A Metaverse-Based Teaching Building Evacuation Training System With Deep Reinforcement Learning

Jinlei Gu, Jiacun Wang, *Senior Member, IEEE*, Xiwang Guo, *Senior Member, IEEE*, Guanjun Liu, *Senior Member, IEEE*, Shujin Qin, *Member, IEEE*, and Zhiliang Bi, *Graduate Student Member, IEEE*

***Abstract*—With the development of IoT, virtual reality, cloud computing, and digital twin technologies, the advent of metaverse has attracted increasing world attention. Metaverse integrates and applies multiple emerging technologies to cloud education, smart health, digital government, and emergency evacuation. Evacuation systems are of great importance to ensure life safety. Due to panic, people in a building may not be able to make the right judgment to choose an optimal path to leave the building in case of an emergency event such as a fire. As a branch of machine learning, deep reinforcement learning (DRL) can model an evacuation scene, collect real-time information, such as crowd distribution and disaster location, find the optimal escape path with a path-planning algorithm, induce the movement state of the crowd through dynamic guidance signs, and improve the evacuation efficiency. In this article, we apply DRL technology to solve the efficient emergency evacuation problem with the help of metaverse and show a training system built upon metaverse that would enable evacuees to choose the most efficient route and leave the building in the least amount of time. The information collected by various sensors, such as video cameras and smoke detectors, can give a whole picture of the status of the building in a real-time manner. The collected data are processed by cloud servers in which a DRL model is trained to dynamically guide evacuees. Experiments in different simulation scenes demonstrate that the proposed method is superior to the traditional static guidance method in saving evacuation time. It can effectively avoid major crowding along the evacuation route and improve evacuation efficiency.**

***Index Terms*—Cyber–physical–social systems, deep *Q*-leaning networks, emergency evacuation, metaverse, three-dimensional (3-D) simulation, virtual reality.**

Manuscript received 15 November 2022; revised 9 December 2022; accepted 16 December 2022. Date of publication 4 January 2023; date of current version 17 March 2023. This article was recommended by Associate Editor F.-Y. Wang. *(Corresponding author: Jiacun Wang.)*

Jinlei Gu and Jiacun Wang are with the Department of Computer Science and Software Engineering, Monmouth University, West Long Branch, NJ 07764 USA (e-mail: s1315289@monmouth.edu; jwang@monmouth.edu).

Xiwang Guo and Zhiliang Bi are with the Computer and Communication Engineering College, Liaoning Petrochemical University, Fushun 113001, China (e-mail: x.w.guo@163.com; 1062814637@qq.com).

Guanjun Liu is with the Department of Computer Science, Tongji University, Shanghai 200092, China (e-mail: liuguanjun@tongji.edu.cn).

Shujin Qin is with the College of Economics and Management, Shangqiu Normal University, Shangqiu 476000, China (e-mail: sjchin@vip.126.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TSMC.2022.3231299.

Digital Object Identifier 10.1109/TSMC.2022.3231299

I. Introduction

METAVERSE originates from Neil Stephenson's science fiction novel named Snow Crash in 1992, which describes a virtual world parallel to the real world [1]. Since its inception, the concept of metaverse has evolved and expanded, such as life-logging [2], augmented reality [3], and the three-dimensional (3-D) virtual world [4]. Metaverse integrates a variety of emerging technologies [5], [6]. While past research has focused on the creation of virtual worlds, such as games [7], with the maturity of VR/AR, 5G, artificial intelligence, and other technologies, metaverse has gradually begun to play a role in political, economic, social, and cultural fields [8]. At the same time, these areas also began to change from the CPS strategy to the CPSS strategy, where CPS is a short for cyber–physical systems, and CPSS stands for cyber–physical–social systems.

CPS is a mechanism that integrates the physical system with the Internet under the control of computer algorithms, so that physical components and software components can run in different space and time scales. However, with the development of big data, IoT, sensor networks, automatic driving, and other technologies, the factor of human society in the system has become more and more important. Human society has begun to occupy the network world. People depend on the network but also interact with the physical world through the network. Therefore, CPSS as new paradigm is proposed [9]. As an extension of CPS, CPSS perfectly integrates the cyber space, physical space, and social space and better realizes the interaction between the virtual world and the real world. The essence of metaverse is the virtual world that can be experienced on the Internet. It turns the human world from the dual space to the ternary space, which can project the physical world and human society into the information space. CPSS is an abstract of metaverse [10].

While modern teaching buildings meet teaching needs, the internal structure of such buildings can be complex. When earthquakes, fires, or other disasters occur, the complex structure in a building hinders the evacuation of people and poses a new threat to life safety [11], [12]. When a disaster occurs, it is difficult for people to accurately find the optimal escape route due to factors, such as lack of understanding of the building environment, limited vision, psychological panic, and so

on [13], [14]. Under the influence of herd mentality, evacuees are easy to form congestion or even stampede, causing greater losses. How to guide people to evacuate in the most effective way is of great significance to protect life and reduce personnel and property losses in disasters.

In the past decades, researchers have developed various crowd evacuation guidance systems, such as those through dynamical signs [15] and by robots [16]. This kind of systems can model the building scene, collect real-time information, such as crowd distribution and disaster location, find the optimal escape path with a path-planning algorithm, and induce the movement state of the crowd through dynamical guidance signs or robots, which effectively improves the escape efficiency of the crowd in an emergency [17]. Although these existing systems have achieved some success, they rely on the large quantity of drill data to formulate corresponding security strategies. The cost of repeated drills is very high, and the actual evacuation scenario is complex. Moreover, with a large manual workload, it is very easy to introduce errors to the collected data [18].

To address this problem, we propose a crowd evacuation guidance approach that uses a deep reinforcement learning (DRL) algorithm to train an evacuation guidance agent. The agent takes the environment information such as architectural plan as an input, constantly learns and explores path-planning methods during the interaction with the environment, finally finds the optimal action strategy, and outputs dynamic guidance signs to evacuate people.

To realize this method, an interactive simulation environment of reinforcement learning (RL) agent based on crowd dynamics simulation of social force model [19] is designed. Benefiting from artificial intelligence support in metaverse, there are enormous advantages and opportunities in implementing evacuation systems. As a kind of artificial intelligence algorithm, DRL can grab vast data from its buffer pool and uses it to train people to evacuate [20] and reduce the complexity of neural networks and improve the practicability of the algorithm in complex building scenes [21], [22]. The main contribution of this article is summarized as follows.

1) A social structure based on metaverse is proposed, which can effectively provide real data for virtual scenes, make DRL more real and efficient, solve the simulation to real problem that DRL faces, and avoid the negative impact that DRL may have when implemented in real world without full test or training.
2) A DRL algorithm with rainbow deep Q-network (DQN) is developed, which combines several advanced DRL algorithms. Compared with traditional DL algorithms, it improves the stability of neural networks and has the best comprehensive training efficiency.
3) In the virtual scene modeling based on the real scene, the effect of crowd evacuation in emergency incidents is simulated and the visualization is achieved by playing back the pixel images, which improves the authenticity of simulation exercises and has guidance significance for people's evacuation in real situations.

The remainder of this article is arranged as follows. In Section II, an overall framework is introduced. In Section III, the teaching building evacuation problem is discussed. In Section IV, a DRL algorithm for emergency evacuation is described. In Section V, emergency evacuation cases are used to compare the effectiveness of the proposed DRL algorithm and other methods. Section VI summarizes this article.

