Advancing Air Combat Tactics with Improved Neural Fictitious Self-Play Reinforcement Learning

Shaoqin He[1,2], Yang Gao[1](✉), Baofeng Zhang[1,2], Hui Chang[1], and Xinchen Zhang[1]

[1] Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
{heshaoqin2021, yang.gao, zhangbaofeng2022, hui.chang, xinchen.zhang}@ia.ac.cn

Abstract. We study the problem of utilizing reinforcement learning for action control in 1v1 Beyond-Visual-Range (BVR) air combat. In contrast to most reinforcement learning problems, 1v1 BVR air combat belongs to the class of two-player zero-sum games with long decision-making periods and sparse rewards. The complexity of action and state space in this game makes it difficult to learn high-level air combat strategies from scratch. To address this problem, we propose a reinforcement learning self-play training framework to solve it from two aspects: the decision model and the training algorithm. Our decision-making model uses the Soft actor-critic (SAC) algorithm, a method based on maximum entropy, as the action control of the reinforcement learning part, and introduces an action mask to achieve efficient exploration. Our training algorithm improves Neural Fictitious Self-Play (NFSP) and proposes the best response history correction (BRHC) version of NFSP. These two components helped our algorithm to achieve efficient training in the high-fidelity simulation environment. The result of the 1v1 BVR air combat problem shows that the improved **NFSP-BRHC** algorithm outperforms both the NFSP and the Self-Play (SP) algorithms.

Keywords: Air Combat, Reinforcement Learning, Neural Fictitious Self-Play.

1 Introduction

The development of intelligent algorithms for unmanned air combat has been a significant research topic for a long time. Since the 1960s, some scholars have been developing intelligent air combat systems, attempting to use artificial intel- ligence methods to replace pilots in air combat decision-making. Prior methods mainly include: expert systems constructed by knowledge from air combat do- main experts[2], decision systems built by heuristic algorithms such as genetic algorithms and fuzzy logic[3], modelling and solving air combat decisions based on game theory[20], supervised learning based on deep learning, using expert- annotated air combat decision data for behavior cloning. In recent years, with the success of Alpha Go, a Go AI developed by the DeepMind team, in defeating the human world champion, more and more reinforcement learning-based algorithm research has also appeared in the air combat domain[19, 11, 5, 15, 14, 13, 17]. The expert system method needs to rely on human experts to design a large number of air combat rules, and the performance of the rule-based air combat intelligent algorithm will not exceed the ability of the designer himself. Heuristic algorithms require experts to design utility functions. The method based on game theory faces the difficulty of modeling real air combat problems and solving Nash equilibrium. Behavioral cloning methods are limited by the size of the data and the ability of the pilots who annotate the data. The current work based on reinforcement learning methods has the problem that the simulation environment is too simple and the target strategy is fixed.

In order to solve these problems, this paper uses reinforcement learning combined with our proposed improved version of the NFSP training method. The contributions of this paper are as follows:

- In the decision-making model, we use the Soft actor-critic (SAC) algorithm, an entropy-based method, for the reinforcement learning aspect of action control, and introduce the mechanism of action mask to help the algorithm to explore efficiently.
- In the training algorithm, we propose an improved version of the NFSP algorithm, called NFSP-BRHC, which is closer to the Nash equilibrium than the original NFSP algorithm.
- Ablation and comparison experiments are conducted in a high-fidelity simulation environment, showing that the NFSP-BRHC training algorithm outperforms the NFSP and SP training algorithms.

2 Related Works

With the continuous innovation of reinforcement learning algorithms, more and more air combat decision-making methods based on reinforcement learning have emerged. Liu[11] used the Deep Q-Learning (DQN)[12] algorithm for short-range maneuvers in air combat, and its training opponent was based on the Min-Max recursive approach with a depth of 5. Yang[19] used the DQN algorithm for the same scenario, and at the same time used the idea of course learning for basic training, and the opponent strategy for subsequent training was based on statistical rule algorithms. Guo[5] verified the effect of Deep Deterministic Policy Gradient (DDPG)[10] algorithm in the same scene. Qiu[15], on the other hand, added the action of firing missiles in a two-dimensional environment using the improved Twin Delayed Deep Deterministic Policy Gradient (TD3)[4] algorithm to learn maneuvers to evade the missiles. Piao[13] used the Proximal Policy Optimization (PPO)[16] algorithm in 1v1 BVR air combat and trained it through the self-play training method. In the Alpha Dogfight Trail held by DARPA, Adrian P. Pope et al.[14] developed an air combat agent based on the layered reinforcement learning architecture of the SAC algorithm[7], and won second place in the competition.

