Article

Crowd Evacuation Guidance Based on Combined Action Reinforcement Learning

Yiran Xue, Rui Wu *, Jiafeng Liu and Xianglong Tang

School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China; xueyiran@hit.edu.cn (Y.X.); jefferyliu@hit.edu.cn (J.L.); tangxl@hit.edu.cn (X.T.)
* Correspondence: simple@hit.edu.cn

Abstract: Existing crowd evacuation guidance systems require the manual design of models and input parameters, incurring a significant workload and a potential for errors. This paper proposed an end-to-end intelligent evacuation guidance method based on deep reinforcement learning, and designed an interactive simulation environment based on the social force model. The agent could automatically learn a scene model and path planning strategy with only scene images as input, and directly output dynamic signage information. Aiming to solve the "dimension disaster" phenomenon of the deep Q network (DQN) algorithm in crowd evacuation, this paper proposed a combined action-space DQN (CA-DQN) algorithm that grouped Q network output layer nodes according to action dimensions, which significantly reduced the network complexity and improved system practicality in complex scenes. In this paper, the evacuation guidance system is defined as a reinforcement learning agent and implemented by the CA-DQN method, which provides a novel approach for the evacuation guidance problem. The experiments demonstrate that the proposed method is superior to the static guidance method, and on par with the manually designed model method.

Keywords: evacuation guidance; crowd simulation; deep Q network; reinforcement learning

check for updates

Citation: Xue, Y.; Wu, R.; Liu, J.; Tang, X. Crowd Evacuation Guidance Based on Combined Action Reinforcement Learning. *Algorithms* **2021**, *14*, 26. https://doi.org/10.3390/a14010026

Received: 28 December 2020
Accepted: 15 January 2021
Published: 18 January 2021

1. Introduction

While large-scale shopping malls, office buildings, and other multi-functional buildings meet diverse needs, the complexity of buildings has gradually increased. When disasters such as earthquakes and fires occur, the complex structures of buildings hinder evacuation and create a new safety threat. It is also difficult for crowds to identify an optimal escape route, owing to people's ignorance of the building environment, their limited vision, and them panicking. Under the influence of herd behavior, survivors are prone to cause congestion, or even trampling, risking significant additional loss of life [1]. Thus, a method for guiding crowd evacuation using the most effective route is of great significance for protecting lives and reducing personal and property losses during disasters.

Researchers have developed several crowd evacuation guidance systems based on dynamic guidance signs [1–4] to assist crowds to evacuate effectively during a disaster. Such systems can model building scenes, collect real-time information such as the disaster location and crowd distribution, use path planning algorithms to determine the optimal escape route, and induce the crowd's movement through dynamic guide signage, thus effectively improving crowd escape efficiency in emergencies. However, existing crowd evacuation guidance systems are inseparable from manual designs based on topological maps or grid forms and the manual input of model parameters according to the scene's characteristics. The manual workload for this process is considerable, and it is easy to introduce human errors, which may interfere with subsequent calculations for steps such as path planning.

Therefore, this study proposed an end-to-end crowd evacuation guidance method—based on deep reinforcement learning algorithms—in response to this problem. An artificial intelligence agent was trained, which only takes the building layout as the input, automatically explores and learns the scenario model and path planning method while interacting with the environment, then discovers the optimal action strategy, and directly outputs dynamic signage information. For the actualization of this method, a reinforcement-learning agent-simulation interactive environment was designed, which is based on a social force crowd dynamics simulation. To solve the problem of "dimensional disaster" that occurs when deep Q network (DQN) [5], a typical method in deep reinforcement learning, is applied to crowd evacuation guidance, this paper proposed a combined action space DQN method (CA-DQN), which reduced the complexity of the network structure and improved the algorithm's practicality in complex architecture scenes.

1.1. Crowd Simulation and Evacuation Guidance

Crowd movement simulation is an essential basis for analyzing and researching crowd behavior characteristics, self-organization, and other crowd evacuation phenomena [6]. Crowd simulation research can be categorized as macroscopic and microscopic models [7]. Macroscopic models mainly examine the crowd's overall state of movement, generally using grid models, such as cellular automata [8]. For example, the data-driven crowd simulation method calculates the velocity field using the fluid dynamics method and acts on the continuum model [9], and the lattice-Boltzmann-based method detects the abnormal movement of crowds [10]. The microscopic models use dynamics methods to simulate the movement characteristics of each individual. Typical methods include social force models that introduce subjective human factors [11].

Researchers hope to improve the evacuation efficiency in simulation research, and model crowd movements that are closer to reality. Therefore, the problem of path planning in crowd evacuation has attracted attention from researchers. Consequently, researchers have used the bee colony algorithm of swarm intelligence to improve path searching [12]. Some researchers combined multiple sensor information, and introduced a path selection method for perceiving disaster locations [13]. The path planning methods in simulation environments integrate environmental information, and then, calculate an escape path that maximizes global evacuation efficiency. In actual scenes, the survivors can only grasp the information around them with their limited vision and experience. Even if the building monitoring system can master an optimized escape route, the system requires a particular means of informing the survivors.

To indicate the escape route, emergency escape signs are generally installed in large buildings. These emergency signs can be divided into static and dynamic guidance signs [14]. In experiments based on real scenes [15] and simulation experiments based on social force models [16], static guidance signs played an important positive role in evacuation efficiency. Unlike static guidance signs that indicate a preset evacuation route, dynamic guidance signs can display additional guidance according to real-time conditions, such as the distribution of people in a disaster scene. Studies show that, when a particular exit is unavailable, dynamic guidance signs can effectively induce people to evacuate from other exits [17]; moreover, when danger occurs, dynamic signs can guide people to avoid unsafe routes [18].

