frontiers in Bioengineering and Biotechnology

ORIGINAL RESEARCH
published: 22 March 2022
doi: 10.3389/fbioe.2022.827408

Pursuit and Evasion Strategy of a Differential Game Based on Deep Reinforcement Learning

Can Xu[1], *Yin Zhang*[2], *Weigang Wang*[1,3]* *and Ligang Dong*[2]

[1]School of Statistics and Mathematics, Zhejiang Gongshang University, Hangzhou, China, [2]School of Information and Electronic Engineering, Sussex Artificial Intelligence Institute, Zhejiang Gongshang University, Hangzhou, China, [3]Collaborative Innovation Center of Statistical Data Engineering, Technology and Application, Zhejiang Gongshang University, Hangzhou, China

Since the emergence of deep neural network (DNN), it has achieved excellent performance in various research areas. As the combination of DNN and reinforcement learning, deep reinforcement learning (DRL) becomes a new paradigm for solving differential game problems. In this study, we build up a reinforcement learning environment and apply relevant DRL methods to a specific bio-inspired differential game problem: the dog sheep game. The dog sheep game environment is set on a circle where the dog chases down the sheep attempting to escape. According to some presuppositions, we are able to acquire the kinematic pursuit and evasion strategy. Next, this study implements the value-based deep Q network (DQN) model and the deep deterministic policy gradient (DDPG) model to the dog sheep game, attempting to endow the sheep the ability to escape successfully. To enhance the performance of the DQN model, this study brought up the reward mechanism with a time-out strategy and the game environment with an attenuation mechanism of the steering angle of sheep. These modifications effectively increase the probability of escape for the sheep. Furthermore, the DDPG model is adopted due to its continuous action space. Results show the modifications of the DQN model effectively increase the escape probabilities to the same level as the DDPG model. When it comes to the learning ability under various environment difficulties, the refined DQN and the DDPG models have bigger performance enhancement over the naive evasion model in harsh environments than in loose environments.

Keywords: dog sheep game, deep reinforcement learning, deep Q network, deep deterministic policy gradient, differential game

OPEN ACCESS

Edited by:
Gongfa Li,
Wuhan University of Science and Technology, China

Reviewed by:
Jing Li,
Tianjin University of Technology, China
Qiang Yin,
Wuhan Polytechnic University, China

***Correspondence:**
Weigang Wang
wangweigang@zjgsu.edu.cn

Specialty section:
This article was submitted to Bionics and Biomimetics, a section of the journal Frontiers in Bioengineering and Biotechnology

Received: *02 December 2021*
Accepted: *10 February 2022*
Published: *22 March 2022*

Citation:
Xu C, Zhang Y, Wang W and Dong L (2022) Pursuit and Evasion Strategy of a Differential Game Based on Deep Reinforcement Learning. Front. Bioeng. Biotechnol. 10:827408. doi: 10.3389/fbioe.2022.827408

1 INTRODUCTION

Bio-inspired differential games have received much attention in recent years due to their broad applications in surveillance (1975) (Lewin and Breakwell, 1975) and air combat (1978, 1981) (Shinar and Gutman, 1978; Shinar, 1981). Differential games were initially proposed by Isaacs (1965). It is a game theory which solves dynamic game problems through differential equations. In each case, autonomous agents divided into two sides, pursuers and evaders, are against each other. This article focuses on the dog sheep game, which is a bio-inspired differential pursuit–evasion game in a constrained environment. The dog chases down the escaping sheep on the circle, while the sheep manages to run away without getting caught. According to the critical game conditions, we analyze

the kinematics of pursuit and evasion strategy. The result shows the circle can be divided into three parts, each representing the position set of sheep based on whether the sheep is able to escape.

Designing reasonable differential equations from a given game scenario is the theme of differential games. Recently, deep reinforcement learning (DRL) methods have applied DNNs to learn agent strategies, which guide agents to interact with the environment. Earlier DRL methods like the deep Q network (DQN) model developed by DeepMind (2013, 2015) (Mnih et al., 2013; Mnih et al., 2015) which combines deep learning with Q-Learning (1992) (Watkins and Dayan, 1992) not only surpassed human-level performance but also outperformed many reinforcement learning methods. The deep deterministic policy gradient (DDPG) model (2015) (Lillicrap et al., 2015) uses off-policy data and the Bellman equation to learn the Q value, and uses the Q-function to learn the policy. The benefit of DRL methods is that it avoids the chaos and potential confusion of manually designed differential equations of each game scenario. Once the game scenario is set up, deep reinforcement learning methods allow us to directly acquire the kinematic strategy of agents end to end. This allows us to directly obtain the pursuit and evasion policy in a much easier way.

