Visual Analysis of Deep Q-network

Dewen Seng, Jiaming Zhang, and Xiaoying Shi*
School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China
[e-mail : sengdw@hdu.edu.cn, z1198647915@gmail.com, shixiaoying@hdu.edu.cn]
*Corresponding author: Xiaoying Shi

Received May 26, 2020; revised September 1, 2020; revised December 16, 2020; accepted February 25, 2021; published March 31, 2021

Abstract

In recent years, deep reinforcement learning (DRL) models are enjoying great interest as their success in a variety of challenging tasks. Deep Q-Network (DQN) is a widely used deep reinforcement learning model, which trains an intelligent agent that executes optimal actions while interacting with an environment. This model is well known for its ability to surpass skilled human players across many Atari 2600 games. Although DQN has achieved excellent performance in practice, there lacks a clear understanding of why the model works. In this paper, we present a visual analytics system for understanding deep Q-network in a non-blind matter. Based on the stored data generated from the training and testing process, four coordinated views are designed to expose the internal execution mechanism of DQN from different perspectives. We report the system performance and demonstrate its effectiveness through two case studies. By using our system, users can learn the relationship between states and Q-values, the function of convolutional layers, the strategies learned by DQN and the rationality of decisions made by the agent.

Keywords: Visual Analytics, Deep Reinforcement Learning, Deep-Q-networks

This research was supported by National Natural Science Foundation of China (No. 61903109).

http://doi.org/10.3837/tiis.2021.03.003 **ISSN : 1976-7277**

1. Introduction

Reinforcement learning (RL) is a prominent learning method to enhance the efficacy of autonomous systems [1]. Such methods build an autonomous agent to achieve the desired goal while interacting with an environment. The agents need to derive efficient representations of the environment from high-dimensional inputs and use these representations to learn an optimal policy.

Traditional reinforcement learning methods extracted hand-crafted features and became unstable when adopting a nonlinear function to approximate the action-value function, which limited their applicability in different domains. As the booming of deep learning, traditional RL methods were optimized by deep neural networks, and deep reinforcement learning (DRL) methods were proposed. The DRL methods have shown powerful performance in a wide range of practical problems, such as dynamic ambulance redeployment [2], intelligent traffic light control [3], large-scale fleet management [4], etc. Deep Q-network (DQN) is an important algorithm in DRL, which starts the revolution of DRL. DQN [5] combines reinforcement learning with the convolutional neural network (CNN), to learn successful policies directly from high-dimensional input images. This method outperforms the best existing reinforcement learning methods on 43 Atari 2600 games without incorporating any additional prior knowledge.

Although DRL models have achieved remarkable performance, the source of their impressive performance remains poorly understood. Visual analytics methods are good at knowledge communication and insight discovery by encoding the original data into meaningful representations [6]. Researchers have developed some visual analytics systems to explain different deep neural networks, such as convolutional neural networks [7, 8], Long Short-Term Memory networks [9, 10], generative adversarial networks [11]. The study of using visual analytics methods to interpret deep reinforcement learning model just gets started. The t-SNE map used in [12] needed to extract hand-crafted features. The saliency map-based methods [13, 14] were sensitive to simple transformation of the input states [15]. Luo et al. [16] used CNN visualization algorithms in the domain of vision-based reinforcement learning, which was aiming at the simulated drone environment.

In this paper, we propose a visual analytics system for an insightful exploration of deep Q-network. Deep Q-network is trained first. Important intermediate data are saved, such as state data, convolutional layer data, Q-values, etc. Then, the trained DQN model is tested to play games, and the related data generated during the testing process are also saved, such as the game images and the executed actions. Finally, four coordinated visual views are designed. The Q-value distribution view helps to understand key strategies learned by the agent and the relationship between states and Q-values. The convolution analysis view helps users to interpret the function of convolutional layers. The time-series view reveals the cyclical pattern executed by the agent and the function of random action. The predicted Q-value probability view assists users in investigating the reason why the agent made a certain decision. These views work together to help users understand the working principle of DQN. Two case studies demonstrate the effectiveness of the proposed system.

The rest of this paper is structured as follows. Section 2 reviews the related work. The analytics tasks and system overview are introduced in Section 3. Section 4 presents the visual designs of our system. Case studies are discussed in Section 5. Section 6 draws our conclusions.

2. Related Work

2.1 Reinforcement Learning Methods

Traditional reinforcement learning methods failed to deal with complex problems since such problems were usually non-Markov and continuous with an unbounded action space. Deep reinforcement learning [17] utilized deep neural networks to approximate the function in traditional RL models, and have been adopted in a wide range of applications. Ji et al. [2] proposed a dynamic ambulance redeployment system by utilizing DRL to learn a deep score network, which not only saved the average pickup time of patients but also improved the ratio of patients being picked up. Lin et al. [4] proposed a contextual multi-agent reinforcement learning framework to tackle the large-scale fleet management problem. Li et al. [18] proposed a contextual cooperative reinforcement learning model to manage the couriers in an express system. Deep reinforcement learning models were also used to improve recommendation performance in an online retail trading platform [19] and provide intelligent traffic light control [3]. Silver et al. [20] developed a hybrid DRL system, AlphaGo, which achieved a 99.8% winning rate against other Go programs and defeated the best professional Go players.

Among all of the DRL methods, deep Q-network [5] is an important model leading the revolution of DRL. DQN used the convolutional neural network to parameterize an approximate value function, and can learn to play a series of Atari 2600 games at a superhuman level directly from high-dimensional observations. The initial DQN model was further optimized, and some modified algorithms were proposed, such as double Q-learning [21], dueling DQN [13], distributional DQN [22], etc.

The above studies focused on proposing DRL approaches to solve different problems, the inherent working principles of DRL models were unexposed.

2.2 Visual Analysis of Deep Neural Networks

Due to the black-box representations in deep neural networks, various visual analysis methods were proposed to understand their performance.

To better analyze the convolutional neural networks, Liu et al. [7] formulated a deep CNN as a directed acyclic graph, and developed a hybrid visualization to disclose the multiple facets of each neuron and the interactions between them. Yosinski et al. [8] visualized the activations and features at different layers for CNN, to help understand how the network works.

To visualize and understand Long Short-Term Memory (LSTM) networks, Karpathy et al. [9] explored the LSTM predictions and their learned representations based on character-level language models, and found these networks learn powerful, long-range interactions. Strobelt et al. [10] presented LSTMVIS, a visual analysis tool for analyzing RNN hidden states.

For DRL model understanding, Zahavy et al. [12] developed a tool to visualize the features learned by DQN, and revealed that the features aggregated the state space in a hierarchical fashion. However, they extracted hand-crafted features from the raw frames, such as player position, enemy position, number of lives, which required a lot of manual work. Aiming at the dueling Q-network, the Jacobian saliency maps were computed for understanding the roles of the value and advantage streams [13]. Focusing on agents trained via the Asynchronous Advantage Actor-Critic (A3C) algorithm, Greydanus et al. [14] described a perturbation-based technique for generating saliency videos of deep RL agents, to interpret the decisions made by agents. The saliency methods [13-14] were sensitive to simple transformation of the input states [15]. Such et al. [23] compared the representations learned by different families of DRL algorithms. Annasamy et al. [24] proposed an interpretable deep Q-network model by combining an attention mechanism and a reconstruction loss, to provide a global explanation of

the model's behavior, which needs to modify the model architecture. Wang et al. [25] proposed DQNViz, a visual analytics system to disclose details of the DQN training process, which enabled users to dive into the large experience space of the agent for model interpretation. Our work is similar to [25]. However, we analyze the DQN model from different aspects. We piecewise analyze the range of Q-values for different games, explore in which condition the agent will obtain higher Q-values or lower Q-values. We also analyze the rationality of decisions made by the agent.