Combining the above-mentioned crowd simulation environment, path planning algorithm, and dynamic guidance signs, researchers have developed several crowd evacuation guidance systems. These systems are based on the building environment model, realize closed-loop feedback from perceiving scene information and evacuation route planning to crowd movement guidance, and have practical value [1]. The dynamic guidance method, which is based on network flow path planning on the topological graph model [2], and the dynamic evacuation system, which uses simulated cameras to collect crowd density information and apply a real-time shortest path algorithm [3], are examples. Some studies have included real buildings and established a parallel emergency evacuation system framework, achieving great practical significance [4].

This type of system's necessary processes include several steps such as inputting a scene layout, manually constructing a topology map or grid model, inputting model parameters according to factors such as path capacity, applying path planning algorithms, and setting dynamic guidance signs. The steps of constructing models and inputting parameters introduce high manual participation and a significant workload, possibly causing errors due to human agency and magnifying them in the subsequent steps to negatively affect the system's evacuation efficiency.

1.2. Deep Reinforcement Learning

Reinforcement learning [19] is a crucial component in the field of artificial intelligence. This method learns mapping from the environment state to action by interacting with the environment and through trial and error, and then, finds the optimal behavioral strategy to maximize the accumulated reward. Combined with deep neural networks, the deep reinforcement learning agent can directly use images as input, and internalize feature extraction and value function estimation in the network structure, which significantly expands the agent's perception and decision-making capabilities. The iconic achievements of deep reinforcement learning include the DQN method, which surpassed the abilities of human players in Atari video games [5]; AlphaGo [20], which defeated top human players in Go; and AlphaStar [21], which attained master-level ranking in StarCraft 2 online battles.

The reinforcement learning model [19] is based on the Markov decision process (MDP), which is described as a four-tuple (S, A, P_a, R_a). S is the set of all states, the state space, and A is the action space. The state transition function $P_a(s, s') = \Pr(s_{t+1} = s' | s_t = s, a_t = a)$ represents the probability of an agent performing action a in state s and the environment entering state s'. The reward function $R_a(s, s')$ represents the agent's instant reward, when it executes action a in state s and the environment enters state s'. The agent observes the environment state s_t at each discrete time step t, and selects action $a_t = \pi(s_t)$ to act on the environment according to strategy $\pi : S \to A$. The environment feeds back reward r_t to the agent and transfers to the next state, s_{t+1}. The interaction process between the agent and environment is shown in Figure 1.

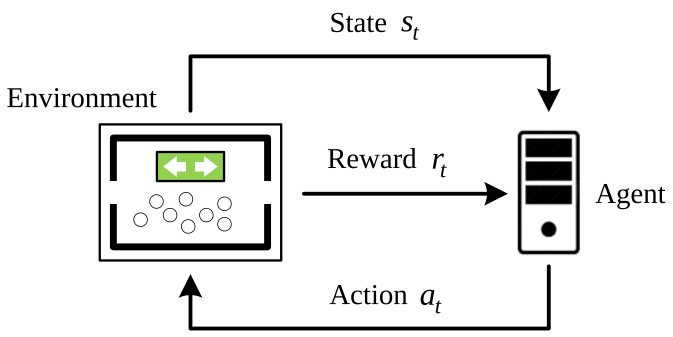


Figure 1. Schematic of reinforcement learning model.

Based on the MDP model of reinforcement learning, the state-action value function is defined, which can also be called the action-value function:

$$Q_\pi(s_t, a_t) = \mathbb{E}_\pi[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots] \tag{1}$$

It represents the expected cumulative reward obtained after performing action a_t in state s_t according to strategy π, where γ is the reward decay coefficient. With the optimal strategy π^*, the optimal action-value function Q^* satisfies the Bellman equation:

$$Q^*(s_t, a_t) = \mathbb{E}_{\pi^*}[r_t + \gamma \max_a Q^*(s_{t+1}, a)] \tag{2}$$

The DQN method, which was developed by the Google DeepMind team [5], uses a deep neural network to approximate the action-value function. It is divided into the current Q network with parameter θ and the target Q network with parameter θ^-, with the current Q network parameters being copied to the target Q network at regular intervals. DQN uses a greedy strategy, always choosing an action with the maximum Q value in the current state, and adds a certain probability to select random actions as the exploration process during training. DQN also uses an experience pool to store and manage samples.

For a sample $e_t = (s_t, a_t, r_t, s_{t+1})$ in a time step, there is an estimate of $Q_\pi(s_t, a_t)$ from Formula (2):

$$r_t + \gamma \max_a Q(s_{t+1}, a; \theta^-) \tag{3}$$

which is called the temporal difference (TD) target. Then, the TD error of this sample is defined as the difference between the TD target and the current value of Q function:

$$\delta_t = r_t + \gamma \max_a Q(s_{t+1}, a; \theta^-) - Q(s_t, a_t; \theta) \tag{4}$$

For a batch of data $B = \{e_1, \ldots, e_t\}$ sampled from the experience pool, the network loss function is defined as the squared error loss $L(\theta) = \mathbb{E}_B(\delta_t^2)$; then, the error backpropagation algorithm is applied to update the network parameters. In the process of minimizing the loss function, the Q function gradually converges to the optimal value, and finally an optimized strategy π is obtained.

