Supplementary Materials for
Decision-making and control with diffractive optical networks

1. Optical forward model

Diffractive optical network (DON) comprises multiple diffractive layers, where each unit on the diffractive layer acts as a neuron. The complex-valued transmission coefficient of the i-th neuron located at (x_i, y_i, z_i) of layer l is defined a

$$t_i^l(x_i, y_i, z_i) = a_i^l(x_i, y_i, z_i) \exp\left(j\phi_i^l(x_i, y_i, z_i)\right), \quad (S1)$$

where a and ϕ denote the modulation functions of amplitude and phase, respectively. Based on the Rayleigh-Sommerfeld diffraction equation, we can consider the neuron as a secondary source of a wave composed of the following optical mode

$$w_i^l(x, y, z) = \frac{z - z_i}{r^2}\left(\frac{1}{2\pi r} + \frac{1}{j\lambda}\right) \exp\left(\frac{j2\pi r}{\lambda}\right), \quad (S2)$$

where λ is the incident wavelength and r is the distance from (x_i, y_i, z_i) to (x, y, z). Therefore, the output function of the neuron can be written as

$$n_i^l(x, y, z) = w_i^l(x, y, z) t_i^l(x, y, z) \sum_k n_k^{l-1}(x, y, z), \quad (S3)$$

where k denotes the set of neurons of layer $l - 1$. These secondary waves diffract between the layers and interfere with each other, forming a complex wave at the surface of the next layer, feeding its neurons. In the training of DONs, the optical forward model can be computed by fast Fourier transform.

2. Proximal policy optimization

Deep reinforcement learning is a subfield of machine learning that integrates deep learning techniques. In traditional reinforcement learning, the agent learns to make decisions based on manually engineered features or tabular representations. However, in many complex tasks, it is difficult to manually design appropriate features, and the state and action spaces may be too large for tabular representations.

Deep reinforcement learning addresses these challenges by using deep neural networks to represent the policy or value function. The neural network takes raw inputs (such as image pixels in a game) and outputs actions or value estimates. The parameters of the neural network are learned from data collected during the interaction with the environment.

In this work, we used proximal policy optimization (PPO) to train the policy model. PPO is a model-free, online, on-policy, policy gradient reinforcement learning method. This algorithm is a type of policy gradient training that alternates between sampling data through environmental interaction and optimizing a clipped surrogate objective function using stochastic gradient descent. The clipped surrogate objective function improves training stability by limiting the new policy far from the old policy. The schematic of the network structure is shown in Figure S2.

The main objective is to learn the parameters θ of the actor by iteratively minimizing a sequence of loss functions, where the loss function is defined as

$$L(\theta) = \mathbb{E}\left(\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}\right)A^\pi(s,a), g\left(\epsilon, A^\pi(s,a)\right)\right), \quad (S4)$$

$$g(\epsilon, A) = \begin{cases}(1+\epsilon) & A \geq 0\\ (1-\epsilon) & A < 0\end{cases}, \quad (S5)$$

where $\mathbb{E}$ indicates the expectation over a finite batch of samples, π_θ is a stochastic policy, A^π is an estimator of the advantage function that describes how much better it is than others on average, ϵ is a hyperparameter, s and a are the state and action, and θ_k is the old parameters before θ.

3. Residual architecture

The residual net (ResNet) is a type of deep learning neural network structure widely used in many computer vision tasks. It has made significant breakthroughs in addressing the issues of vanishing and exploding gradients during the training of deep neural networks. The key to the ResNet is the introduction of residual modules, which are designed with shortcut connections. This involves introducing an unmodulated channel to preserve the original input features, then performing a sum operation with the output features of modulated channel. We note that when there is an angle between the polarization direction of incident light and the extraordinary axis of the liquid crystal of SLM, some light will not be modulated and reflected directly to the camera, thus preserve the features of the original input light signal and creating a shortcut connection without the need to add additional devices. Specifically, we can represent the Jones matrix of the SLM as $\begin{bmatrix}e^{i\Phi} & 0\\ 0 & \zeta\end{bmatrix}$, where ζ is a constant. This represents that the y-polarization channel applies the same modulation to each neuron of the input image, thus preserving the original features.