The first four works[11, 19, 5, 15] all use rule-based target strategy as opponents for training, which suffers from many problems such as overfitting and being limited by the target strategy. The last two works[13][14] use a training framework based on self-play to achieve autonomous evolution of air combat AI, but this training method has the problem that the game may be caught in a loop and cannot converge to the Nash equilibrium.

Different from the above methods, the training method used in our study is an improved version of NFSP. We improve the NFSP algorithm to make it converge faster and closer to the approximate Nash equilibrium during the training process. In terms of exploring the state space, we propose action masks to help the agent improve the exploration efficiency. These improvements allow agents to converge faster during training and perform better in high-fidelity simulation environment.

3 Proposed Method

In this section, the BVR air combat problem will be modeled as a Markov game[1]. Then, the decision model and network architecture of the BVR air combat agent we use are described. Next, an action mask is introduced to help AI better explore the state space. Finally, several self-play algorithms are discussed, and a Neural Fictitious Self-Play algorithm based on best response history correction (NFSP-BRHC) is proposed for the tactical evolution of air combat agents.

3.1 BVR Air Combat Modeling

In this paper, the 1v1 BVR air combat problem is modeled as a fully competitive multi-agent Markov game. We can use the tuple $(s_0, S, A, P, r, \gamma)$ to define this Markov game, where s_0 is the initial state, S is the joint state space, A is the joint action space, $P: S \times A \to S$ presents the state transition probability, $r: S \times A \to R$ presents the reward function, and $\gamma \in (0,1]$ is the discount factor. For agent i, the state space is S^i, the action space is A^i, the reward is r_i, and its policy π_i is the mapping of $S^i \times A^i \to [0,1]$. The goal of each agent in the Markov game is to maximize the cumulative discounted reward: $R_i = E_{\rho_{\pi_i}}\left[\sum_{t=0}^{T} \gamma^t r_i(s_t, a_t)\right]$, where T represents the time horizon, ρ_{π_i} denotes the probability distribution of trajectories produced by following policy π_i.

State space.

The state space of BVR air combat mainly contains two parts: the state S_i that can be obtained directly and the state S_{id} that needs to be obtained indirectly by further calculation. S_i includes opposing aircrafts' three-dimensional coordinates x, y, z, three angles ϕ_p, ϕ_r, ϕ_y, indicating the attitude, velocities v_x, v_y, v_z in the direction of the three axes, magnitude of the combined velocity $|v|$, radar lock signal lo and the number of remaining missiles m. S_{id} includes opposing aircrafts' distance R, the projected distances r_x, r_y, r_z in the direction of the three axes, the closing rate R', the aspect angle AA, the antenna train angle ATA, the elevation angle EA, the heading crossing angle HCA, and finally the energy-related velocity squared E and velocity squared difference E_d. The entire state space is defined as follows:

$$\begin{aligned} S^i = [&x, y, z, \phi_p, \phi_r, \phi_y, v_x, v_y, v_z, |v|, lo, m, R, \\ &r_x, r_y, r_z, R', AA, ATA, EA, HCA, E, E_d]. \end{aligned} \tag{1}$$

There are 38 states in total. All states are inclusive of both sides except for the distance R, the projected distances r_x, r_y, r_z, the closing rate R', the elevation angle EA, the heading crossing angle HCA and the velocity squared difference E_d.

Action Space.

Fighter pilots can make many complex tactical maneuvers during their missions, and these complex tactical maneuvers are composed of basic atomic actions. So as long as the action space contains these basic atomic actions, it can cover higher-level complex maneuvers. Referring to the seven basic maneuvers [automated maneuvering decisions for air-to-air combat] designed by NASA scholars, the action space we designed contains seven basic actions: uniform straight flight, upward flight, downward flight, left-turn flight, right-turn flight, accelerate, decelerate, and a launch action to control the missile, as shown in Table 1.

Table 1. Action Space.

Category	Serial	Basic Atomic Action
Maneuvers	a_1	uniform straight flight
	a_2	upward flight
	a_3	downward flight
	a_4	left-turn flight
	a_5	right-turn flight
	a_6	accelerate
	a_7	decelerate
Attack	a_8	launch missile

Reward Definition.