DQN has made breakthroughs in tasks such as mastering Atari video games that use images as input. Based on DQN, researchers have proposed improvement methods such as the double DQN (DDQN) method, which uses the current Q network to select the target action [22], and priority experience replay method [23], which uses the TD error to set the priority of the sample in the experience pool.

However, the action space output by DQN is discrete, and an output layer node is used to evaluate each possible action combination. Therefore, when the action dimension increases, the network complexity will increase exponentially. In the crowd evacuation guidance problem, the agent uses the display state of dynamic guidance signs as the output action, and the discrete action of each sign forms an independent action dimension. When there are numerous dynamic guidance signs in complex architectural scenes, the output layer of the DQN will become excessively large for the algorithm to actualize.

2. Methods

2.1. Reinforcement Learning Model for Crowd Evacuation Guidance

In the crowd evacuation guidance problem, the evacuation guidance system is regarded as an agent, and the environment, with which the agent interacts, includes the actual building scene and people moving within the building. An image is used to represent the architectural scene. Various sensors, such as cameras, collect the crowd's movement state and draw this data into the scene image. As shown in Figure 2, The scene image now contains the information that is required in the current environment, which is defined as the environment state $s_t \in S$ of the MDP. The evacuation guidance system displays signals through dynamic guidance signs to guide and direct crowd movement. Therefore, the agent action $\boldsymbol{a_t} \in A$ corresponds to the guide sign signal, $\boldsymbol{a_t}$ is a discrete vector, each dimension corresponds to a guide sign, and the component is a sign display state (left or right). Owing to the complexity of the environment and crowd movement, the state transition function $P_a(s, s')$ is unknown; hence, the agent must learn and adapt during the interaction process. The design of the reward function determines the optimization direction and learning purpose of the agent. In the crowd evacuation problem, the reward function should be designed according to the number of people who successfully evacuated or the time required for evacuation. This paper defines $r_t = -1$, which indicates that a fixed penalty is given at each time step. The agent's learning goal is to minimize the cumulative penalty, that is, the shortest evacuation time for the entire crowd.

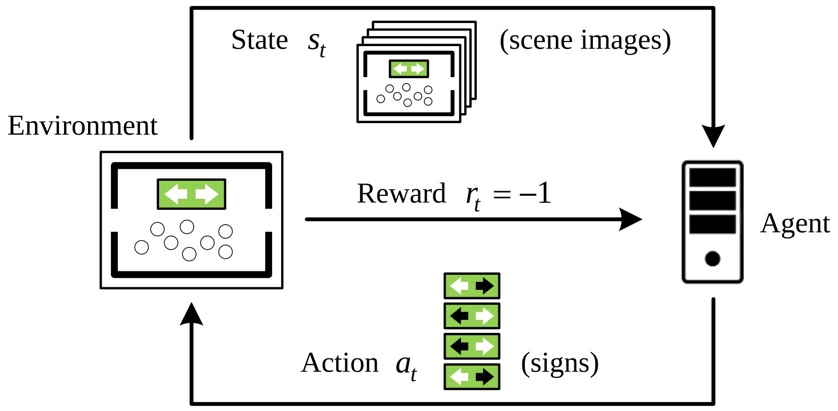


Figure 2. Reinforcement learning model for evacuation guidance.

The training process for reinforcement learning agents requires continuous interaction with the environment, as well as learning through exploration and trial and error. The required interaction scale is large, generally over tens of thousands of cycles and millions of time steps. Moreover, the agent's lack of knowledge in the initial training period may cause more potential dangers. Therefore, the reinforcement learning agent used for the intelligent evacuation guidance system must be trained in a simulation environment, and deployed in the actual building after the training is completed.

The evacuation guidance system agent explores and learns through various interactions with the simulation environment, and finally obtains an approximate optimized strategy $\pi(s)$. There is no need to manually design the building route topology graph or grid model during the learning process. The agent can autonomously discover and optimize the guidance strategy, and the design of additional intermediate algorithms, such as path planning, is not required. In practical applications, the building's map is inputted into the simulation system, and the reinforcement learning agent is pre-trained through simulation to obtain the optimal strategy in this scene. After being deployed to the actual building, the sensors collect crowd movement information at each moment using additional computer vision algorithms, and draw an image combined with the building's map as the agent's observation state input. The agent calculates the action vector according to the previously optimized strategy, and the dynamic guidance signs display the corresponding signal to realize practical guidance for the crowd's evacuation.

2.2. Combined Action Space DQN

In the network structure, DQN uses a multi-layer convolutional neural network to process the image input, and then, connects a multi-layer fully connected neural network with each neuron in the output layer, corresponding to a possible combination of discrete actions. For the components of the action, the total action space is the Cartesian product of each dimension's action space. When the action space has n dimensions and each dimension has m discrete actions, the DQN network needs m^n output layer nodes to correspond to value $Q(s, a)$ of different actions. Therefore, as the number of action dimensions increases, the DQN network structure's complexity increases exponentially, making the algorithm unrealizable. Meanwhile, an excessive number of output layers will also reduce sample utilization and the difficulty in updating network parameters. This phenomenon is called the "dimensional disaster" of DQN.

