A Deep Reinforcement Learning Based Intelligent Decision Method for UCAV Air Combat

Pin Liu(✉) and Yaofei Ma

School of Automation Science and Electrical Engineering,
Beihang University, Beijing, China
{liu0pin,mayaofeibuaa}@163.com

Abstract. Based on deep reinforcement learning, an intelligent tactical decision method is proposed to solve the problem of Unmanned Combat Aerial Vehicle (UCAV) air combat decision-making. The increasing complexity of the air combat environment leads to a curse of dimensionality when using reinforcement learning to solve the air combat problem. In this paper, we employed the deep neural network as the function approximator, and combined it with Q-learning to achieve the accurate fitting of action-value function, which is a good way to reduce the curse of dimensionality brought by traditional reinforcement learning. In order to verify the validity of the algorithm, simulation of our deep Q-learning network (DQN) is carried out on the air combat platform. The simulation results show that the DQN algorithm has a good performance in both the reward and action-value utility. The proposed algorithm provides a new idea for the research of UCAV intelligent decision.

Keywords: UCAV · Intelligent decision · Deep reinforcement learning · Combat simulation

1 Introduction

For the purpose of ensuring pilots' zero casualty in the future air combat, Unmanned Aerial Vehicle (UAV) will be gradually transformed into the protagonist of directly implementing the air combat mission from battlefield's supporting role, and Unmanned Combat Aerial Vehicle (UCAV) may even become the main force of air combat replacing manned aircraft [1]. At present, the man-in-the-loop ground station control system cannot meet the needs of increasingly complex and changeable environment of modern air combat [2], so it is necessary to study an intelligent decision method for UCAV air combat, which can evaluate the air combat situation and generate the corresponding maneuver command to perform combat missions autonomously. Expert system, one of the traditional methods in intelligent air combat maneuvering decision, can only be used to solve the known problems, but still need people involved when encountering the unknown problems, which is not fully autonomous decision-making [3]. In addition, the influence diagram [4], differential game [5], genetic algorithm [6], artificial neural network [7], which are widely used in the study of air combat maneuvering decision, are mostly applied to fighter's assistant decision or the UAV decision of ground station control mode. It remains to be verified whether these methods meet the requirements of UCAV online decision-making.

Recently, the successful application of machine learning techniques in many fields, such as games [8, 9], intelligent driving [10, 11], robots [12, 13], etc., has provided a new idea for the research of intelligent air combat decision-making. It is a powerful way to apply reinforcement learning (RL) in air combat domain since it can interact with the environment through trial and error, and obtain the optimal strategy through iteration [14].

However, traditional RL approach is not suitable for solving the large-scale Markov decision processes (MDP) like air combat due to computational overload caused by the curse of dimensionality. The approximate learning method, combining approximate technology with reinforcement learning, can alleviate the problem caused by curse of dimensionality to a certain extent, in which the key operation is to approximate the value function or the state space through a function approximator, to avoid accurate solution [15, 16]. The performance of the approximate learning method is determined by the ability of function approximator, including linear fitting function [17], nonlinear fitting function [18], classifier [19], neural network [20], etc., but these methods may not accurate enough to identify the complex state space. In this paper, a multi-layer stacking deep neural network (DNN) is employed as the function approximator to represent the complex state space accurately, due to its robust and accurate capacity. A number of typical researches using the combination of deep learning and reinforcement learning in Atari game and AlphaGo [8, 9], provide a good thinking for applying this approach to air combat missions.

In addition, for the purpose of maximizing the cumulative reward, the tradeoff of exploration and exploitation is the content we need to study. On the one hand, it is necessary to choose the action of highest reward using the learned experiences, so that the system moves to a better state. On the other hand, what we need is to fully grasp the environmental information and avoid falling into local optimum. Only by fully exploring the environment and using what we have learned can we maximize the cumulative reward. Consequently, we improved the algorithm by combining the deep learning with reinforcement learning and verify the feasibility of the algorithm by the battle in the air combat simulation platform. It lays the foundation for the further realization of UCAV autonomous air combat capability.

In this paper, the 1 versus 1 UCAV air combat scene is modeled in Sect. 2, including aircraft dynamics equation, states, actions, features and rewards. In Sect. 3, the reinforcement learning method is briefly reviewed, and the deep Q-learning network is built using some measures, such as experience replay and $\varepsilon - greedy$ policy. In Sect. 4, the experiment results validate the effectiveness of the proposed algorithm.