This study explores the use of DRL methods for guiding ignorant sheep to escape from the circle, rather than a handcraft for each game scenario. Powerful as the DQN model may seem, adjustments and refinements are still required when implementing it in various game scenarios such as the experience replay and the carefully designed reward mechanism. In this study, a delicate reward mechanism is applied to help train the DQN model. According to the cost function in the differential pursuit–evasion game (Yong, 2014), a time-out reward strategy is introduced to help agents obtain the optimal policy that minimizes the cost function. Due to its discrete action space, it is not practical to let the sheep choose the exact steering angle in the dog sheep game with limited computing power. Therefore, an attenuation mechanism of the steering angle is proposed based on the kinematic theory of the dog sheep game. This reduces the action space dimension significantly without compromising the chance of evasion for the sheep too much. Another way to overcome the defect of the discrete action space is to adopt different methods. This study adopts the powerful deep deterministic policy gradient (DDPG) model, which is actor-critic based and model-free. When simulating the dog sheep game, the model is trained with no knowledge in advance. Generally speaking, we mainly focus on two DRL methods to this end, DQN and DDPG, each with a different rationality. Later, we make some adjustments and refinements to improve the performance of DRL methods.

This work addresses the aforementioned research problems by implementing DRL methods for pursuit–evasion problems. Considering the dog sheep game is set in a constrained environment, the first problem this study addresses is solving the kinematic pursuit and evasion policies based on the differential pursuit–evasion game theory. DRL has shown its powerful performance in games, and the question is whether it is possible to endow the sheep the ability to escape. Therefore, the second problem is to set up a reinforcement learning environment and implement appropriate DRL methods to this particular differential game. However, some DRL methods may need adjustments when being applied to various scenarios. Hence, the third problem is to optimize the performance of DRL methods with the idea from the differential games theory. We summarize the contributions of this work as follows:

- For the first problem, by finding the equilibrium point in the dog sheep game, this study successfully establishes the kinematic pursuit and evasion policies.
- For the second problem, a delicate reward mechanism is introduced according to related theories of differential pursuit–evasion games.
- For the third problem, due to the defect of DQN whose the action space is discrete, an attenuation mechanism of the steering angle is proposed.
- By quantifying the environment difficulty, this study evaluates the performance and learning ability of our refined DQN model and DDPG model and other baseline models.

The rest of the article is organized as follows. In **Section 3**, the mathematical form of differential pursuit–evasion game theory is introduced. The kinematic pursuit and evasion strategy is solved in **Section 4**, while the implementation of DRL methods for evasion policy learning is elaborated in **Section 5**.

2 RELATED WORK

Researchers have studied the implementations of DRL methods for differential pursuit–evasion games and improved them in many aspects. Isaacs (1965) first proposed the theory of differential games. In his book, Isaacs proposed a classic "homicidal chauffeur" problem which is a classic differential pursuit–evasion game. In this game, a slow but maneuverable pedestrian is against a driver with a faster but less maneuverable vehicle, attempting to run over the pedestrian. Merz (1971) presented the complete solution for the "homicidal chauffeur" game. Another classic game scenario is the game in constrained environments. For example, agents in Sundaram et al.'s (2017) study are constrained to road networks. The control policies of our research are based on the kinematic pursuit and evasion game theory. Apart from differential games, kinematic analysis is applied to various research areas, such as robotic control (2021) (Liu et al., 2021a; Liu et al., 2021b; Xiao et al., 2021; Zhao et al., 2022).