3. Task Analysis and System Overview

3.1 Background: Deep-Q-network

In the reinforcement learning setting, an agent observes its current state s_t at each time step t, selects an action a_t according to a potentially stochastic policy π, receives a reward r_t, and transits to the next state s'. The future reward at time t is defined as: $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where T is the terminal step, and $\gamma \in [0,1)$ is the discount factor. The Q-function of a policy π is defined as:

$$Q^{\pi}(s,a) = E[R_t \mid s_t = s, a_t = a, \pi] \tag{1}$$

The optimal action-value function $Q^*(s,a) = \max_{\pi} Q^{\pi}(s,a)$ obeys the Bellman optimality equation:

$$Q^*(s,a) = E_{s'}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a \right] \tag{2}$$

Deep Q-Network [5] adopts the convolutional neural network to approximate the optimal action-value function, which takes the raw image as input and outputs the expected reward for individual actions. The goal of DQN is to minimize the loss function at iteration *i* as follows:

$$L_i(\theta_i) = E_{(s,a,r,s')\square ER}\left[\left(y_i^{DQN} - Q(s,a;\theta_i) \right)^2 \right] \tag{3}$$

with

$$y_i^{DQN} = r + \gamma \max_{a'} \hat{Q}(s',a';\theta_i^-) \tag{4}$$

where Q and $\hat{Q}$ represent the online network and the target network with parameters θ_i and θ_i^-, respectively.

DQN has two important ingredients: target network and experience replay (ER). The parameters of target network θ_i^- are only updated from θ_i every fixed number of iterations. The agent's experiences in the form $e = (s,a,r,s')$ are stored in the experience replay memory and sampled uniformly at random to update the network.

3.2 Task Analysis

To help users understand the internal operation of Deep Q-network better, a list of analytical tasks is listed below:

T.1 Analyze the experience space and the relationship between states and Q-values, to interpret the large experience space of the agent, and in which condition the agent will obtain higher Q-values or lower Q-values.

T.2 Illustrate the effects of the convolutional layers, to understand the function of the convolutional layers, and the influence factors for selecting actions by the agent.

T.3 Visualize the agent's behavior along the timeline, to discover if the strategy used by the agent has a fixed pattern, and learn the function of random actions.

T.4 Visualize the predicted Q-value probability distribution over actions, to show the rationality of decisions made by the agent.

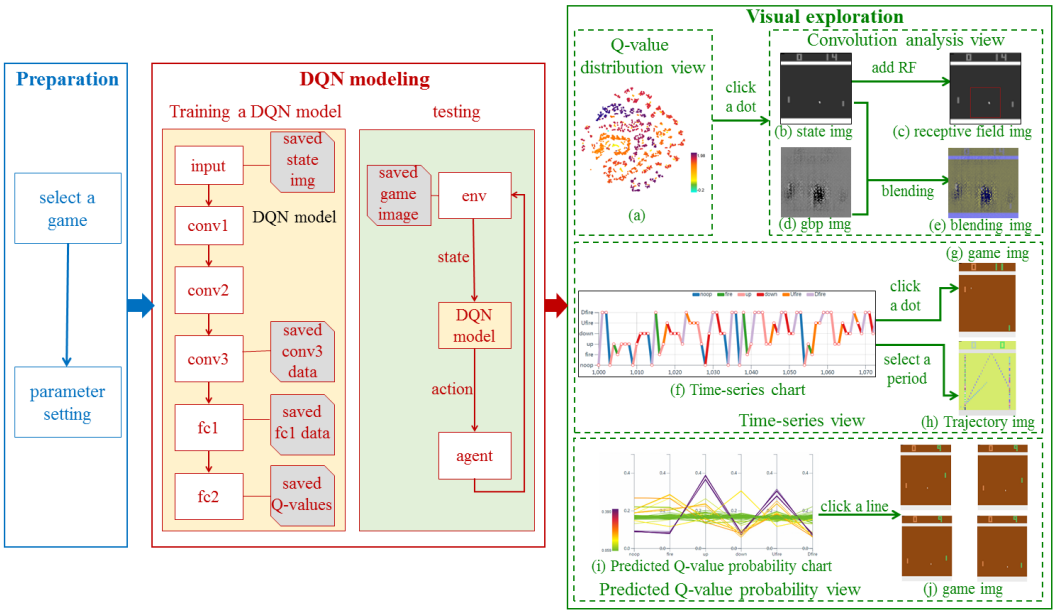


Fig. 1. The system pipeline

3.3 System Overview

Aiming to complete the above tasks, we design a visual analytics system to explore deep Q-Network. **Fig. 1** illustrates the system pipeline, including three phases:

Preparation During the preparation phase, users select a game to analyze and set the running parameters.

DQN modeling In this phase, a DQN model is trained first, containing three convolutional (conv) layers and two fully connected (fc) layers. The intermediate results are stored, including the state image (img), the data processed by the convolutional layers or the fully connected layers. The obtained DQN model is tested for different games in Atari 2600 game environment (env). The predicted Q-value probability distribution, the performed actions and the game images during the testing process are saved for further analysis.

Visual exploration In this phase, four visual views are designed to help users understand the DQN model. The Q-value distribution view helps users observe the experience space and understand the rules of Q-value acquisition (**T.1**). When clicking a dot in the Q-value distribution view, two images (blending image and receptive field image) are shown. The

receptive field image highlights the region in the input image that affects the neuron with maximal activation in the convolutional layer, to help understand where the neuron is looking at. The blending image is generated from the guided-backpropagation (gbp) image and the state image, which identifies the important pixels that influence the agent's behavior (**T.2**). The time-series view visualizes the actions along the timeline. When clicking the dot on the time-series view, the current game image is shown. When choosing a period of time, the trajectory image containing the agent's movement is shown. This view helps users learn the agent's action patterns and the function of random action (**T.3**). The predicted Q-value probability view visualizes the predicted Q-value distribution over actions, which provides an opportunity to interpret the decisions made by the agent (**T.4**).

4. Visual Designs

In this section, four visual views are designed to help users understand the DQN model.

4.1 Q-value Distribution View

The Q-value distribution view abstracts one state into a point in a two-dimensional space and shows the distribution of different Q-values. Filter operation is provided for selecting Q-values within the expected intervals.

Near the end of training, the states in the replay memory are collected. To be more specific, the neural activations for 10000 game states of the first fully connected layer $hdata \in \mathbf{R}^{10000\times512}$ are stored, seen in (5). Each data item corresponds to one state with 512 features, which can be regarded as a point in the high-dimensional space. We apply t-Distributed Stochastic Neighbor Embedding (t-SNE) [26] to map the feature matrix $hdata$ in the high-dimensional space to a two-dimensional space as shown in (6):

$$hdata = fc1\big(conv3(conv2(conv1(X)))\big) \tag{5}$$

$$rdata = t\_SNE(hdata) \tag{6}$$

where $rdata$ is the matrix after dimensionality reduction and X is the input image.

Based on $rdata$ and the Q-values predicted by the Q-network, the Q-value distribution view is rendered, which is shown in **Fig. 1(a)**. One dot represents a specific state, whose position is decided by $rdata$ and whose color is decided by its estimated Q-value. A gradient color scheme (purple-red-yellow-green-cyan) is used to encode the Q-value. The purple dot encodes a higher Q-value, while the cyan dot encodes a lower Q-value. The t-SNE embeddings cluster states with similar characteristics. Filter operation is provided in this view. Users can modify the range of Q-values to update this view. Only dots falling within that range are shown. When clicking a dot in the Q-value distribution view, the corresponding Q-value, receptive field image and blending image will be shown. The latter two will be introduced in the next section in detail.

4.2 The Convolution Analysis View

The convolution analysis view provides two images: the receptive field (RF) image and the blending image. They are together to illustrate what information is used for the agent to make decisions.