2 UCAV Air Combat Model

UCAV air combat platform involves two opponent planes (the roles of red and blue; red is the side that applies the approach proposed in this paper). The aircraft dynamics equation is shown as follows.

$$\begin{aligned}\dot{\psi} &= \frac{g}{v_{xy}} \tan(\varnothing) \\ \dot{x} &= v \cdot \cos(\psi) \\ \dot{y} &= v \cdot \sin(\psi) \\ \dot{h} &= v_z,\end{aligned} \tag{1}$$

where v is the velocity of the aircraft, and $v = \sqrt{v_z^2 + v_{xy}^2}$, in which v_z is in the vertical direction and v_{xy} is in the horizontal direction. $\psi \in [-\pi, \pi]$ is the yaw angle at which the flight of the aircraft deviates from north (along the y axis). $\varnothing \in [-\pi, \pi]$ is the roll angle of clockwise direction, whose value is changed to add or subtract change rate following the left or right scrolling of the aircraft. g is the gravitational acceleration.

Firstly, we can get each state in the air combat model, described with 12-dimension vectors.

$$\boldsymbol{s} = \{\boldsymbol{x_r}, \boldsymbol{y_r}, \boldsymbol{h_r}, \varnothing_{\boldsymbol{r}}, \boldsymbol{\psi_r}, \boldsymbol{v_{zr}}, \boldsymbol{x_b}, \boldsymbol{y_b}, \boldsymbol{h_b}, \varnothing_{\boldsymbol{b}}, \boldsymbol{\psi_b}, \boldsymbol{v_{zb}}\}, \tag{2}$$

where subscript r and b represent red and blue respectively. And state moves from s to s' after executing action of red and blue (u_r, u_b) with Eq. (1).

$$\boldsymbol{s} \underset{\boldsymbol{u_r, u_b}}{\longrightarrow} \boldsymbol{s}'. \tag{3}$$

The actions taken in the simulation model consist of five single actions. The move list is shown as follows.

$$movelist = [turn_{left_{up}}, turn_{right_{up}}, turn_{left_{down}}, turn_{right_{down}}, maintain]. \tag{4}$$

Secondly, the features were extracted by preprocessing the state data for the purpose of making the convergence of approximate function faster and closer to the real utility. The chosen features in this paper are related to pilot's experience of determining the air combat situation, shown in Table 1.

$$\boldsymbol{state} \overset{\varnothing(\mathrm{s})}{\rightarrow} \boldsymbol{feature}, \tag{5}$$

where aspect angle (AA) is the angle between the connecting line from target to attacker and the tail direction in body longitudinal axis of the target plane. Antenna train angle (ATA) is the angle between the direction of nose in body longitudinal axis of attacking plane and its radar's line of sight (LOS). Relative range (R) is the relative distance [21]. Their geometric description is shown in Fig. 1.

Table 1. Feature vectors

Features	Description
AA^+	Symbol of AA
$\lvert AA \rvert$	Absolute value of AA
ATA^+	Symbol of ATA
$\lvert ATA \rvert$	Absolute value of ATA
R	The relative distance between the two planes
$\dot{A}A$	The changing ratio of AA
$A\dot{T}A$	The changing ratio of ATA
ϕ_r	Red roll angle
ϕ_b	Blue roll angle

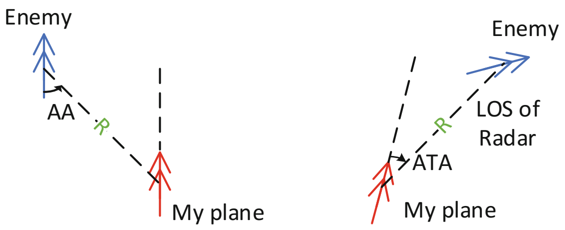

Fig. 1. The description of *AA* and *ATA* (Color figure online)

The reward function is also defined according to features.

$$R' = \left[\frac{\left(1 - \frac{|AA|}{\pi}\right) + \left(1 - \frac{|ATA|}{\pi}\right)}{2} \right] e^{-\left(\frac{|R| - R_d}{k\pi}\right)}, \tag{6}$$

where R_d is the best shooting distance of airborne weapon. The constant k is used to adjust the weight of the distance factor in the reward function (unit is m/rad).