Deep neural networks have shown their advantages over traditional methods in multiple research areas. For example, they have been adopted in semantic analysis (2021) (Chen et al., 2021a; Chen et al., 2021b; Jiang et al., 2021b; Chen et al., 2021c) and image recognitions (2021) (Hao et al., 2021; Jiang et al., 2021a; Yang et al., 2021). The DRL method, which combines DNNs and reinforcement learning, was first proposed by DeepMind (2013, 2015) (Mnih et al., 2013; Mnih et al., 2015). Their method called DQN is the combination of Q-learning and deep neural network. It shows excellent performance in the Atari

game. The emergence of the DQN leads to a number of similar research studies based on DRL methods. Shao et al. (2019) surveyed the application of DRL methods in video games. Value-based, policy gradient, and model-based DRL methods are applied to various video games such as Atari, Minecraft, and StarCraft. For example, Wei et al. (2018) employed convolutional neural networks trained with refined DQN to play the snake game. DRL has been a long-standing research area when it comes to artificial intelligence in differential games. Jiang et al. (2020) introduced an approximate soft policy iteration-based reinforcement learning method which used a value neural network to provide cooperative policy for two pursuers versus an evader. Lin (1992) proved that the experience replay helps the network train faster and smoother.

To overcome the defect of the discrete action space, researchers hypothesized several methods with a continuous action space. The actor-critic (A3C) method is utilized in many related research studies. For example, Perot et al. (2017) adopted the A3C with CNN + LSTM as the state encoder to play racing games. Lixin Wang et al. (2019) used a fuzzy deterministic policy gradient algorithm to obtain the specific physical meaning for policy learning in a pursuit–evasion game. Lillicrap et al. (2015) first introduced the DDPG methods with the continuous action space. Maolin Wang et al. (2019) implemented the DDPG model to an open pursuit–evasion environment to learn the control strategy. Several researchers (Lowe et al., 2020; Singh et al., 2020; Wan et al., 2021) proposed an actor-critic multi-agent DDPG algorithm to preprocess actions of multiple agents in the virtual environment.

3 PRELIMINARY

This article focuses on the dog sheep game, in which the dog is the pursuer and the sheep is the evader. The game takes place in a circle. The dog is only allowed to pursue inside the circle, while the sheep is randomly born inside the circle. The distance between the sheep and the center of the circle is non-decreasing; otherwise, it would cause confusion when solving the kinematic evasion strategy. The objective of the sheep is to obtain an evasion strategy that maximizes its chance of escaping, while the objective of the dog is exactly the opposite.

The dog sheep game perfectly fits the differential game theory (1992) (Lin, 1992). In a differential pursuit–evasion game, the pursuer and the escaper have their own strategy and target. The following $x, X()$ each refers to the state and the state trajectory; α_1, A_1, u_2, U_2 each represents the strategy used and the strategy set for the pursuer and the escaper; and t and $\mathcal{T}(x, \alpha_1)$ are the game time and the capturing time of a game, respectively.

$$\begin{cases} \dot{X}(t) = f(X(t), \alpha_1[u_2](t), u_2(t)) & t \in [0, \infty) \\ X(0) = x & x \in R^n \end{cases}. \tag{1}$$

For the pursuer, let M be a moving target set. For any $(t, x) \in R^+ \times R^n$, the objective is to find an $\alpha_1 \in A_1$ such that for any $u_2 \in U_2$,

$$X(t; x, \alpha_1(u_2), u_2) \in M(t) \quad t \in [0, \mathcal{T}(x, \alpha_1)]. \tag{2}$$

In an actual pursuit game situation, the pursuer expects the expression to be true. The game can be defined in five different levels: capturable, locally capturable, globally capturable, small time locally capturable (STLC), and small time globally capturable (STGC). We define

$$\begin{cases} P(t; M) = \{x \in R^n | \exists \alpha_1 \in \mathcal{A}_1, T(x, \alpha_1) \le t\}, \\ P(M) = P(\infty; M), \\ M \subseteq P(M). \end{cases} \tag{3}$$

Then the capturability can be described as follows (Yong, 1986):

$$\begin{cases} locally\ capturable \Leftrightarrow \exists \mathcal{O}(M) \subseteq P(\varepsilon; M), \\ globally\ capturable \Leftrightarrow R^n \subseteq P(M), \\ STLC \Leftrightarrow \forall \varepsilon > 0,\ \exists \mathcal{O} \subseteq P(\varepsilon, M), \\ STGC \Leftrightarrow \forall \varepsilon > 0,\ R^n \subseteq P(\varepsilon; M). \end{cases} \tag{4}$$