When applying crowd evacuation guidance, the agent action is defined as the guide signs' display states. Even if each guide sign has only two states, left and right, for n guide flags, the total action space capacity will reach as much as 2^n. To solve this problem, the CA-DQN was proposed. Since the guide signs' information corresponding to the action a_{t-1} in the previous time step has been included in the scene image s_t, and the signs do not change frequently, $a_t \approx a_{t-1}$. Therefore, it can be approximately considered that under the condition of observation s_t, the strategy of an action dimension is independent of other action dimensions, and an independent neural network structure can be used. As shown in Figure 3, for independent action dimensions, each dimension corresponds to a group of nodes in the output layer of the Q function network, and each group contains all discrete actions in this dimension. This change can be seen as setting the value function $Q_d(s, a^{(d)}; \theta)$ for each action dimension d, and sharing a set of network parameters. At this time, the number of nodes in the network's output layer is the sum of the number of discrete actions in each dimension, and the growth rate of the number of output layer nodes decreases from exponential growth to linear growth. For example, the number of output layer nodes required for n guiding signs is $2n$.

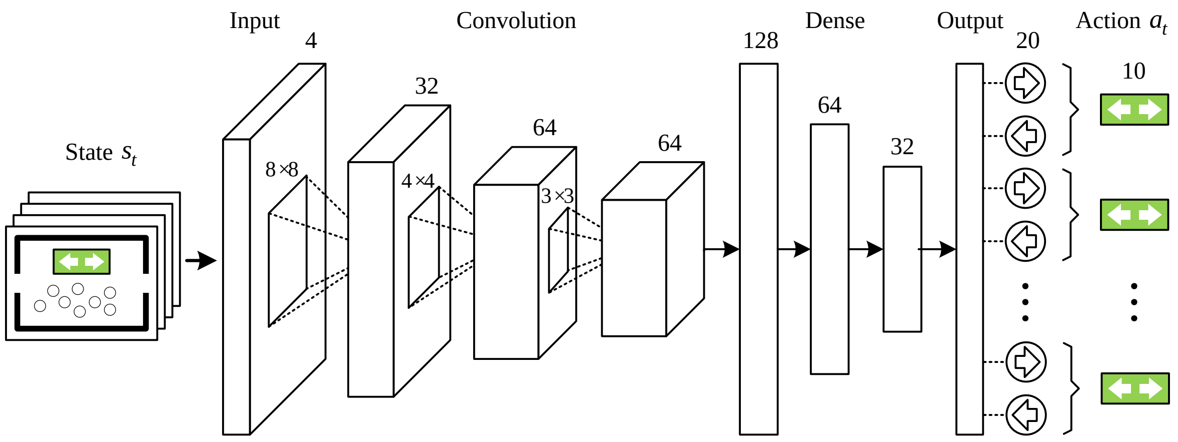


Figure 3. Network structure of combined action-space deep Q network (CA-DQN).

In CA-DQN, the agent's action

$$\boldsymbol{a_t} = (a_t^{(1)}, a_t^{(2)}, \ldots, a_t^{(D)}) \tag{5}$$

is a D-dimensional combined action vector. For each dimension d, the agent uses a greedy strategy to select the actions.

$$a_t^{(d)} = \arg\max_a Q_d(s_t, a; \theta) \tag{6}$$

Referring to Formula (4), for a sample $e_t = (s_t, a_t, r_t, s_{t+1})$, this paper define the TD error in each dimension:

$$\delta_t^{(d)} = r_t + \gamma \max_a Q_d(s_{t+1}, a; \theta^-) - Q_d(s_t, a_t^{(d)}; \theta) \tag{7}$$

Combined with the DDQN algorithm that was proposed in [22] to select actions at time $t+1$ with the current Q network to avoid overestimation, the TD error is further defined as follows:

$$\delta_t^{(d)} = r_t + \gamma Q_d(s_{t+1}, \arg\max_a Q_d(s_{t+1}, a; \theta); \theta^-) - Q_d(s_t, a_t^{(d)}; \theta) \tag{8}$$

Then, for a set of samples $B = \{e_1, \ldots, e_t\}$ from the experience pool, the loss function of the neural network is defined as the arithmetic mean of the squared error loss.

$$L(\theta) = \mathbb{E}_B\left[\frac{1}{D}\sum_d \left(\delta_t^{(d)}\right)^2\right] \tag{9}$$

According to the loss Formula (9), the neural network is trained with the error backpropagation algorithm. Subsequently, for each sample, an output layer node is selected for each dimension of the action, which participates in the calculation of the TD error and backpropagation of network error, such that there are a total of D output layer nodes that can be updated. Compared with each sample in DQN that can only update one output layer node, CA-DQN improved the utilization efficiency of the samples.

2.3. Priority Experience Playback of CA-DQN

DQN randomly samples from the experience pool, regardless of the sample differences, and has low sample utilization efficiency. Using the preferred experience playback method [23], with the sample TD error defined by Formula (4), the sampling priority is

$$p_t = (|\delta_t| + \varepsilon)^{\alpha} \tag{10}$$

where ε and α are constants. A sample with a more significant absolute value of the TD error implies that it contains more useful information and should be given a higher sampling priority, which can improve sample utilization and training efficiency.

The TD error of a sample in CA-DQN is defined by Formula (8), which is a vector

$$\boldsymbol{\delta_t} = (\delta_t^{(1)}, \ldots, \delta_t^{(d)}) \tag{11}$$

When the preferred experience playback method is applied, the sample priority can be defined as the maximum absolute TD error of each dimension (called the CA-DQN-max method).

$$p_t = \left(\max_d \left| \delta_t^{(d)} \right| + \varepsilon \right)^{\alpha} \tag{12}$$

The maximum value can highlight samples with a greater value for training in a particular action dimension. However, the dimension with the max TD error may cover up other dimensions, thereby decreasing the training stability and efficiency.