In a convolutional neural network, the receptive field is defined as the region in the input image that a particular CNN's feature is looking at [27]. In other words, it represents the region

mapped on the input image for the pixels in different feature maps. A receptive field of a feature is described by its central location and its size. We use (7) and (8) to calculate the receptive field:

$$r_{l+1} = r_l + \left((k_{l+1} - 1) * \prod_{i=0}^{l} s_i \right) \tag{7}$$

$$start_{l+1} = start_l + \left((\frac{k_{l+1} - 1}{2} - p_{l+1}) * \prod_{i=0}^{l} s_i \right) \tag{8}$$

where r_{l+1} is the receptive field of layer $l+1$, $r_0 = 1$. k_{l+1} and p_{l+1} are the kernel size and padding size of layer $l+1$, respectively. s_i is the stride of layer i. Layer 0 is the input gray-scale image. $start_{l+1}$ represents the central location of the receptive field in the input image for the upper-left pixel in the $(l+1)$-*th* feature map. $start_0 = (0.5, 0.5)$ is the central location of the upper-left pixel in the input image.

Through the experiments, we find that the receptive field of the neuron with the largest activation in the third convolutional layer captures useful information. Therefore, we use it to generate the receptive field image. We first calculate the size of the receptive field for the third convolutional layer. Then, we locate the coordinates of the neuron with the largest activation in layer 3 and calculate its receptive field in the input image. Lastly, we use a red box to mark the receptive field on the input state image to generate the receptive field image (**Fig. 1(c)**). The marked region contains the units that affect the neuron with the largest activation in layer 3.

To detect important pixels that influence the agent's actions, guided-backpropagation (gbp) method [28] is adopted to visualize high-level features learned by the convolutional neural network. The gbp method combines deconvnet and backpropagation, which uses the gradient values and features of forward propagation as the thresholds. The input state image f^0 (**Fig. 1(b)**) is processed through forward pass to obtain the feature map f^l for layer l, and the gradient values for layer $l+1$ are calculated as R^{1+1}. Then, R^{1+1} is processed by (9):

$$R^l = (f^l > 0) \cdot (R^{l+1} > 0) \cdot R^{l+1} \tag{9}$$

Two thresholds are used to select the pixels in R^{l+1}. The positions of pixels which are greater than 0 in f^l and R^{l+1} are retained, and the remaining pixels are set to be 0, to obtain R^l through back propagation. Finally, the reconstructed image R^0 is calculated through back propagation, also called as the gbp image (**Fig. 1(d)**). The nonzero positions in the gbp image identify the concepts learned by neurons in higher layers, which affects the agent's action. The darker pixels are more important. We further blend the state image and the gbp image to generate a blending image (**Fig. 1(e)**). From which, we can observe the important pixels in the game environment, to find important factors that affect the agent's action.

4.3 Time-series View

The time-series view shows the executed actions as time evolves, to illustrate the agent's behavior driven by the DQN model.

During the testing process, we save the intermediate data of game execution for a period of time. Three kinds of data are saved: 1) The agent's actions, which are related to the game running environment. 2) The randomness of action, which is determined by a pre-defined probability value. 3) The game image, which shows the current running status of the game.

Based on the saved data, the time-series chart (**Fig. 1(f)**) is rendered. The *x*-axis represents the timesteps, while the *y*-axis represents the action space. For example, the Breakout game has four kinds of actions: no-operation (noop), firing the ball (fire), moving left (left) and moving right (right), the values of *y*-axis are noop, fire, left and right. The dot in the time-series chart represents a state in one timestep, whose *y*-axis value indicates the upcoming action. A red solid dot is used to denote the state at the beginning of a game, while the red hollow dot represents other states. The line between two dots denotes the action executed between two states. Different color lines are used to represent different actions. The legend for actions is shown above. If the previous action is generated randomly, then a gray rectangle is drawn behind the dot.

When clicking one dot, the game image (**Fig. 1(g)**) for the current timestep is shown. When selecting a period of time, a video clip illustrating the game execution process is generated. The game images during that period are added by corresponding pixels to generate a trajectory image (**Fig. 1(h)**), which shows the trajectory of object movement.

4.4 Predicted Q-value Probability View

To help interpret the decisions made by the agent, we design a predicted Q-value probability view to show the predicted condition of the trained agent during the testing process.

During the testing process, the predicted Q-value probability distributions over actions for different timesteps are stored first. Then, we use softmax operation to normalize the obtained Q-values. After that, a predicted Q-value probability chart based on parallel coordinates visualization is designed (**Fig. 1(i)**). Each axis corresponds to one kind of action, while the axis value corresponds to the predicted Q-values over different actions. A line depicts the predicted condition for a certain timestep. To make it easier to distinguish the lines, we calculate the variance of the predicted Q-values over different actions for each line, and assign the line color according to its variance. A gradient color scheme (purple-red-yellow-green) is used to encode the variance. The purple line encodes higher variance, indicating that the agent is aware of which action to perform. The green line encodes lower variance, indicating that the agent will obtain similar Q-values no matter which action is performed. Users need to input the timesteps to be observed. When clicking a certain line, the corresponding game images of four consecutive frames will be shown (**Fig. 1(j)**), to help users determine how the game is playing.

5. Case Studies

Two case studies were conducted to evaluate the effectiveness of our system.

5.1 Experiment Setting

We used Atari 2600 game environment for the experiments. The parameters were set as follows: replay memory = 200000, learning rate = 0.0001, discount factor = 0.99. The initial random probability is set to be 1, which decreases as the game runs, and decreases to 0.01 finally. Two games 'Pong' and 'Breakout' were used for evaluation.

Pong game Pong is one kind of confrontational game. The agent plays ping-pong game with the computer. The left and right paddle are controlled by the computer and the agent, respectively. The player gets 1 point if the opponent misses the ball, and the game terminates when one player gets 21 points first. The agent has 6 actions, which will be described in the analysis process.

Breakout game Breakout is a non-confrontational game. The agent gets a reward by firing the ball to hit the bricks on the screen. The agent will receive 1, 4 and 7 points when the ball hits bricks in the bottom two rows, middle two rows, and top two rows, respectively. The agent

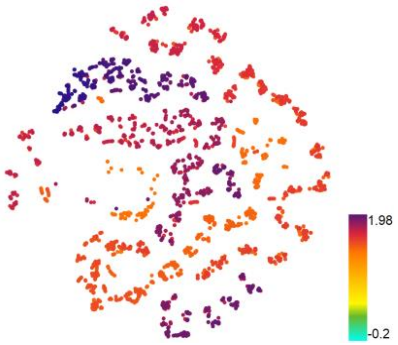
has 4 actions and owns 5 lives in each game episode. If the agent doesn't catch the ball with the paddle, it loses one life.

5.2 Case Study 1: The Analysis of Pong Game

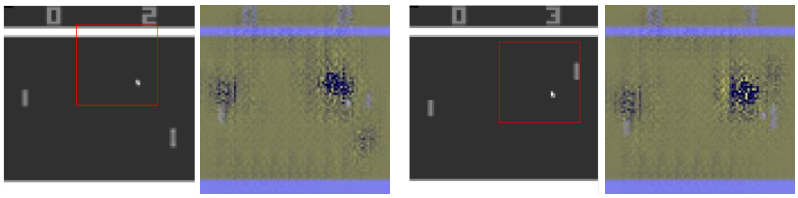
5.2.1 The Analysis of Q-value Distribution

We first analyze the Pong game. **Fig. 1(a)** shows its Q-value distribution view. To better examine the states with different Q-values, we filter Q-values for further observation. In Pong game, if the computer cannot catch the ball, the agent receives one point. Therefore, we first observe the condition when $Q \geq 1$. **Fig. 2(a)** shows the Q-value distribution view when $Q \geq 1$. Most dots fall into this interval. Because the Pong game is relatively easy for the agent, the trained powerful DQN model can obtain a high Q-value. When clicking two dots in **Fig. 2(a)**, the corresponding RF images and blending images are shown. Seen from the RF images on left of **Fig. 2(b)** and **Fig. 2(c)**, we find that these states are common scenarios during the game execution process. The ball bounces between two paddles.