In addition, we test the prediction accuracy of DON and Res DON using MNIST dataset, and the results are shown in Table S1. It can be seen that the prediction accuracy of DON using residual structure is improved compared to the general structure, except when the network contains fewer layers, which also corresponds to the property that residual structure can train deeper networks.

4. Training of network

We use the open-source machine learning framework Pytorch to build the training algorithm. The training process is implemented using Python (v3.9.13), Gym (v0.21.0) environment, and Pytorch (v1.11.0) framework on a desktop computer with Intel Core i7-13700K CPU, Nvidia GeForce RTX 4070 Ti GPU, and 16 GB RAM.

We train each DON individually. The structure of each network is almost identical, except that the number of neurons in each layer is determined by the resolution of the SLM and the aspect ratio of the game, which are 1,080×1,080, 1,152×1,080, and 1,620×1,080 for Tic-Tac-Toe, Super Mario Bros., and Car Racing respectively. In the iterative process of the DON, α is 0.5, the learning rate is 0.01, the number of training epochs is 1,000, and constrain the phase-modulation range to 0–2π for

training. In addition, due to the unbalanced number of occurrences of output action, such as jump and crouch are far rarer than running in Super Mario Bros., we balance these actions by altering the weight each training example carries when computing the loss. Training of the network takes about tens of minutes. After training, the performance of networks is verified by interacting with the game environment. The phase profiles of networks are shown in the Figure S6–S8.

In addition, parameter initialization, as the starting point of network training, determines the initial training position. The choice greatly affects the convergence speed and the final training results. We evaluate the effect of different parameter initializations on the results. The results show that the normal initialization works best; see Figure S5 for details.

5. Modeling of control policy

We train different control policies on each game using the same network architecture, learning algorithms, and hyperparameter settings. That shows our approach is robust enough to work on various games while incorporating only minimal prior knowledge. PPO uses two neural network architectures to design and optimize the policy: the critic network and the actor-network. Both networks are adapted during training, but only the actor-network is deployed as the control policy. Specifically, for the actor-network, the input to the CNN consists of an 84×84 grayscale image produced by the preprocessing. The first hidden layer convolves 32 filters of 8×8 with stride 4 with the input image. The second hidden layer convolves 64 filters of 4×4 with stride 2. This is followed by a third convolutional layer convolving 64 filters of 3×3 with stride 1. The final hidden layer is fully-connected with 512 latents. Each hidden layer is followed by a rectified linear unit (ReLU). The output layer is a fully-connected linear layer with a single output for each valid action. The critic network is also a CNN which is about the same as the actor-network, except the output represents the discounted expected future reward for various actions. The reward functions are detailed in Supplementary Note 6. The learning rate is 0.001, and the discount is 0.99. In addition, because consecutive frames do not vary much, we can skip 4 intermediate frames without losing much information to accelerate training. Train of the control policy takes about an hour to a dozen hours. After the training, we save each game frame and the corresponding action for DON training.

6. Reward function

The goal of the agent is to maximize some notion of cumulative reward over a trajectory, and the reward function is the primary basis for altering the policy by making the task goal specific and numerical. Therefore, the reward function is critically important in reinforcement learning. In order to be effective, these approximate reward functions must also be positively correlated with the true reward. Combining the requirements of games, we set different reward functions for each game.

For Tic-Tac-Toe, since the goal is to achieve a win rate as high as possible, a reward of $+1$ for each win game, 0 for draw, and -1 for lose as a penalty.

For Super Mario Bros., the reward function assumes the objective of the game is to move as far right as possible, as fast as possible, without dying. To train the policy, some independent rules compose the reward. First, the difference in agent x position values between states is used as a reward, which in this case is the instantaneous velocity for the given step. Second, the difference in the game clock between frames is used as a reward prevents the agent from standing still. In addition, we set a large reward of 100 for passing the shortcut and a reward of -1 for jumping and crouching to avoid unnecessary actions. Finally, a death penalty of -15 penalizes the agent for dying in a state.

For Car Racing, the reward is -0.1 every frame and $+1000/N$ for every track tile visited, where N is the total number of tiles visited in the track.