The better way is to define sample priority as the mean absolute value of the TD error of each dimension (called the CA-DQN-mean method).

$$p_t = \left(\frac{1}{D} \sum_d \left| \delta_t^{(d)} \right| + \varepsilon \right)^{\alpha} \tag{13}$$

The mean value collects information on the sample importance from all dimensions, and calculates a more accurate priority under the influence of random interference. Consequently, it helps maintain the training process's stability and obtain a more effective training result.

3. Results

3.1. Experiment Design and Implementation

In this study, a crowd dynamics simulation system based on the social force model [3], is used as the agent's interactive environment. A typical multi-room, dual-exit indoor scene is constructed, in which the guidance signs are marked as green arrows, and the exits are marked as green rectangles. The simulation system takes the display states of the evacuation guidance signs as input. In the simulation, the individuals first determine a subjective driving force direction based on the nearby guidance signs and the distance to each exit. When the simulated individuals do not see the evacuation guidance signs, they choose the closest exit and escape according to the statically shortest route. When the individuals notice the evacuation guidance sign, they escape in the direction indicated by the sign, unless they are close to an exit, in which case they may ignore the sign. When congestion occurs, the individual's speed is limited, and the subjective driving force that tends to get rid of congestion is added. Then, the dynamic simulation calculates the speed and displacement of the individual movement, simulating the collision between people and people, and people and walls. Individuals within the exit area are deemed to have successfully escaped. At the end of a simulation step, the crowd's positions and the signs' states are painted in a scene image as output, which is passed to the reinforcement learning agent as input to its neural network. The simulation system is based on C++ and the Qt library.

As shown in Figure 4, the simulation scene's size is 29.2 m × 19.7 m, and the size of the floor map is 499 × 337 pixels. The scene contains two exits and six rooms, where the upper and lower channels connect the rooms and exits. Each channel is set to have five dynamic guidance signs, and the signs can indicate one of two directions. The number of people is 200, and their initial positions are randomly distributed in a circular range; the range of the distribution center and radius is $x \in (100, 140)$, $y \in (60, 280)$, and $r \in (100, 200)$, and the maximum individual movement speed is 5 m/s. Each time step of the simulation system dynamics calculation is 40 ms, and the upper limit of the simulation time is 100 s.

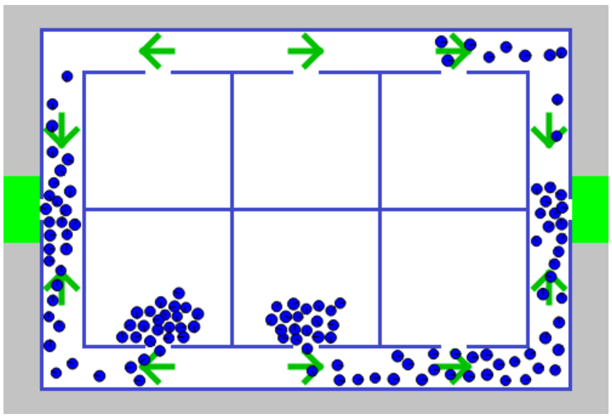


Figure 4. Simulation environment.

The CA-DQN method is implemented based on Python, the TensorFlow platform, and OpenAI/baseline library. In each time step of the reinforcement learning agent, the simulation system first performs a five-step calculation, simulating the crowd's movement within 200 ms. The last four images obtained are down-sampled into 1/2-size grayscale images, and combined into a four-channel image of 249 × 168 pixels, which is input to the agent's Q network as state s_t. The Q network structure is shown in Figure 3, comprising a three-layer convolutional neural network and three-layer fully connected neural network. The 20 neurons in the output layer are divided into ten groups. From each group, the larger output value is selected as the display signal for the dynamic guide sign. By combining them, a ten-dimensional discrete output vector is formed as agent action a_t. a_t acts on the simulation system to change the direction displayed by the ten guide signs, thereby directing the crowd's movement. At this time, the interaction between the agent and simulation environment completes a cycle. The agent's reward for each step is fixed at -1, implying that it obtains a reward of -5 per second.

The agent's training goal is to reduce the overall evacuation time. For example, the exits' congestion and the imbalanced exit utilization lead to longer evacuation time and lower rewards, giving the agent negative feedback. The agent will learn how to avoid this situation and find the optimal strategy through trial and error.

Among the training parameters, the batch size is 64, the learning rate is 10^{-5}, the total time step is 10^7, the experience pool's sample size is 10^5, and the current Q network parameters are copied to the target Q network every 2×10^4 steps. The experimental hardware platform is an AMD Threadripper 2990WX CPU, NVIDIA RTX 2080Ti GPU, and 128 GB memory.

3.2. *Experimental Results and Analysis*

As the original DQN method would require $2^{10} = 1024$ output layer nodes for the experiments conducted herein, compared with the 20 nodes for CA-DQN, the DQN network is too large to implement under existing conditions. Therefore, in this study, the chosen method is based on static guidance signs, an evacuation guidance algorithm based on topology map modeling, and dynamic Dijkstra shortest path method [3] for comparison. The static guidance signs indicate the nearest exit, regardless of the real-time distribution of the crowd. Because the individuals in this simulation environment escape to the nearest exit if they cannot find a guide sign, the static guide sign's role in this simulation environment is similar to having no guide sign. The dynamic Dijkstra shortest path method requires experts to manually build a topology map model based on its channel structure. They have to set up multiple virtual camera nodes, count the crowd density on different paths, adjust the real-time weights of each edge, and use the Dijkstra algorithm for path planning to achieve effective crowd evacuation.