For any $x \in R^n$, $a_1 \in A_1$, and $u_2 \in U_2$, the minimum terminating time $T(x)$ can be defined as follows:

$$\begin{cases} T(x; \alpha_1[u_2], u2) = \inf\{t \ge 0 |\ \mathrm{d}(X(t; x, \alpha_1[u_2], u_2), M) = 0, \\ \mathcal{T}(x) = \inf_{\alpha_1 \in \mathcal{A}_1} \sup_{u_2 \in U_2} T(x; \alpha_1[u_2], u_2). \end{cases} \tag{5}$$

Then, we introduce the cost function $J(x; \alpha_1[u_2], u_2)$:

$$J(x; \alpha_1[u_2], u_2) = \int_0^{T(x; \alpha_1[u_2], u_2)} e^{-s} ds = 1 - e^{-T(x; \alpha_1[u_2], u_2)} \tag{6}$$

The projective of the pursuer in the different pursuit games is to minimize the terminating time and the cost function when the game is capturable.

Symmetrically, the problem for the evader can be proposed in the following way:

$$\begin{cases} \dot{X}(t) = f(X(t), u_1(t), \alpha_2[u_1](t)) & t \in [0, \infty), \\ X(0) = x & x \in R^n. \end{cases} \tag{7}$$

Considering M as the terminating set and $R^n \backslash M$ as the survival set. For any $x \in R^n \backslash M$ and any $u_1 \in U_1$, find an $\alpha_2 \in A_2$ that makes the following expression true:

$$X(t; x, u_1, \alpha_2[u_1]) \notin M \quad t \in [0, \infty) \tag{8}$$

Correspondingly, the ability to evade (Lin, 1992) can be summarized as evadable and uniformly evadable. If a game is considered to be evadable from M, for any $u_1 \in \mathcal{U}_1$ and $x \in R^n \backslash M$, there exists an $\alpha_2 \in \mathcal{A}_2$. For those games to be uniformly evadable from M, there exists a $\delta > 0$ and an $\alpha_2 \in \mathcal{A}_2$, the following expression holds:

$$d(X(t; x, u_1, \alpha_2[u_1]), M) \ge \delta, \quad \forall t \ge 0,\ u_1 \in \mathcal{U}_1. \tag{9}$$

In the evasion circumstances, the minimum terminating time and the cost function have similar definition to those in pursuit games. The purpose of the evader is to find out an α_2 in the evadable game with less terminating time and cost.

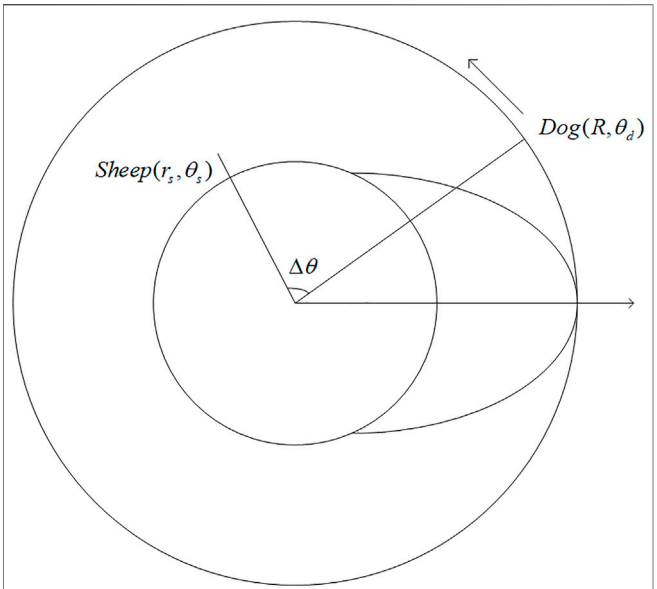

FIGURE 1 | Diagram of the dog's pursuit strategy.

