Optics EXPRESS

Optimization of colour generation from dielectric nanostructures using reinforcement learning

IMAN SAJEDIAN,[1] TREVON BADLOE,[1] AND JUNSUK RHO[1,2,*]

[1] *Department of Mechanical Engineering, Pohang University of Science and Technology (POSTECH), Pohang 37673, Republic of Korea*
[2] *Department of Chemical Engineering, Pohang University of Science and Technology (POSTECH), Pohang 37673, Republic of Korea*
**jsrho@postech.ac.kr*

Abstract: Recently, a novel machine learning model has emerged in the field of reinforcement learning known as deep Q-learning. This model is capable of finding the best possible solution in systems consisting of millions of choices, without ever experiencing it before, and has been used to beat the best human minds at complex games such as, Go and chess, which both have a huge number of possible decisions and outcomes for each move. With a human-level intelligence, it has solved the problems that no other machine learning model has done before. Here, we show the steps needed for implementing this model to an optical problem. We investigate the colour generation by dielectric nanostructures and show that this model can find geometrical properties that can generate much purer red, green and blue colours compared to previously reported results. The model found these results in 9000 steps from a possible 34.5 million solutions. This technique can easily be extended to predict and optimise the design parameters for other optical structures.

1. Introduction

Plasmonic structures can be used to create high resolution images beyond the diffraction limit of light [1–4]. Due to their small dimensions, the resolution can be as high as 100k dots per inch [5–7]. Unfortunately, they suffer from optical losses and poor colour saturation which leads to a small colour gamut. Many designs using different types of shapes, structures and materials have been proposed to expand the achievable colour gamut as much as possible [8–11]. One design of note is based on an all dielectric nanostructure using silicon rods with an antireflective layer [8]. The colour gamut achieved was wider than previous reports, but the geometrical parameters could be optimised further to reach much purer colours. In this paper, we show a technique using reinforcement learning [12–14], which belongs to the artificial intelligence family, to find the best geometrical parameters to achieve the purest possible red, green and blue colours for a given structure.

We see colours from objects due to their specific reflection or transmission spectra [4,15]. Based on their absorptive and scattering properties, different objects shape the light reaching our eyes in different ways. This fact is used in structural colour printing to design periodic, metallic [2,5,16] or dielectric [17–19] nanostructures of specific dimensions to produce the desired reflection spectrum, and therefore colour. The resonances from the nanostructure can be easily controlled by changes in the geometrical properties, such as periodicity, shape, and size. Due to fabrication limitations, so far only simple shapes such as circular rods, square rods and crosses have been utilised.

Recently, neural networks have been used in the design and inverse design of nano-photonic structures [20–23], and for the design of chiral metamaterials [24]. Two methods are usually used; simple neural networks consisting of hidden layers, and generative adversarial networks (GANs). Both have the undesirable property of only being able find parameters

#348664 https://doi.org/10.1364/OE.27.005874
Journal © 2019 Received 19 Oct 2018; revised 26 Nov 2018; accepted 4 Dec 2018; published 18 Feb 2019

inside the limits of the data that they have been trained with, and are unable to generate new points outside the extremities of this data. To find the extreme limits of a specific design, a reinforcement learning method should be used.

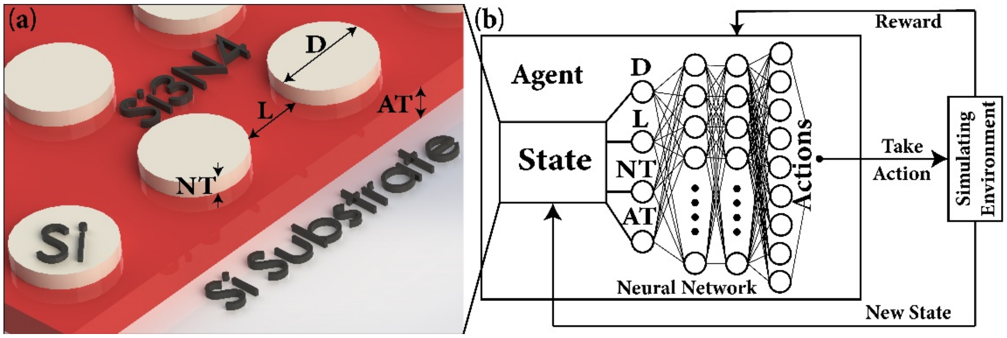

Fig. 1. The all-dielectric structure and reinforcement model. (a) A diagram of the structure used. Silicon (Si) nanodisks on an antireflective layer of silicon nitride (Si_3N_4), on an Si substrate. The diameter and thickness of nanodisks are given by D and NT, and the distance between them is given by L. The thickness of Si_3N_4 layer is given by AT. (b) A schematic of the design of the reinforcement model.