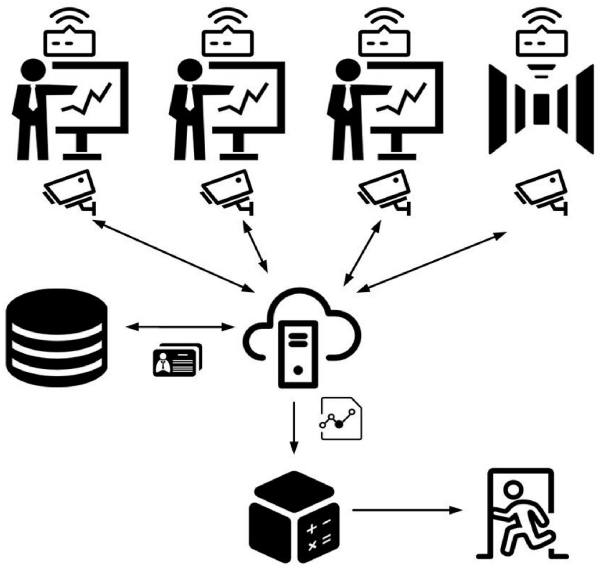


Fig. 1. Illustration of teaching building evacuation. Evacuees are in classrooms and hallways. The floor information and each evacuee's identity are stored in a database. The dynamic distribution of evacuees over the floor is collected by sensors and sent to the server in the cloud for processing with a learning system, which eventually outputs an optimal evacuation path for every evacuee.

II. Overall Framework

The big idea of using metaverse for dynamic evacuation of teaching buildings is illustrated in Fig. 1. By using the technology of the IoT [23], we can obtain the specific distribution of people in the building by placing cameras or other sensors in each classroom and hallway. We can also build a threat situation information scene through smoke or fire sensors, which is helpful for the evacuation system to analyze whether a path can be passed or predict whether a collapse might occur. Through the remote connection between the cloud servers and the databases, we can identify each person, so as to check the number of evacuated people and ensure the accuracy of the model. Moreover, the social relationship between people will also affect the path judgment of people who escape. For example, when the disaster occurs, some people may choose to search their lovers or friends rather than follow the guidance to evacuate.

The collected data is submitted to the cloud servers for analysis, which outputs a series of model parameters such as the scene layouts. The evacuation system takes these as the input and uses the DRL algorithm to automatically learn the training model and explore the optimal path-planning strategy. Finally, an evacuation model will be gained, which reflects how to guide the crowd to the escape exit with maximum efficiency under the current building layout and crowd distribution.

For DRL, it usually requires a simulation environment to simulate the real physical world, which coincides with the

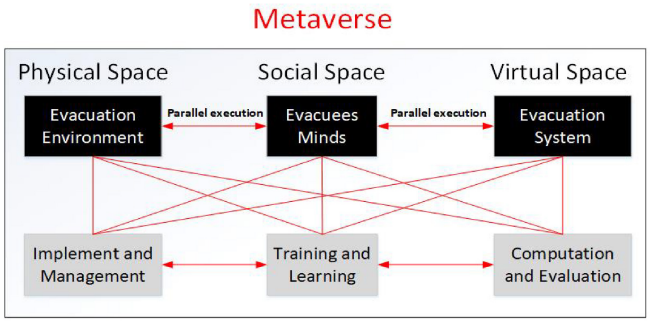


Fig. 2. Schematics of the evacuation system in metaverse. It is an integration of physical space, social space, and virtual space, which are running in parallel and interacting with each other.

concept of the metaverse. If safety and robustness are not taken into account, DRL may have disastrous effects when implemented in real life [24]. In a multiagent system, each participant has his own intention and wants to optimize his own interests. That is why we need metaverse to build a social structure that is efficient, fair, and safe in the multiagent interaction scenario.

Simulation to reality is the main bottleneck for the smooth implementation of DRL in areas other than games [25] due to low computing efficiency and low sampling efficiency and the fact that DRL strategies based only on rewards lack consideration of safety and robustness. To address the issue of simulation to reality, we build a virtual–physical–social space in the metaverse, which consists of three modules, namely, artificial societies, computational experiments, and parallel execution, as illustrated in Fig. 2. Computational experiments and parallel execution improve calculation and sample efficiency. Artificial societies improve the reliability and security of the algorithm. The physical, virtual, and social phases in the metaverse can actually reduce the gap between the virtual world and reality in the absence of computing resources and data [26].

Different theoretical frameworks produce different algorithm designs: from theory to algorithm, safe and robust algorithms are designed, and the third layer tries to make a social decision that can not only improve efficiency but also be fair and safe under the scenario of multiagent interaction [27]. Presently, there are some outstanding research results in the algorithm design of DRL, but the social decision making of the third level is just kicked-off, and its development needs the support of metaverse. The metaverse is responsible for connecting reality and algorithms. Algorithms are learned in metaverse. The metaverse reflects reality more, the algorithms learned are more suitable for the real world.

III. Problem Description

Previous studies have shown that in emergency events, it is not the sudden disaster that affects the safe evacuation of the crowd, but the lack of evacuation experience that makes the crowd panic in the evacuation process and leads to congestion and stampede accidents, which results in inefficient evacuation. In order to solve this problem, some methods are proposed, including robot-led and dynamic sign-based. However, robot-led evacuation is difficult to adapt to the changing environment effectively, and robot navigation is prone to accidents in disasters, resulting in crowd congestion, which has great risks [28]. Therefore, from the perspective of efficiency and safety, this article mainly studies evacuation strategies of dynamic signs in complex virtual scenes.

Most buildings are equipped with emergency escape signs to direct personnel to the nearest exit in case of emergency. Most emergency escape signs are static signs, while dynamic signs are not different from static signs in ordinary times [29]. Static sign guidance plays a key role in the first minute after the evacuation drill starts. Unlike static guidance signs, dynamic guidance signs do not evacuate people according to the predetermined route, instead, they can provide different guidance information according to the real-time situation, such as crowding and stampede. In a disaster scenario, research shows that when one exit is crowded or evacuation efficiency is low, dynamic guidance signs can effectively guide people to evacuate through other exits. They also guide people to move away from unsafe or dangerous routes.

In the actual scene, when classes are on session, teaching buildings often have a large number of people (students and instructors) inside. Because of the differences between different disasters and different scenarios, the evacuation process will be complicated by overcrowding. This presents a challenge for evacuees to get out quickly. In order to improve the efficiency of evacuation, many drills are needed, which can be extremely costly. Some disasters, such as earthquake and flood, are difficult to reproduce, while some disasters, such as fire, often lack authenticity and enough participants to drill. Therefore, we construct a virtual scene instead of a real one to study crowd evacuation in an emergency.

The mental state and behavior of evacuees in the real world are even more complex and difficult to express in the virtual environment, and the problem of the lack of reality and verisimilitude in the virtual scene begins to emerge. In order to break through this bottleneck, researchers have put forward many different solutions, among which, with the rise of machine learning, RL has become the main direction to solve this problem. The agent gets the returned information through interaction with the environment and makes decisions and actions based on the information. Such a model is closer to the human thinking process.

Crowd motion simulation is a key part of the evacuation study. The macroscopic model and microscopic model are used. The macroscopic model usually regards the crowd as a whole, such as fluid mechanics model [30], potential energy field model [31], etc. The microscopic model can consider the interaction between each individual and the environment from the perspective of individual. Typical methods include cellular automata model [32] and social force model. The social force model uses the mathematical model of crowd behavior to characterize the interaction between people. In simulated evacuation, it not only shows the interaction between individuals and the environment but also reflects the internal relationship between individuals. In this study, we use it as the simulation method to improve the efficiency of agent evacuation and conform to the actual situation.