We analyze the following three intervals for Q<1: $Q < 0.5$, $Q \in [0.5, 0.9)$, and $Q \in [0.9, 1)$. The Q-value distribution view only contains two cyan dots and two yellow dots when $Q < 0.5$. By checking the corresponding RF images for these dots, we find that the agent fails to catch the ball in such conditions. Because the trained agent owns a powerful performance, it rarely misses the ball. Thus, there are very few dots representing such states. Left of **Fig. 3(a)** shows one example. For $Q \in [0.5, 0.9)$, the dots representing states near the end of the game. At that time, the agent has obtained a high score, such as 19 points or 20 points. If it can gain one or two more points, the game terminates. The Q-value is relatively low since the long-term reward is low. Left of **Fig. 3(b)** and **Fig. 3(c)** show the RF images for two representative dots. **Fig. 4(a)** and **Fig. 4(b)(c)** show the Q-value distribution view and RF images when $Q \in [0.9, 1)$, respectively. At that time, one player has just won one point, and the ball is not on the screen.

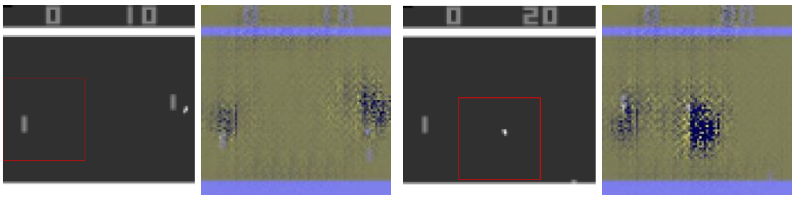


(a) Q-value distribution view

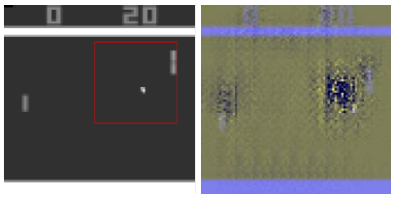


(b) Q = 1.553 (c) Q =1.531

Fig. 2. The analysis of Q-values for Pong game ($Q \geq 1$)

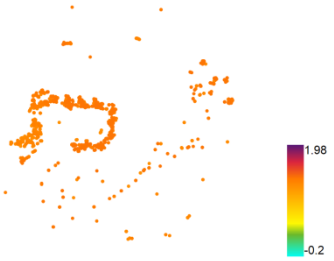


(a) Q = -0.121 (b) Q = 0.720

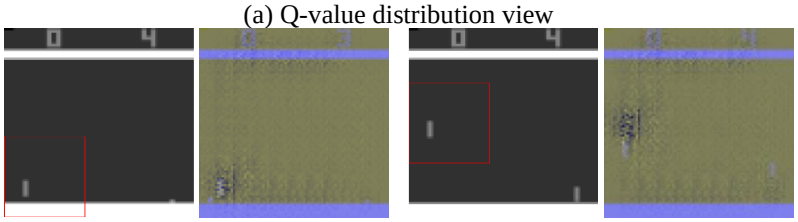


(c) Q = 0.836

Fig. 3. The analysis of Q-values for Pong game ($Q < 0.9$)



(a) Q-value distribution view



(b) Q = 0.933 (c) Q = 0.996

Fig. 4. The analysis of Q-values for Pong game ($Q \in [0.9,1)$)

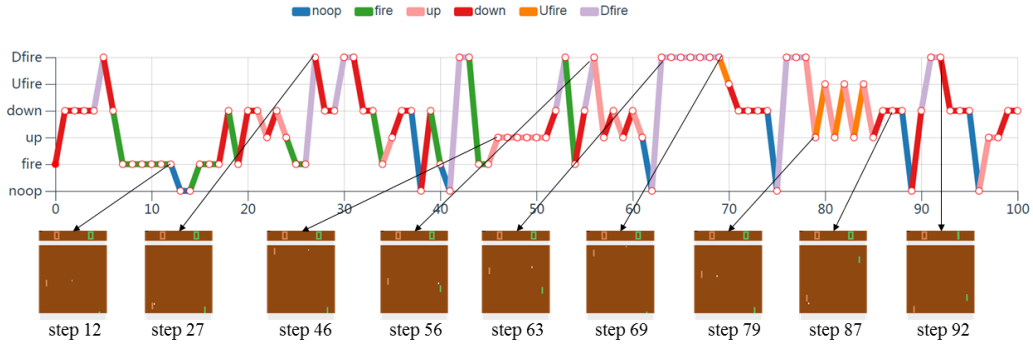
5.2.2 The Analysis of Convolutional Layer Function

Next, we observe the RF images and blending images to learn the function of the convolutional layers. When the game executes normally, the corresponding RF images (left of **Fig. 2(b)(c)** and left of **Fig. 3(b)(c)**) mark the regions of receptive field for the neuron with maximal activation in the third convolution layer, which captures the position of the ball. Seen from the related blending images, we find that the ball position and opponent's paddle position are the most important pixels, which have the dominant influence on the final decision of the agent. Seen from the pixels' colors, we know that the ball is more important than the opponent. During the start stage of the game, the ball is not on the screen, and the receptive field only captures the opponent's paddle position (left of **Fig. 4(b)(c)**). The corresponding blending images also highlight the opponent's paddle positions (right of **Fig. 4 (b)(c)**). We also find that the receptive field fails to capture the ball position in **Fig. 3(a)**, which causes the agent to miss the ball, and the opponent gets one point. Therefore, the Q-value is negative.

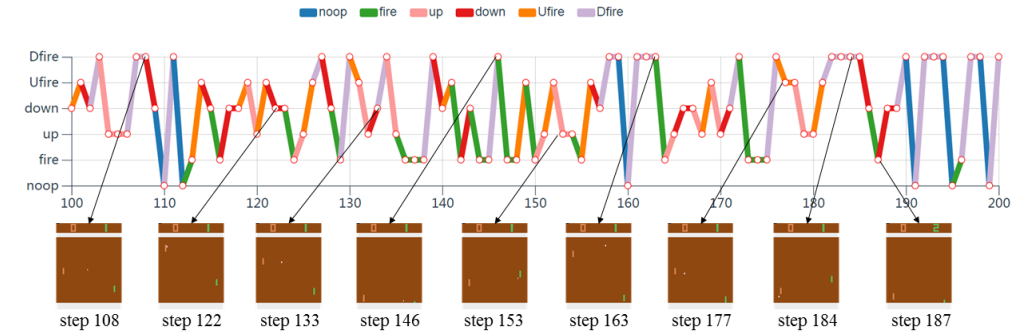
5.2.3 The Analysis of Agent Behavior

In Pong Game, there have six kinds of actions. In time-series charts (**Fig. 5**), different colors are used to identify different actions: blue for no-operation (noop), green for firing the ball (fire), pink for moving up (up), red for moving down (down), orange for moving up and fire the ball (Ufire) and purple for moving down and fire the ball (Dfire).

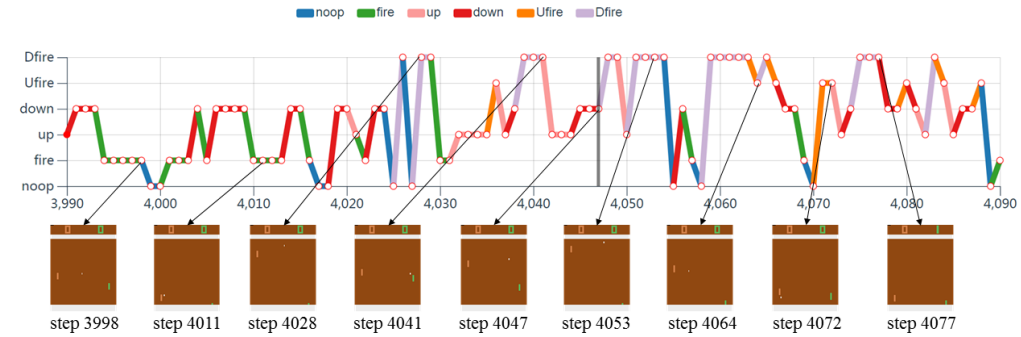
Since the solid red dot indicates the start of a new game, we analyze the first 200 steps for two games starting from the solid red dots. The time steps are from 0 to 200, and 3990 to 4190. **Fig. 5** shows the relevant time-series charts. By checking the game images, we find that the agent gets two points from 0 to 200 steps. The first point starts from step 0 and ends up with step 91. As seen from **Fig. 5(a)**, the score in the top right corner of the screen changes from 0 to 1 in step 92. The second point starts from step 92 and ends up with step 186 (**Fig. 5(b)**). During 3990 to 4190 steps, the agent also gains two points. The first point covers the range from step 3990 to step 4076 (**Fig. 5(c)**), and the second point from 4077 to 4171(**Fig. 5(d)**).