3 Method

3.1 Reinforcement Learning

Reinforcement learning (RL) is one of machine learning methods that agent learns a mapping from environmental state to behavior by interacting with the environment, whose goal is to maximize the cumulative reward. The schematic diagram of the basic principle is shown in Fig. 2.

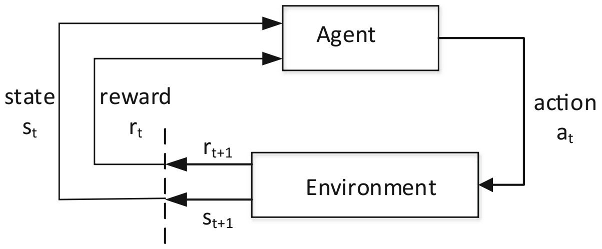

Fig. 2. Reinforcement learning framework

RL's mathematic model is Markov decision process (MDP), and air combat can be described with five-tuples $\{S, A, P, R, V\}$ in MDP. S is the set of states, $S = \{s\}$. A is all optional actions in aircraft flight, $A = \{a\}$. $P\left(s; a; s'\right) \in [0, 1]$ is the state transition probability from s to s' by performing action a. $R(s)$ is the reward of the current state s. $V(s)$ is the utility function of the state s [22]. The goal of agent is to obtain optimal policy to maximize the total expected reward:

$$V(s) = max_{\pi} \mathrm{E}\left[\mathrm{R}(s_0, a_0) + \gamma \mathrm{R}(s_1, a_1) + \gamma^2 \mathrm{R}(s_2, a_2) + \cdots\right], \tag{7}$$

where $\gamma \in (0, 1)$ is the discount factor to make sure V converges eventually, which indicates that the latter state has a smaller impact on the total reward. The state-to-action mapping is defined as policy function $\pi : S \to A$. In order to evaluate the quality of policy, a value function $V(s)$ or state-action value function $Q(s, a)$ is defined as the expected discounted cumulative reward when performing policy π at the state s.

$$V(s) = E\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots | s_0 = s, \pi\right] \tag{8}$$

$$\begin{aligned} V^{\pi}(s_0) &= R(s_0) + \gamma\left(E\left[R(s_1) + \gamma R(s_2) + \gamma^2 R(s_3) + \cdots\right]\right) \\ &= R(s_0) + \gamma V^{\pi}(s') \end{aligned} \tag{9}$$

$$V^{\pi}(s) = R(s) + \gamma \sum\nolimits_{s' \in S} P^{a}_{s,s'} V^{\pi}(s'). \tag{10}$$

There is an optimal policy π to get an optimal $V^*(s)$:

$$V^*(s) = max_{\pi} V^{\pi}(s). \tag{11}$$

The Bellman equation of the above formula is as follows.

$$V^*(s) = R(s) + max_{a \in A} \gamma \sum\nolimits_{s' \in S} P^{a}_{s,s'} V^*(s'). \tag{12}$$

Then the optimal policy $\pi^* : S \to A$ is as follows.

$$\pi^*(s) = \underset{a \in A}{arg\,max}\, V^*(s) \tag{13}$$

The relationship between value function and RL is shown in Fig. 3. RL updates the value function by evaluating the policy, and uses the value function to determine the policy. The optimal value function and policy can be found step by step through this iteration.

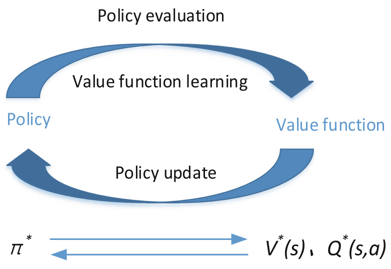

Fig. 3. The relationship between value function and RL

In this paper, we select Q-learning, one of off-policy RL algorithms, to interact with air combat environment, which replaces the value function $V(s)$ with the state-action value $Q(s,a)$ and update $Q(s,a)$ with $Q^*(s,a)$.

$$Q(s,a) = r + \gamma \sum_{s' \in S} P^a_{s,s'} \max_{a'} Q(s', a' \mid a' \in A) \tag{14}$$

$$Q^*(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma \max_{a' \in A(s)} Q\left(s', a'\right) - Q(s,a)) \tag{15}$$

where α is the learning rate, and $\alpha = 1.0$ in this paper so that the iterative formula is shown as follows.

$$Q^*(s,a) \leftarrow r + \gamma \max_{a' \in A(s)} Q\left(s', a'\right) \tag{16}$$

