box, the temperature distribution is far less disturbed by the presence of the periodic composite material box, compared to those in (a) and (b), which signifies the match between the effective thermal conductivity achieved by our proposed mechanism and the thermal conductivity of the background. The design parameter values are shown in Table II.

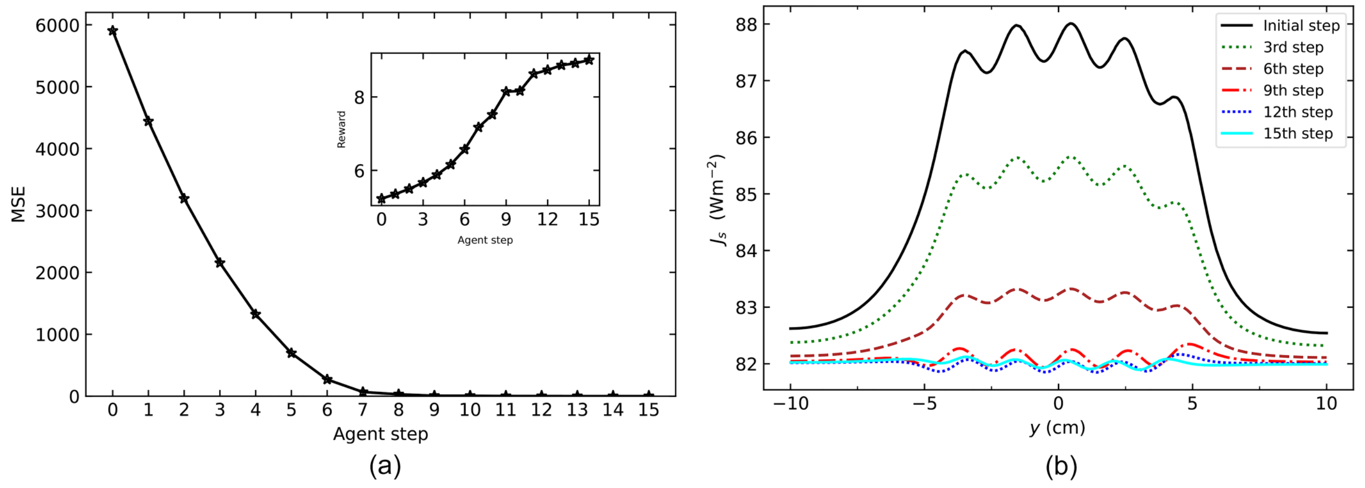

FIG. 7. Another episode that ends with the terminating step finding a design parameter set that achieves thermal transparency at the baseline located at $x = -6$ cm with heat fluxes $J_S \approx 82\,\mathrm{Wm}^{-2}$. The episode consists of an initial step and 15 subsequent steps. (a) The MSE between the response vector in each step and the response vector in the perfect thermal transparency case. Inset: The corresponding reward value in each step calculated using Eq. (2). (b) The response vector for the initial, 3rd, 6th, 9th, 12th, and 15th steps converging to the near perfect response vector. The FEM simulation of the configuration in the initial step is presented in Figs. 8(a) and 8(b), and the FEM simulation of the configuration in the 15th step is presented in Figs. 8(c) and 8(d).

algorithm samples a batch from the replay buffer and trains the Q-networks.

Inspired by our previous work,[26] we realize that this typical RL training workflow can be tremendously accelerated by inserting a pre-training phase between the two aforementioned phases. In this pre-training phase, the training examples in the replay buffer are used again to train a feed-forward neural network that takes design parameters as input and predicts the (reduced) response vector. The architecture of this feed-forward neural network is illustrated in Fig. 3(c). We then use the trained NN to act as a pseudo-environment with which the agent interacts. The key improvement comes from making an NN inference to obtain the reduced response vector that is several orders faster than taking a FEM-based evaluation. Although one would expect some errors in an inferred response vector deviating from the ground-truth response vector calculated by a FEM simulation, the inferred response vector is accurate enough to let the agent know the approximate landscape of the response space and train the Q-networks accordingly to adapt to it. We call this phase the NN-accelerated pre-training phase of the RL training.

Once the agent is adequately trained in the NN-accelerated pre-training phase, which is signified by the diminishing yet stabilizing training loss for the Q-networks, we switch the environment and let the agent interact with the real FEM-based environment, which further improves the policy of the agent. We call this phase the FEM-based fine-tuning phase. In this phase, we start to search for the optimal design parameter set to achieve thermal transparency. We define the criterion for finding the optimal design parameter set to be having an MSE less than 1 from the perfect response vector. To give an idea how stringent this criterion is, a randomly generated design parameter set usually corresponds to a MSE on the order 10^3.