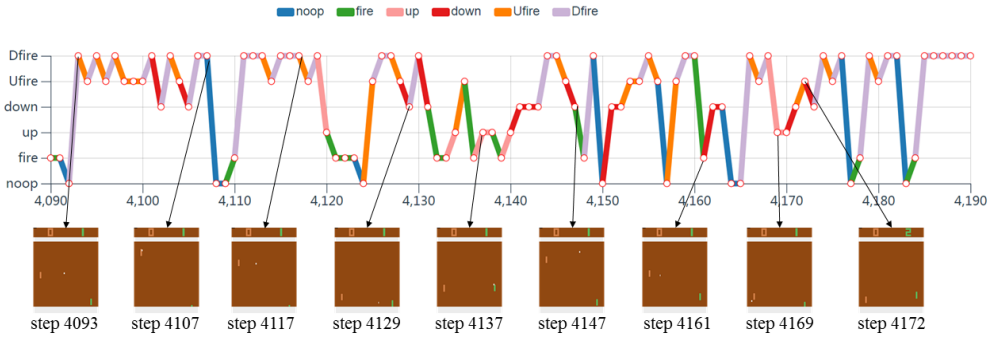
(a) Step 0 to step 100



(b) Step 100 to step 200

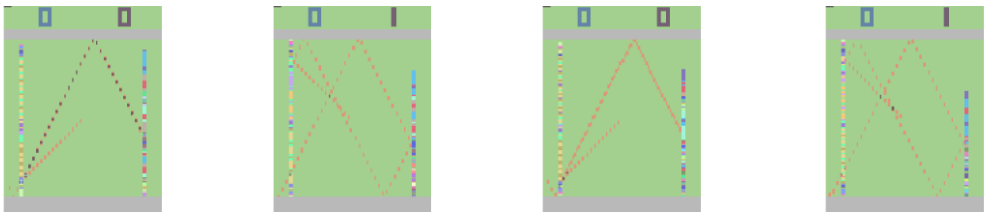


(c) Step 3990 to step 4090



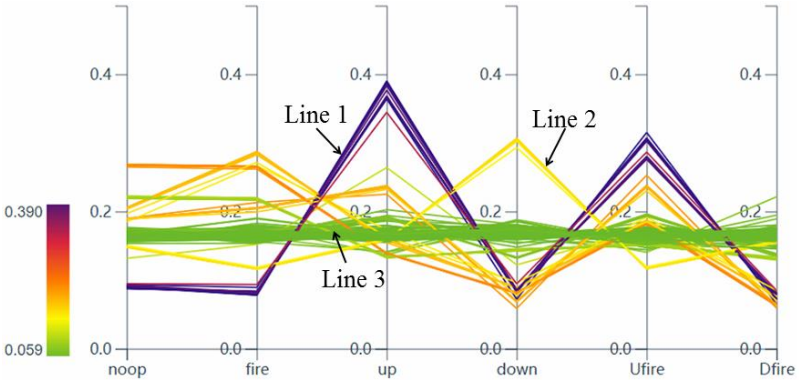
(d) Step 4090 to step 4190

Fig. 5. The time-series charts

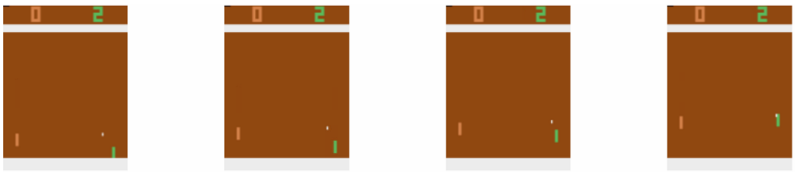


(a) Step 0 to 91 (b) Step 92 to186 (c) Step 3990 to 4076 (d) Step 4077 to 4171

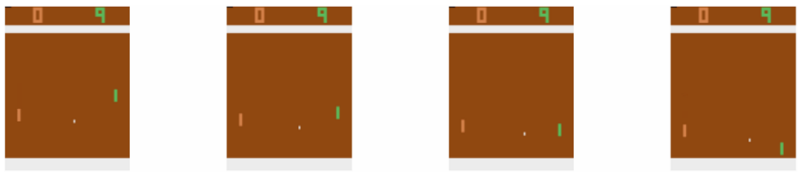
Fig. 6. The trajectory images



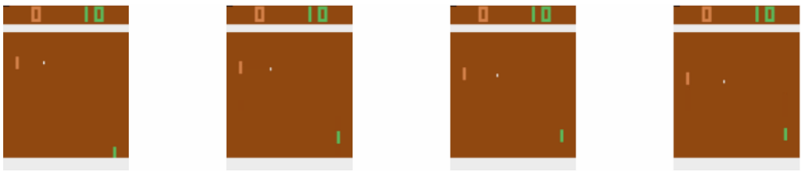
(a) The predicted Q-value probability chart



(b) The game images for line 1



(c) The game images for line 2



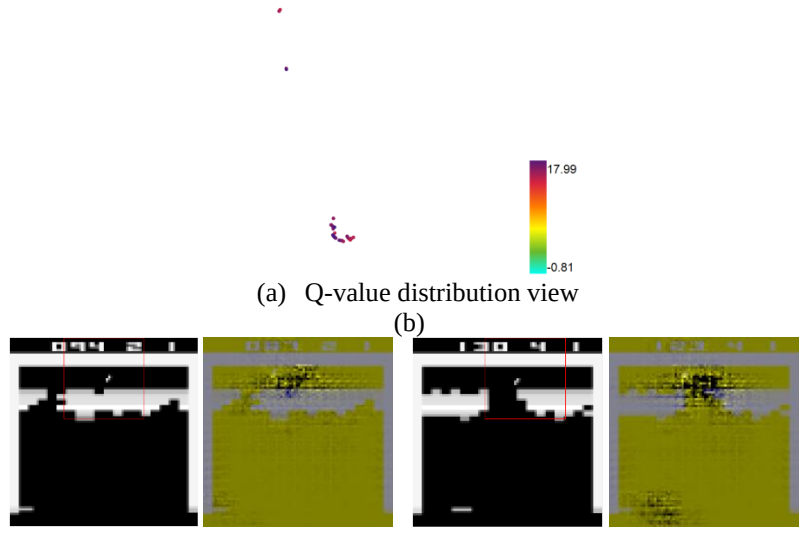
(d) The game images for line 3

Fig. 7. The predicted Q-value probability view for Pong game

Next, we explore the strategies developed by the agent through the time-series view. In the time-series charts (**Fig. 5**), some keyframes are shown to illustrate the game states. Users can also watch the ball movements within the selected timesteps in terms of video clips. **Fig. 6** shows the trajectory images for the time steps when one point is decided. From **Fig. 6(a)** and the game images in **Fig. 5(a)**, we can know the moving trail of the ball from step 0 to step 91, during which the agent gains the first point. The ball starts from the middle and goes to the bottom left. Then it goes from the bottom left to the top firing by the opponent, and flies to the agent after a bounce. After that, the agent fires the ball, and the ball flies to the opponent following the original trajectory. Since the opponent does not catch the ball, the agent gains one point. The trajectory in **Fig. 6(a)** has an overlap. Seen from **Fig. 5(a)(c)** and **Fig. 6(a)(c)**, we find that although the executed actions are different for two time periods when the agent gains the first point, the trajectories of balls are similar. The trajectories of balls are also similar during the time periods when the agent gains the second point, seen from **Fig. 5(b)(d)** and **Fig. 6(b)(d)**. It indicates that the agent can follow the same optimal strategy in the process of the game running to gain scores, and win the game quickly.