Reinforcement learning, which belongs to the artificial intelligence family, is a method in which an agent tries to find the best decision at each step, based on the given rewards. Reinforcement learning has been shown to be capable of beating many Atari games [12,25–27] and even beating human masterminds at games such as Go, backgammon and chess [28–31]. In these games, there are millions of different possible states that the agent could find themselves in, and the program should find the best action to take at each step, to maximise the chance of gaining the reward in the end. There are many branches of reinforcement learning like swarm intelligence and genetic algorithms but human-level intelligence was achieved recently by deep Q-learning [12]. A detailed benchmark of this model compared to other reinforcement learning methods and an expert human player on 50 different Atari games was done in that work. Deep Q-learning managed to achieve very high scores even compared to an expert human being. The question may be asked of how the benchmark of Atari games can be valid for an optical problem. The simple answer is because of the nature of neural networks. Since neural networks input and output values are basic numerical data, the origin of the data is not important. In other words, it is a framework that can be used for any subject. Similar neural networks can be used in image processing, cancer diagnostics, sales prediction, text translating and physical problems [20,21]. In this paper, we use deep Q-learning to optimise the dimensions for a specific silicon (Si)-based, all dielectric colour printing design, show in Fig. 1(a). This model can easily be extended to other optical problems.

2. Methods

Below is a simple example of how reinforcement learning works. To optimise the size of the radius of a nanostructure, first, starting from an arbitrary radius, the size should be gradually increased or decreased while receiving the feedback at every stage. By finding a connection between the radius and results, a relationship can be formed. For example, assume the goal is to achieve 100 points, and a few simulations are done as is shown in Table 1.

Table 1. Steps for a simplified reinforcement learning algorithm

Step	*Current state*	*Action*	*New state*	*Reward for the new state*
1	*Radius = 10 nm*	*Algorithm start point*	*Radius = 10 nm*	*10 points*
2	*Radius = 10 nm*	*Decrease the radius*	*Radius = 5 nm*	*5 points*
3	*Radius = 5 nm*	*Increase the radius*	*Radius = 10 nm*	*10 points*
4	*Radius = 10 nm*	*Increase the radius*	*Radius = 15 nm*	*15 points*

At the end of step 4, human intuition tells us that a radius of 100 nm will likely give 100 points. Exactly the same happens in reinforcement learning, after doing some exploration and learning the relationship between the action and the feedback, it heads towards the goal. The key difference is that the agent can only move in steps defined by the given actions until it reaches the goal. In this simple example, the model would learn that increasing the radius leads to a higher score, and would therefore keep increasing the radius until the goal is met. Now we discuss how reinforcement learning can be applied to a physical problem.

In order to apply reinforcement learning to a physical problem, an environment should be created that takes an action from the agent and gives a reward based on that action. Based on the reward, a neural network decides the best action at each step and keeps updating itself to continuously keep training over different states. A schematic of the algorithm is shown in Fig. 1(b). The actions are defined as specific changes in the geometrical properties of the structure, and the reward is defined based on the colour difference between the target colour and the generated colour by the structure.

2.1 Deep Q-learning model

The deep Q-learning model can learn the best way to act in any given situation. This method has had huge success in completing many Atari games without having to change the algorithm. This is a huge leap towards a general artificial intelligence model that can solve any given problem, like a human brain [12].

This algorithm can be explained best by using the example of playing computer games. The game has an environment that changes, depending on the actions, i.e. the input from a game controller given to it. For each action, there is a change of state in the game. At each point, the game gives feedback though a reward system, such as, gaining points or losing a life. To write an algorithm to beat a game, we should consider all different actions at all possible states, and their relative rewards. The rules for how the actions are chosen is called a policy. The set of states, actions, and policies, form a Markov decision process, as shown below [13]:

$$s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_{n-1}, a_{n-1}, r_n, s_n \tag{1}$$

where, s is the state, a is the action and r is the reward. S_n is the terminating state, which will be the target, in terms of the computer game example this would be the end state of the game. At state s_0, the total reward from taking specific actions can be given by the following expression, known as the discounted future reward:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^{n-t} r_n = r_t + \gamma(r_{t+1} + \gamma(r_{t+2} + \ldots)) = r_t + \gamma R_{t+1} \tag{2}$$