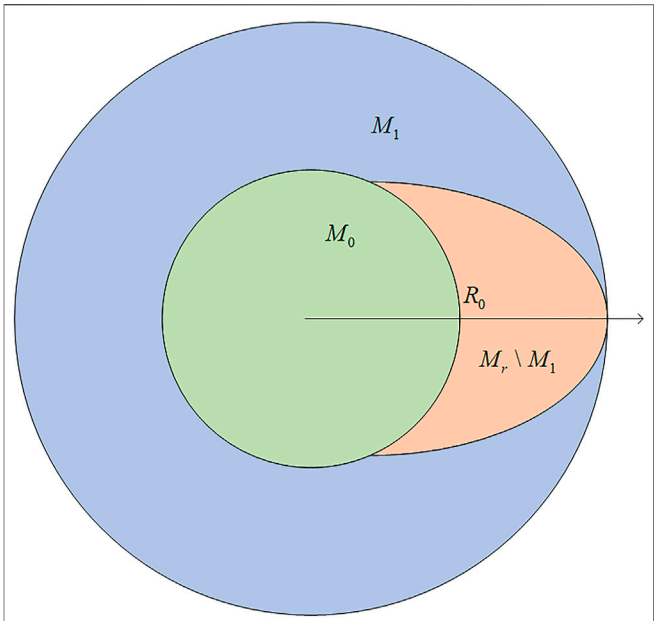

FIGURE 2 | Sheep divided into three parts based on its relative position.

$$T(x;u_1,\alpha_2[u_1]) = \inf\{t>0 X(t;x,u_1,\alpha_2[u_1]) \in M\},$$
$$J(x;u_1,\alpha_2[u_1]) = \int_0^{T(x;u_1,\alpha_2[u_1])} e^{-s}ds = 1 - e^{-T(x;u_1,\alpha_2[u_1])}. \quad (10)$$

The idea of the cost function is to minimize the cost of the evader, provided it can escape successfully. Later in this article, a reward mechanism shares the similar philosophy with the cost function here.

4 MODELING AND KINEMATIC ANALYSIS OF DOG SHEEP GAME

4.1 Kinematic Analysis of Dog Pursuit Game

In the dog sheep game, the game environment is set to be a circle of radius R in which the dog runs at a constant speed of V_d. The sheep also runs at a constant speed of v_s, while its original position is set to be randomly distributed inside the circle. For convenience, given the dog and the sheep coordinates of (R_d, θ_d) and (r_s, θ_s), $(R_d = R, r_s \in [0, R), \theta_d, \theta_s \in [0, 2\pi))$. In case if the sheep aborts its attempt to escape, the distance between the sheep and the center of the circle r_s is constrained to be monotonically non-decreasing. It is assumed that the dog and the sheep both have the ability to adjust their heading directions without losing any speed. The game terminates as soon as the sheep gets caught or successfully escapes.

Under the aforementioned conditions, this section concentrates on the optimal pursuit strategy for the dog that maximizes its chance of successful pursuit and the kinematic evasion policy for the sheep, provided the dog adopts the optimal tactics. When the sheep escapes from the circle, the quickest way is to run outward along the radius where it is located. Hence, the ideal situation for the dog is to arrive at the intersection of the certain radius and the circle (R, θ_s) sooner than the sheep does, supposing that the sheep does not adjust its heading direction, which is not often the case. Therefore, the heading direction of the dog may change dynamically, but it is always the one that reduces the inferior angle $\Delta\theta$ formed by the radii where the dog and the sheep are located. Thus, the pursuit policy α_1 for the dog can be summarized as follows:

$$\begin{cases} \theta_s - \theta_d \in (-2\pi, -\pi] \cup (0, \pi] & \textit{The dog runs clockwise,} \\ \theta_s - \theta_d \in (-\pi, 0) \cup (\pi, 2\pi) & \textit{The dog runs anticlockwise,} \\ & \textit{otherwise Stay where it is.} \end{cases} \quad (11)$$

From the perspective of differential games, the dog's kinematic pursuit strategy α_1 simply depends on the relative position of the sheep and the dog. **Figure 1** portrays the pursuit strategy for the dog.

Considering the influence of the relative position of the sheep on the pursuit strategy, the capturability of the game needs to be discussed based on different origin positions of sheep. This inferior angle $\Delta\theta$ is determined by the angular velocity of the dog and the sheep, each being represented as ω_d and ω_s, respectively. The closer the sheep is to the circle, the slower will be its angular velocity ω_s. Correspondingly, ω_s may go to infinity as long as the sheep is close enough to the center of the circle $(r_s \to 0)$. Thus, there exists an r_0 that meets the condition $\omega_s = \omega_d$ if $r_s = r_0$. The set $M_0 = \{(r_s, \theta_s) \mid 0 \le r_s < r_0\}$, which is the green part in **Figure 2**, and the sheep located in it is capable to run at a larger angular velocity. This means the sheep has the ability to adjust $\Delta\theta$ as it wishes. Supposing the sheep has infinite physical power, it could run to the place with the greatest chance of escaping and then start running outward. So, when the sheep is born inside the circle of radius r_0, the game is capturable

when the sheep has no way of escaping if the following expression holds:

$$\frac{R-r_0}{v_s} \geq \frac{\pi R}{V_d}. \tag{12}$$

This means the sheep could not escape even at the place with the greatest chance of escaping $\{(r_s, \theta_s) \mid |\theta_s - \theta_d| = \pi\}$.