5.2.4 The Analysis of Predicted Q-value Probability Distribution

To further analyze why the agent makes such decisions, we observe the predicted Q-value probability view. The predicted Q-value probability chart in **Fig. 7(a)** shows the overall probability distribution of the predicted Q-values. We choose different lines to see their detailed game conditions. The purple lines always have two peaks. Take line 1 for example, the corresponding game images (**Fig. 7(b)**) shows that the paddle tries to move up towards the ball to catch the ball. Therefore, the Q-values for action “up” and “Ufire” are high. Line 2 has a high value for “down” action, and the game images (**Fig. 7(c)**) shows that the paddle moves down towards the ball to catch the ball. The Q-values of green lines distributes equally, which indicates that the current action has little impact on the final score. We observe the corresponding game conditions for one green line, and find that the ball is far away from the paddle (**Fig. 7(d)**). Thus, the action of the paddle in the current step is less important.

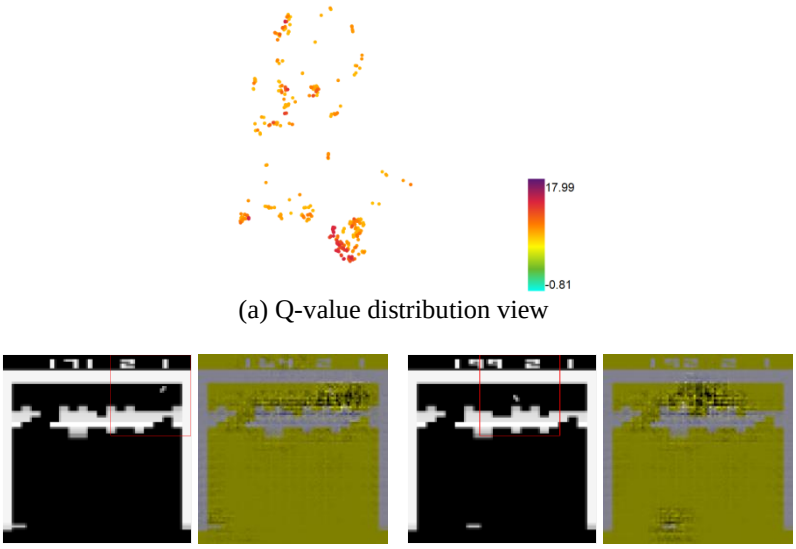


(a) Q-value distribution view

(b)

(b) Q = 16.144 (c) Q = 15.795

Fig. 8. The analysis of Q-values for Breakout game ($Q \geq 14$)



(a) Q-value distribution view

(b) Q = 10.585 (c) Q = 9.056

Fig. 9. The analysis of Q-values for Breakout game ($Q \in [8,14)$)

5.3 Case Study 2: the Analysis of Breakout Game

5.3.1 The Analysis of Q-value Distribution

In the second case study, the Breakout game is analyzed. Since the distribution of Q-values is more complicated, we filter Q-values to better observe the agent's actions in different ranges.

Fig. 8(a) shows the Q-value distribution view when $Q \geq 14$. Seen from the selected RF images (left of **Fig. 8(b)(c)**), we find that the agent is very smart by developing the tunneling strategy. In such conditions, the agent has just dug a tunnel through the bricks, and the ball is

on the top of bricks. The ball can bounce between the ceiling and the top rows of bricks to hit bricks with high scores. Such a strategy can avoid the paddle to miss the ball, and the agent receives high immediate rewards.

Fig. 9(a) illustrates the Q-value distribution view for $Q \in [8,14)$. **Fig. 9(b)** and **Fig. 9(c)** show two selected states. In such conditions, the agent has dug the tunnel for some time. Some bricks in the top two layers have been destroyed. Therefore, the estimated Q-values are lower than that of $Q \geq 14$.

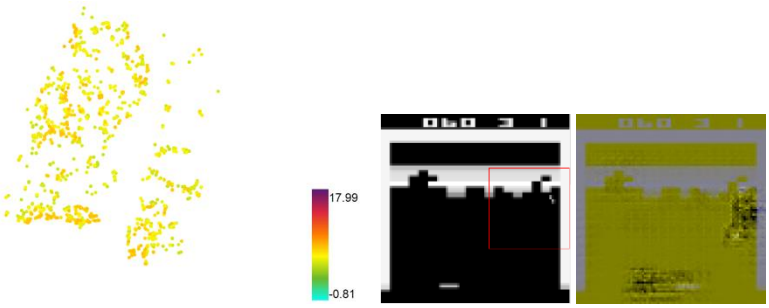
Fig. 10(a) shows the Q-value distribution view when $Q \in [5,8)$. At that time, the game is running and the bricks in the bottom layers with lower scores have been destroyed a lot. The agent begins to hit the bricks with higher scores. However, since the agent has not yet dug the tunnel (left of **Fig. 10(b)(c)**) or has not sent the ball into the tunnel (left of **Fig. 10(d)**), the estimated Q-values are not very high.

The green dots in the Q-value distribution view (**Fig. 11(a)**) represent the initial states when $Q \in [2,5)$, which has the largest proportion. The RF images and blending images for two selected dots are shown in **Fig. 11(b)(c)**. The ball is firing the bricks in the bottom rows, and the estimated Q-values are low.

Fig. 12(a) shows the Q-value distribution view for $Q < 2$. Most dots represent terminated states of the game, as shown in left of **Fig. 12(b)(c)**. Because there left a few bricks, the long-term rewards are very low. There is another special case where Q-value is very low. The agent fails to catch the ball, and a game with a new life begins (**Fig. 12(d)**). Since there has no ball on the screen and the immediate reward is not available, the estimated Q-values are very low.

5.3.2 The Analysis of Convolutional Layer Function

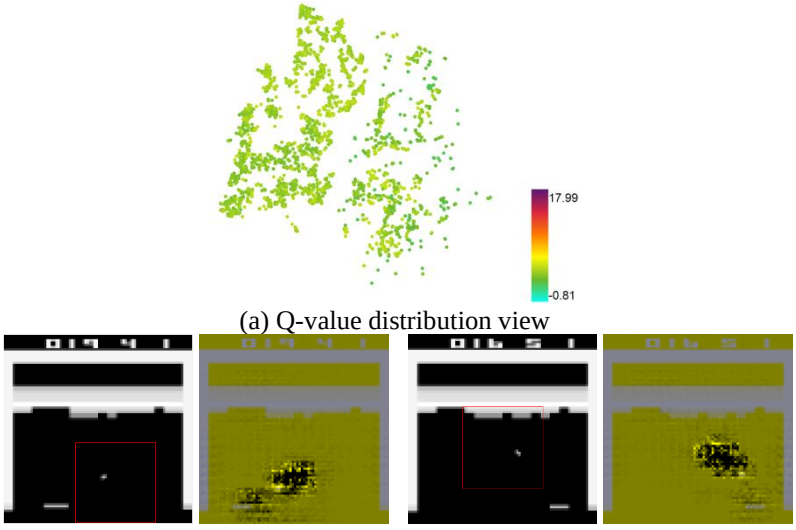
Next, we observe the function of convolutional layers through the RF images and blending images. Seen from **Fig. 8** to **Fig. 12**, we find that the receptive fields in RF images capture the ball when the balls are on the screen. We also explore which pixels are important for the agent to make decisions by observing the blending images. In general, the ball has the greatest impact on the agent, followed by the paddle, seen from the black pixels in right of **Fig. 8(b)(c)** and right of **Fig. 11(b)(c)** for example. When the game is nearly over, there left a few bricks on the screen, seen from right of **Fig. 12(b)(c)**. The positions of bricks are also shown in the blending images. In other words, the ball, paddle and bricks jointly influence the behavior of the agent when there remain a few bricks.