3.2 Deep Q-Learning Network

In this paper, we built a deep Q-learning network (DQN) to approximate the action-value function, which combines deep neural network with Q-learning algorithm. The deep neural network (DNN) is a multi-layer structure and the connections between layers are realized by the weights of multiple neurons. In the supervised learning, DNN provide a complex and nonlinear hypothesis function $h_{w,b}(x)$ to fit the data by training the sample set $(x^{(i)}, y^{(i)})$. The parameters of model are recorded as (W, b), in which W represents the connecting weights between neurons and b is offset vector. In this paper, activate function $f : \mathbb{R} \rightarrow \mathbb{R}$ is described as Rectified Linear Unit (ReLU) because of its quick convergence velocity. And fully-connected layer is used to achieve the connection between the layers. And in the back propagation, we need define a loss function to evaluate the difference between the output value of the network and the real value, and optimize the objective function by parameter optimization, such as stochastic gradient descent, to update the weights of the network. Dropout is used as a measure to prevent overfitting. The parameters of function approximator are shown in Table 2.

Table 2. The parameters of function approximator

Name	Description
Input layer	A feature vector of 18 dimensions
1st-hidden layer	The neuron number of 90 dimensions
2nd-hidden layer	The neuron number of 36 dimensions
Output layer	The Q-value of 5 actions
Activate function	ReLU
Loss function	MSE
Optimization function	RMSProp
Dropout	The probability is 0.2

Some problems will be faced when combining deep learning with reinforcement learning, including the strong correlations between the samples, noisy and delayed reward signal and the changing data distribution with the new behavior. The method of experience replay is proposed to overcome these problems, in which a replay memory is established as $M_t = \{e_1, \cdots, e_t\}$ to store the agent's experiences $e_t = \{s_t, a_t, r_t, s_{t+1}\}$ at each-time step in each training epoch (where the end of an epoch occurs when a terminal state is reached). Then we obtain the experiences <s, a, r, s′> from the pool of stored samples using uniform random sampling and use them to do mini-batch update. During the experiment, we did not store all experiences due to memory limitations, but maintained an experience queue M of length N following the rules of first in first out, sampling samples from the queue when the update was performed. This approach improves the utilization of data and avoids falling into local minimum.

In addition, $\varepsilon - greedy$ policy is proposed to balance the exploration and exploitation by randomly selecting action. When selecting possible actions, agent will select a best action from the current model with a probability of $1 - \varepsilon$, and select a random action with probability ε. The value of ε will gradually decrease over time. Initial value of ε is 1.0.

$$\varepsilon = max\left(\varepsilon - \frac{1}{total\_steps}, 0.1\right) \tag{17}$$

$$action = \begin{cases} randint(0,5) & random() < \varepsilon \\ argmaxQ & others \end{cases} \tag{18}$$

where the value of ε will remain when reduced to 0.1. Action is selected from all possible actions to collect environmental information for follow-up learning at the beginning of ε close to 1. With the value of ε decreasing, agent performs its learned strategies when having more accurate prediction.

The process of DQN algorithm is shown below.

```
Deep Q-learning Algorithm
Initialize replay memory $M$ to capacity $N$
For epoch=1,T do
  Initialize state $s_1 = \{x_1\}$ and feature $f_1 = \emptyset(s_1)$
  While (status == 1)
    Initialize action-value function Q with random weights
    With $\varepsilon - greedy$ policy to select an action $a_t$
    Execute action $a_t$ in simulation and observe reward $r_t$ and
    next state $s'$and preprocessed feature sequence $\emptyset(s')$
    Sample random mini-batch of transitions from $M$
    Set $y_i = \begin{cases} r & terminal\ state \\ r + \gamma maxQ(s', a') & non - terminal\ state \end{cases}$
    Transform sampled data into training tuple of the model
    $< x_k, y_k > \; x_k = \{\emptyset(s)\}, y_k = y_i$
    Performs RMSProp on MSE to update the network weights
    with $< x_k, y_k >$
    Save model weights
  Break when step_to_stop <=0
End for
```

In this paper, we applied DQN approach to air combat scenario to evaluate our agent. The overall framework of air combat model is shown in Fig. 4. With the agent's action, the UCAV Air Combat Model can generate a series of continuous states