where γ is the discount factor, a value between 0 and 1 which can be optimised as needed. The algorithm should maximise the discounted future reward (R_t). In deep Q-learning, this is achieved by defining a function, $Q\ (s,\ a)$, which represents the maximum discounted future reward for each action at a given state. At each state, the action with the highest Q is chosen, this is called the policy. This function is optimised by a neural network at each step, to find the policy that can choose an action that achieves the highest future reward, even for an unknown state. The policy is defined as:

$$\pi(s) = \arg\max_a Q(s,a) \quad (3)$$

Argmax$_a$ chooses the action for which Q is maximised. Analogous to the discounted future reward (Eq. (3)), this Q is defined as:

$$Q(s,a) = r + \gamma \max_{a'} Q(s',a') \quad (4)$$

This is known as the Bellman equation [13].

Table 2. Definitions of actions used in the reinforcement model.

Action No.	Action Definition
0	Decrease the spacing between disks (L) by 5nm. (min 5nm)
1	Increase the spacing between disks (L) by 5nm. (max 500nm)
2	Decrease the nanodisks diameter (D) by 5nm. (min 10nm)
3	Increase the nanodisks diameter (D) by 5nm. (max 500nm)
4	Decrease the nanodisks thickness (NT) by 5nm. (min 5nm)
5	Increase the nanodisks thickness (NT) by 5nm. (max 500nm)
6	Decrease the antireflective layer thickness (AT) by 5nm. (min 10nm)
7	Increase the antireflective layer thickness (AT) by 5nm. (max 200nm)
8	Do nothing

The final step is to connect a deep neural network, which connects the states to the Q functions. Neural networks need databases to train from. First, the model attempts to create data by exploring. After this the trained model can predict new states. To create the initial database, a method known as ε-greedy exploration was used. For each iteration, a random number (between 0 and 1) is chosen, if this number is smaller than ε (which is also between 0 and 1), then a random action is performed, if it is bigger than ε, an action determined by the network is performed. At the beginning of the learning process, ε was set to a number near 1 then decreased at each step, to a non-zero minimum, which always allows the model some chance to explore. Using this method, a database of random states and actions from which the model can be trained was built. All the states, actions and rewards are added to memory, from which the model picks states at random to be trained on. This assures that the model remembers what it has done before and that the prediction is unbiased. A flowchart for this model is shown in Fig. 1(b).

Here, we used an improved version of deep Q-learning known as double deep Q-learning (DDQN) [32]. This version is shown to have a better performance over normal deep Q-learning and less overestimation in many cases. To apply this method, two similar models are created instead of one main model. One of them trains the target, known as the target model, and one acts as the main model, as shown in Fig. 2. At each iteration, the weights of the target model are gained from the combination of the main model and the target model weights, by the following formula:

$$Tw = Mw \times \tau + Tw \times (1-\tau) \quad (5)$$

Here Tw is the target model weights, Mw is the main model weights and τ is a hyper parameter that should be tuned by the performance of the model. The target, which is the prediction of the model for each action (the action with the highest target will be chosen) is obtained from the addition of the targets predicted from the old state (from previous step) and the targets predicted from the new state (from the current step, known as Q values) at each iteration. The main model is then trained on the current state and the target as is shown in Fig. 2.

To apply these techniques to optical problems, the following steps should be taken:

1. Define an environment which takes an action and gives a reward.

2. Define the actions.

3. Define a reward system.

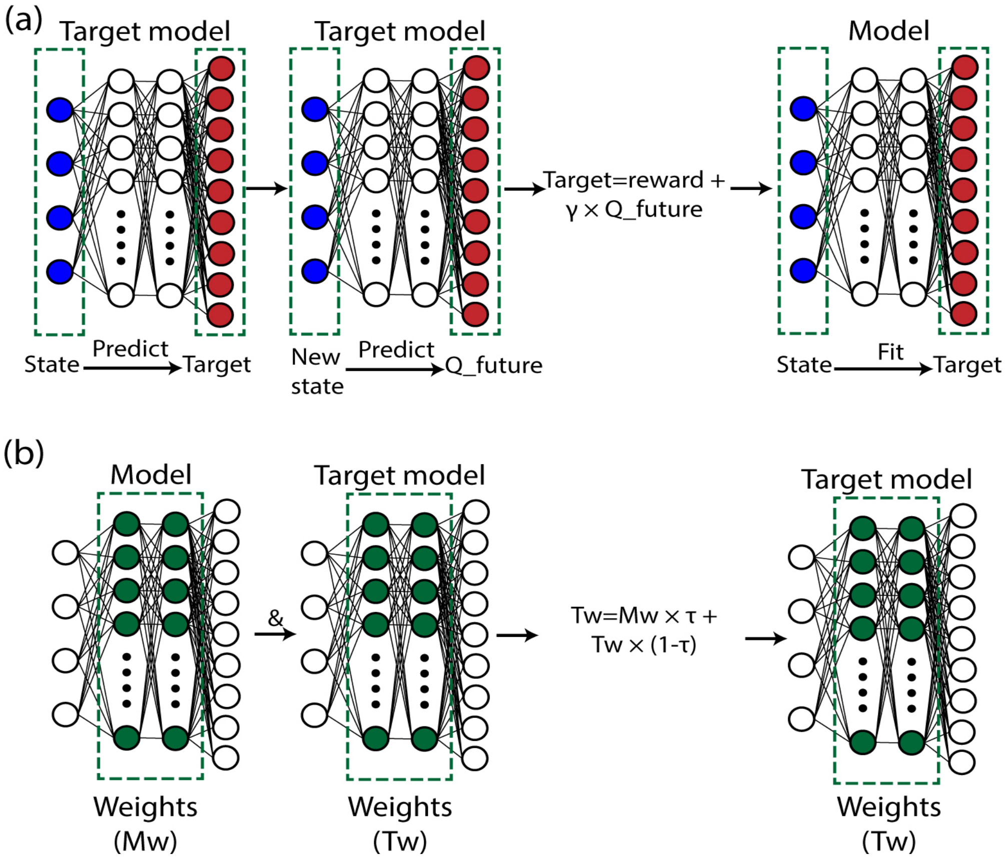

Fig. 2. The double deep Q-learning model structure. Two similar models are created. One for predicting targets named as target model and one as the main model. (a) The targets are found by the combination of the reward and the Q-values predicted by the target model. The targets found by the target model are then used to train the main model. This procedure is repeated at each iteration. (b) The weights from the main model and the target model are combined to update the target model weights at each iteration.

2.2 The environment and actions

The structures design is shown in Fig. 1(a). This structure was proposed in recently published work [8]. Here, we show that the structure can produce much purer red, green and blue responses than what was reported in the paper, by further optimising the geometrical properties. The structure consists of silicon (Si) nanodisks on an antireflective silicon nitride (Si_3N_4) film on an Si substrate. In the original paper, the thickness of the antireflective layer and nanodisks were kept constant, while the diameter of the nanodisks was varied from 40 to 270 nm and the spacing between the disks from 10 to 120 nm. Here, all geometrical parameters are variable and only limited by minimum and maximum sizes over a large range and the minimum step size. The geometrical variables and limits were as follows:

- Spacing between disks (L): 5 – 500 nm; step size: 5 nm; number of steps: 99.

- Nanodisks diameter (D): 10 – 500 nm; step size: 5 nm; number of steps: 98.
- Nanodisks thickness (NT): 5 – 500 nm; step size: 5 nm; number of steps: 99.
- Antireflective layer thickness (AT): 10 – 200 nm, step size: 5 nm; number of steps: 38.

Giving a total number of states = 99 × 98 × 99 × 38 = 36,498,924. Simulating this number of possible states would require a significant amount of time and computational effort.

In order to create an environment for reinforcement learning, actions and rewards must be defined. Using the variables and limits set out above, 9 actions were defined, as shown in Table 2.

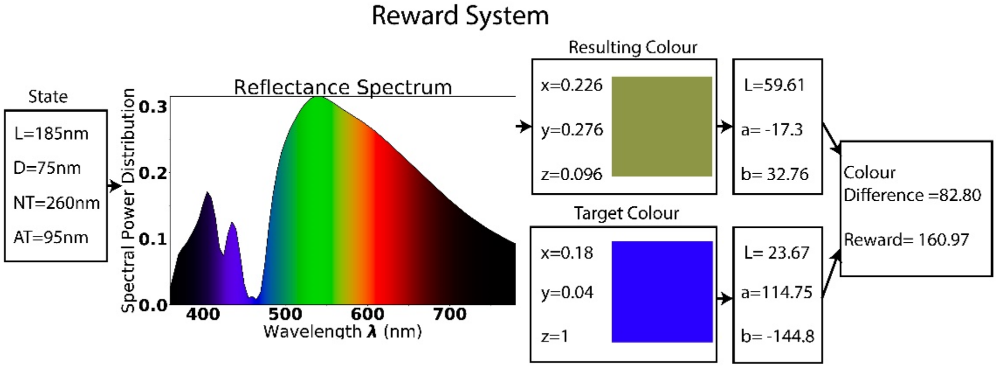

Fig. 3. The reward system used for the reinforcement model. Each state is converted to a reflectance spectrum by simulation software. The reflected spectrum is converted to XYZ colour values. The XYZ colour values are then converted to L*a*b* colour values. The reward is calculated from the colour difference between the resulting colour from the spectrum and the target colour.