(a) Q-value distribution view (b) Q = 6.37

(c) Q = 7.92 (d) Q = 7.71

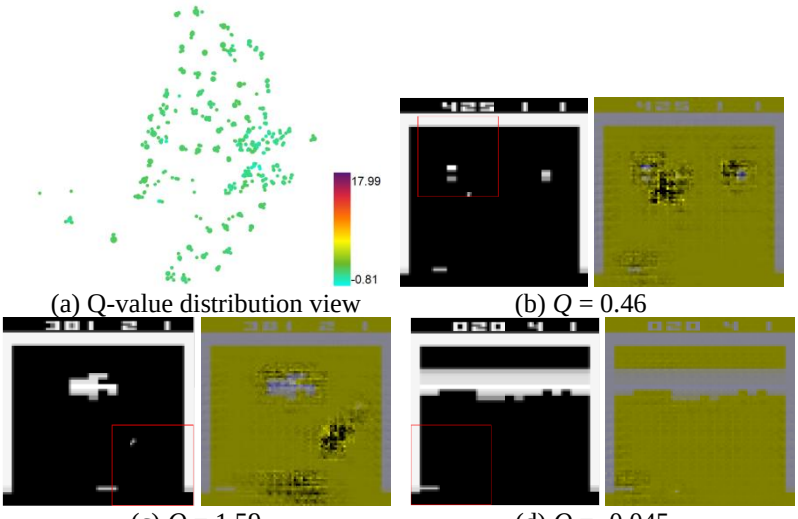
Fig. 10. The analysis of Q-values for Breakout game ($Q \in [5,8)$)



(a) Q-value distribution view

(b) Q = 4.02 (c) Q = 4.68

Fig. 11. The analysis of Q-values for Breakout game ($Q \in [2,5)$)



(a) Q-value distribution view (b) Q = 0.46

(c) Q = 1.59 (d) Q = -0.045

Fig. 12. The analysis of Q-values for Breakout game ($Q < 2$)

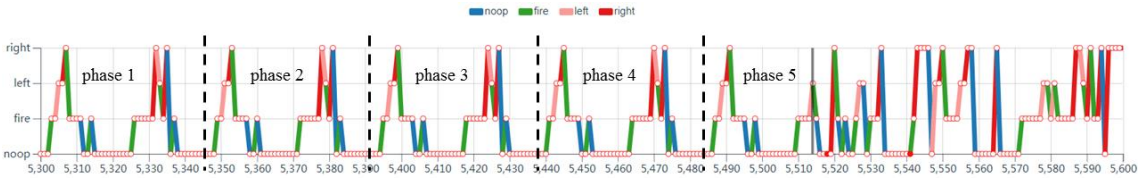
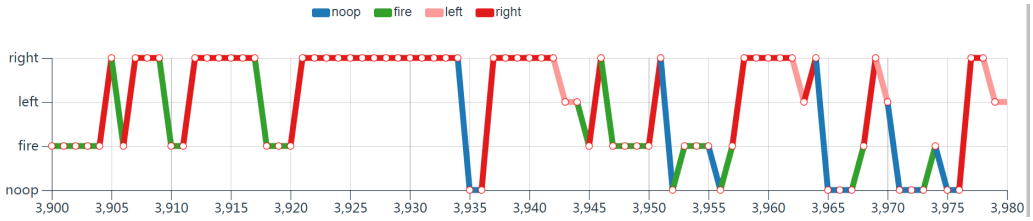
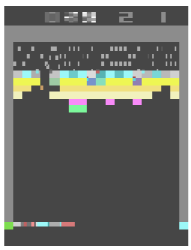


Fig. 13. The function of random action



(a) Time-series chart



(b)Trajectory image

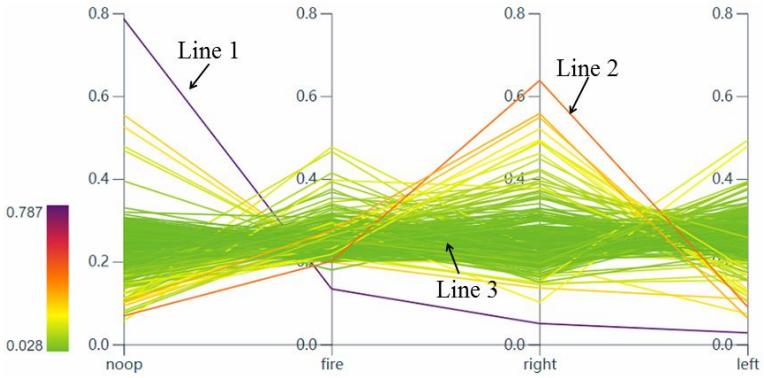
Fig. 14. The analysis of tunneling strategy

5.3.3 The Analysis of Agent Behavior

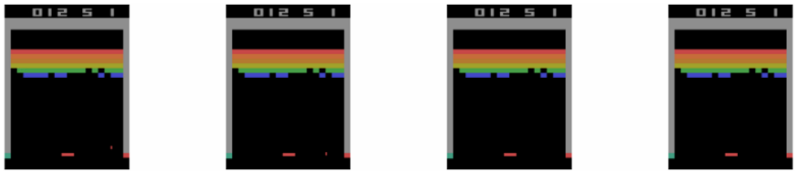
In Breakout game, there have four kinds of actions. Different colors are used to identify different actions: blue for no-operation (noop), green for firing the ball (fire), pink for moving left (left) and red for moving right (right).

Fig. 13 shows the time-series chart from step 5300 to step 5600. We find that the agent performs a series of periodical actions. The dotted black lines divide these time steps into five phases. The actions in the first four phases have the same patterns, indicating that the agent is trapped in a loop. In the fifth stage, the agent's movement pattern changes significantly due to the random action (marked by the gray bar), and the loop is jumped out. Random actions prevent the agent from trusting existing strategies too much and help break the loop.

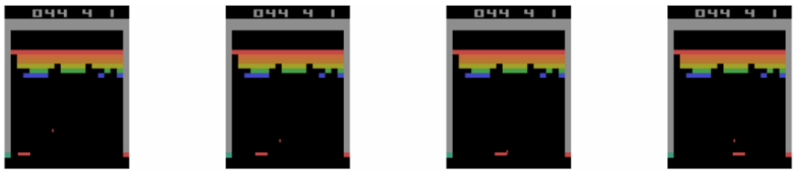
Fig. 14(a) and **Fig. 14(b)** show the time-series chart and the trajectory image from step 3900 to step 3980, respectively. Before then, the agent has carved a tunnel on the left of the screen and transported the ball into the tunnel. After that, the ball bounces between the ceiling and the top two rows to hit bricks with high scores. We can see from **Fig. 14(b)** that the ball bounces on the top of bricks. This strategy helps the agent gain high scores quickly.



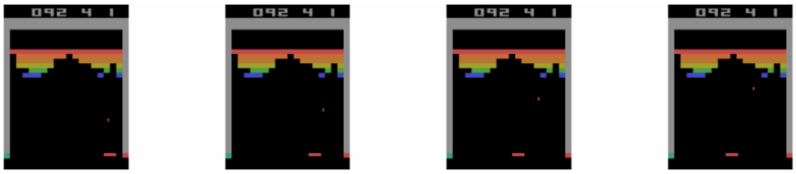
(a) The predicted Q-value probability chart



(b) The game images for line 1



(c) The game images for line 2



(d) The game images for line 3

Fig. 15. The predicted Q-value probability view for Breakout game

5.3.4 The Analysis of Predicted Q-value Probability Distribution

We also observe the predicted Q-value probability view for Breakout game. The predicted Q-value probability chart demonstrates the overall probability distribution of the predicted Q-values (**Fig. 15(a)**). Three typical lines are chosen for further analysis. Line 1 has a high Q-value for "noop" action. By checking the game images (**Fig. 15(b)**), we find that the paddle has missed the ball in such a condition. For high Q-value with action "right", the game images indicate that the paddle tries to move right towards the ball to catch the ball (**Fig. 15(c)**). The Q-values of green lines distributes equally. Take line 3 for example (**Fig. 15(d)**), the ball is moving towards the bricks to hit them. Thus, the paddle can perform any action in the current timestep.