In past work, some relied on fine-tuned per-step reward signals from human experts, while others were based on sparse reward signals from important events[13]. Per-step reward signals can solve the problem of cold start, but are difficult to adjust and performance is limited by the designer's subjective understanding of the air combat advantage. Important event rewards reflect objective air combat events, but are too sparse and suffer from the cold start problem. Therefore, we combine these two types of rewards. The rewards at each step with small weights guide the agent to explore the tendency to explore more valuable states to solve the problem of cold start, and the objective important event rewards help the agent correctly recognize the causal relationship between winning and policy.

The rewards for each step include whether the radar detects the opponent, speed advantage in the direction of the enemy's defense line and a fixed negative reward to encourage the agent to end the fight as soon as possible. Rewards for important events include sparse rewards for launching missiles and rewards for successfully evading missiles. The rewards for launching missiles are negative, which can make the agent more cautious about launching missiles. This setting is because each agent only carries two missiles and needs to maximize the benefits of launching. Rewards for important events also include rewards for crossing the enemy's line, rewards for launching missiles and hitting the enemy to win, and penalties for going out of bounds.

The coefficient relationship between the rewards of each step plus the rewards of the missile and the reward of the final result is such that the winning side will receive much larger rewards than the losing side. The reward settings are described in Table 2 ,where n denotes the max step number in the simulation.

Table 2. Reward Definition.

Category	Event	Weight
Per-step Rewards	Detect	0.5
	Be detected	-0.5
	Speed advantage	[-1,1]
	fixed negative	-0.1
Import Event Rewards	Launch missile	-50
	Escape missile	50
	Cross the line first	2n
	Cross the line late	-2n
	Hit	2n
	Be hit	-2n
	Out of bounds	-4n

3.2 Decision Model

Soft actor-critic.

Soft actor-critic (SAC)[7] is a maximum entropy reinforcement learning (MERL) algorithm developed based on the idea of maximum entropy, which outputs a stochastic distributed policy function similar to the PPO algorithm. The difference is that the SAC algorithm is an off-policy actor-critic algorithm. Compared with PPO, the sample efficiency of SAC is higher. Compared with the DDPG algorithm, which belongs to the off-policy actor-critic algorithm, SAC uses a stochastic policy, while DDPG uses a deterministic policy. The stochastic policy can help the agent to explore better and converge more stably, while the deterministic policy is more likely to fall into a local optimum, and its performance and convergence are also unstable. And what makes SAC most different from other reinforcement learning algorithms is that it maximizes the entropy of the policy while optimizing it to maximize the cumulative benefit. This makes the stochastic policy behave more randomly while ensuring the cumulative return, which can more fully explore the state space and further enhance the stability of the algorithm's performance. After introducing the maximum entropy, the training goal of the SAC actor is to maximize the cumulative reward and the expectation of the policy entropy, which is expressed as follows:

$$\pi^*_{MaxEnt} = \underset{\pi}{\operatorname{argmax}} \sum_t \ \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[r(s_t, a_t) + \alpha H\big(\pi(\cdot\ |s_t)\big) \right], \tag{2}$$

where ρ_π denotes the distribution of state-action pairs that the agent encounters under the control of policy π, α is the temperature coefficient, which is used to adjust the degree of emphasis on entropy.

Due to the different optimization goals, the value iteration formula of maximum entropy reinforcement learning is slightly changed compared with standard reinforcement learning. The Bellman equation for value iteration of the soft Q function and soft V function in the SAC algorithm is as follows:

$$Q^{\pi}_{soft}(s_t, a_t) = \mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)}\left[r(s_t, a_t) + \gamma V^{\pi}_{soft}(s_{t+1})\right], \quad (3)$$

$$V^{\pi}_{soft}(s_t) = \mathbb{E}_{a_t\sim\pi}\left[Q^{\pi}_{soft}(s_t, a_t) - \alpha\log\pi(a_t|s_t)\right]. \quad (4)$$

The critic network corresponds to the soft Q function, and its training goal is to minimize the following loss function:

$$J_Q = \mathbb{E}_{(a_t,s_t)\sim D}\left[\frac{1}{2}\left(Q(s_t, a_t) - \widehat{Q}(s_t, a_t)\right)\right], \quad (5)$$

where D represents the replay buffer, $\widehat{Q}$ is the target critic network.