2.3 Reward system

The goal of the model to find the structure that generates the closest possible colour to whatever we define as the target. The reward was defined based on the smallest colour difference in L*a*b* space, between the simulated response and red, green and blue, defined as purest red, green and blue. To calculate the colour difference, first the reflection spectrum of the structure was required. The current state, the set of L, D, NT, and AT values, were sent to the simulation software. Using these values, the reflection spectrum was simulated and extracted. To convert this spectrum to XYZ tristimulus colour values, defined by CIE [33], a python package named colour-science for colour operations [34] was used. As XYZ values are not linearly comparable to their respective colour, the colour difference between the target and the reflection spectrum was calculated in L*a*b* colour space using CIE delta E 2000(CIEDE2000) [35]. Since the colour difference gets smaller as the target colour is reached, but the reward should increase reward, the following formula was used to calculate the reward:

$$reward = (200 - CIEDE(obtained\ color - target\ color))^3 / 10000 \qquad (6)$$

200 is big enough to make the colour difference positive. To amplify the reward as the model gets closers to the target, the difference was raised to the power of three. Dividing by 10,000 was done to make the results easier to read. After this process, the higher reward has more value than lower reward. An example of this process is shown in Fig. 3.

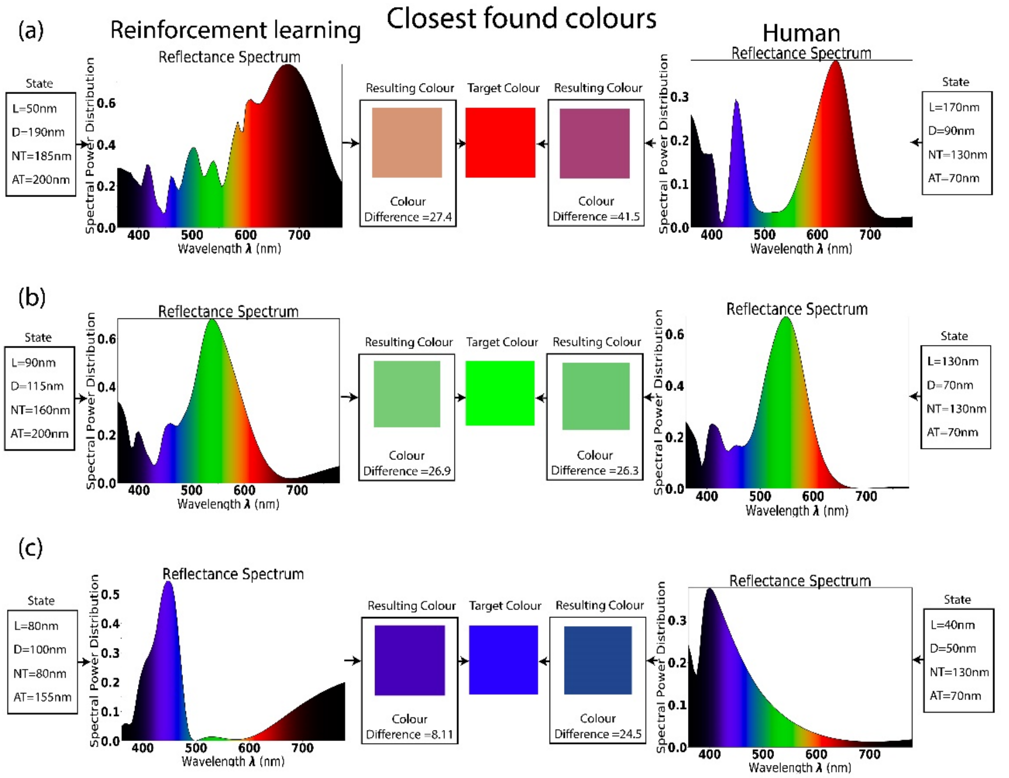

Fig. 4. The geometrical properties of the closest colours found by reinforcement learning (RL), for red (a), green (b) and blue (c) colours compared to those found by human researchers (calculated from geometrical properties of best found colours given in [8]). As can be seen the RL model found much closer colours for red and blue and almost the same green.