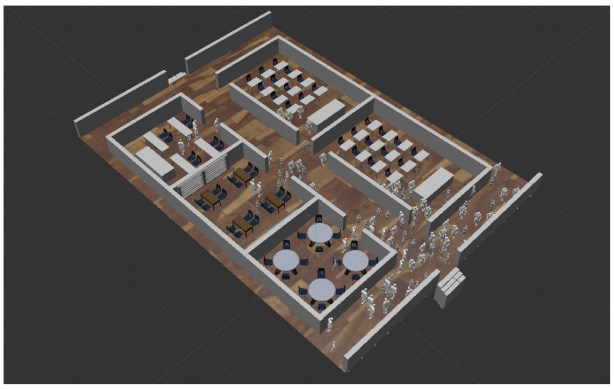


Fig. 3. View of the evacuation floor layout. There are five classrooms and two exits.

Building evacuation guidance involves three types of objects, namely, building scene, evacuees, and intelligent evacuation guidance system. The existing research usually defines each fleeing individual as an agent and studies the individual's action strategy and motion state. RL requires agents to constantly interact with the environment and summarize the best behavior decisions at each step through trial and error. It requires a very large scale of interaction, which accumulates over millions of time steps. In these studies, each individual is regarded as independent, which increases the computational cost of simulation. In this article, we treat the evacuation guide system as an agent. In a given environment, the location of an evacuee is determined by its speed. The collisions should be avoided in the evacuation process. In the training stage, each evacuee is regarded as an idealized person. After several interactions with the environment, they can follow the guidance of dynamic signs to promote evacuation.

The 3-D view of an example evacuation floor plan in this study is shown in Fig. 3. The virtual evacuation scene is created with the Cinema 4-D toolset, which is a professional 3-D modeling, animation, simulation, and rendering software solution. We have tried other building information model (BIM) tools, such as 3-D Max, and we found Cinema 4-D has the best performance. However, due to huge computation load, it is not feasible for a DRL algorithm to learn from a 3-D virtual scene efficiently. Therefore, we convert (project) a 3-D scene to a two-dimensional (2-D) view.

IV. DRL for Evacuation Guidance

A. Deep Reinforcement Learning

RL is one of the important components in the field of artificial intelligence [33], [34], [35], [36], [37]. RL consists of two parts: 1) agent and 2) environment. In the process of RL, agents interact with the environment. After an agent obtains a state of the environment, it bases on the state to output an action. Then, this action is executed in the environment, which will output the next state and the reward brought by the current action according to the action taken by the agent. The purpose of agents is to get as many rewards from the environment as possible. Compared with other machine learning methods, such as manifold learning [38] and deep learning [39], RL focuses more on long-term rewards. This makes it popular in many fields, including robot simulation [40], disassembly line balancing [26], [41], and intelligent games [42], [43].

The Markov decision process (MDP) is the basic framework of RL. It can be described as a five tuple (S, A, P, R, γ), where S is a finite set of all states, A is a finite set of all actions, and the state transition function $P(s_{t+1} = s'|s_t = s, a_t = a)$ means that the future state s' depends not only on the current state s but also on the actions a taken by the agent in the current state. The reward function $R(s_t = s, a_t = a) = E[r_t|s_t = s, a_t = a]$ indicates that the current state s and the action a determine the current reward. γ is the discount factor that is in the range of $\in [0, 1]$. The policy $\pi(a|s) = P(a_t = a|s_t = s)$ defines what actions should be taken in a certain state. It represents how to select an action from all possible actions. For example, there may be a probability of 0.7 to go left, and a probability of 0.3 to go right for an evacuee.

Some RL algorithms have been proposed to study the path-planning problem of crowd evacuation. Specifically, the appropriate reward function is designed to simulate the interaction between agents and the environment, so that agents can correctly find the target point and avoid collision with obstacles. This way of path finding is very similar to the human decision-making model. Typical tabular algorithms, such as Q-learning method [44], are widely used in the path planning of agent evacuation. However, in real scenarios, a large-scale evacuation drill has a large number of people and a complex environment. When the traditional RL algorithm based on Q-table is used to solve such problems, it is easy to cause large state space and dimension disaster by the huge Q-table created in high-dimensional scenarios.

In recent studies, researchers combined the deep neural network (DNN) with RL to come up with DRL for dealing with high-dimensional data [45]. Because of the representation learning ability of DNN [46], DNN has replaced Q-table, breaking the limit of the number of states. The DRL agent can directly take images as input, and get rid of the dilemma of too small Q-learning state space. It also has the strong perception ability of deep learning and the decision-making ability of RL. This enables it to perform well on many problems that cannot be solved by traditional methods.

DQN, as the first proposed framework of DRL, is an improved model based on Q-learning, which trains agents through experience playback. In 2015, DeepMind team from Google [47] proposed the DQN method that solved the problem of state dimension explosion and achieved some remarkable achievements by using DQN, such as training AlphaGo to beat the world champion in Go and introducing AlphaStar that reached the master rank in StarCraft 2.

The DQN we use is composed of the current Q network and the target Q network. Although the two Q networks have the same structure, their parameters are different and denoted by θ and θ^-, respectively. The current Q network is responsible for calculating the reward value of the agent action, and the target Q network is responsible for assisting in calculating the expected maximum reward value in the next step of the action. The current Q network parameters are synchronized with the target Q network at regular intervals. The strategy π of DQN

is a ϵ-greedy strategy. It has both exploration and utilization. The probability of random action is determined by the set ϵ.

DQN introduces an experience replay buffer to learn from memories rather than directly learn from the current feedback. Some minibatches are used randomly from it for training. For a sample e_t in a time step, the experience replay buffer stores state s_t, action a_t, reward r_t, and new state s_{t+1} after action. The loss function for each sample is calculated as

$$L_t = E\Big[(Q_t - Q(s_t, a_t; \theta))^2\Big] \tag{1}$$

where Q_t is the target Q value and calculated as

$$Q_t = r_t + \gamma \max_{a_{t+1}} Q\big(s_{t+1}, a_{t+1}; \theta^-\big). \tag{2}$$

The Q value is updated as follows:

$$Q^*(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\big[r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\big] \tag{3}$$

where r is the reward value obtained from the environment, which is also stored in the experience replay buffer, γ is the learning rate, θ and θ^- are the neural network parameters, s_{t+1} is the next state, and a_{t+1} is the action with the largest reward in all outputs after the next state is reached.

The training process of DQN is shown in Algorithm 1, in which the function *randominit* initializes the simulation environment and the function *agentpolicy* is for the agent to select actions by ϵ-*greedy* strategy. For each step, we have *state_new*, *reward*, and *end* which judge whether evacuation is finished. The function *caltderror* is to calculate TD error according to (1). The function *calpriority* calculates the priority of each sample and *randomsample* is used for sampling according to priority. After that, the TD error is calculated and the loss function is calculated according to function *calloss*, and the parameter θ is updated by the *backpropagation* function.

On the basis of DQN, researchers have proposed several advanced DRL algorithms, including Double DQN [48] that decouples the target value, Dueling DQN [49] that decouples the action value, and rainbow DQN [50] that integrates several improved DRL methods. Rainbow DQN has shown significant improvements in reward and convergence speed compared with other algorithms. Given its powerful capabilities, rainbow DQN has been widely used in panoramic video streaming [51], traffic congestion control [50], and autonomous car racing [52]. This article adopts a rainbow DQN algorithm, which mainly combines double DQN and dueling DQN, to improve the learning speed and stability of the model.