There is one subtle problem regarding the size for the replay buffer for different phases. During the first phase when the replay buffer is being filled with FEM-based training samples, we limit the size of the replay buffer to 1000. The same 1000 training samples are used to train the feed-forward NN in the second phase, the NN-accelerated pre-training phase. Furthermore, these 1000 training samples are used a third time to train the autoencoder for dimensionality reduction of the response space. Then, we use the DDQN algorithm to train the agent interacting with the pseudo-environment while the replay buffer continues to be filled with training examples generated by the interactions between the agent and the pseudo-environment as we lift the limit of the size

TABLE III. The design parameters of two systems (C and D) to achieve thermal transparency at the baseline located at x = –6 cm with heat fluxes $J_S \approx 82\,\mathrm{Wm}^{-2}$. System C corresponds to Figs. 8(a) and 8(b), and system D corresponds to Figs. 8(c) and 8(d). dy and R are in cm; $\kappa_{\rho\rho}$, $\kappa_{\theta\theta}$, κ_b, and κ_m are in $\mathrm{Wm}^{-1}\,\mathrm{K}^{-1}$; and the other design parameters are unitless.

Design parameter	dy	R	ϕ	p_a	p_b	e	$\kappa_{\rho\rho}$	$\kappa_{\theta\theta}$	κ_b	κ_m
C	1.02	1.06	−0.0378	0.285	0.0540	1.78	0.721	0.735	0.344	0.410
D	0.90	1.06	−0.352	0.0747	0.0540	1.78	0.621	0.535	0.244	0.410