The agent only needs to be previously trained once for a scene, and the training process takes about three days. The training curve in Figure 5 shows that the agent reaches the optimal strategy after approximately 30,000 training periods, and when the number of interactions between the agent and simulation environment is approximately 6.4×10^6 time steps. When the agent executes the optimized strategy obtained from training, for a 200 ms step, the execution of strategy takes 2.4 ms, and the crowd simulation takes 14.3 ms. The processing speed can meet the real-time requirement of actual deployment.

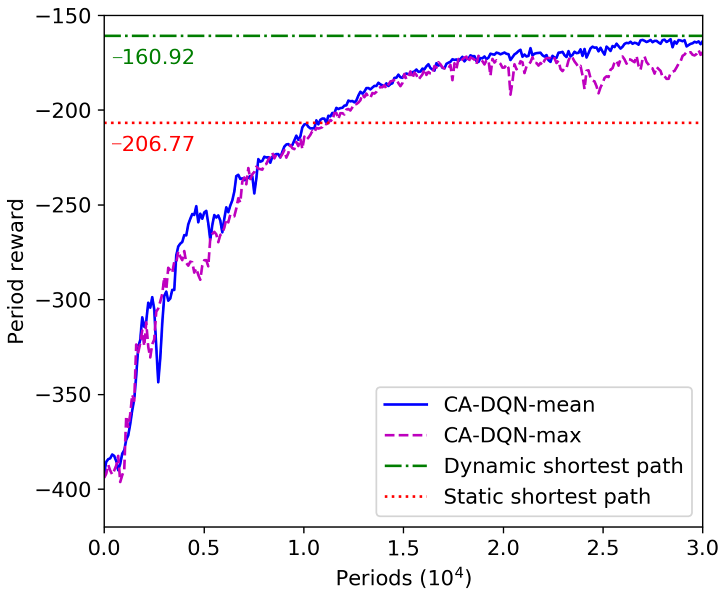


Figure 5. Training curve of the agent.

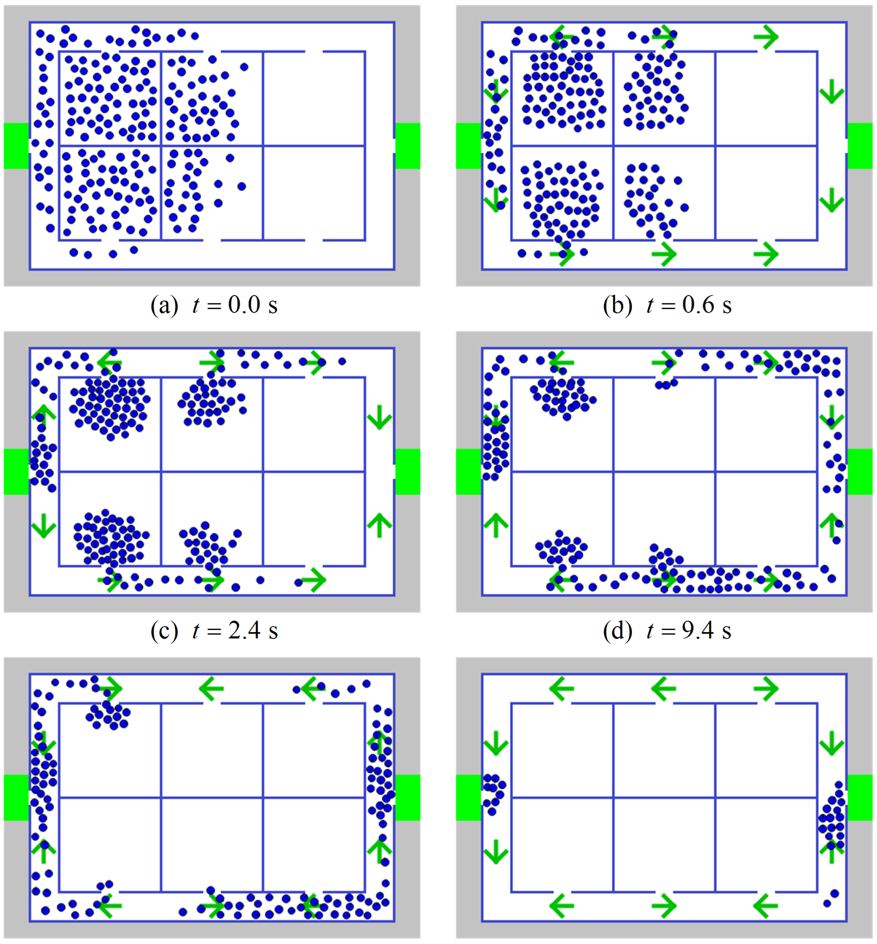
The CA-DQN-mean method is more stable in the later stage of training; moreover, the training effect of the CA-DQN-mean method is better than that of the CA-DQN-max method. This result indicates that under the influence of random interference, the CA-DQN-mean method can calculate the sample importance more accurately with the information of all the dimensions. Consequently, it is more appropriate to define the sample priority as the average of absolute TD errors in each dimension.