The agent policy approximates the energy-based policy (EBP)[6] by minimizing the KL divergence between the policy represented by the state-conditioned stochastic neural network[6] and EBP. Finally, the loss function of the actor network is derived as follows:

$$J_{\pi} = \mathbb{E}_{s_t\sim D,a_t\sim\pi}\left[\log\pi(a_t|s_t) - \frac{1}{\alpha}Q(s_t, a_t)\right]. \quad (6)$$

Many performance-improving techniques are used in SAC, including drawing on the double Q network[18] and target Q network in DQN to alleviate the problem of overestimation of the Q value. One of the most important improvements is the automatic entropy adjustment mechanism. α, as a hyperparameter that controls MERL's emphasis on entropy, has a significant impact on the performance of the algorithm. It has different suitable values in different reinforcement learning tasks or different training stages of the same task[7]. The authors of SAC propose an improvement to automatically adjust alpha to keep entropy always greater than a threshold while maximizing the expected reward. The final loss function on alpha is as follows:

$$J_{\alpha} = \mathbb{E}_{a_t\sim\pi_t}[-\alpha\log\pi_t(a_t|s_t) - \alpha\mathcal{H}_0], \quad (7)$$

where $\mathcal{H}_0$ denotes the threshold of entropy.

Action Mask.

Due to the huge combination of action and state space in 1v1 BVR air combat, in order to improve the efficiency of training, the method of action mask is proposed to prune the exploration of reinforcement learning. There is not much expert knowledge involved in the action mask, and it is mainly used to eliminate some unreasonable places in the training process: 1) About going out of bounds, whether it is falling into the sea or going out of bounds in the horizontal direction. By designing the action mask to realize turning or climbing in advance to avoid failure caused by going out of bounds. 2) Regarding launching missiles, avoid launching missiles when you are in a state where launching missiles is impossible to hit or when there are no missiles.

Action masks help the algorithm cut down on exploration and use limited resources to explore more meaningful states.

Network Structure.

The network structure of the reinforcement learning part of a single agent is shown in Fig. 1. The observed value of the environment is composed of two parts: direct observation O_d and indirect observation O_{id} which needs further calculation. After normalization, S_d and S_{id} are obtained respectively, and these two parts constitute the state of the whole problem. Then input the state to an actor network and four critic networks (the other three critic networks with the same structure are omitted in the figure). The actor network and the critic network are composed of four layers of fully connected layers, the number of hidden units in each layer is 512, and the activation function is ReLU. In addition, there is a module called Action Mask that calculates the action mask of the current state based on the environment observation.

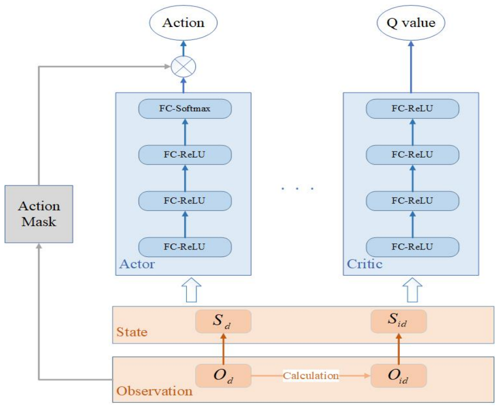

Fig. 1. The neural network structure of the RL part of the agent

3.3 Neural Fictitious Self-Play with Best Response History Correction

The training methods used in previous work[13,14,17] on the application of reinforcement learning to air combat are often based on self-play. However, Using self-game in a non-transitive game will make the trained models of each generation fall into a cycle. In short, there are three policies A, B, and C in a non-transitive game, and $A > C$ cannot be obtained when $A > B$ and $B > C$ are satisfied.

Fictitious Self-Play[8] can alleviate this problem, and it has been proved that it can converge to approximate Nash equilibrium in imperfect-information poker games. In the Fictitious self-play, the current policy of the agent is obtained by the weighted average of the best response(BR) and the historical average policy. Among them, the BR is the best response of the agent to the opponent's historical average policy, and the historical average policy is the historical average of BR for one's own side.