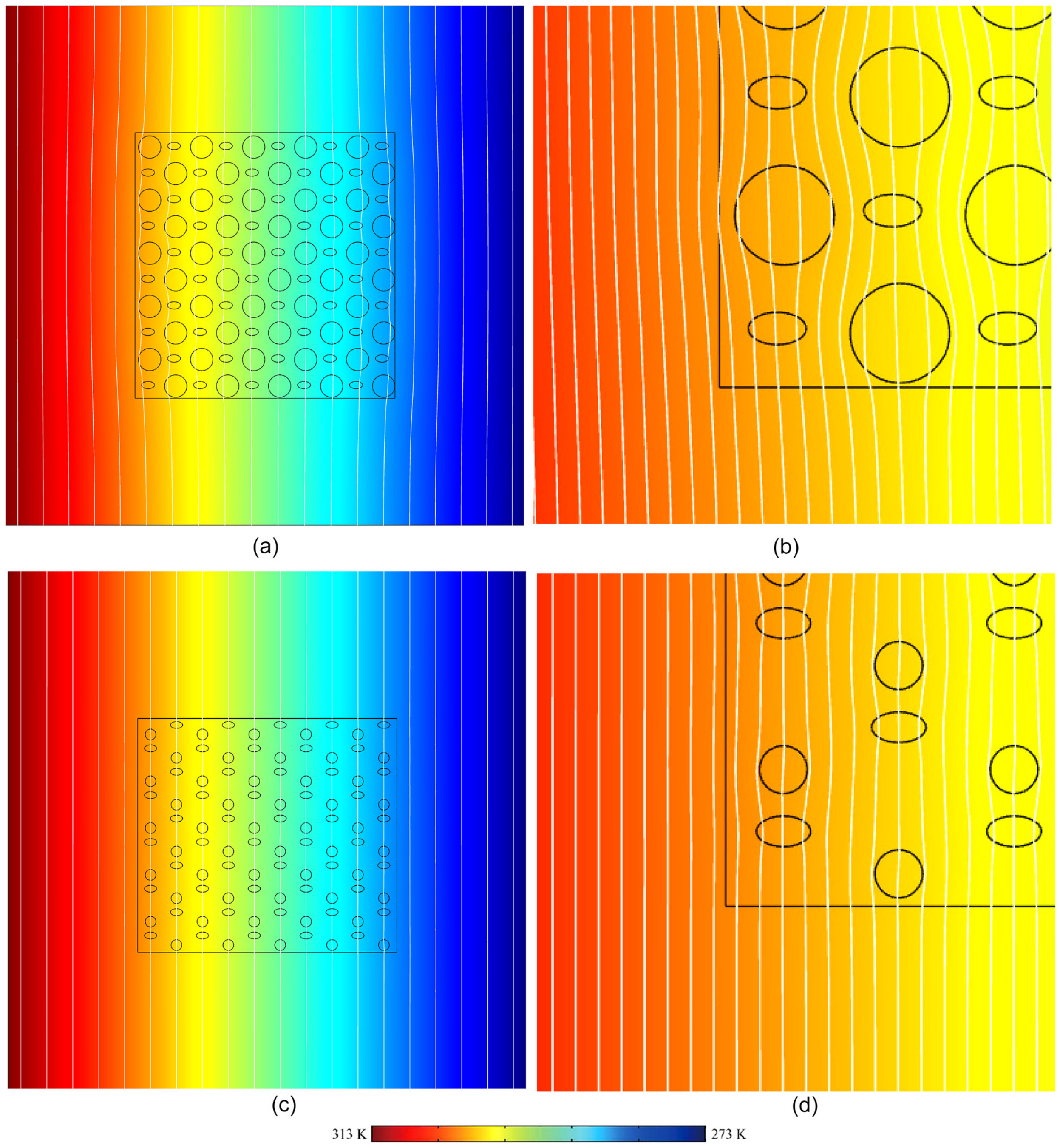

FIG. 8. Finite-element simulations of thermal transparency with heat fluxes $J_S \approx 82\,\mathrm{Wm}^{-2}$. The size of the simulation box is $20 \times 20\,\mathrm{cm}^2$, and the periodic composite material is around $10 \times 10\,\mathrm{cm}^2$. The baseline for measuring heat fluxes is located at $x = -6$ cm (not shown on the figures), and the origin is located at the center of the simulation box. White lines represent isotherms. (a) The design parameter corresponds to the initial step in the episode shown in Fig. 7. The calculated MSE [defined in Eq. (1)] is 5903.62. (b) The zoom-in of the left-bottom corner of the periodic composite material box in (a). (c) The design parameter corresponds to the terminating step in the episode shown in Fig. 7. The calculated MSE [defined in Eq. (1)] is 0.99. (d) The zoom-in of the left-bottom corner of the periodic composite material box in (c). The temperature isotherms in (c) and (d) demonstrate that in the vicinity of the periodic composite material box, the temperature distribution is far less disturbed by the presence of the periodic composite material box, compared to those in (a) and (b), which signifies the match between the effective thermal conductivity achieved by our proposed mechanism and the thermal conductivity of the background. The design parameter values are shown in Table III.

of the replay buffer to 100 000. When we proceed to the last phase of FEM-based fine-tuning, we assemble a replay buffer with an initial size of 2000, half of which are training samples from the first phase and the other half are from interactions between the agent and the pseudo-environment during the second phase. Thus, we keep all the FEM-based training samples and downsample the training samples generated by the interactions between the agent and the pseudo-environment from ∼100 000 to 1000. As the RL training in the FEM-based fine-tuning continues, the replay buffer is being filled with more FEM-based training samples.

The rationale behind varying the replay buffer size is simply to balance the accuracy and speed. During the first phase when the FEM-based training samples are costly to generate, we have to limit the replay buffer to a relatively small size. During the NN-accelerated pre-training phase when generating training samples by NN inference is several orders cheaper, we assign a much larger limit to the size of the replay buffer to allow the agent to be trained on a much larger yet less accurate training set. During the last phase of FEM-based fine-tuning, although new FEM-based training samples are trickling into the replay buffer, we still have to downsample the training samples generated by the interactions between the agent and the pseudo-environment. Otherwise, the replay buffer would be dominated by the less accurate training samples generated by NN inference, which gives the agent a scarce chance to be exposed to the accurate FEM-based environment.

We employ the commercial software packages COMSOL Multiphysics[65] for FEM simulations and MATLAB[66] as the interface between the COMSOL engine and the Python environment for running the RL training. The environment which wraps around the COMSOL engine is written in a mixture of Python/MATLAB/Java.