As shown in Table 1, a 100-cycle evacuation simulation was performed for different evacuation methods using new random crowd distribution parameters. The bold numbers indicate the best period reward and the shortest evacuation time. The average period reward of the agent's optimal strategy was −158.25, implying that the average evacuation time was 31.65 s, which was better than the 41.35 s obtained using static guidance signs and 32.18 s obtained using the dynamic Dijkstra shortest path method. This demonstrates that the intelligent evacuation guidance agent based on CA-DQN can effectively guide an evacuation.

Table 1. Comparison of evacuation times using different methods.

Method	Period Reward	Evacuation Time (s)
Static sign	−206.77	41.35
Dynamic shortest path	−160.92	32.18
CA-DQN-mean	−**158.25**	**31.65**
CA-DQN-max	−160.40	32.08

Figure 6 shows a typical evacuation process. Figure 6a is the crowd's initial distribution, which is primarily in the four rooms on the left. Without dynamic guidance, the static evacuation strategy with the shortest distance to the exit will cause congestion at the exit on the left, and the right exit will not be used effectively. In Figure 6b, the agent perceives the crowd's distribution in the scene image with CNN and leads some of the crowd in the upper left room to the left exit and the rest of the crowd to the right exit. Notice that the side signs for the lower channel lead to an unexpected direction. This phenomenon is because when survivors are very close to the exit in the simulation environment, they will ignore the signs and go directly to the exit. Such signs have little influence on the evacuation efficiency, and it is hard to obtain feedback for training. For the same reason, the left side signs in Figure 6c are mostly ignored.



(a) $t = 0.0$ s (b) $t = 0.6$ s (c) $t = 2.4$ s (d) $t = 9.4$ s (e) $t = 13.8$ s (f) $t = 28.0$ s

Figure 6. Six moments in a typical evacuation process.

In Figure 6d,e, the congestion at the left exit has been relieved. The right exit is expected to have more people being evacuated; consequently, the agent will lead the remaining people in the lower-left area to the left exit. Notice the first and third signs in the upper channel of Figure 6e seem to be unexpected. This phenomenon is because jamming will slow down the crowd in the simulation environment. The agent learned this feature and tries to delay part of the crowd to avoid exit congestion. Finally, in Figure 6f, the crowd is evacuated from both sides of the exit simultaneously, indicating that the crowd evacuation guide agent has maximized the crowd evacuation efficiency. The signs of empty areas in Figure 6f will not affect the evacuation efficiency, and the agent cannot learn from feedback; therefore, the signs in these areas are uncertain.

The number of people initialized in the simulation scene was changed, and 100 cycles of evacuation simulation were run. A comparison of the evacuation effects of different methods is shown in Figure 7. When the number of people was small, each channel could remain unobstructed, and the static guidance method was effective. When the number of people increased, the static guidance method was affected more, while CA-DQN and the dynamic shortest path method avoided crowd congestion. When the number of people increased to more than 80, the three dynamic methods' evacuation effect was better than the static method. The CA-DQN-mean method, which used the average value of TD errors in each dimension to define the sample priority according to Formula (13), performed better than the dynamic shortest path method. The effect of the CA-DQN-max method, which was defined by Formula (12), was equivalent to that of the dynamic Dijkstra shortest path method.

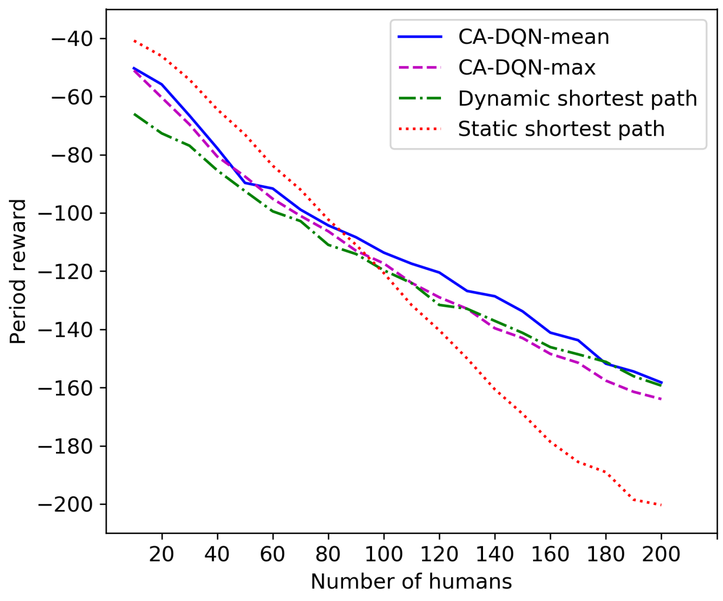


Figure 7. Period reward with varying number of humans.

The experimental results show that, compared with static signs that cannot perceive crowd distribution information, the proposed CA-DQN reinforcement-learning-based crowd evacuation guidance method can dynamically adjust the display signals of the guidance signs, and effectively improve the efficiency of crowd evacuation. Compared with the dynamic Dijkstra shortest path method that is based on topological map modeling, the proposed method demonstrated higher evacuation guidance efficiency, while avoiding the workload and potential manual errors of artificial topology map construction.

4. Conclusions

This study analyzed crowd evacuation guidance using dynamic guidance signs. This paper proposed a crowd evacuation method based on combined action space deep reinforcement learning to overcome the disadvantages of existing methods. Previous methods required the manual design of topological graph models or grid models and independent path planning algorithms, resulting in a large manual workload and easily introducing human errors. Through end-to-end deep learning, the agent explores and learns building structure and path planning methods on its own during the training process, and automatically corrects cognitive errors through environmental feedback, thereby identifying the optimal evacuation guidance strategy.