Running inside the ring defined as M_r formed by the outside circle and the circle of radius r_0, the sheep's angular velocity ω_s is smaller than ω_d. Considering the condition in which sheep is born inside this ring, there exists a set where any sheep located in it has no chance of escaping as long as the dog adopts the optimal pursuit strategy. We define this set as M_1 and it can be represented as follows:

$$M_1 = \left\{ (r_s, \theta_s) r_0 \leq r_s < R, \ |\theta_s - \theta_d| \leq \frac{(R_d - r_s)V_d}{v_s R_d} \right\}. \tag{13}$$

The game is capturable to the pursuer when sheep is located in M_1, which is the red part in **Figure 2**. For the sheep located in $M_r \backslash M_1$, which is the blue part in **Figure 2**, there exist some strategies that make it impossible for the dog to chase it down.

Figure 2 portrays the sheep in different parts of the circle to help better analyze the capturability of the game.

4.2 Kinematic Modeling of Sheep Evasion Game

In this part of the article, we focus on the kinematic evasion model of sheep in the dog sheep game under the circumstances that the dog adopts the optimal strategies to chase down the sheep. As discussed earlier, kinematic analyses for the sheep located in M_0, M_1, and $M_r \backslash M_1$ should be separately conducted.

4.2.1 Sheep Located in M_0

For the sheep whose initial states x are in M_0, its kinematic evasion models can be uniformly discussed, since its relatively larger angular velocity endows it the ability to adjust $\Delta\theta$ and r_s. This simplifies the analysis of game's ability to evade and sheep's evasion strategy from M_0 to those from $\{(r_s, \theta_s) \mid r_s = r_0\}$. As discussed before, the game is evadable if the following expression holds:

$$\frac{R-r_0}{v_s} < \frac{\pi R}{V_d}. \tag{14}$$

Assuming the game is evadable from M_0, the strategy for sheep is to utilize its advantage to the dog in angular velocity to change the angle $\Delta\theta$ until the following evasion condition is satisfied:

$$\frac{R-r_s}{v_s} < \frac{|\theta_s - \theta_d| R}{V_d} \quad r_s \in [0, r_0]. \tag{15}$$

Once the aforementioned expression holds, the evasion strategy for the sheep is to run straight outward.

4.2.2 Sheep Located in M_1

Next, we focus on the sheep born inside M_1. Theoretically speaking, it has no chance of escaping, provided the dog makes no mistake, that is, for any $\alpha_2 \in \mathcal{A}_2$, $x \in M_1$, there exists an $u_1 \in U_1$, and $M_1 \cap (R_n \backslash M) = \varnothing$ holds. This means the game is not evadable from M_1.

4.2.3 Sheep Located in $M_r \backslash M_1$

The only one scenario remains is when the sheep is born inside $M_r \backslash M_1$. In this scenario, the sheep has the opportunity to escape successfully in many different ways, among which running outward along the radius where it is located is the quickest. In the meantime, the intersection of the circle and the certain radius $E_0(R, \theta_{E_0})$ is defined. There exists an open neighborhood $(\theta_{E_0^-}, \theta_{E_0^+})$ of θ_{E_0} such that for the sheep running straight toward the point inside this neighborhood, the dog would not be able to hunt it down. Define $E_0^-(R, \theta_{E_0^-})$, $E_0^+(R, \theta_{E_0^+})$, and the distance between the sheep and E_0^-, E_0^+ as $d_{sE_0^-}$, $d_{sE_0^+}$ for convenience. Specific details are shown in **Figure 3**.