7. Orthogonality of diffractive optical network

The inner product of any two light fields in the DON is invariant, and the operation to an optical field is a unitary transformation. Here, we define two similarity coefficients: field similarity coefficient and amplitude similarity coefficient. The projection from the unit vector along Q to the unit vector along W is

$$p_{QW} = \langle e_W, e_Q \rangle = \frac{\langle W, Q \rangle}{\sqrt{\langle Q, Q \rangle \langle W, W \rangle}}, \tag{S6}$$

where e_W and e_Q are the unit vectors along W and Q. The field similarity coefficient F_{QW} is defined as the square of the norm of p_{QW} as

$$F_{QW} = \left| p_{QW} \right|^2 = p_{QW}^* p_{QW} = \frac{\langle Q, W \rangle \langle W, Q \rangle}{\langle Q, Q \rangle \langle W, W \rangle}, \tag{S7}$$

which representing the similarity between the fields Q and W.

Due to the detector senses the amplitude of light, and the similarity of fields does not mean the similarity of amplitudes. We define the amplitude similarity coefficient between Q and W as

$$I_{QW} = \frac{\langle |Q|, |W| \rangle \langle |W|, |Q| \rangle}{\langle |Q|, |Q| \rangle \langle |W|, |W| \rangle} = \frac{\langle |Q|, |W| \rangle \langle |W|, |Q| \rangle}{\langle Q, Q \rangle \langle W, W \rangle}, \tag{S8}$$

which representing the similarity of amplitude distributions between Q and W. It can be seen that

$$0 \leq F_{QW} \leq I_{QW} \leq 1. \tag{S9}$$

Thus, according to the invariant inner product in the DON, the amplitude similarity coefficient is also constant no matter how we optimize the network. This means that if the input images are similar, the output has only tiny differences in intensity distribution. Therefore, the DON is unsuitable for predicting states with only local differences or high similarity.

Our results also confirm this feature of DON. In playing Tic-Tac-Toe, we counted the turn where the mistake occurred, as shown in Figure S3. We can easily find that the mistakes are concentrated in 4th turn. This is because as the number of marks increases, some states with high field similarity coefficients will appear, and the network has difficulty identifying these states to take the right action.

In addition, in playing Super Mario Bros., although the coherent frames of the game are highly similar, the entire scene is in constant motion, and some of the items change (e.g., gold coins), which significantly reduces the field similarity coefficient. Moreover, in playing Car Racing, we use the difference as a mechanism to trigger actions to identify tiny differences in intensity distribution, further avoiding the above problem.

8. Inverse prediction

In order to perceive the operation mechanism of the DON, we use inverse prediction to understand what features the DON has learned in Super Mario Bros. The method to map these features back to the input pixel space, showing what input pattern originally caused a given signal in the output. Specifically, the method similar to the iterative process of the network, except that the updated object is changed from the phase profile to the input image. Here, they are not used in any learning capacity, just as a probe of an already trained neural network. Considering that the signal of crouching only once in the game, the results can be well demonstrated. We input the maximal output signal of crouch into the trained DON, and by error back-propagation, get the input image that can make the signal of crouch maximal.

Although we cannot fully explain the black box of DON, through inverse prediction, we found that the features learned by DON show global property, learning the rough outline of the input image but ignoring the critical parts, which is different from the human visual system. This also gives guidance on how to get a better network: one of the simplest ideas is to make the contrast of the input image higher, so that the features are more distinct, and more generalized features can be learned.

9. Experimental setup

A He-Ne laser (HNL100L, Thorlabs) is used as the light source. A DMD (DLP7000, Texas Instruments) mounted on a controller board (F4100, X-digit), consists of 1,024×768 micromirrors with a pitch of 13.68 μm, which can modulate the incident light at a maximum speed of 18 kHz and display 8-bit grayscale images. A liquid-crystal-on-silicon (LCoS) SLM (PLUTO-2.1, HOLOEYE Photonics) with a pixel size of 8 μm and a resolution of 1,920×1,080 is used to modulate the phase of the wavefront. A grayscale CMOS camera (GS3-U3-51S5M, Teledyne FLIR) with a resolution of 2,448×2,048 and a pixel size of 3.45 μm is used to capture the output. Self-developed Python scripts control the above devices.

10. Adaptive training for digital micromirror device

Digital micromirror device is a binary device that steers light by using binary pulse width modulation (PWM) for each micromirror. Therefore, during the training process, we need to simulate the fast rotation of the micromirror in order to match the actual experimental system.