B. RL Model for Crowd Evacuation Guidance

In the context of RL for evacuation, the RL agent refers to the evacuation system, as shown in Fig. 4. Its environment is mainly transformed from a 3-D simulation environment into a 2-D plan [53], including the layout of the floor, people who are inside, and some inaccessible blocks affected by disasters. The cloud server is responsible for converting a 3-D scene to a 2-D plane, and the sensor network and panoramic cameras

Algorithm 1 Training Process of the Agent

```
Input: Environment env
Output: Neural network parameters θ*
Initialize current Q network parameters θ
Initialize target Q network parameters θ⁻
Initialize replay buffer pool
While steps < total_steps:
  state, reward, end ← env.randominit()
  While not end:
    action ← agentpolicy(state, θ)
    state_new, reward, end ← env.step(action)
    td_error ← caltderror(state, action, reward,
state_new, θ, θ⁻)
    priority ← calpriority(td_error)
    pool.append(state, action, reward,
state_new, priority)
    state ← state_new
    steps ← steps + 1
    s_t, a_t, r_t, s_{t+1} ← pool.randomsample(batch_size)
    td_error ← caltderror(s_t, a_t, r_t, s_{t+1}, θ, θ⁻)
    loss ← calloss(td_error)
    θ ← backpropagation(θ, loss)
    every certain steps: θ⁻ ← θ
  end
  If the performance improves: θ* ← θ
end
```

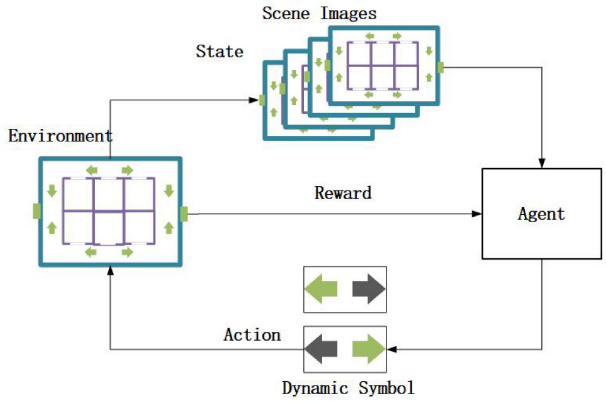


Fig. 4. RL model for evacuation guidance.

are responsible for collecting the movement track and distribution of people, Hierarchical processing is required in case of multiple layers.

The scene images contain the required information for the environmental states $s_t \in S$ in MDP. Under ideal conditions, people observe the dynamic signs and follow the guidance of the evacuation system. Therefore, the agent action $a_t \in A$ corresponds to the guidance sign signal. a_t is a discrete vector, usually composed of the directions of the sign. The state transition function $P_a(s_t, s_{t+1})$ is usually unknown due to the complex environment and crowd movement, which requires agents to learn and adapt in the process of interaction. The design of the reward function determines whether the agent finally achieves its goal, and directly affects the convergence

speed and final performance of the algorithm. In the crowd evacuation problem, the reward function is often related to the total evacuation time and the total number of people evacuated. The reward function $R_a(s_t, s_{t+1})$ is defined for each evacuee. The reward value is related to evacuation time. The agent's reward function is the sum of all evacuees' reward values. The agent's learning goal is to maximize this reward, so as to reduce the evacuation time of all people.

In the simulated environment, we assume that the total evacuation of the crowd requires m frames, which can be equally specified with a total time t. In each frame, evacuees positions are updated by interacting with the environment, including state, action, reward, and other information, until all people are evacuated or the preset number of frames is reached. At this time, the system stores these information in the experience replay buffer. The priority of these data is determined by temporal difference error. In each training, we use them from the experience pool according to the priority to improve the sample utilization.

However, there are also problems in the DQN algorithm. There is a max function in the loss function, which will result in maximization bias. It overestimates the Q value of the action. The error value increases with the number of actions. This will cause fatal problems in crowd evacuation. However, by using the calculation method of the target value in the double DQN algorithm, we only need to use the current network to determine the action a_t with maximum Q value and the target network to compute the target Q value by the action a_t. Instead of building a new network, we directly input the new state into the old neural network and return the maximum Q value, so the new loss function should be

$$L_t = E\Big[\big(r_t + \gamma Q_t\big(s_{t+1}, \max_{a_t}(Q(s_{t+1}, a_t)); \theta^-\big) - Q(s_t, a_t; \theta)\big)^2\Big]. \tag{4}$$

C. State Space, Action Space, and Reward Function

1) State Space: The state space is mainly used to store image information when evacuees interact with the environment. It provides data needed by the current Q network. Although the establishment of 3-D virtual scene can be basically realized under the technology of metaverse, image processing by a neural network will occupy a large amount of computer memory space and computing resources, so we need to convert the 3-D virtual scene into a 2-D plane, Only the last three frames are selected as the current state. The size of the input image after processing is 84 × 84 pixels, and the value of each pixel is 0–255. We set different pixels for the background, evacuees, the wall, the disaster impact area, and other information, which can significantly speed up the simulation process and reduce the requirement on hardware.

2) Action Space: The action space is a collection of actions taken by an agent according to its own state. We take the evacuation system as an agent. The actions it can take are modify the direction of dynamic signs. The action space stores discrete vectors composed of all dynamic sign directions. By reducing the action interval, discrete actions can approximate continuous actions.

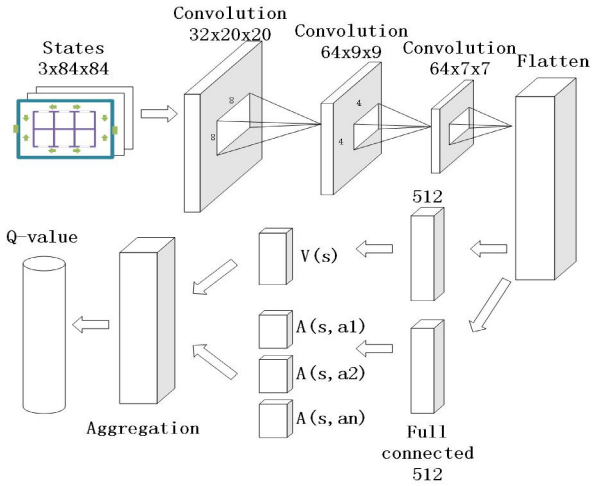


Fig. 5. Structure of the proposed DQN for evacuation. The input S_t is first processed by three convolutional layers. At the fully connected layer, the result is flattened and then processed by two fully connected layers to separately estimate the state value $V(s)$ and the advantage function $A(s, a)$ for each action. The two parts are eventually aggregated to output the Q value.

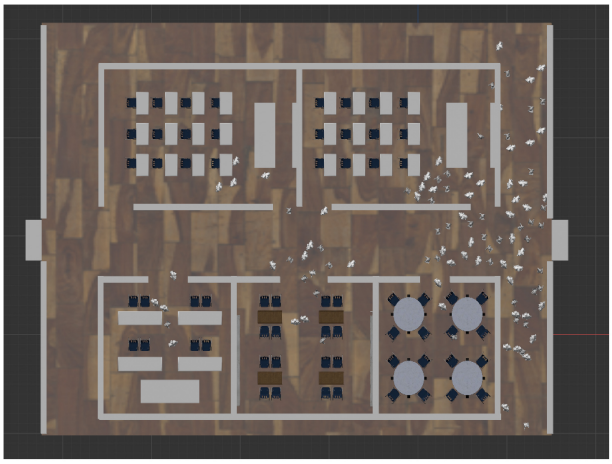


Fig. 6. Bird's-eye view of the evacuation floor plan.