We are able to solve d_{sE^-} and d_{sE^+} first and then $\theta_{E_{0^-}}$ and $\theta_{E_{0^+}}$:

$$\begin{cases} d_{sE_0^-} = \sqrt{r_s^2 \sin^2\left(\theta_s - \theta_{E_0^-}\right) + \left(R - r_s \cos\left(\theta_s - \theta_{E_0^-}\right)\right)^2} \\ d_{sE_0^+} = \sqrt{r_s^2 \sin^2\left(\theta_s - \theta_{E_0^+}\right) + \left(R - r_s \cos\left(\theta_s - \theta_{E_0^+}\right)\right)^2} \end{cases}. \tag{16}$$

By definition, if the sheep runs straight toward E_0^- and E_0^+, the dog would catch it just in time. Therefore, we have

$$\begin{cases} \frac{d_{sE_0^-}}{v_s} = \frac{\left|\theta_d - \theta_{E_0^-}\right| R}{V_d} \\ \frac{d_{sE_0^+}}{v_s} = \frac{\left|\theta_d - \theta_{E_0^+}\right| R}{V_d} \end{cases} \Rightarrow \begin{cases} d_{sE_0^-} = \frac{\left|\theta_d - \theta_{E_0^-}\right| v_s R}{V_d} \\ d_{sE_0^+} = \frac{\left|\theta_d - \theta_{E_0^+}\right| v_s R}{V_d} \end{cases}. \tag{17}$$

So, the polar coordinates of E_0^- and E_0^+ can be solved as follows:

$$\begin{cases} \theta_{E_0^-} = \theta_d - sgn(\sin(\theta_s - \theta_d)) \cdot \frac{V_d \sqrt{r_s^2 \sin^2\left(\theta_s - \theta_{E_0^-}\right) + \left(R - r_s \cos\left(\theta_s - \theta_{E_0^-}\right)\right)^2}}{v_s R} \\ \theta_{E_0^+} = \theta_d + sgn(\sin(\theta_s - \theta_d)) \cdot \frac{V_d \sqrt{r_s^2 \sin^2\left(\theta_s - \theta_{E_0^+}\right) + \left(R - r_s \cos\left(\theta_s - \theta_{E_0^+}\right)\right)^2}}{v_s R} \end{cases}. \tag{18}$$

For the sheep whose initial position is $M_r \backslash M_1$, if it does not run straight to the inferior arc formed by E_0^- and E_0^+, the evasion time is limited to the open interval $(\frac{R-r_s}{v_s}, min(\frac{d_{sE_0^-}}{v_s}, \frac{d_{sE_0^+}}{v_s}))$ or it would miss the chance of escaping.

5 IMPLEMENTATION OF DEEP REINFORCEMENT LEARNING METHODS FOR DOG SHEEP GAME

5.1 Technical Foundation of Deep Reinforcement Learning Methods

Deep reinforcement learning (DRL) has shown its advantages in various games over the past few years, especially since DeepMind creatively introduced DQN (2013) (Jiang et al., 2021a), which combines reinforcement learning and deep learning. Recently,

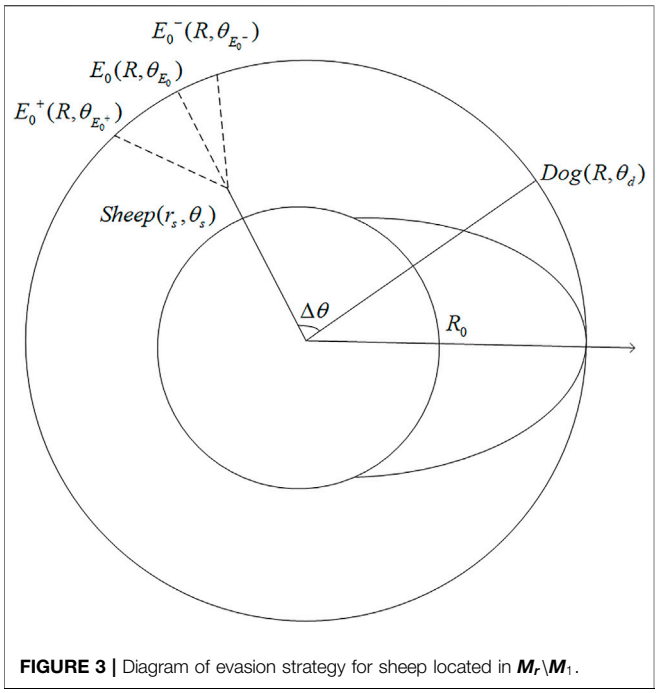

FIGURE 3 | Diagram of evasion strategy for sheep located in $M_r \backslash M_1$.

many similar research studies on implementations of DRL for different game scenarios emerged.