Specifically, because the DMD micromirrors can be either on or off, an image is created by turning on the mirrors corresponding to the bit set in a bit-plane. With binary PWM, the intensity level is reproduced by controlling the amount of time the micromirror is on. Thus, for a grayscale image, we can generate multiple binary images by converting the value of each pixel to a duty ratio, as shown in Figure S4(a). After that, the results can be computed by averaging the results of these binary images. Although the method has unavoidable inaccuracies, it works and has been experimentally validated.

However, using an amplitude spatial light modulator (SLM) instead of DMD is obviously a more accurate and convenient way, and this is what we will do in our future work.

11. Implementation of our approach on metasurfaces

Here, we have also implemented our proposed network on metasurfaces, the schematic of the metasurface as shown in Figure S9(a) and (b). By adjusting the diameter of these nanopillars, we can modify the transmitted phase of meta-atoms covering almost 2π and then construct the specific metasurfaces according to the relations between phase profiles and geometry parameters. Figure S9(c) shows the phase and transmittance vs. geometry parameters.

Figure S10–S12 show that the metasurface-based DON was similarly successful for the three games in the paper. The phase profiles of networks are shown in the Figure S13–S15. Despite the absence of photoelectric activation and residual architecture, similar results were achieved using more layers and additional mechanisms. The results revealing the potential of all-optical on-chip integrated artificial intelligence systems to solve complex real-world problems.

Table S1. The accuracy of MNIST

Network	Number of layers	Accuracy (%)
DON	3	94.64
	5	96.51
	7	96.74
Res DON	3	94.44
	5	97.12
	7	97.47

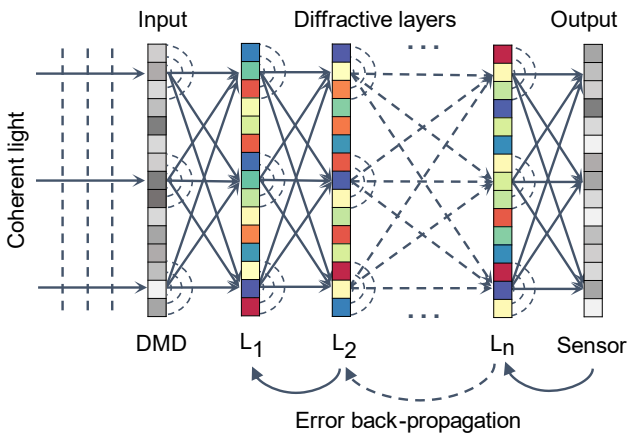

Figure S1. Schematic of DON. A DON comprises multiple diffractive layers, where each point on a given layer acts as a neuron with a complex-valued transmission coefficient. The transmission coefficients of each layer can be trained by using error back-propagation to perform a function between the input and output planes of the network.

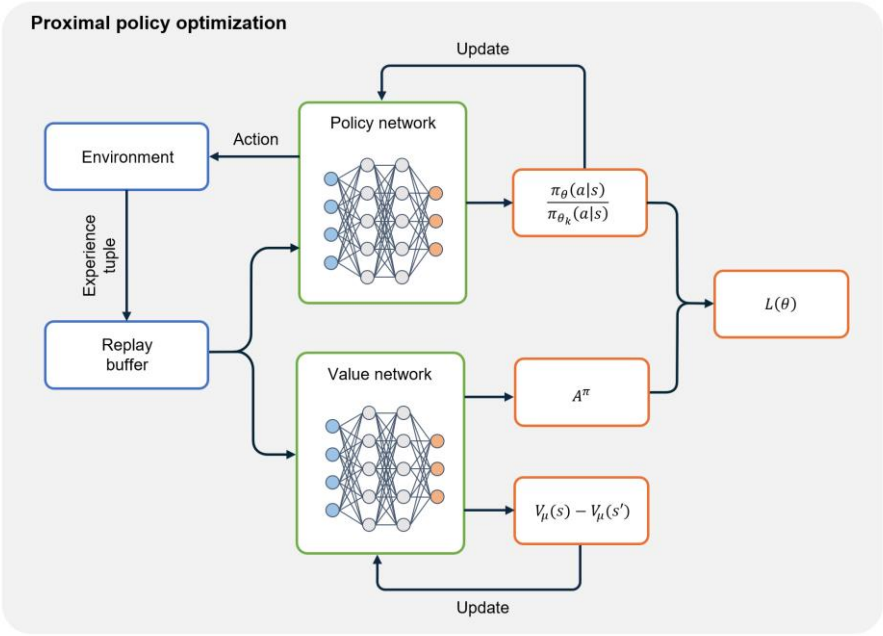

Figure S2. The schematic of PPO algorithm. Where V_μ is value function.