In the Neural Fictitious Self-Play[9], the best response and the historical average policy are represented by the neural network, as shown in Fig. 2. Each agent in the Neural Fictitious Self-Play consists of a best response policy network and a historical average policy network. The best response policy network is learned and trained by a reinforcement learning algorithm, such as SAC with an action mask used in this paper. The historical average policy network is trained by supervised learning, and the

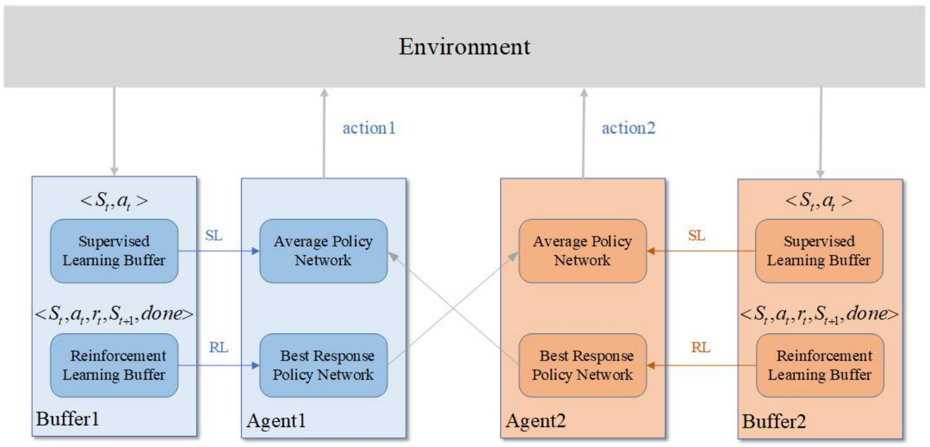
Environment
action1
action2
$<S_t,a_t>$
Supervised Learning Buffer
SL
Average Policy Network
$<S_t,a_t,r_t,S_{t+1},done>$
Reinforcement Learning Buffer
RL
Best Response Policy Network
Buffer1
Agent1
Average Policy Network
SL
$<S_t,a_t>$
Supervised Learning Buffer
$<S_t,a_t,r_t,S_{t+1},done>$
Best Response Policy Network
RL
Reinforcement Learning Buffer
Agent2
Buffer2

Fig. 2. The NFSP framework

training samples come from the past average behavior of the best response policy network. The supervised learning part uses the logarithmic loss function:

$$J_{\Pi} = \mathbb{E}_{(s,a)\sim M_{SL}}[-log\Pi(a|s)]. \tag{8}$$

The final output policy of the agent trained by NFSP is a weighted mixture of the best response policy and the historical average policy, where the weight is η.

Fictitious Play can converge to Nash equilibrium in theory, but Neural Fictitious Self-Play uses the neural network to approximate the best response policy, and there is a certain gap between the obtained and real best response. Because each time the best response policy network is used to approximate the best response, the network may not converge completely, and there will always be a deviation. For example, in the game of rock-paper-scissors, the opponent's strategy is fixed rock, and our best response policy network outputs a policy of 25\% rock, 5\% scissors, and 70\% paper after a round of training. This is not the best response. If we do not choose scissors because of the 30\% probability, then the supervised learning buffer used to learn the historical average policy will be mixed with many non-best response samples, which will cause errors in the learning of the historical average policy. Therefore, we propose an NFSP with the best response history correction (**NFSP-BRHC**) to solve this problem.

NFSP-BRHC no longer stores all the output history of the best response strategy network into the supervised learning buffer of the historical average policy, but stores the policy of the final winning episode output by the best response network into the SL buffer. Because in the two-person zero-sum game with long-term sparse rewards, the real reward signal is only the final victory or failure. In this game, the final winning decision sequence is more consistent with the definition of the best response than the failed decision sequence generated due to incomplete convergence, and it is also more valuable in training the historical average policy.

In addition, considering that there are few samples in the supervised learning (SL) buffer in the early stage of training, so the best response strategy should account for a greater weight in the mixed strategy output. With the increase of training times, the best response policy samples in the SL buffer continue to expand, and the weight of the average policy should continue to increase. In this way, the second improvement of the NFSP-BRHC algorithm is obtained, and the fixed mixing coefficient is improved to decay to a threshold with the number of training generations.

The improvements to the SL buffer samples and the attenuated mixing coefficients constitute the complete NFSP-BRHC training algorithm. Combining NFSP-BRHC and SAC algorithm with action mask, we get NFSP-BRHC with SAC, shown in Algorithm 1.

1v1 BVR air combat is a two-person zero-sum game with long-term sparse rewards, and other auxiliary rewards are set to help the algorithm converge faster and explore better. In the experimental part, we verified the effectiveness of the algorithm improvement on the 1v1 BVR air combat problem.