3) Reward Function: The reward function is the basis for judging whether the system strategy is good or not. As a crowd evacuation system in the teaching building, it aims to enable people in the building to evacuate in a short time. In order to make the reward function more reasonably, it is determined by the total reward of all evacuees in the scene. We consider four states for each evacuee: reaching the obstacle position, reaching the exit position, crowding, and passing smoothly. These four states are divided into four reward functions. The total reward for each evacuee R_{total} is the sum of all four rewards

$$R_{\text{total}} = R_{\text{goal}} + R_{\text{collision}} + R_{\text{crowd}} + R_{\text{smooth}} \tag{5}$$

where R_{goal} depends on the distance between the evacuee and the exit, $R_{\text{collision}}$ depends on the distance between the evacuee and the obstacle, and R_{crowd} and R_{smooth} are mainly determined by the comparison between the current movement speed and the previous fastest speed. Considering these factors, the

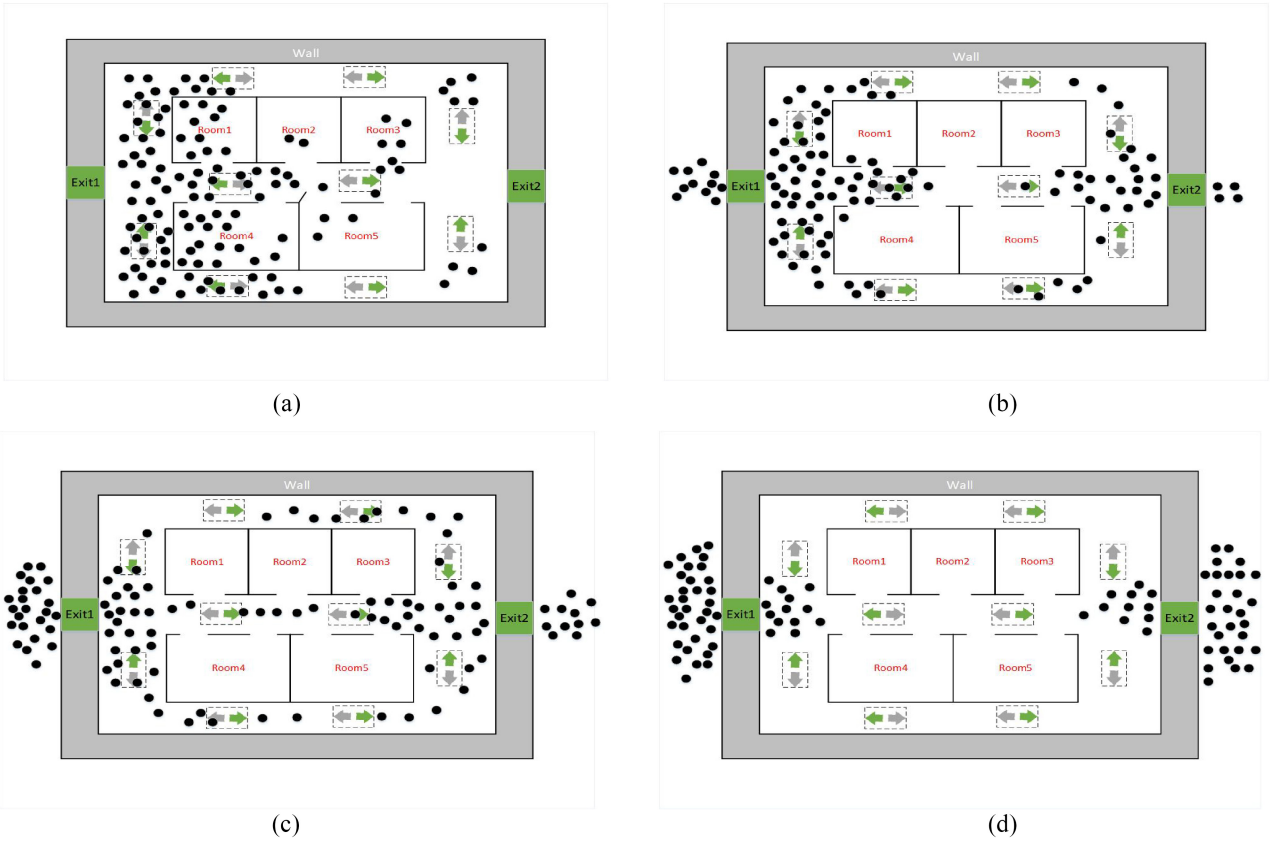


Fig. 7. Evacuation process in the simple scenario. (a) Initial crowd distribution. (b) Crowd evacuation under guidance system in 40 s. (c) Crowd evacuation under guidance system in 80 s. (d) Crowd evacuation under guidance system in 120 s.

agent can learn the path allocation strategy that maximizes the evacuation efficiency.

D. Network Architecture

As mentioned earlier, we only intercept the last three frames. The image resolution is 84 × 84 pixels. In DNN, the input size is 3 × 84 × 84, while our action space is a discrete vector composed of left and right. As shown in Fig. 5, we used three convolutional layers to input S_t. At the fully connected layer, the result from the previous convolutional layer is flattened and then connected to the most basic neural network. Dueling DQN changes one fully connected layer of the original DQN into two fully connected layers to separately estimate the state value $V(s)$ and the advantage function $A(s, a)$ for each action. The two parts are eventually aggregated to output the Q value.

V. Experimental Design and Result

We set the simulation scene as an area of 20.4 m × 20.1 m in a teaching building of a university. There are two exits, one on the left side and the other one on the right side. There are five classrooms and three corridors with 120 people. Metaverse technologies are used to draw virtual scenes. Fig. 6 shows the bird's-eye view of the overall scene. The cloud server compresses the 3-D scene into a 2-D plane.

Table I provides some basic parameters used in the experiment. The learning rate is a parameter used to adjust the network weight. The lower the learning rate, the more important the old information is, and it takes longer time to converge. Under the condition that other super parameters are the same, we tested the learning rate $a = 0.002$, 0.001, 0.0005, and 0.0001. Of which 0.0001 has the best convergence effect. Therefore, the learning rate was set to 0.0001.

TABLE I
Parameter in DQN Algorithm

Parameters	Value	Parameter	Value
Learning rate	0.0001	Buffer size	100000
Batch size	128	Target Q Update Frequency	1000
Discount factor	0.99	Image size	84x84

The discount factor indicates the degree of foresight. The smaller the value, the more important the present reward is than the future reward. We set it to 0.99. The buffer size determines the number of sample data to be stored. We set the size to 100 000. It extracts samples by priority to ensure the uniform distribution of training data samples. In addition, we set the batch size to 128. The target Q update adds stability to the algorithm. The slower the update, the slower the algorithm converges. Through experiments, we set the target Q update frequency to 1000 iterations. The input image shall ensure the image clarity and provide the basic information of the environment. However, if the input image size is too large, the training efficiency will be reduced. Therefore, based on the environment in this article, the selected input image size is 84 × 84 pixels.