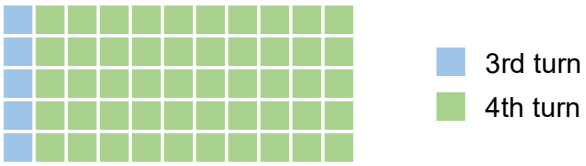

Figure S3. In which turn mistakes occurred. A total of 55 mistakes for the prediction of the O piece, 5 in the 3rd turn and 50 in the 4th turn.

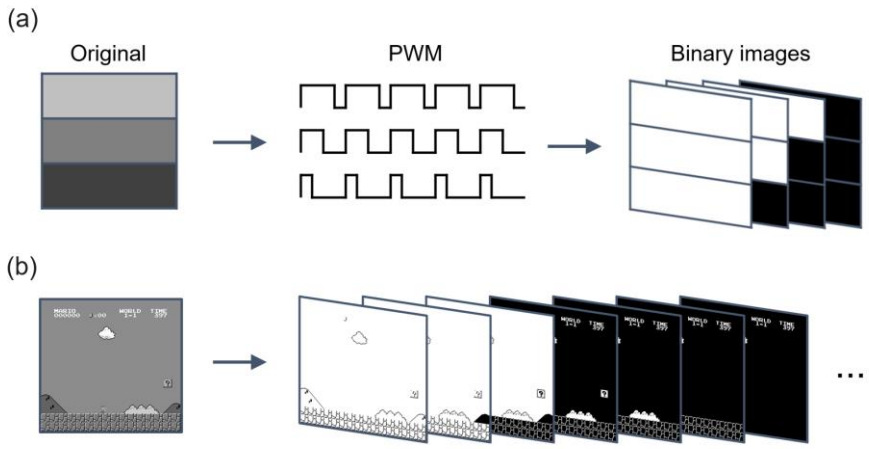

Figure S4. Adaptive training for digital micromirror device. (a) A grayscale image can be converted to multiple binary images by converting the value of each pixel to a duty ratio. Then, by means of these binary images, we can simulate the fast rotation of the micromirror. (b) An example of the method.

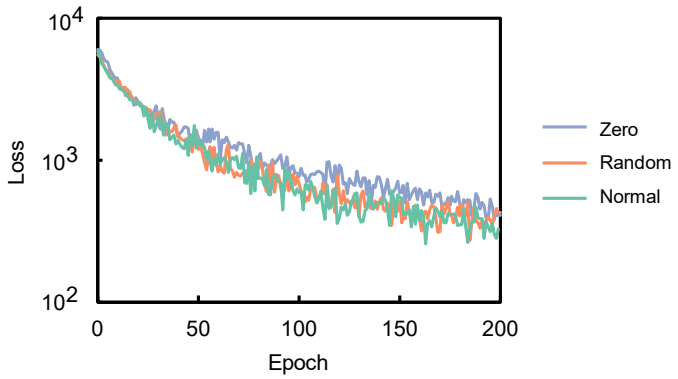

Figure S5. Effect of different parameter initializations. The training loss curves for Car Racing with different parameter initializations. We trained five times with different random seeds for each parameter initialization, and the curve corresponds to the average loss over the five trials. The results show that the loss decreases the fastest with normal initialization.