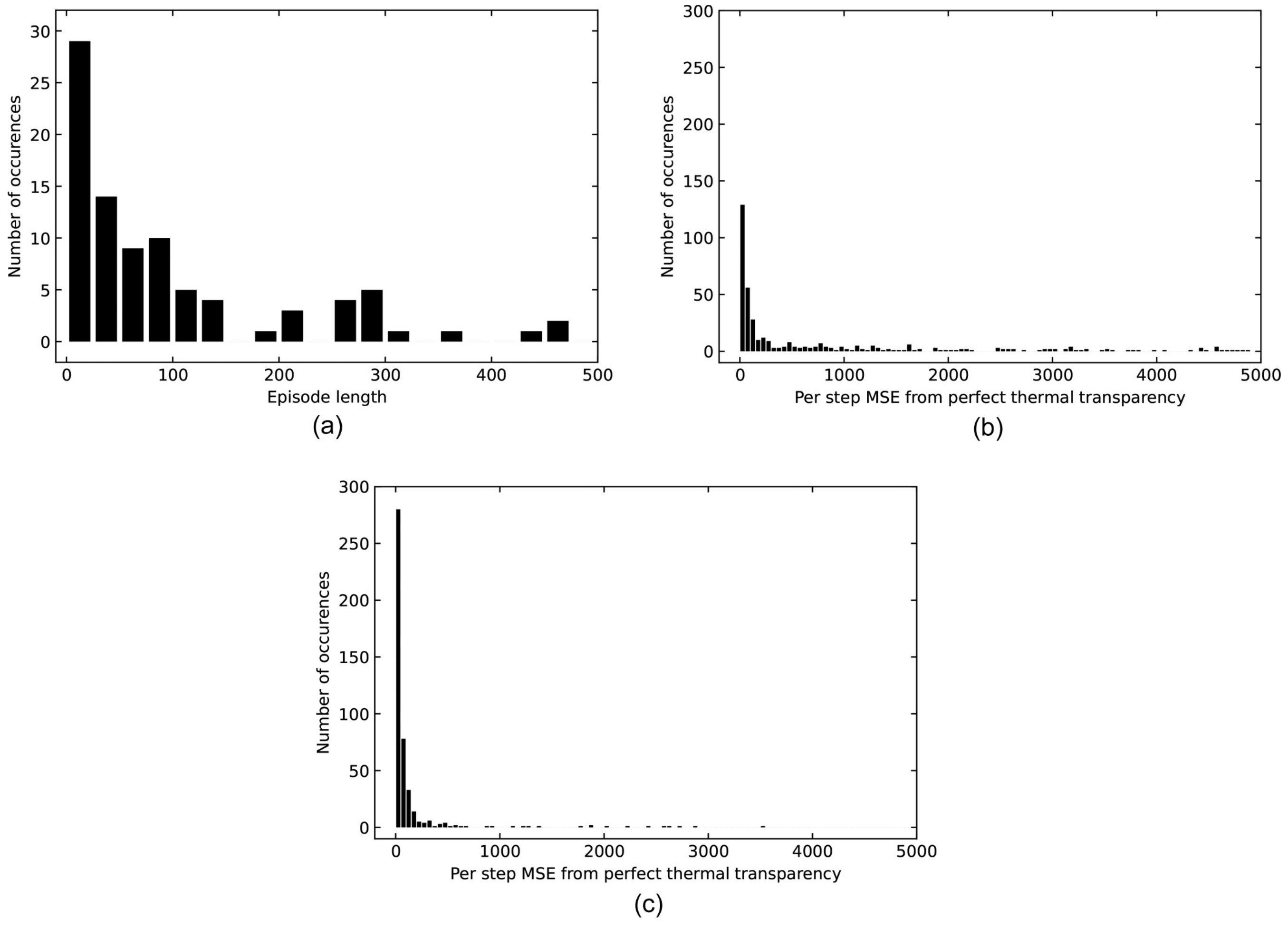

FIG. 9. (a) The histogram of the length of all episodes the agent experienced in the FEM-based RL fine-tuning phase. (b) The histogram of the MSE between the response vector and the perfect thermal transparency case in the first five steps of all episodes the agent experienced in the FEM-based RL fine-tuning phase. (c) The histogram of the MSE between the response vector and the perfect thermal transparency case in the last five steps of all episodes the agent experienced in the FEM-based RL fine-tuning phase.

The RL training is performed with the framework of TensorFlow 2.1[68] and TF-Agents 0.4.0.[69] We also employ Pandas[70] for data analysis. The FEM simulations and RL training were primarily done on a dual-socket 24 physical core Xeon Haswell-EP server.

The architecture of the Q-networks is a simple feed-forward NN (19–100–50–50–18). The dimension of the input layer for the Q-networks comes from the fact that the observation vector is 19-dimensional. The dimension of the output layer for the Q-networks arises from the 18 possible actions to take (9 design parameters to vary and either increase or decrease by a predefined stride). We employ the RMSProp optimizer with a learning rate of 2.5×10^{-4}, a discount factor of 0.95 for the history/coming gradient, and zero momentum. We set the target Q-network update frequency to 2000, following the classical papers by DeepMind.[50] We use Huber loss[71] to evaluate temporal difference (TD) errors. We use 0.99 as the discount factor for future rewards. We also employ a ϵ-greedy strategy to balance exploration and exploitation, where ϵ has a decaying value from 1.0 to 0.01 over 250 000 steps.