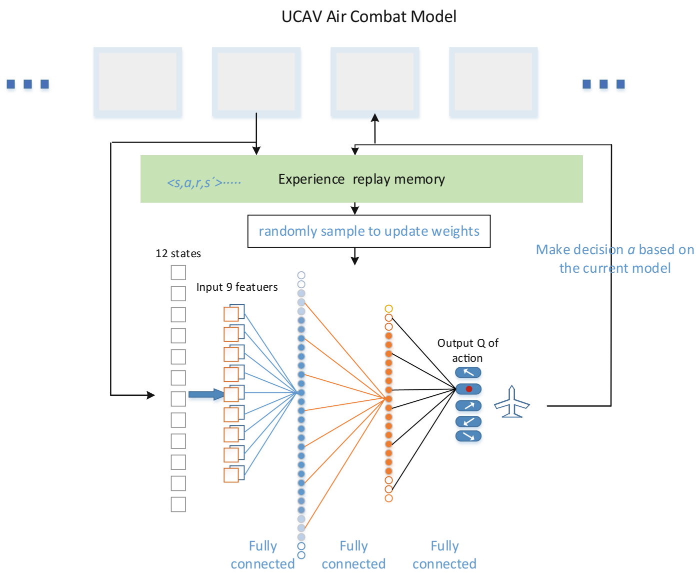

Fig. 4. The framework of air combat model

preprocessed as the inputs. The network is able to output the Q values of five actions, only receiving the 9-dimonsion features. Then observed variables (current state, action, reward, next state) are stored in the experience queue, from which the empirical samples are sampled. These samples are transformed into tuples of training network to update the weights according to the gradient information.

4 Experiment

To verify the validity of the algorithm, we take advantage of DQN agent fighting with an enemy using the Min-Max algorithm in the air combat platform. The Min-Max algorithm can make decisions by looking into future for steps [17]. Its depth is set as 3 steps in the simulation experiment taking into account the factors of computation time and decision-making effect. We use the belligerent results of two agents at the same simulation model, to demonstrate that our approach robustly learns more successful policies with the following minimal prior knowledge: features and all possible actions.

The experiments are arranged as follows. The initial states are selected randomly in Table 3.

Table 3. Initial states

State	Value
x_r, x_b	[−20, 20]
y_r, y_b	[−20, 20]
h_r, h_b	[−10, 10]
$\varnothing_r, \varnothing_b$	0, horizontal fly
ψ_r, ψ_b	[−π, π]
v_{zb}, v_{zr}	0, horizontal fly

The implementation of the experiment consists of two parts, training and testing, after building DQN model. The training set is used to adjust parameters and improve design, and testing set is used to evaluate the ability of DQN agent's decision. Winning or losing is determined by whether to enter the dominant area (that is $0.1 < R < 3$ and $|AA| < 1.105$ and $|ATA| < 0.523$). The value of *is_win* is set to 1 when red is dominant position and is set to -1 when blue is dominant position.

In the training section, we trained 5000 air combat epochs, each of which consists of one hundred steps, which means the Q-function is approximated after 500000 steps of sampling. All hyperparameters in training model are provided in Table 4, which consists of their values and descriptions.

In the testing section, we train 100 air combat epochs of 100 steps using the model of every 5000 sampling steps to observe the merits of the strategy. The decision performance is measured with 3 metrics: The average reward per epoch is defined that the cumulative rewards of all steps in each war round are divided by the number of fighting steps at the current round; The max Q-value, represents the expected discount

Table 4. List of hyperparameters and their values in training model

Hyperparameters	Value	Description
γ	0.9	Discount factor used in Q-learning update
ε	(0.1, 1.0)	The range of ε
Epochs	5000	The number of war rounds (where the end of an epoch occurs when a terminal state is reached)
Batch size	500	The bitch of random sampling from experience replay memory
Buffer size	50000	The length of replay memory
Observe	2000	The start size of replay
Total steps	5000 * 100	The number of all states

reward using the best action to the state S under the current model parameters; and the total number of winning, losing and drawing in each epoch are recorded.

In order to observe the stability of the algorithm, the average reward in the training process is represented through the broken line graph in Fig. 5. We use the weight parameters of model every 5000 steps to test 100 air combat epochs and get average reward of each model. It can be seen that the value of average reward is gradually increasing with the optimization of model parameters, though there is still noise due to random action selection of $\varepsilon - greedy$. This demonstrates that the level of agent's decision-making is generally improved as the training epoch increases.