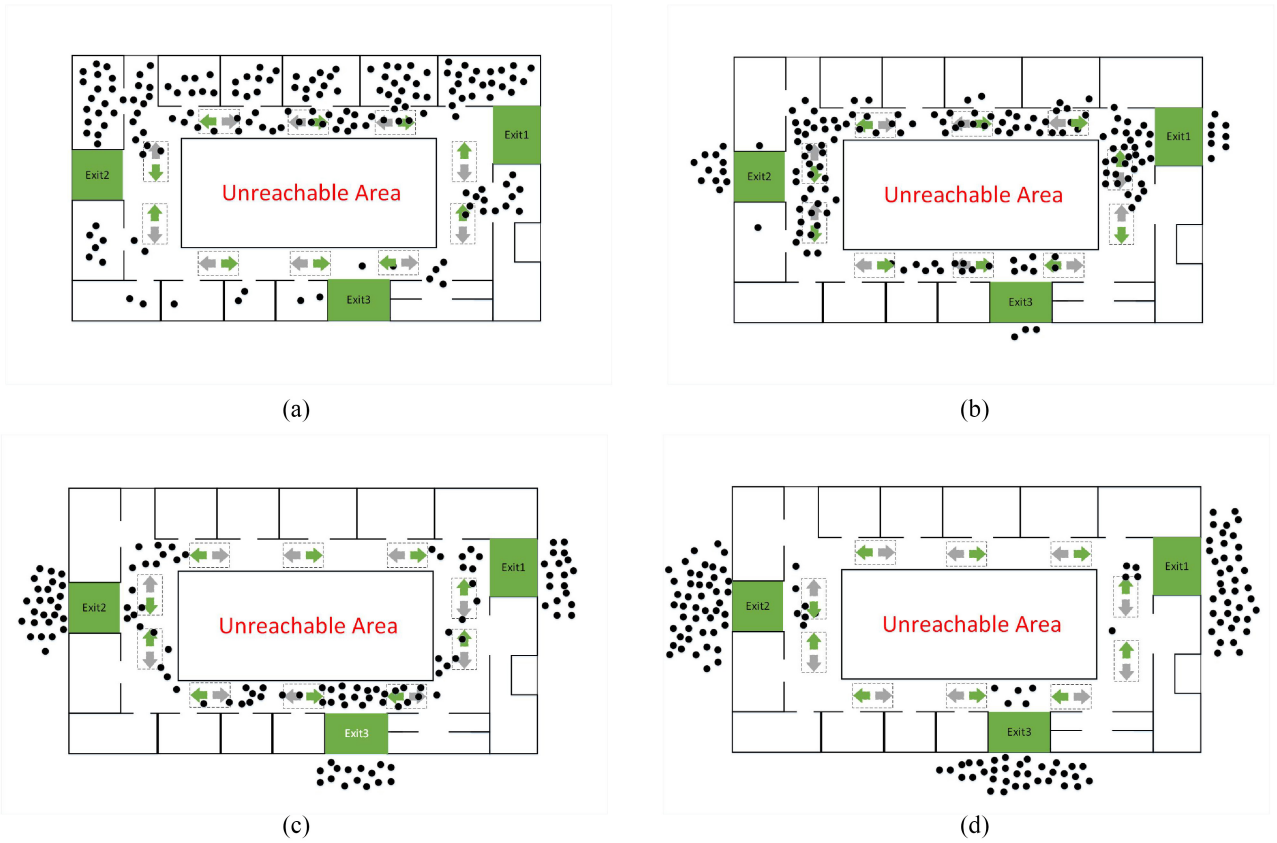


Fig. 8. Evacuation process in the second scenario. (a) Initial crowd distribution. (b) Crowd evacuation under guidance system in 60 s. (c) Crowd evacuation under guidance system in 120 s. (d) Crowd evacuation under guidance system in 200 s.

Fig. 7 illustrates an instance of the evacuation process. The green rectangles represent the exits, the green arrows represent the dynamic guidance signs, and the gray color of blocks represent the wall.

Fig. 7(a) shows the initial distribution of the crowd. The crowd is mainly distributed in or near the two rooms on the left. If there is no dynamic guidance, the crowd will evacuate according to the static sign with the shortest distance to the left exit, which will lead to excessive congestion of people on the left exit and insufficient use of the right exit.

When the simulation is started, the agent first guides people on both sides to the nearest exits according to the distance between the crowd and the nearest exit. Then, when the agent senses the crowd distribution and finds that the evacuation efficiency of the left exit is much lower than that of the right exit, it begins to guide the crowd in the three corridors to the right exit, reducing the congestion of the left exit and improving the overall efficiency. Fig. 7(b) shows the scene 40 s after the simulation starts.

After the dynamic guidance, the congestion in the area near the left exit has been relieved as the pressure of the following crowd has dissipated. But the evacuation efficiency of the right exit is still low. The direction of the signs on the three corridors remains unchanged. At this time, more people have been evacuated from the left exit than from the right. Fig. 7(c) shows the scene 80 s into the simulation.

Finally, Fig. 7(d) shows the scene that the simulation has been running for 120 s. The crowd has basically completed the evacuation from the exits on both sides at the same time. Almost the same number of people have successfully evacuated from the two exits, indicating that the evacuation efficiency has been maximized under the help of the DRL agent.

To ensure that the proposed dynamic guidance system works for different evacuation scenarios, we test it with a more complex floor plan, which is depicted in Fig. 8(a). There are 15 rooms and three exits in the floor. The initial distribution of the crowd is mainly concentrated at one side of the scene. After the simulation starts, the agent senses the crowd distribution and directs the crowd in the upper part to Exits 1 and 2. The evacuees located in the lower section are guided to the lower right exit (Exit 3).

Fig. 8(b) shows the situation 60 s after the simulation is started. Most of the crowd are gathered at the left and right exits, the agent senses that the efficiency of the two exits is beginning to decline. From Fig. 8(b) and (c), by changing the signs dynamically, part of the crowd effectively moves to Exit 3 to avoid further congestion around the two upper exits.

Fig. 8(c) shows the scene of for 120 s after the simulation is started. At this time, the number of people at Exit 3 has reached full load, while there are relatively few people at Exit 1 and Exit 2. Dynamic signs are changed to guide the

TABLE II
EVACUATION TIME UNDER DIFFERENT METHODS (SECONDS)

Methods	Evacuation time in the first scenario	Evacuation time in the second scenario
Rainbow DQN	122.54	231.76
Static guidance sign	155.18	278.38
Shortest-path algorithm	128.83	242.87

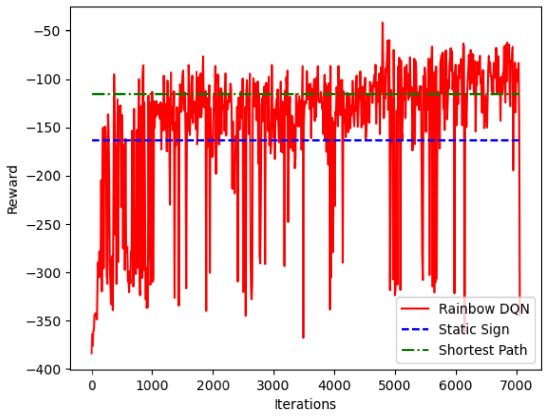


Fig. 9. Training curve of the DRL agent in the first scenario.

rest of the people in the corridor near Exit 3 to the left and right exits.

Fig. 8(d) shows when the evacuation is nearly complete. All the three exits have almost the same few number of evacuees moving toward them, a strong indication that the agent has achieved the maximum efficiency of crowd evacuation over the three exits.

In a static guidance sign method, the shortest path is calculated either automatically or manually. Each sign points to the nearest exit, and the evacuation path is calculated only once regardless of the dynamic change of the crowd distribution, and the signs do not change during the simulation process. We set the number of people in a simple scenario to 80, and in a complicated scenario to 160. The training curve of the first scenario is shown in Fig. 9, in which the DRL agent approaches the optimal strategy after about 2000 iterations, and the reward of the rainbow DQN is superior to that of static signs and shortest path algorithm. The average evacuation time is determined by the average number of pixel images which playing back. As shown in Table II, the two scenarios above are used to simulate different evacuation methods. Under the same parameters, the average evacuation time of rainbow DQN is better than that of static guidance signs and dynamic shortest path algorithm. It shows that the rainbow DQN algorithm can actually reduce the evacuation time.

VI. CONCLUSION

This article addresses the problem of teaching building evacuation guidance by using dynamic navigation signs. In view of the shortcomings of existing methods that require extensive exercise data and manual design work, a dynamic evacuation system based on metaverse is proposed with a rainbow DQN as the simulation engine. Through the DRL algorithm, the agent learns the path-planning method during the training process and accumulates experience through the interaction with the environment. The optimal evacuation guidance strategy is obtained in the learning process eventually.

In our future work, we will apply the metaverse-based evacuation approach to other scenarios, such as buildings with more complex floor layouts, or some people on wheelchairs among the evacuees, or there are falls in the evacuation process. All these add extra challenges to the evacuation.

REFERENCES

[1] S.-M. Park and Y.-G. Kim, "A metaverse: Taxonomy, components, applications, and open challenges," *IEEE Access*, vol. 10, pp. 4209–4251, 2022.