IV. RESULTS

In the FEM-based RL fine-tuning phase, we record all the episodes that the agent experienced. To obtain a comparison with our previous work, we still aim to achieve thermal transparency at the baseline located at $x = -6$ cm with heat fluxes $J_S \approx 82\,\mathrm{W m^{-2}}$. Out of the 95 episodes we recorded, 9 episodes ended with a terminating step in which a design parameter set that could achieve thermal transparency was found. The other episodes ended with a terminating step in which the design parameter set had at least one parameter out of the range defined in Table I.

Here, we present two exemplary episodes that achieve our final objective of realizing thermal transparency. The first episode consists of six steps. Figure 5 demonstrates the MSE between the response vector and the ideal case response vector (flat values of $J_S = 82\,\mathrm{W m^{-2}}$) for each step. The reward value, which is the negative of the logarithmic MSE plus a constant, is also presented in the inset. The shape of the response vectors are also shown in Fig. 5(b). We perform FEM simulations with the design parameter sets in the initial step and the terminating step, respectively, with the values for the design parameters shown in Table II and the temperature distribution and isotherms shown in Fig. 6. The agent only changes the values for $\kappa_{\rho\rho}$ and κ_b.

The second episode consists of 16 steps. Figure 7 demonstrates the MSE between the response vector and the ideal case response vector (flat values of $J_S = 82\,\mathrm{W m^{-2}}$) for each step. The reward value, which is the negative of the logarithmic MSE plus a constant, is also presented in the inset. The shape of the response vectors are also shown in Fig. 7(b). We perform FEM simulations with the design parameter sets in the initial step and the terminating step, respectively, with the values for the design parameters shown in Table III and the temperature distribution and isotherms shown in Fig. 8. In this episode, the agent varies more design parameters, including dy, ϕ, p_a $\kappa_{\rho\rho}$, $\kappa_{\theta\theta}$, and κ_b.

Figure 9(a) shows a histogram for the lengths of all 95 episodes. The vast majority of episodes have a length shorter than 100 steps. In Fig. 9(b), we collect the first five steps from all episodes and plot a histogram of the MSE between the corresponding response vector and the ideal case response vector (flat values of $J_S = 82\,\mathrm{W m^{-2}}$). In Fig. 9(c), we plot a similar histogram, but for the last five steps of all episodes. Comparing Figs. 9(b) and 9(c) indicates that the trained agent has the capability of varying the design parameters toward achieving thermal transparency with statistical significance.

V. DISCUSSION AND CONCLUSION

Thermal transparency has potential applications for reducing thermal stress concentration. In general, a non-uniform thermal field may damage devices due to thermal stress concentration. The proposed scheme can remove the distortion of thermal profiles and reduce the thermal stress concentration, so it can thermally protect devices, i.e., thermal protection. Moreover, thermal transparency also provides insights into thermal camouflage because it is a special kind of thermal camouflage to some extent.

There are two basic methods to realize anisotropic thermal conductivities in experiments. One method is based on natural materials such as fiber materials and porous materials that commonly have anisotropic thermal conductivities themselves.[72] The other method is to design microstructures in the materials with isotropic thermal conductivities. Common microstructures include layered structures, whose anisotropic thermal conductivities are a result of the different capabilities of heat transfer along the directions of series and parallel connections.[73] To realize the designed parameters, layered structures are good candidates. Since the effect of thermal transparency is mainly determined by the relative sizes of particles and matrices, we can scale up the whole system to make it easy to fabricate the required microstructures. A more convenient scheme lies in the 3D printing skill. As long as the microstructures are well designed, the samples can be fabricated with the anisotropic thermal conductivities induced by the microstructures.[74]

In this work, we demonstrate a RL-based approach for solving the inverse design problem of deciding the design parameter set for realizing thermal transparency with a periodic interparticle system. The DDQN algorithm we employ can train an intelligent agent that is able to vary the design parameters toward the realization of thermal transparency. In our approach, we carefully design the reward function, the composition of observations of the environment for the RL training. We implement the environment to mimic the physical world by using a COMSOL-based FEM engine and a MATLAB/Python wrapper. We devise an efficient scheme of making most use of training samples in the replay buffer and achieve balance between accuracy and speed. To the best of our knowledge, the use of an NN-based pseudo-environment to pre-train the RL agent to accelerate the whole workflow has not been attempted by others previously. In the FEM-based RL fine-tuning phase, we find several episodes with terminating steps in which the design parameter set for realizing thermal transparency is found. It demonstrates the full power of our trained RL agent.