Through the above analysis, users are able to learn the rewards gained by an agent under different states and the function of convolutional layers. The convolutional layers can identify interpretable and high-level patterns to assist in making decisions. The time-series view and the predicted Q-value probability view help users interpret the used strategies and the reason why the decisions are made by the agent.

6. Conclusion and Future Work

In this paper, we propose a visual analytics system for understanding deep Q-network. Important data generated from the model training and testing process are saved. Four visual views are designed to illustrate the operating principle of deep Q-network from different perspectives. Two case studies have demonstrated the effectiveness of our system in understanding the used strategies and the important visual features learned by the agent. The system can also help to explore the fixed pattern, the function of random actions and the decisions made by the agent during the game execution process. The studied deep Q-network only controls one agent. In a more complicated environment, multi-agents often cooperate to complete a task. In terms of future research, we plan to study the operating principle of multi-agent deep reinforcement learning, to explore the ways of cooperation among multiple agents.

References

[1] S. G. Khan, G. Herrmann, F. L. Lewis, T. Piep, and C. Melhuish, "Reinforcement learning and optimal adaptive control: An overview and implementation examples," *Annual Reviews in Control*, vol. 36, no. 1, pp. 42-59, 2012. Article (CrossRef Link)

[2] S. Ji, Y. Zheng, and Z. Wang, "A Deep Reinforcement Learning-Enabled Dynamic Redeployment System for Mobile Ambulances," in *Proc. of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 3, no. 1, pp. 1-20, 2019. Article (CrossRef Link)

[3] H. Wei, G. Zheng, H. Yao, and Z. Li, "Intellilight: A reinforcement learning approach for intelligent traffic light control," in *Proc. of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2496-2505, 2018. Article (CrossRef Link)

[4] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient large-scale fleet management via multi-agent deep reinforcement learning," in *Proc. of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1774-1783, 2018. Article (CrossRef Link)

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, and J. Veness, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529-533, 2015. Article (CrossRef Link)

[6] F. Hohman, M. Kahng, R. Pienta, and D. H. Chau, "Visual analytics in deep learning: An interrogative survey for the next frontiers," *IEEE Transactions on Visualization Computer Graphics*, vol. 25, no. 8, pp. 2674-2693, 2019. Article (CrossRef Link)

[7] M. Liu, J. Shi, Z. Li, J. Zhu, and S. Liu, "Towards better analysis of deep convolutional neural networks," *IEEE Transactions on Visualization Computer Graphics*, vol. 23, no. 1, pp. 91-100, 2017. Article (CrossRef Link)

[8] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," in *Proc. of the 31st International Conference on Machine Learning*, pp. 1-12, 2016. Article (CrossRef Link)

[9] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015. Aritcle (CrossRef Link)

[10] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush, "Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 667-676, 2018. Article (CrossRef Link)

[11] M. Kahng, N. Thorat, D. Chau, F. Viegas, and M. Wattenberg, "Gan lab: Understanding complex deep generative models using interactive visual experimentation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 310-320, 2019. Article (CrossRef Link)
[12] T. Zahavy, N. Ben-Zrihem, and S. Mannor, "Graying the black box: Understanding DQNs," in *Proc. of the 33rd International Conference on Machine Learning*, pp.1899-1908, 2016. Aritcle (CrossRef Link)
[13] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. of the 33rd International Conference on Machine Learning*, pp. 1995-2003, 2016. Aritcle (CrossRef Link)
[14] S. Greydanus, A. Koul, J. Dodge, and A. Fern, "Visualizing and understanding atari agents," *arXiv preprint arXiv:1711.00138*, 2017. Aritcle (CrossRef Link)
[15] P. J. Kindermans, S. Hooker, J. Adebayo, M. Alber, K. Schutt, S. Dahne, D. Erhan, and B. Kim, "The (un)reliability of saliency methods," *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pp. 267-280, 2019. Article (CrossRef Link)
[16] J. Luo, S. Green, P. Feghali, G. Legrady, and K. Koc, "Visual Diagnostics for Deep Reinforcement Learning Policy Development," *arXiv preprint arXiv:1809.06781*, 2018. Aritcle (CrossRef Link)
[17] N. D. Nguyen, T. Nguyen, and S. Nahavandi, "System design perspective for human-level agents using deep reinforcement learning: A survey," *IEEE Access*, vol. 5, pp. 27091-27102, 2017. Article (CrossRef Link)
[18] Y. Li, Y. Zheng, and Q. Yang, "Efficient and Effective Express via Contextual Cooperative Reinforcement Learning," in *Proc. of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 510-519, 2019. Article (CrossRef Link)
[19] S. Y. Chen, Y. Yu, Q. Da, J. Tan, and H. Huang, "Stabilizing reinforcement learning in dynamic environment with application to online recommendation," in *Proc. of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1187-1196, 2018. Article (CrossRef Link)
[20] D. Silver, A. Huang, and C. J. Maddison, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, pp.484-489, 2016. Article (CrossRef Link)
[21] H. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proc. of the 30th AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, pp. 2094-2100, 2016. Article (CrossRef Link)
[22] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *Proc. of the 34th International Conference on Machine Learning*, pp. 449-458, 2017. Aritcle (CrossRef Link)
[23] F. P. Such, V. Madhavan, R. Liu, R. Wang, P. S. Castro, Y. Li, J. Zhi, L. Schubert, M. Bellemare, J. Clune, and J. Lehman, "An atari model zoo for analyzing, visualizing, and comparing deep reinforcement learning agents," in *Proc. of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, 2019. Aritcle (CrossRef Link)
[24] R. M. Annasamy and K. Sycara, "Towards better interpretability in deep q-networks," in *Proc. of the 33rd AAAI Conference on Artificial Intelligence*, vol. 33, no. 1, pp. 4561-4569, 2019. Article (CrossRef Link)
[25] J. Wang, L. Gou, H. W. Shen, and H. Yang, "Dqnviz: A visual analytics approach to understand deep q-networks," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 1, pp. 288-298, 2019. Article (CrossRef Link)
[26] L. Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579-2605, 2008. Aritcle (CrossRef Link)
[27] W. Luo, Y. Li, R. Urtasun, and R. Zemel, "Understanding the Effective Receptive Field in Deep Convolutional Neural Networks," in *Proc. of the 30th Annual Conference on Neural Information Processing Systems (NIPS)*, pp. 4905-4913, 2016. Aritcle (CrossRef Link)
[28] J. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *Proc. of the 3rd International Conference on Learning Representations (ICLR)*, 2015. Aritcle (CrossRef Link)

Dewen Seng received the Ph.D. degree from University of Science and Technology Beijing in 2005. He is currently an associate professor with the School of Computer Science and Technology, Hangzhou Dianzi University. His research interests are in intelligent transportation system, mobile internet, and data mining.

Jiaming Zhang received the B.S. degree from Hangzhou Dianzi University. He is currently pursuing the M.S. degree in computer technology with the School of Computer Science and Technology, Hangzhou Dianzi University. His research interests are in visual analytics and reinforcement learning.

Xiaoying Shi received the Ph.D. degree in control theory and control engineering from Zhejiang University of Technology in 2014. She is currently a lecturer at School of Computer Science and Technology, Hangzhou Dianzi University, China. Her research interests include visual analytics, data mining and urban transportation.