Algorithm 1 NFSP-BRHC with SAC

```
Execute RUN for every agent in the game $\Gamma$
function RUN($\Gamma$)
  Initialize replay memories $M_{RL}$ and $M_{SL}$
  Initialize average-policy network $\Pi(a|s;\theta^{\Pi})$
  Initialize actor $\pi(a|s;\theta^{\pi})$, critic $Q(s,a;\theta^{Q})$ in best response network
  Initialize target critic network parameters $\theta^{Q'} \leftarrow \theta^{Q}$
  Initialize RL mixed weight parameter $\eta$
  for each episode do
    Observe initial state $s_1$ and reward $r_1$
    Calculate action mask $m_s$
    Set policy $\sigma \leftarrow \begin{cases} \pi * m_s & \text{with prob} \quad \eta \\ \Pi * m_s & \text{with prob} \quad 1-\eta \end{cases}$
    Make $\eta$ decay with episode
    for $t = 1, ..., T$ do
      Sample action $a_t$ from policy $\sigma$
      Execute action $a_t$ and observe reward $r_{t+1}$ and next state $s_{t+1}$
      Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $M_{RL}$
      if best response policy $\pi$ is selected and win then
        Store behaviour tuple $(s_t, a_t)$ in $M_{SL}$
      end if
      Update
      critic $\theta^{Q_i} \leftarrow \theta^{Q_i} - \beta_Q \nabla_{Q_i} J(Q_i) \quad i = 1,2$
      actor $\theta^{\pi} \leftarrow \theta^{\pi} - \beta_{\pi} \nabla_{\pi} J(\pi)$
      temperature coefficient $\alpha \leftarrow \alpha - \beta_{\alpha} \nabla_{\alpha} J(\alpha)$
      average-policy $\theta^{\Pi} \leftarrow \theta^{\Pi} - \beta_{\Pi} \nabla_{\Pi} J(\Pi)$
      target critic $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$
    end for
  end for
end function
```

4 Experiments

1v1 BVR air combat is a two-person zero-sum game with long-term sparse rewards, and other auxiliary rewards are set to help the algorithm converge faster and explore better. In the experimental part, we verified the effectiveness of the algorithm improvement on the 1v1 BVR air combat problem.

4.1 Simulation Environment and Problem Setting

The 1v1 BVR air combat simulation environment in this article is a high-fidelity 6-DOF simulation environment developed based on C++ language, and the flight dynamics models of the corresponding aircraft and missiles are consistent with those in the DCS world. The aircraft used in our experiments is the F-16 fighter jet. The battlefield for BVR air combat is set as a rectangular area with a length of 100km from north to south and a width of 50km from east to west. The bases of both sides are 10km in size from north to south, and 50km in width from east to west, which is located at the southernmost and northernmost points of the battlefield respectively. The two opposing sides carry two AIM-120 and set off from the center of their respective bases at a height of 3km. If the aircraft is out of bounds, it will be directly judged to be a failure. The out-of-bounds actions include the position exceeding the battlefield boundary and crashing into the sea.

4.2 Experiment Setup

In the experiments, the discount factor is set as 0.99 and the SAC algorithm in the reinforcement learning part uses the Adam optimizer with an initial learning rate of α_π 5.0e-5 for the actor network and α_Q 8.0e-5 for the critic network with a target entropy of -10. The neural network in the supervised learning part contains five hidden layers and uses the Adam optimizer with an initial learning rate of α_Π 5.0e-5. The probability of the BR policy being selected in the NFSP-BRHR, η, is initially 1 and decays to a threshold of 0.1 as the experiment proceeds. In the experiments, each action lasts 0.5s, which is an appropriate value, considering the difference between BVR air combat and Dog Fight. If it is too long, it will be difficult to learn complex maneuvering behaviors, and if it is too short, it will be difficult to train.

4.3 Ablation and Contrast Experiments

In a two-person zero-sum game, comparing the rewards of algorithms is pointless. We use the winning rate of simulation experiments between agents trained by different training algorithms as a comparison standard.

Action Mask Ablation Experiment.

To demonstrate the effectiveness of the action mask, we conduct ablation experiments. Using the SP, NFSP, and NFSP-BRHC three algorithms to train a total of 5000 episodes, about two million (2M) steps.

Finally, the trained red/blue model with the action mask module and the red/blue model without the action mask module of the same training algorithm is simulated 100 times for BVR air combat, and the average winning rate is calculated as shown in Fig. 3. (a).