One advantage of our RL-based approach over our previous autoencoder-based approach is that the new approach requires a significantly fewer computational resource. Most of the computational cost arises from the first phase of the RL training (i.e., filling the replay buffer with 1000 FEM training samples) and the last phase of the RL training, FEM-based RL fine-tuning, where several thousand

FEM simulations are performed. This is in sharp contrast to our previous work[26] where 50 000 FEM simulations were performed.

In theory, with adequate computational resources, the NN-accelerated RL pre-training phase could be "ablated," and the RL training could still converge. However, according to our experience with a number of trial and error iterations to obtain convergence in the RL training, the NN-accelerated RL pre-training phase is an integral part of the whole RL training process. Constrained by our computational resources, we simply could not afford to fill the replay buffer only with the accurate yet very expensive FEM-based training samples. Without the NN-accelerated RL pre-training phase, only FEM-based training samples can go into the replay buffer, which takes too long to make the size of the replay buffer sufficiently large to train the RL agent using a DDQN algorithm.

Furthermore, our RL-based approach can be readily extended to solve inverse design problems with constraints, which cannot be easily addressed by our previous autoencoder-based approach. For example, thermal metamaterials with certain ranges of thermal conductivities could be costly to fabricate, and it would be preferable to find a set or multiple sets of design parameters that avoid the said ranges of thermal conductivities. In our RL-based approach, one can manipulate the reward function to tell the agent to avoid a certain range for a specific design parameter while the agent is searching the design space, to satisfy the predefined constraints. Considering that one often encounter inverse design problems with constraints in real-world applications, our RL-based approach could broaden the applications of thermal metamaterials significantly.

RL, compared to supervised learning and unsupervised learning, is an intrinsically different paradigm for acquiring machine intelligence. The superior efficiency of our RL-based approach, compared to our previous supervised learning-based approach, is difficult to explain using technical details, as one could not draw a direct apple-to-apple comparison between two different paradigms. However, a high-level comparison between the two approaches is possible. A very recent article by Silver *et al.*,[75] "Reward is enough," defines the essence of intelligence as "the computational part of the ability to achieve goals in the world," and declares that RL, as a formalization of the problem of goal-seeking intelligence, as the ultimate way to achieve artificial general intelligence (AGI). They introduce a hypothesis for achieving AGI: Reward-is-Enough. Under this hypothesis, once a reward is adequately defined, and algorithms that can train an agent acting in its environment to maximize the reward is implemented, any kind of human level or even super-human level machine intelligence can be achieved. This hypothesis, from a high-level perspective, explains the efficacy of an RL-based approach compared to a traditional supervised learning-based approach. In simple words, an RL-based approach is more akin to the nature of human intelligence, while a traditional supervised learning-based approach is more mechanistic. In summary, an RL-based approach, though relatively challenging to implement, could offer intrinsic benefits in terms of computational efficiency, as it captures the very nature of intelligence.

A known limitation of our RL-based approach is that the initiation scheme for generating the first time step of each episode may have a large room to improve. Currently, the design parameter set is randomly generated for the first time step of each episode. However, most episodes end up with not finding an optimal design parameter set. As the RL training in the final fine-tuning goes, the agent knows more and more about the landscape of the response space. Therefore, in theory, a more clever initiation scheme could be designed to make more efficient use of the information the agent knows, to make a more educated guess of the initial design parameter set for an episode. This new initiation scheme could tremendously reduce the number of fruitless episodes that can be known *a priori* to have a high probability to get stuck in a region where finding an optimal design parameter is highly unlikely.

Our RL-based approach is extensible to other inverse design problems in the field of thermal metamaterials, such as deciding the design parameter set for thermal cloaks. Furthermore, our RL-based approach shows the great potential of RL, as a framework to acquire machine intelligence, to solve a broad spectrum of inverse design problems that could be found in fields far beyond metamaterial designs.