[2] J. Han, J. Yun, J. Jang, and K.-R. Park, "User-friendly home automation based on 3D virtual world," *IEEE Trans. Consum. Electron.*, vol. 56, no. 3, pp. 1843–1847, Aug. 2010.

[3] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Comput. Graph. Appl.*, vol. 21, no. 6, pp. 34–47, Nov./Dec. 2001.

[4] A. Hendaoui, M. Limayem, and C. W. Thompson, "3D social virtual worlds: Research issues and challenges," *IEEE Internet Comput.*, vol. 12, no. 1, pp. 88–92, Jan./Feb. 2008.

[5] F.-Y. Wang, "Parallel intelligence in metaverses: Welcome to Hanoi!" *IEEE Intell. Syst.*, vol. 37, no. 1, pp. 16–20, Jan./Feb. 2022.

[6] Q. Xiao, C. Li, Y. Tang, and L. Li, "Meta-reinforcement learning of machining parameters for energy-efficient process control of flexible turning operations," *IEEE Trans. Autom. Sci. Eng.*, vol. 18, no. 1, pp. 5–18, Jan. 2021.

[7] K. Getchell, I. Oliver, A. Miller, and C. Allison, "Metaverses as a platform for game based learning," in *Proc. 24th IEEE Int. Conf. Adv. Inf. Netw. Appl.*, 2010, pp. 1195–1202.

[8] Y. Wang et al., "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Commun. Surveys Tuts.*, early access, Sep. 7, 2022, doi: 10.1109/COMST.2022.3202047.

[9] F.-Y. Wang, N.-N. Zheng, D. Cao, C. M. Martinez, L. Li, and T. Liu, "Parallel driving in CPSS: A unified approach for transport automation and vehicle intelligence," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 4, pp. 577–587, Oct. 2017.

[10] X. Wang, J. Yang, J. Han, W. Wang, and F.-Y. Wang, "Metaverses and DeMetaverses: From digital twins in CPS to parallel intelligence in CPSS," *IEEE Intell. Syst.*, vol. 37, no. 4, pp. 97–102, Jul./Aug. 2022.

[11] Z. Wang, C. Zhang, Z. Zheng, and L. Li, "Research on path planning algorithm for crowd evacuation," *Symmetry*, vol. 13, no. 8, p. 1339, 2021.

[12] D. Zhang, G. Huang, C. Ji, H. Liu, and Y. Tang, "Pedestrian evacuation modeling and simulation in multi-exit scenarios," *Physica A, Stat. Mech. Appl.*, vol. 582, Nov. 2021, Art. no. 126272.

[13] J. Wang, "Patient flow modeling and optimal staffing for emergency departments: A Petri net approach," *IEEE Trans. Comput. Social Syst.*, early access, Jul. 4, 2022, doi: 10.1109/TCSS.2022.3186249.

[14] L.-W. Chen and J.-X. Liu, "Time-efficient indoor navigation and evacuation with fastest path planning based on Internet of Things technologies," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 51, no. 5, pp. 3125–3135, May 2021.

[15] W. Shi et al., "Variable message sign and dynamic regional traffic guidance," *IEEE Intell. Transp. Syst. Mag.*, vol. 1, no. 3, pp. 15–21, Nov. 2009.

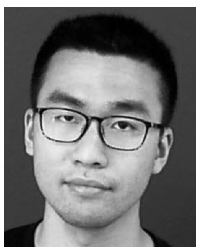
[16] E. Boukas, I. Kostavelis, A. Gasteratos, and G. C. Sirakoulis, "Robot guided crowd evacuation," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 2, pp. 739–751, Apr. 2015.

[17] L. Huang, J. Gong, and W. Li, "A perception model for optimizing and evaluating evacuation guidance systems," *ISPRS Int. J. Geo-Inf.*, vol. 10, no. 2, p. 54, 2021.

[18] H. Jiang, "Mobile fire evacuation system for large public buildings based on artificial intelligence and IoT," *IEEE Access*, vol. 7, pp. 64101–64109, 2019.

[19] R. Mehran, A. Oyama, and M. Shah, "Abnormal crowd behavior detection using social force model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 935–942.