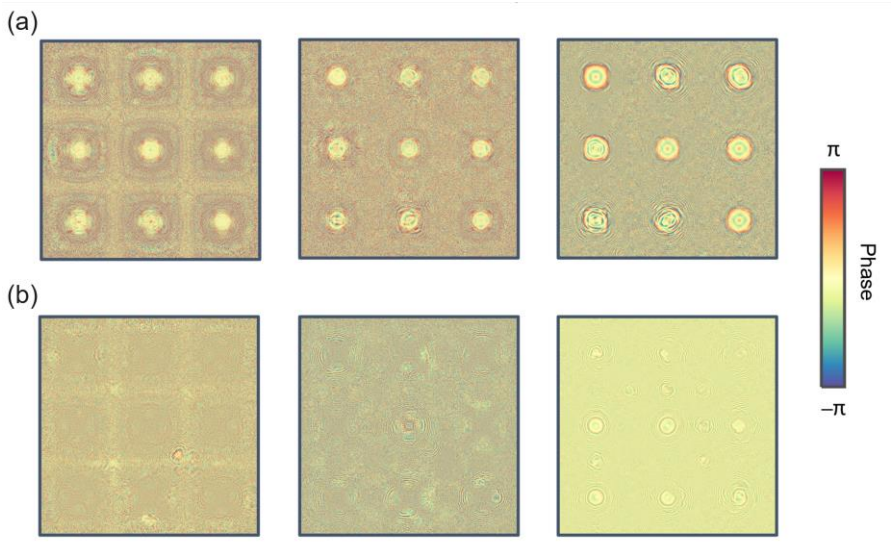

Figure S6. Phase profiles for playing Tic-Tac-Toe. The phase profiles of each layer of DON for Tic-Tac-Toe, (a) for the prediction of the X piece, and (b) for O.

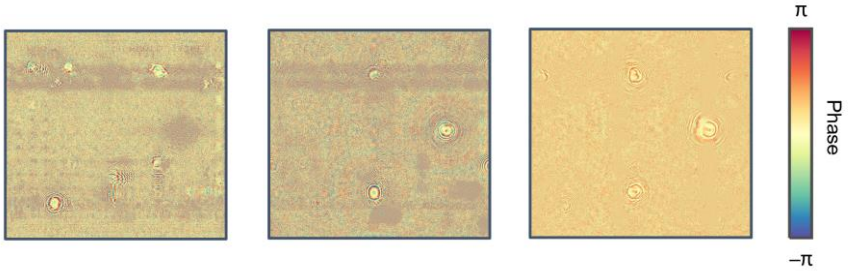

Figure S7. Phase profiles for playing Super Mario Bros. The phase profiles of each layer of DON for Super Mario Bros.

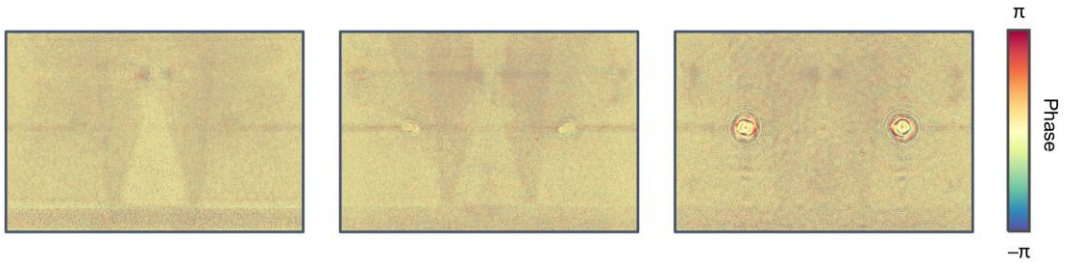

Figure S8. Phase profiles for playing Car Racing. The phase profiles of each layer of DON for Car Racing.

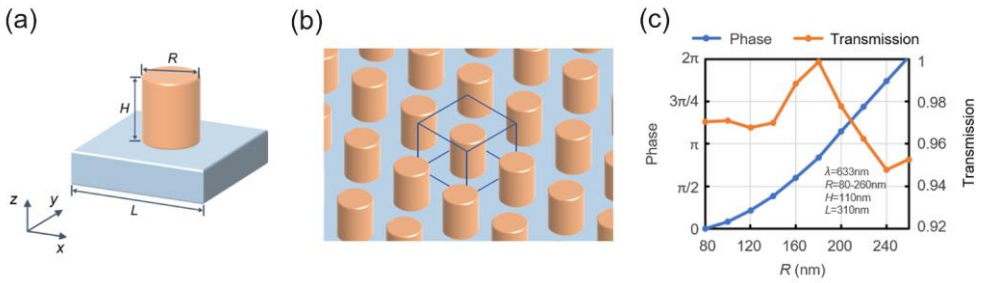

Figure S9. Schematic of the metasurface. (a) The unit cell with a tunable diameter R. Each unit cell acts as a neuron that has phase profiles trained by machine learning. (b) The schematic of the metasurface, and the layer-to-layer distance is 50 μm. (c) The phase and transmittance of metasurface.