ACKNOWLEDGMENTS

We acknowledge financial support from the National Natural Science Foundation of China under Grant Nos. 11725521 and 12035004, the Science and Technology Commission of Shanghai Municipality under Grant No. 20JC1414700, the Education Department of Sichuan Province under Grant No. 18ZB0273, and the Leshan Science and Technology Bureau under Grant No. 15NZD100. We thank DAI, Gaole for his contribution of Python codes for detecting particle overlapping.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1]C. Z. Fan, Y. Gao, and J. P. Huang, Appl. Phys. Lett. **92**, 251907 (2008).
[2]T. Y. Chen, C. N. Weng, and J. S. Chen, Appl. Phys. Lett. **93**, 114103 (2008).
[3]S. Narayana and Y. Sato, Phys. Rev. Lett. **108**, 214303 (2012).
[4]R. Schittny, M. Kadic, S. Guenneau, and M. Wegener, Phys. Rev. Lett. **110**, 195901 (2013).
[5]H. Y. Xu, X. H. Shi, F. Gao, H. D. Sun, and B. L. Zhang, Phys. Rev. Lett. **112**, 054301 (2014).
[6]T. C. Han *et al.*, Phys. Rev. Lett. **112**, 054302 (2014).
[7]Y. G. Ma, Y. C. Liu, M. Raza, Y. D. Wang, and S. L. He, Phys. Rev. Lett. **113**, 205501 (2014).
[8]G. Dai and J. Huang, J. Appl. Phys. **124**, 235103 (2018).
[9]S. Yang, L. Xu, and J. Huang, J. Appl. Phys. **125**, 055103 (2019).
[10]S. Yang, L. Xu, G. Dai, and J. Huang, J. Appl. Phys. **128**, 095102 (2020).
[11]R. S. Kapadia and P. R. Bandaru, Appl. Phys. Lett. **105**, 233903 (2014).
[12]L. J. Xu, S. Yang, and J. P. Huang, Phys. Rev. E **98**, 052128 (2018).
[13]X. He and L. Z. Wu, Phys. Rev. E **88**, 033201 (2013).
[14]L. W. Zeng and R. X. Song, Appl. Phys. Lett. **104**, 201905 (2014).
[15]T. Z. Yang *et al.*, Adv. Mater. **27**, 7752 (2015).
[16]R. Z. Wang, L. J. Xu, Q. Ji, and J. P. Huang, J. Appl. Phys. **123**, 115117 (2018).
[17]T. C. Han, X. Bai, J. T. L. Thong, B. W. Li, and C. W. Qiu, Adv. Mater. **26**, 1731 (2014).
[18]X. He and L. Z. Wu, Appl. Phys. Lett. **105**, 221904 (2014).
[19]T. Z. Yang, Y. Su, W. Xu, and X. D. Yang, Appl. Phys. Lett. **109**, 121905 (2016).
[20]R. Hu *et al.*, Adv. Mater. **3**, 1707237 (2018).
[21]S. L. Zhou, R. Hu, and X. B. Luo, Int. J. Heat Mass Transf. **127**, 607 (2018).