[20] C. Huang, H. Zhang, L. Wang, X. Luo, and Y. Song, "Mixed deep reinforcement learning considering discrete-continuous hybrid action space for smart home energy management," *J. Modern Power Syst. Clean Energy*, vol. 10, no. 3, pp. 743–754, 2022.
[21] J. Li, Y. Hu, and J. Li, "Rapid risk assessment of emergency evacuation based on deep learning," *IEEE Trans. Comput. Social Syst.*, vol. 9, no. 3, pp. 940–947, Jun. 2022.
[22] L. Jin, J. Li, Z. Sun, J. Lu, and F.-Y. Wang, "Neural dynamics for computing perturbed nonlinear equations applied to ACP-based lower limb motion intention recognition," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 8, pp. 5105–5113, Aug. 2022.
[23] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of Things for smart cities," *IEEE Internet Things J.*, vol. 1, no. 1, pp. 22–32, Feb. 2014.
[24] Z. Zhang, J. Liu, G. Liu, J. Wang, and J. Zhang, "Robustness verification of swish neural networks embedded in autonomous driving systems," *IEEE Trans. Comput. Social Syst.*, early access, Jun. 9, 2022, doi: 10.1109/TCSS.2022.3179659.
[25] J. Revell, D. Welch, and J. Hereford, "Sim2Real: Issues in transferring autonomous driving model from simulation to real world," in *Proc. SoutheastCon*, 2022, pp. 296–301.
[26] F.-Y. Wang, Y. Tang, X. Liu, and Y. Yuan, "Social education: Opportunities and challenges in cyber-physical-social space," *IEEE Trans. Comput. Social Syst.*, vol. 7, no. 2, pp. 191–196, Apr. 2019.
[27] S. Zheng and H. Liu, "Improved multi-agent deep deterministic policy gradient for path planning-based crowd simulation," *IEEE Access*, vol. 7, pp. 147755–147770, 2019.
[28] Z. Zhou, P. Zhu, Z. Zeng, J. Xiao, H. Lu, and Z. Zhou, "Robot navigation in a crowd by integrating deep reinforcement learning and online planning," *Appl. Intell.*, vol. 52, no. 13, pp. 15600–15616, Oct. 2022.
[29] M. Zhou, H. Dong, P. A. Ioannou, Y. Zhao, and F.-Y. Wang, "Guided crowd evacuation: Approaches and challenges," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 5, pp. 1081–1094, Sep. 2019.
[30] S. Mukherjee, D. Goswami, and S. Chatterjee, "A Lagrangian approach to modeling and analysis of a crowd dynamics," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 45, no. 6, pp. 865–876, Jun. 2015.
[31] M. Zhou, H. Dong, Y. Zhao, P. A. Ioannou, and F.-Y. Wang, "Optimization of crowd evacuation with leaders in urban rail transit stations," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 12, pp. 4476–4487, Dec. 2019.
[32] L. D. O. Carneiro, J. B. Cavalcante-Neto, C. A. Vidal, and T. B. Dutra, "Crowd evacuation using cellular automata: Simulation in a soccer stadium," in *Proc. Symp. Virtual Augmented Reality*, 2013, pp. 240–243.
[33] T.-Y. Mu, A. Al-Fuqaha, and K. Salah, "Automating the configuration of MapReduce: A reinforcement learning scheme," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 11, pp. 4183–4196, Nov. 2020.
[34] L. Huo, Z. Wang, M. Xu, and Y. Song, "A task-agnostic regularizer for diverse subpolicy discovery in hierarchical reinforcement learning," *IEEE Trans. Syst., Man, Cybern., Syst.*, early access, Sep. 30, 2022, doi: 10.1109/TSMC.2022.3209070.
[35] X. You, X. Li, Y. Xu, H. Feng, and J. Zhao, "Toward packet routing with fully-distributed multi-agent deep reinforcement learning," in *Proc. Int. Symp. Model. Optim. Mobile, Ad Hoc, Wireless Netw. (WiOPT)*, 2019, pp. 1–8.
[36] X. Xu, L. Zuo, X. Li, L. Qian, J. Ren, and Z. Sun, "A reinforcement learning approach to autonomous decision making of intelligent vehicles on highways," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 10, pp. 3884–3897, Oct. 2020.
[37] J. Liang, Y. Tang, R. Hare, B. Wu, and F. Y. Wang, "A learning-embedded attributed petri net to optimize student learning in a serious game," *IEEE Trans. Comput. Social Syst.*, early access, Dec. 23, 2021, doi: 10.1109/TCSS.2021.3132355.
[38] H. Han, W. Li, J. Wang, G. Qin, and X. Qin, "Enhance explainability of manifold learning," *Neurocomputing*, vol. 500, pp. 877–895, Aug. 2022.
[39] B. Hu and J. Wang, "Deep learning based hand gesture recognition and UAV flight controls," *Int. J. Autom. Comput.*, vol. 17, no. 1, pp. 17–29, 2020.
[40] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14413–14423, Dec. 2020.
[41] X. Guo, T. Wei, J. Wang, S. Liu, S. Qin, and L. Qi, "Multiobjective U-shaped disassembly line balancing problem considering human fatigue index and an efficient solution," *IEEE Trans. Comput. Social Syst.*, early access, Nov. 8, 2022, doi: 10.1109/TCSS.2022.3217101.
[42] J. Li, H. Modares, T. Chai, F. L. Lewis, and L. Xie, "Off-policy reinforcement learning for synchronization in multiagent graphical games," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2434–2445, Oct. 2017.
[43] F. Christopher, Y. Tang, A. Johnson, and T. Bielefeldt, "Balancing fun and learning in a serious game design," *Int. J. Game Based Learn.*, vol. 4, no. 4, pp. 37–57, 2014.
[44] A. Kumar, A. Zhou, G. Tucker, and S. Levine, "Conservative *Q*-learning for offline reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1179–1191.
[45] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.
[46] W. Zhang, J. Wang, and F. Lan, "Dynamic hand gesture recognition based on short-term sampling neural networks," *IEEE/CAA J. Automatica Sinica*, vol. 8, no. 1, pp. 110–120, Jan. 2021.
[47] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.
[48] J. Pan, X. Wang, Y. Cheng, and Q. Yu, "Multisource transfer double DQN based on actor learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2227–2238, Jun. 2018.
[49] B.-A. Han and J.-J. Yang, "Research on adaptive job shop scheduling problems based on dueling double DQN," *IEEE Access*, vol. 8, pp. 186474–186495, 2020.
[50] M. Nawar, A. Fares, and A. Al-Sammak, "Rainbow deep reinforcement learning agent for improved solution of the traffic congestion," in *Proc. 7th Int. Japan–Africa Conf. Electron., Commun., Comput. (JAC-ECC)*, 2019, pp. 80–83.
[51] G. Xiao, M. Wu, Q. Shi, Z. Zhou, and X. Chen, "DeepVR: Deep reinforcement learning for predictive panoramic video streaming," *IEEE Trans. Cogn. Commun. Netw.*, vol. 5, no. 4, pp. 1167–1177, Dec. 2019.
[52] K. Güçkıran and B. Bolat, "Autonomous car racing in simulation environment using deep reinforcement learning," in *Proc. Innovations Intell. Syst. Appl. Conf. (ASYU)*, 2019, pp. 1–6.
[53] Z. Lu et al., "Multiobjective evolutionary design of deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 25, no. 2, pp. 277–291, Apr. 2021.



Jinlei Gu received the B.S. degree in computer science from the Changshu Institute of Technology, Suzhou, China, in 2021. He is currently pursuing the M.S. degree in computer science with Monmouth University, West Long Branch, NJ, USA.

His research interests are reinforcement learning algorithms and emergency evacuation.

Jiacun Wang (Senior Member, IEEE) received the Ph.D. degree in computer engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1991.

He is currently a Professor of Software Engineering with Monmouth University, West Long Branch, NJ, USA. From 2001 to 2004, he was a member of scientific staff with Nortel Networks, Richardson, TX, USA. Prior to joining Nortel Networks, he was a Research Associate with the School of Computer Science, Florida International University, Miami, FL, USA. He authored *Timed Petri Nets: Theory and Application* (Kluwer, 1998), *Real-Time Embedded Systems* (Wiley, 2018), and *Formal Methods in Computer Science* (CRC, 2019), edited *Handbook of Finite State Based Models and Applications* (CRC, 2012), and published over 160 research papers in journals and conferences. His research interests include software engineering, discrete-event systems, formal methods, machine learning, and real-time distributed systems.

Dr. Wang was an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS—PART C: APPLICATIONS AND REVIEWS, and is currently an Associate Editor of IEEE/CAA JOURNAL OF AUTOMATICA SINICA. He has served as the general chair, program chair, and special sessions chair or program committee member for many international conferences.

Xiwang Guo (Senior Member, IEEE) received the B.S. degree in computer science and technology from the Shenyang Institute of Engineering, Shenyang, China, in 2006, the M.S. degree in aeronautics and astronautics manufacturing engineering from Shenyang Aerospace University, Shenyang, in 2009, and the Ph.D. degree in system engineering from Northeastern University, Shenyang, in 2015.

He is currently an Associate Professor with the College of Computer and Communication Engineering, Liaoning Shihua University, Fushun, China. From 2016 to 2018, he was a Visiting Scholar with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA. He has authored over 60 technical papers in journals and conference proceedings, including IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON SYSTEM, MAN AND CYBERNETICS: SYSTEMS, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, and IEEE/CAA JOURNAL OF AUTOMATICA SINICA. His current research interests include Petri nets, remanufacturing, recycling and reuse of automotive, and intelligent optimization algorithm.

Guanjun Liu (Senior Member, IEEE) received the Ph.D. degree in computer software and theory from Tongji University, Shanghai, China, in 2011.

He was a Postdoctoral Research Fellow with the Singapore University of Technology and Design, Singapore, from 2011 to 2013, and a Postdoctoral Research Fellow with the Humboldt University of Berlin, Berlin, Germany, from 2013 to 2014, supported by the Alexander von Humboldt Foundation. He is currently a Professor with the Department of Computer Science, Tongji University. He has authored over 130 papers and three books. His research interests include Petri net theory, model checking, machine learning, cyber–physical systems, workflow, and credit card fraud detection.

Shujin Qin (Member, IEEE) received the B.S. degree in information and computing science from Tianjin Polytechnic University, Tianjin, China, in 2008, the M.S. degree in operational research and cybernetics from the University of Science and Technology Liaoning, Anshan, China, in 2011, and the Ph.D. degree in system engineering from Northeastern University, Shenyang, China, in 2019.

He joined Shangqiu Normal University, Shangqiu, China, in 2019, where he is currently a Lecturer of Logistics Management. His current research interests include large-scaled integer programming, cutting stock problem, vehicle routing problem, traveling repairman problem, and intelligent optimization algorithm.

Zhiliang Bi (Graduate Student Member, IEEE) received the B.S. degree in computer science from Qilu Normal University, Jinan, China, in 2021. He is currently pursuing the M.S. degree in computer science with the College of Computer and Communication Engineering, Liaoning Petrochemical University, Fushun, China.

His research interests are in machine learning and artificial intelligence.