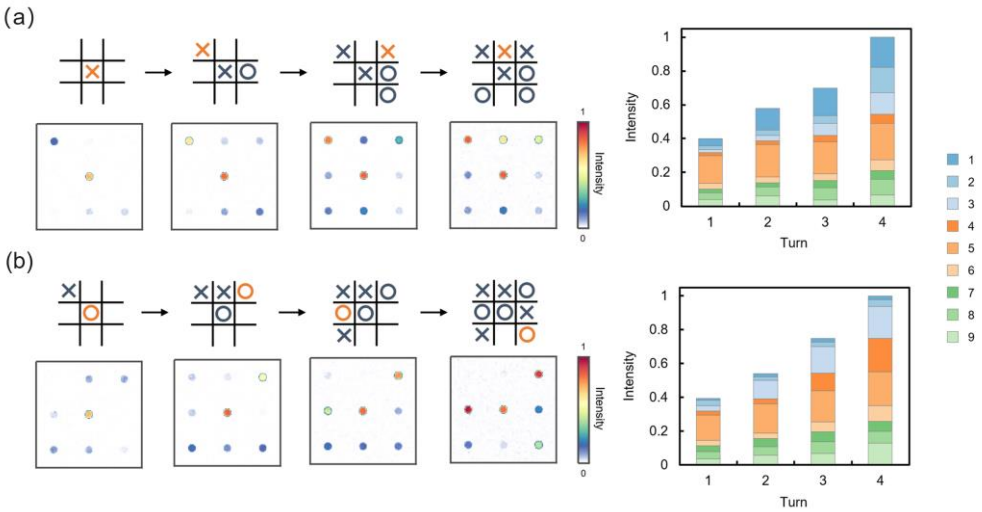

Figure S10. Playing Tic-Tac-Toe with metasurface-based DON. The DON consists of cascaded three-layer metasurfaces, the neuron numbers per layer of DON is 150×150.

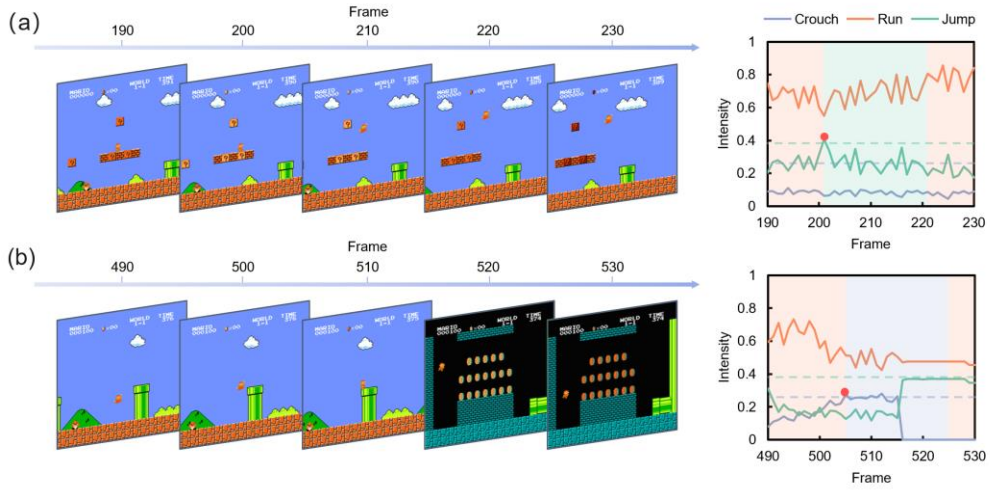

Figure S11. Playing Super Mario Bros. with metasurface-based DON. The DON consists of cascaded five-layer metasurfaces, the neuron numbers per layer of DON is 256×240. Unlike in the

paper, the most optimal action that Mario chooses to take is predicted by referring to the preset threshold measured in the training process.