[22]L. J. Xu, R. Z. Wang, and J. P. Huang, J. Appl. Phys. **123**, 245111 (2018).
[23]L. J. Xu and J. P. Huang, Phys. Lett. A **382**, 3313 (2018).
[24]L. Xu, R. Wang, and J. Huang, J. Appl. Phys. **123**, 245111 (2018).
[25]L. Xu, S. Yang, and J. Huang, Phys. Rev. Appl. **11**, 034056 (2019).
[26]B. Liu, L. Xu, and J. Huang, J. Appl. Phys. **129**, 065101 (2021).
[27]D. Liu, Y. Tan, E. Khoram, and Z. Yu, ACS Photonics **5**, 1365 (2018).
[28]J. Peurifoy *et al.*, Sci. Adv. **4**, eaar4206 (2018).
[29]Z. Liu, D. Zhu, S. P. Rodrigues, K.-T. Lee, and W. Cai, Nano Lett. **18**, 6570 (2018).
[30]M. H. Tahersima *et al.*, Sci. Rep. **9**, 1368 (2019).
[31]T. Zhang *et al.*, Photonics Res. **7**, 368 (2019).
[32]W. Ma, F. Cheng, and Y. Liu, ACS Nano **12**, 6326 (2018).
[33]Y. Qu, L. Jing, Y. Shen, M. Qiu, and M. Soljačić, ACS Photonics **6**, 1168 (2019).
[34]S. Inampudi and H. Mosallaei, Appl. Phys. Lett. **112**, 241102 (2018).
[35]H. Kabir, Y. Wang, M. Yu, and Q.-J. Zhang, IEEE Trans. Microw. Theory Tech. **56**, 867 (2008).
[36]G. Fujii, Y. Akimoto, and M. Takahashi, Appl. Phys. Lett. **112**, 061108 (2018).
[37]G. Fujii and Y. Akimoto, Appl. Phys. Lett. **115**, 174101 (2019).
[38]G. Fujii and Y. Akimoto, Opt. Lett. **44**, 2057 (2019).
[39]G. Fujii and Y. Akimoto, Int. J. Heat Mass Transf. **137**, 1312 (2019).
[40]G. Fujii and Y. Akimoto, Int. J. Heat Mass Transf. **159**, 120082 (2020).
[41]G. Fujii and Y. Akimoto, Phys. Rev. E **102**, 033308 (2020).
[42]R. Hu *et al.*, Phys. Rev. X **10**, 021050 (2020).
[43]R. Hu *et al.*, Nano Energy **72**, 104687 (2020).
[44]W. Sha, Y. Zhao, L. Gao, M. Xiao, and R. Hu, J. Appl. Phys. **128**, 045106 (2020).
[45]Y. Kiarashinejad, S. Abdollahramezani, and A. Adibi, NPJ Comput. Mater. **6**, 12 (2020).
[46]R. Bellman, *Dynamic Programming*, Dover Books on Computer Science Series (Dover Publications, 2003).
[47]R. Bellman, R. Bellman, and K. M. R. Collection, *Adaptive Control Processes: A Guided Tour* (Princeton Legacy Library, Princeton University Press, 1961).
[48]R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. (MIT Press, 2018).
[49]V. Mnih *et al.*, arXiv 1312.5602 (2013).
[50]V. Mnih *et al.*, Nature **518**, 529 (2015).
[51]D. Silver *et al.*, Nature **529**, 484 (2016).
[52]D. Silver *et al.*, Nature **550**, 354 (2017).
[53]L. Lin, "Reinforcement learning for robots using neural networks," Ph.D. thesis (Carnegie Mellon University, 1993).
[54]S. Levine, C. Finn, T. Darrell, and P. Abbeel, arXiv 1504.00702 (2016).
[55]D. Kalashnikov *et al.*, arXiv 1806.10293 (2018).
[56]B. R. Kiran *et al.*, arXiv 2002.00444 (2021).
[57]T. Théate and D. Ernst, arXiv 2004.06627 (2020).
[58]E. S. Ponomarev, I. V. Oseledets, and A. S. Cichocki, J. Commun. Technol. Electron. **64**, 1450 (2019).
[59]I. Sajedian, H. Lee, and J. Rho, Sci. Rep. **9**, 10899 (2019).
[60]H. van Hasselt, A. Guez, and D. Silver, arXiv 1509.06461 (2015).
[61]C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. thesis (King's College, Cambridge, UK, 1989).
[62]M. Riedmiller, "Neural fitted q iteration—First experiences with a data efficient neural reinforcement learning method," in *Machine Learning: ECML 2005*, edited by J. Gama, R. Camacho, P. B. Brazdil, A. M. Jorge, and L. Torgo (Springer, Berlin, 2005), pp. 317–328.
[63]L.-J. Lin, Mach. Learn. **8**, 293–321 (1992).
[64]G. Dulac-Arnold, D. J. Mankowitz, and T. Hester, arXiv:1904.12901 (2019).
[65]See http://www.comsol.com/ for more information.
[66]See http://www.matlab.com/ for more information.
[67]G. E. Hinton and R. R. Salakhutdinov, Science **313**, 504 (2006).
[68]M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," software available from tensorflow.org (2015).
[69]S. Guadarrama *et al.*, "TF-Agents: A library for reinforcement learning in tensorflow," see https://github.com/tensorflow/agents (2018) (accessed 25 June 2019].
[70]The Pandas Development Team, Pandas-dev/pandas, Pandas, 2020.
[71]P. J. Huber, Ann. Math. Stat. **35**, 73 (1964).
[72]N. Athanasopoulos and N. J. Siakavellas, Adv. Eng. Mater. **17**, 1494 (2015).
[73]Q. Ji *et al.*, Int. J. Heat Mass Transf. **169**, 120948 (2021).
[74]Y.-G. Peng, Y. Li, P.-C. Cao, X.-F. Zhu, and C.-W. Qiu, Adv. Funct. Mater. **30**, 2002061 (2020).
[75]D. Silver, S. Singh, D. Precup, and R. S. Sutton, Artif. Intell. **299**, 103535 (2021).