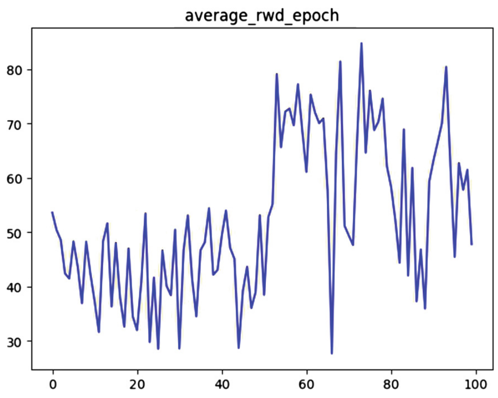

Fig. 5. Average reward per epoch

In order to evaluate the strategy more effectively, we observe the learning process through another metric, that is action-value function. Before testing, a fixed set of air combat states is collected to sample with 1 versus 1 air combat. Then we observe the decision making of sampling states with the weight parameters of model every 5000 steps, which can be represented by the max Q value. The changes of this variable is shown in Fig. 6. It can be seen that the max Q value is gradually increasing with the improvement of air combat results. At the same time, the max Q value is gradually converged to stability to demonstrate the feasibility of the learning model using deep neural network to fit Q function.

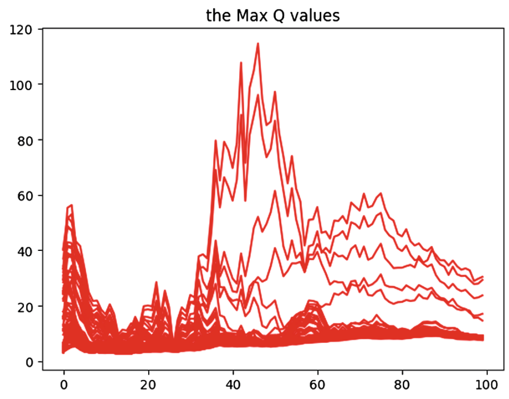

Fig. 6. Max Q value

The numbers of winning, losing and drawing are represented with red solid line, blue dotted line, and green point broken line in Fig. 7, respectively. It is clear that the numbers of winning and losing are almost the same at the beginning and the numbers of winning are gradually increasing with the optimization of model parameters.

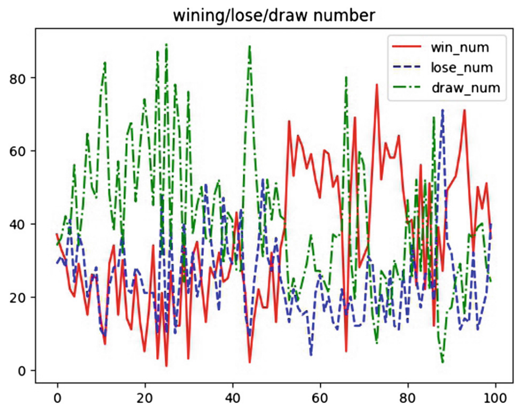

Fig. 7. Winning/lose/drawing numbers (Color figure online)

These results show that the approach combining deep neural network with reinforcement learning is effective and of high performance to resolve air combat problem. The plane using DQN approach has a better ability of air combat decision-making comparing with the plane using Min-Max approach.

5 Conclusion and Future Works

This paper studied the problem of UCAV air combat and employed an improved Q-learning approach to interact with environment of 1 versus 1 air combat. The deep neural network (fully connected layer) was used as the function approximator of DQN approach. The selections of features and reward function depend on the pilot's empirical indicators judging air combat situation in the real world. The DNQ approach involves the iteration of fitting functions and design of deep neural network. To break the strong correlation between samples and improve the stability of fitting function, we used experience replay, $\varepsilon - greedy$ policy, dropout, etc., to optimize the model structure and get a better convergence. According to the simulation results, the decision-making performance of DQN approach is quicker and more effective than Min-Max recursive approach.

In the future work, we plan to employ the features extracted by convolution neural network to replace artificial features for which the current features are too dependent on subjective experience. It is worthy looking forward to extract better features automatically by adding convolution neural network into the current network structure. And one of the follow-up works is to compare with other methods of decision-making.