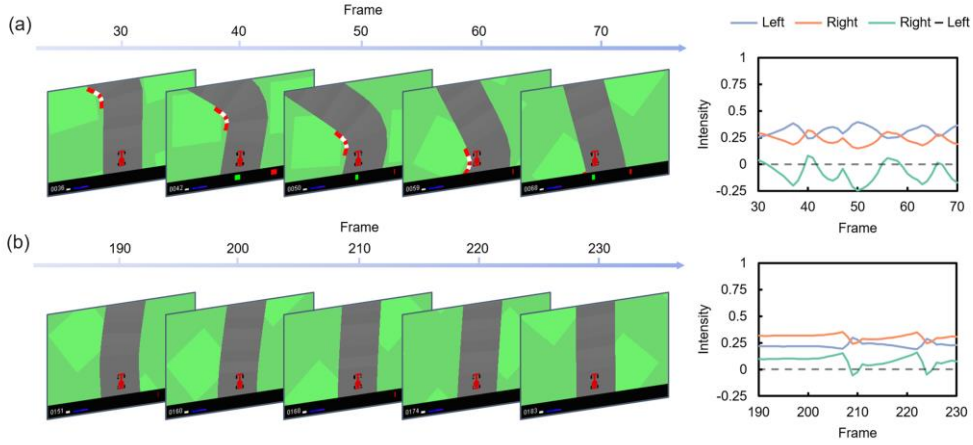

Figure S12. Playing Car Racing with metasurface-based DON. The DON consists of cascaded four-layer metasurfaces, the neuron numbers per layer of DON is 300×200.

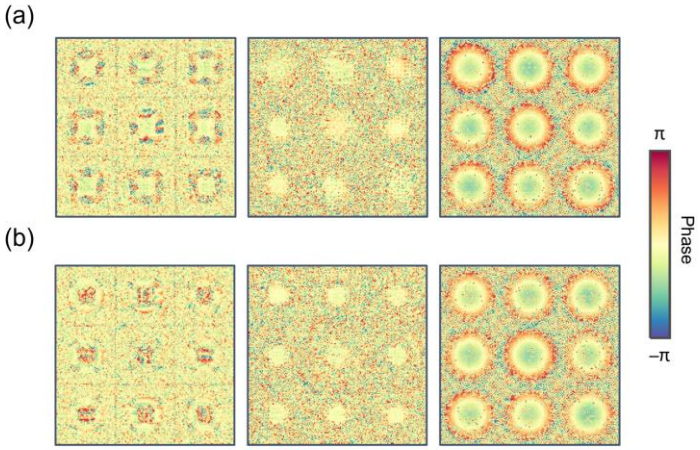

Figure S13. Phase profiles of metasurfaces for Tic-Tac-Toe.

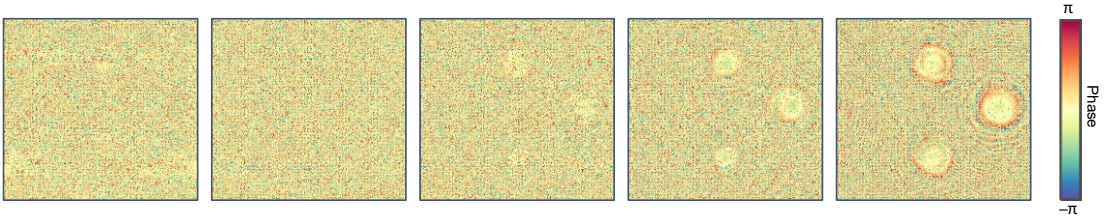

Figure S14. Phase profiles of metasurfaces for Super Mario Bros.

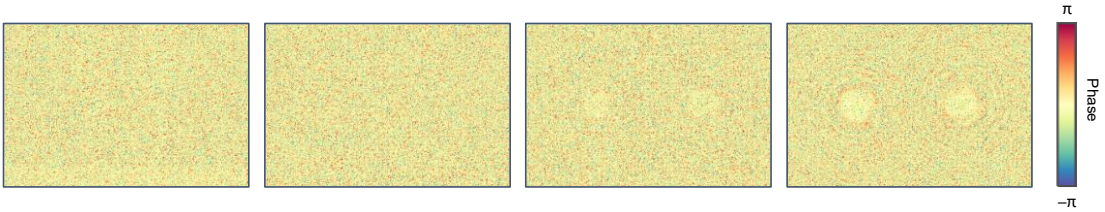

Figure S15. Phase profiles of metasurfaces for Car Racing.