To sum up, the process can be defined as follows:

1- Start from a reference geometrical point in the first step, or the provided geometrical data from the DDQN.

2- Create a simulation from the geometrical data provided from the previous step.

3- Run the simulation and save the reflection data.

4- Extract the colour from reflection data and return the reward to the DDQN.

5- If the algorithm reaches the target colour, exit the code.

6- Save the reward if it is the new maximum.

7- The DDQN does the training and picks the correct action.

8- Create new geometrical data from the given action.

9- Repeat from step 1.

Here, the model had 3 hidden layers with 24, 48, 24 neurons each, with an Adam optimiser with a learning rate of 0.005. The epsilon decay rate was 0.995 starting from 0.95 to 0.1.

3. Results

All the simulations of this project were done in a commercial FDTD package, Lumerical. A computer with a strong CPU and GPU were used for this project. This ensures fast FDTD simulations and a fast machine learning procedure at the same time. A machine with a 16-core 3.40 GHz processor, 64 GB of RAM, and a NVIDIA GTX 1080ti GPU with 11GB DDR5X RAM was used. The code for reinforcement learning and plots were done in Python with the help of TensorFlow, Keras and colour packages. Pure red, green and blue colours were set as targets. As is shown in Fig. 4 the colours found by reinforcement learning in the red and blue cases are much closer to the pure colours than those found in the previous research and the green colour is almost the same. The reflection spectra and geometrical properties of the results are also shown. The best geometrical properties found by this model are L = 50 nm, D = 190 nm, NT = 185 nm and AT = 200 nm for red, L = 90 nm, D = 115 nm, NT = 160 nm and AT = 200 nm for green, L = 80 nm, D = 100 nm, NT = 80 nm and AT = 155 nm for blue.

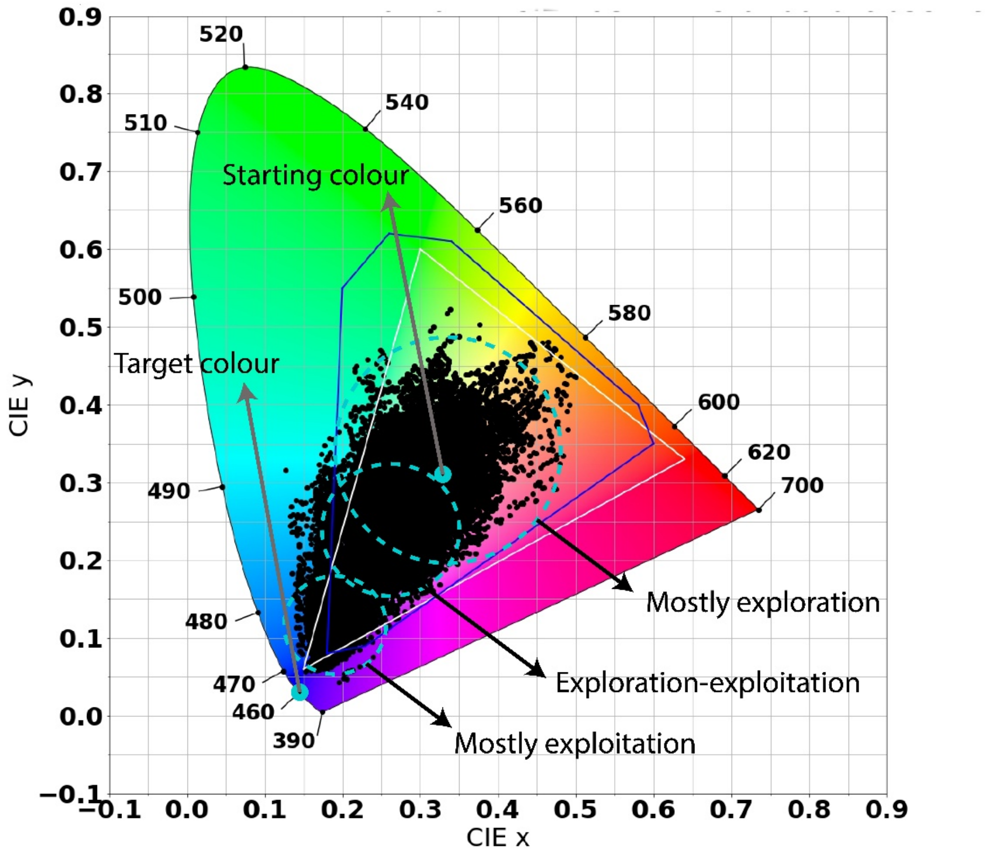

Fig. 5. The progress of the DDQN model to reach pure blue. Each black dot shows a single structure. The model starts from an arbitrary point and after some exploration goes towards the target colour. The white triangle is the sRGB coverage and the blue outline is the colour coverage found in the previous research.