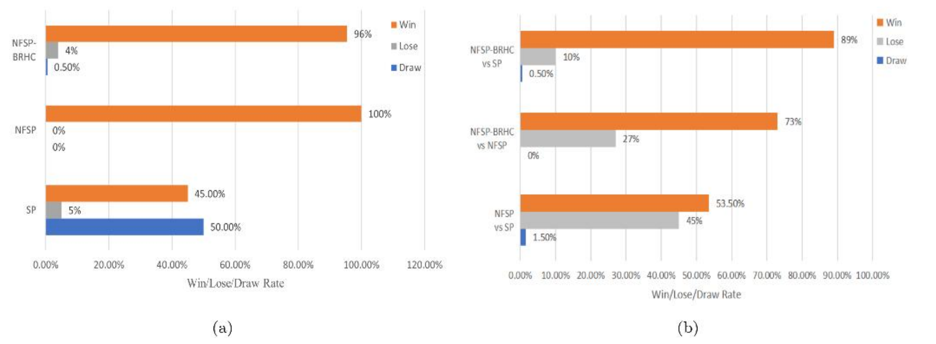

Fig. 3. a) The ablation experiment of Action Mask. b) Comparative experiment with baseline training algorithm

It can be found that the effect of the policy model with the action mask module trained by three different training methods is better than that without the action mask module. The experimental results prove that the action mask can indeed effectively help the agent to prune the exploration space, and use more resources in the places that need to be explored more, thereby helping the agent to achieve a higher winning rate.

Comparison with Baseline Algorithms.

After the same number of training times as in the ablation experiment, 100 BVR air combat simulations are performed between the models obtained by the three training algorithms of SP, NFSP, and NFSP-BRHC. The average winning rate is finally shown in Fig. 3. (b).

It can be found that the relationship among the effects of the three training algorithms is satisfied: NFSP-BRHC > NFSP > SP, which is consistent with our expectations. The model trained by NFSP-BRHC will be closer to the Nash equilibrium and converge faster, while the model trained by NFSP will deviate from the Nash equilibrium due to the error strategy generated when the best response network does not fully converge. However, the models trained by these two training methods are better than SP.

The experimental results confirm the analysis we conducted in the method section, and prove the effectiveness of our proposed NFSP-BRHC training method.

4.4 Tactical Evolution During Training

As shown in Fig. 4, during the training, the agents adapt to each other's policies and continuously realize the evolution of tactics.

These behaviors show that the tactics of the agent trained by the NFSP-BRHC algorithm are constantly evolving during the training process. The policies of the two sides are adapting to each other during the training, and they are constantly approaching the Nash equilibrium in the process of adaptation. This phenomenon of mutual adaptation of policies also confirms our analysis in the Methods section.

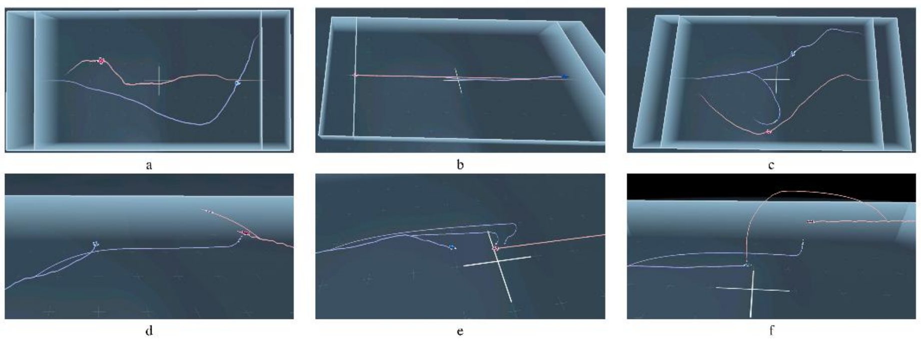

Fig. 4. a) Initially, the agent did not launch missiles or fly directly towards the finish line. b) As training progressed, the agent learned to accelerate at the start to maintain level flight and win the race. c) However, after learning to launch missiles, the strategy of accelerating towards the line became ineffective, and the agent learned to judge possible missile trajectories to escape them. d) The agent mastered missile launching, often performing a crank maneuver to evade opponent missiles launched at a closer range. e) The agent learned to launch two missiles at different positions. f) Additionally, the agent learned to climb first in the game and increase the threat of launching missiles, which also helped it gain a situational advantage.