To overcome the "dimension disaster" problem caused by the number of output guidance signs when the typical DQN method in deep reinforcement learning is applied to crowd evacuation problems, this paper proposed the CA-DQN method. It reduces the network structure complexity of the output action dimension from exponential growth to linear growth, and improves the usability of reinforcement learning methods in complex scenarios and large-scale crowd evacuation problems. According to the different definitions of sample priority, the CA-DQN method is divided into CA-DQN-max and CA-DQN-mean methods.

The experiments on crowd simulation based on the social force model showed that the proposed method effectively improved the crowd evacuation efficiency, and reduced the evacuation time compared with static guidance signs. Moreover, the results of the proposed method are comparable to those of the dynamic shortest path method based on manual modeling. Between the CA-DQN-max and CA-DQN-mean methods, CA-DQN-mean showed advantages in training stability and result strategy's efficiency, which indicates that it is more appropriate to define the sample priority as the average of absolute TD errors in each dimension.

Future work will further improve the training efficiency of reinforcement learning agents in complex scenarios, including more rooms, multi-layer building scenes, and obstacles. Moreover, restrictions will be imposed on the frequency of output signal changes, making it easier to understand in real scenarios.

Author Contributions: Conceptualization, Y.X.; methodology, Y.X. and R.W.; software, Y.X.; validation, Y.X., R.W. and J.L.; resources, J.L. and X.T.; writing—original draft preparation, Y.X. and J.L.; writing—review and editing, Y.X., R.W., J.L. and X.T.; visualization, Y.X.; supervision, R.W. and X.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China grant number 61672190.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ran, H.; Sun, L.; Gao, X. Influences of intelligent evacuation guidance system on crowd evacuation in building fire. *Autom. Constr.* **2014**, *41*, 78–82. [CrossRef]
2. Desmet, A.; Gelenbe, E. Capacity based evacuation with dynamic exit signs. In Proceedings of the 2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS), Budapest, Hungary, 24–28 March 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 332–337.
3. Zhao, W.; Liu, C.; Lian, X.; Xue, Y.; Guo, Y. Simulation of crowd movement and design and implementation of evacuation optimization method. *J. Syst. Simul.* **2014**, *26*, 523–529. (In Chinese)
4. Dong, H.; Ning, B.; Chen, Y.; Sun, X.; Wen, D.; Hu, Y.; Ouyang, R. Emergency management of urban rail transportation based on parallel systems. *IEEE Trans. Intell. Transp. Syst.* **2012**, *14*, 627–636. [CrossRef]
5. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
6. Yang, S.; Li, T.; Gong, X.; Peng, B.; Hu, J. A review on crowd simulation and modeling. *Graph. Model.* **2020**, *111*, 101081. [CrossRef]
7. Xu, M.L.; Jiang, H.; Jin, X.G.; Deng, Z. Crowd simulation and its applications: Recent advances. *J. Comput. Sci. Technol.* **2014**, *29*, 799–811. [CrossRef]
8. Muramatsu, M.; Irie, T.; Nagatani, T. Jamming transition in pedestrian counter flow. *Phys. A Stat. Mech. Appl.* **1999**, *267*, 487–498. [CrossRef]

9. Bisagno, N.; Conci, N.; Zhang, B. Data-driven crowd simulation. In Proceedings of the 2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Lecce, Italy, 29 August–1 September 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
10. Xue, Y.; Liu, P.; Tao, Y.; Tang, X. Abnormal prediction of dense crowd videos by a purpose–driven lattice boltzmann model. *Int. J. Appl. Math. Comput. Sci.* **2017**, *27*, 181–194. [CrossRef]
11. Helbing, D.; Molnar, P. Social force model for pedestrian dynamics. *Phys. Rev. E* **1995**, *51*, 4282. [CrossRef]
12. Liu, H.; Xu, B.; Lu, D.; Zhang, G. A path planning approach for crowd evacuation in buildings based on improved artificial bee colony algorithm. *Appl. Soft Comput.* **2018**, *68*, 360–376. [CrossRef]
13. Cuesta, A.; Abreu, O.; Balboa, A.; Alvear, D. Real-time evacuation route selection methodology for complex buildings. *Fire Saf. J.* **2017**, *91*, 947–954. [CrossRef]
14. Zhou, M.; Dong, H.; Ioannou, P.A.; Zhao, Y.; Wang, F.Y. Guided crowd evacuation: Approaches and challenges. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 1081–1094. [CrossRef]
15. Fu, L.; Cao, S.; Song, W.; Fang, J. The influence of emergency signage on building evacuation behavior: An experimental study. *Fire Mater.* **2019**, *43*, 22–33. [CrossRef]
16. Yuan, Z.; Jia, H.; Zhang, L.; Bian, L. A social force evacuation model considering the effect of emergency signs. *Simulation* **2018**, *94*, 723–737. [CrossRef]
17. Galea, E.R.; Xie, H.; Deere, S.; Cooney, D.; Filippidis, L. Evaluating the effectiveness of an improved active dynamic signage system using full scale evacuation trials. *Fire Saf. J.* **2017**, *91*, 908–917. [CrossRef]
18. Cho, J.; Lee, G.; Lee, S. An automated direction setting algorithm for a smart exit sign. *Autom. Constr.* **2015**, *59*, 139–148. [CrossRef]
19. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
20. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
21. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [CrossRef] [PubMed]
22. Hasselt, H.v.; Guez, A.; Silver, D. Deep reinforcement learning with double Q-Learning. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 2094–2100.
23. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.