The whole training was done over 18 attempts containing 500 steps each, in a total of 9,000 steps for each colour. It took around one week to run the model each time, with almost all the time being used on the simulations. Figure 5 shows the DDQN's progress to reach the target of blue from a starting point.

A drawback of this method is that it only works for one target at a time. So, now it is not possible to find the best structures for red, green and blue colours simultaneously. This is because the agent can only go towards one goal at a time. To find three targets at a time, three agents would be needed, i.e. running the whole code three times as we did here. As with other machine learning methods there are many hyper parameters to tune to achieve the best model. Developing a working model takes a long time, but once a successful model is created, it can easily be extended to work for similar problems.

4. Discussion

Deep Q-learning was used to find the best parameters to design structures for a physical problem, namely all-dielectric reflective colour filters. The model was able to find much more optimised dimensions for designing the structures to achieve purer red, green and blue colours compared to the ones achieved in previous work. Using deep Q-learning allowed the optimisation of the best structure for each colour in a few number of steps compared to the total number of possibilities. Since the DDQN acts like an intelligent sweep, it is useful in situations where many options are available to explore. Although the coding of a DDQN is hard compared to other similar machine learning methods, once it is written, it can easily be extended to other physical problems and used as a tool for optimising nanosurface and nanostructure designs.

Funding

National Research Foundation grants (NRF-2017R1E1A1A03070501, NRF2018M3D1A1058998, NRF-2015R1A5A1037668 & CAMM-2019M3A6B3030637) funded by the Ministry of Science and ICT, Republic of Korea.

Acnolwedgments

J.R. and I.S. conceived the concept and initiated the project. I.S. designed the research and performed numerical calculations. I.S., J.R. and T.B. wrote the manuscript. All authors contributed to the discussion, analysis and confirmed the final manuscript. J.R. guided the entire the project.

References

1. Y. Gu, L. Zhang, J. K. Yang, S. P. Yeo, and C.-W. Qiu, "Color generation via subwavelength plasmonic nanostructures," Nanoscale **7**(15), 6409–6419 (2015).
2. S. J. Tan, L. Zhang, D. Zhu, X. M. Goh, Y. M. Wang, K. Kumar, C.-W. Qiu, and J. K. Yang, "Plasmonic color palettes for photorealistic printing with aluminum nanostructures," Nano Lett. **14**(7), 4023–4029 (2014).
3. S. Yokogawa, S. P. Burgos, and H. A. Atwater, "Plasmonic color filters for CMOS image sensor applications," Nano Lett. **12**(8), 4349–4354 (2012).
4. A. Kristensen, J. K. Yang, S. I. Bozhevolnyi, S. Link, P. Nordlander, N. J. Halas, and N. A. Mortensen, "Plasmonic colour generation," Nat. Rev. Mater. **2**(1), 16088 (2017).
5. X. M. Goh, Y. Zheng, S. J. Tan, L. Zhang, K. Kumar, C.-W. Qiu, and J. K. Yang, "Three-dimensional plasmonic stereoscopic prints in full colour," Nat. Commun. **5**(1), 5361 (2014).
6. X. Zhu, C. Vannahme, E. Højlund-Nielsen, N. A. Mortensen, and A. Kristensen, "Plasmonic colour laser printing," Nat. Nanotechnol. **11**(4), 325–329 (2016).
7. A. S. Roberts, A. Pors, O. Albrektsen, and S. I. Bozhevolnyi, "Subwavelength plasmonic color printing protected for ambient use," Nano Lett. **14**(2), 783–787 (2014).
8. Z. Dong, J. Ho, Y. F. Yu, Y. H. Fu, R. Paniagua-Dominguez, S. Wang, A. I. Kuznetsov, and J. K. W. Yang, "Printing beyond sRGB color gamut by mimicking silicon nanostructures in free-space," Nano Lett. **17**(12), 7620–7628 (2017).
9. P. R. Wiecha, A. Arbouet, C. Girard, A. Lecestre, G. Larrieu, and V. Paillard, "Evolutionary multi-objective optimization of colour pixels based on dielectric nanoantennas," Nat. Nanotechnol. **12**(2), 163–169 (2017).
10. L. Wang, R. J. H. Ng, S. Safari Dinachali, M. Jalali, Y. Yu, and J. K. Yang, "Large area plasmonic color palettes with expanded gamut using colloidal self-assembly," ACS Photonics **3**(4), 627–633 (2016).
11. T. D. James, P. Mulvaney, and A. Roberts, "The plasmonic pixel: large area, wide gamut color reproduction using aluminum nanostructures," Nano Lett. **16**(6), 3817–3823 (2016).

12. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," Nature **518**(7540), 529–533 (2015).
13. R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction* (MIT, 1998).
14. L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," JAIR **4**, 237–285 (1996).
15. K. Busch, G. Von Freymann, S. Linden, S. Mingaleev, L. Tkeshelashvili, and M. Wegener, "Periodic nanostructures for photonics," Phys. Rep. **444**(3-6), 101–202 (2007).
16. K. Kumar, H. Duan, R. S. Hegde, S. C. Koh, J. N. Wei, and J. K. Yang, "Printing colour at the optical diffraction limit," Nat. Nanotechnol. **7**(9), 557–561 (2012).
17. X. Zhu, W. Yan, U. Levy, N. A. Mortensen, and A. Kristensen, "Resonant laser printing of structural colors on high-index dielectric metasurfaces," Sci. Adv. **3**(5), e1602487 (2017).
18. Y. Shen, V. Rinnerbauer, I. Wang, V. Stelmakh, J. D. Joannopoulos, and M. Soljacic, "Structural colors from Fano resonances," ACS Photonics **2**(1), 27–32 (2015).
19. H. Kim, J. Ge, J. Kim, S.-e. Choi, H. Lee, H. Lee, W. Park, Y. Yin, and S. Kwon, "Structural colour printing using a magnetically tunable and lithographically fixable photonic crystal," Nat. Photonics **3**(9), 534–540 (2009).
20. D. Liu, Y. Tan, E. Khoram, and Z. Yu, "Training deep neural networks for the inverse design of nanophotonic structures," ACS Photonics **5**(4), 1365–1369 (2018).
21. J. E. Peurifoy, Y. Shen, L. Jing, F. Cano-Renteria, Y. Yang, J. D. Joannopoulos, M. Tegmark, and M. Soljacic, "Nanophotonic inverse design using artificial neural network," in *Frontiers in Optics*, (Optical Society of America, 2017), FTh4A. 4.
22. I. Malkiel, A. Nagler, M. Mrejen, U. Arieli, L. Wolf, and H. Suchowski, "Deep learning for design and retrieval of nano-photonic structures," https://arxiv.org/abs/1702.07949 (2017).
23. J. Peurifoy, Y. Shen, L. Jing, Y. Yang, F. Cano-Renteria, B. G. DeLacy, J. D. Joannopoulos, M. Tegmark, and M. Soljačić, "Nanophotonic particle simulation and inverse design using artificial neural networks," Sci. Adv. **4**, eaar4206 (2018).
24. W. Ma, F. Cheng, and Y. Liu, "Deep-Learning-Enabled On-Demand Design of Chiral Metamaterials," ACS Nano **12**(6), 6326–6334 (2018).
25. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," http://arXivpreprintarXiv:1312.5602 (2013).
26. V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016), 1928–1937.
27. T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," http://arXivpreprintarXiv:1509.02971 (2015).
28. D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, and T. Graepel, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," http://arXivpreprintarXiv:1712.0181 (2017).
29. G. Tesauro, "Temporal difference learning and TD-Gammon," Commun. ACM **38**(3), 58–68 (1995).
30. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," Nature **529**(7587), 484–489 (2016).
31. D. Silver, R. S. Sutton, and M. Müller, "Reinforcement Learning of Local Shape in the Game of Go," in *IJCAI*, 2007), 1053–1058.
32. H. Van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," in *AAAI*, (Phoenix, AZ, 2016), 5.
33. C. CIE, *Commission internationale de l'eclairage proceedings* (Cambridge University, 1932).
34. T. M. Mansencal, M. Parsons, M. Canavan, L. Cooper, S. Shaw, N. Wheatley, K. Crowson, K. L. Ofek, "Colour Science for Python 0.3.11," (2018).
35. G. Sharma, W. Wu, and E. N. Dalal, "The CIEDE2000 color□difference formula: Implementation notes, supplementary test data, and mathematical observations," Color Research & Application: Endorsed by Inter□ Society Color Council, The Colour Group (Great Britain), Canadian Society for Color, Color Science Association of Japan, Dutch Society for the Study of Color, The Swedish Colour Centre Foundation, Colour Society of Australia, Centre Français de la Couleur **30**, 21–30 (2005).