5 Conclusion

In this paper, we propose a new training method NFSP-BRHC. This method corrects the best response historical samples during training and uses decayed best response weights, making the averaging policy much closer to the historical average of best responses. And, we combine the NFSP-BRHC algorithm and SAC with action mask and apply it to 1v1 BVR air combat decision-making. The experimental results show that in 1v1 BVR air combat, the agent trained by our proposed training algorithm has more advantages than the agents trained based on SP or NFSP.

In addition, the NFSP-BRHC training framework can be used not only in 1v1 BVR air combat but also in other sparsely rewarded two-person zero-sum games. In the future, we intend to apply the NFSP-BRHC training framework to more games that conform to this definition to further verify the effectiveness of the training framework and continue to improve it.

Acknowledgement. This work was supported by National major science and technology plan project, National Defense Science and Technology Foundation Reinforcement Program Key Project, the Strategic Priority Research Program of the Chinese Academy of Sciences.

References

1. Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., Mordatch, I.: Emergent complexity via multi-agent competition. arXiv preprint arXiv:1710.03748 (2017)
2. Burgin, G.H.: Improvements to the adaptive maneuvering logic program. Tech. rep. (1986)
3. Ernest, N., Carroll, D., Schumacher, C., et al.: Genetic fuzzy based artificial intelligence for unmanned combat aerial vehicle control in simulated air combat missions. Journal of Defense Management 6(1), 2167–0374 (2016)
4. Fujimoto, S., Hoof, H., Meger, D.: Addressing function approximation error in

actor-critic methods. In: International conference on machine learning. pp. 1587–1596. PMLR (2018)

5. Guo, J., Wang, Z., Lan, J., Dong, B., Li, R., Yang, Q., Zhang, J.: Maneuver decision of uav in air combat based on deterministic policy gradient. In: 2022 IEEE 17th International Conference on Control & Automation (ICCA). pp. 243–248. IEEE (2022)
6. Haarnoja, T., Tang, H., Abbeel, P., Levine, S.: Reinforcement learning with deep energy-based policies. In: International conference on machine learning. pp. 1352–1361. PMLR (2017)
7. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International conference on machine learning. pp. 1861–1870. PMLR (2018)
8. Heinrich, J., Lanctot, M., Silver, D.: Fictitious self-play in extensive-form games. In: International conference on machine learning. pp. 805–813. PMLR (2015)
9. Heinrich, J., Silver, D.: Deep reinforcement learning from self-play in imperfect information games. arXiv preprint arXiv:1603.01121 (2016)
10. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
11. Liu, P., Ma, Y.: A deep reinforcement learning based intelligent decision method for ucav air combat. In: Modeling, Design and Simulation of Systems: 17th Asia Simulation Conference, AsiaSim 2017, Melaka, Malaysia, August 27–29, 2017, Proceedings, Part I 17. pp. 274–286. Springer (2017)
12. 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 518(7540), 529–533 (2015)
13. Piao, H., Sun, Z., Meng, G., Chen, H., Qu, B., Lang, K., Sun, Y., Yang, S., Peng, X.: Beyond-visual-range air combat tactics auto-generation by reinforcement learning. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2020)
14. Pope, A.P., Ide, J.S., Mićović, D., Diaz, H., Rosenbluth, D., Ritholtz, L., Twedt, J.C., Walker, T.T., Alcedo, K., Javorsek, D.: Hierarchical reinforcement learning for air-to-air combat. In: 2021 international conference on unmanned aircraft systems (ICUAS). pp. 275–284. IEEE (2021)
15. Qiu, X., Yao, Z., Tan, F., Zhu, Z., Lu, J.G.: One-to-one air-combat maneuver strategy based on improved td3 algorithm. In: 2020 Chinese Automation Congress (CAC). pp. 5719–5725. IEEE (2020)
16. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
17. Sun, Z., Piao, H., Yang, Z., Zhao, Y., Zhan, G., Zhou, D., Meng, G., Chen, H., Chen, X., Qu, B., et al.: Multi-agent hierarchical policy gradient for air combat tactics emergence via self-play. Engineering Applications of Artificial Intelligence 98, 104112 (2021)
18. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 30(2016)
19. Yang, Q., Zhang, J., Shi, G., et al.: Maneuver decision of uav in short-range air combat based on deep reinforcement learning. IEEE Access 8, 363–378 (2019)
20. Zheng, H., Deng, Y., Hu, Y.: Fuzzy evidential influence diagram and its evaluation algorithm. Knowledge-Based Systems 131, 28–45 (2017)