References

1. Schmitt, M.N.: Unmanned combat aircraft systems (Armed Drones) and international humanitarian law: simplifying the oft benighted debate. J. Soc. Sci. Electron. Publish. **2**, 1006–1013 (2012)
2. Tsach, S., Peled, A., Penn, D., et al.: Development trends for next generation of UAV systems. In: AIAA Infotech@Aerospace 2007 Conference and Exhibit, Rohnert Park, California (2007)
3. Burgin, G., Sidor, L.B.: Rule-based air combat simulation. Technical report, TITAN-TLJ-H-1501, Titan Systems Inc., La Jolla, Calif, USA (1988)
4. Virtanen, K., Raivio, T., Hamalainen, R.P.: Modeling pilot's sequential maneuvering decisions by a multistage influence diagram. J. Guidance Control Dyn. **27**(4), 665–677 (2004)
5. Perelman, A., Shima, T., Rusnak, I.: Cooperative differential games strategies for active aircraft protection from a homing missile. In: AIAA Guidance, Navigation, and Control Conference, Toronto, Ontario, Canada (2010). J. Guidance Control Dyn. **34**(3), 761–773 (2010)
6. Nohejl, A.: Grammar-based genetic programming. Department of Software and Computer Science Education (2011)
7. Schvaneveldt, R.W., Goldsmith, T.E., Benson, A.E., et al.: Neural Network Models of Air Combat Maneuvering. New Mexico State University, New Mexico (1992)
8. Mnih, V., Kavukcuoglu, K., Silver, D., et al.: Human-level control through deep reinforcement learning. J. Nature **518**(7540), 529–533 (2015)
9. Silver, D., Huang, A., Maddison, C.J., et al.: Mastering the game of Go with deep neural networks and tree search. J. Nature **529**(7587), 484–489 (2016)
10. Liu, H.L., Taniguchi, T., Takano, T., et al.: Visualization of driving behavior using deep sparse autoencoder. In: IEEE Intelligent Vehicles Symposium, pp. 1427–1434 (2014)

11. Liu, H.L., Taniguchi, T., Tanaka, Y., et al.: Visualization of driving behavior based on hidden feature extraction by using deep learning. J IEEE Trans. Intell. Transp. Syst. **99**, 1–13 (2017)
12. Levine, S., Wagener, N., Abbeel, P.: Learning contact-rich manipulation skills with guided policy search. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 156–163. IEEE, Seattle (2015)
13. Levine, S., Pastor, P., Krizhevsky, A., et al.: Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. arXiv preprint arXiv:1603.02199 (2016)
14. Roessingh, J.J., Merk, R.J., Huibers, P., et al.: Smart Bandits in air-to-air combat training: combining different behavioural models in a common architecture. In: 21st Annual Conference on Behavior Representation in Modeling and Simulation, Amelia Island, Florida, USA (2012)
15. McGrew, J.S., How, J.P., Williams, B., et al.: Air-combat strategy using approximate dynamic programming. J. Guidance Control Dyn. **33**(5), 1641–1654 (2012)
16. Teng, T.H., Tan, A.H., Tan, A.H., et al.: Self-organizing neural networks for learning air combat maneuvers. In: The 2012 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2012)
17. Ma, Y., Ma, X., Song, X.: A case study on air combat decision using approximated dynamic programming. Math. Probl. Eng. **2014** (2014)
18. Maei, H.R., Szepesvári, C., Bhatnagar, S., et al.: Convergent temporal-difference learning with arbitrary smooth function approximation. In: Advances in Neural Information Processing Systems, pp. 1204–1212. MIT Press, Vancouver (2009)
19. Duan, H., Shao, X., Hou, W., et al.: An incremental learning algorithm for Lagrangian support vector machines. J. Pattern Recogn. Lett. **30**(15), 1384–1391 (2009)
20. Lange, S., Riedmiller, M., Voigtländer, A.: Autonomous reinforcement learning on raw visual input data in a real world application. In: Proceedings of the International Joint Conference on Neural Networks, pp. 1–8. IEEE, Brisbane (2012)
21. Yang, D., Ma, Y.: An air combat decision-making method based on knowledge and grammar evolution. In: Zhang, L., Song, X., Wu, Y. (eds.) AsiaSim/SCS AutumnSim -2016. CCIS, vol. 644, pp. 508–518. Springer, Singapore (2016). doi:10.1007/978-981-10-2666-9_52
22. Ma, Y., Gong, G., Peng, X.: Cognition behavior model for air combat based on reinforcement learning. J. Beijing Univ. Aeronaut. Astronaut. **36**(4), 379–383 (2010)