DQN was initially presented by DeepMind (2013) (Jiang et al., 2021a) to play the Atari video game, and it successfully learned the control policy and remarkably outperformed human-level performance. Excellent as DQN may seem, adjustments are still required when it is applied to other game scenarios such as the design of reward systems and the adoption of replay buffers (2018) (Wei et al., 2018).

As a value-based method, it evaluates each possible action at certain stages in the game based on the Q-value. When an agent is at state s_t and takes an action a_t, $Q(s_t, a_t)$ is used to estimate the value at this certain time step t. Then it chooses the suitable action based on the action select strategy. After the execution of action a at state s, the reward is r_t, and the game goes to the next state s_{t+1}. In order to enable the neural network to learn from experiences, the experience replay is introduced. A four-tuple (s_t, a_t, r_t, s_{t+1}) is stored in the experience replay buffer, and the network would sample and learn from it.

5.2 Implementation of Deep Q Network for Dog Sheep Game

In this subsection, we propose DQN to the dog sheep game. Technical details and model optimizations will be elaborated as well.

5.2.1 Environment Settings

Environment settings are of great significance since the environment is the one the agent interacts with. To simulate the dog sheep game, our game environment consists of basic parameters, initializations, and environment updates.

5.2.1.1 Basic Parameters and Initializations

In the dog sheep game environment, the radius of the circle R is set to be 200. Since determining whether the game terminates requires judging whether they have physical contact or not, it would be inappropriate to consider both sides as particles. Apart from the sizes of both agents, their running speed should be specified as well. For relatively bigger sizes or a smaller dog sheep speed ratio, the difficulty for sheep to evade increases considerably. Hence, proper sizes and speed should be initialized to prevent the environment from being partial to one side in the pursuit–evasion game. After a number of experiments and simulations, we set the size of both agents as 10 and the dog speed $V_d = 16$ to make sure each episode ends at around 11 steps. The speed of the sheep is determined by the speed ratio $\eta = \frac{V_d}{v_s}$, which is a key parameter to evaluate the environment difficulty. The exact value of η will be given later.

5.2.1.2 Environment Updates

As the simulation proceeds, whether the game is terminated should be checked every time the game enters a new time step. Then the sheep and the dog could take actions resulting in the changes of their positions if the game is not finished. Their next positions can be deduced using their current positions and their heading directions.

As it is discussed previously, the dog's heading direction is decided by its capture strategy, while the sheep's heading direction is something to be learned in the DRL model. For DQN, the action space is discrete. If we allow the sheep to adjust its deflection angle in degree, this means there would be 181 different possible actions at each time step. This will lead to the exponential growth of the requirement for computational power. Therefore, we adjust the action space and set it to be two dimensional and come up with an attenuation mechanism of the deflection angle. The sheep in the game is only left with the choice to turn left or right, while the steering angle $\theta_{sa} \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ is determined by a function set in advance. Given the fact that ω_s decreases as r_s grows and equals to ω_d when $r_s = r_0$, it may be quite costly if the sheep chose to make a 90° turn when it is close to the circle of the radius R. For the sheep inside or on the circle of radius r_0, the value of θ_{sa} has no negative influence on their chances of escaping. However, due to the fact that ω_s is at a disadvantage when $r_s \in (r_0, R)$, the sheep should make its escape as soon as possible, which means θ_{sa} should be relatively small. As discussed previously, we introduce the steering angle function as follows:

$$\theta_{sa} = \frac{\pi}{2} \cdot \underbrace{\left(1 - \frac{r_s}{R}\right)}_{\text{Distance discount}} \cdot \underbrace{|\cos(0.5(\theta_s - \theta_d))|}_{\text{Angle discount}}. \tag{19}$$

The attenuation mechanism of the deflection angle consists of a distance discount and an angle discount. As the sheep runs outward, its maximum steering angle is restricted by the distance discount. This prevents the sheep from making sharp turns at unsuitable locations. However, it would be unnecessary for the sheep that already is an ideal initial position to turn its heading direction. So the angle discount is introduced to prevent that from happening. When the sheep is at an ideal position where $|\theta_s - \theta_d|$

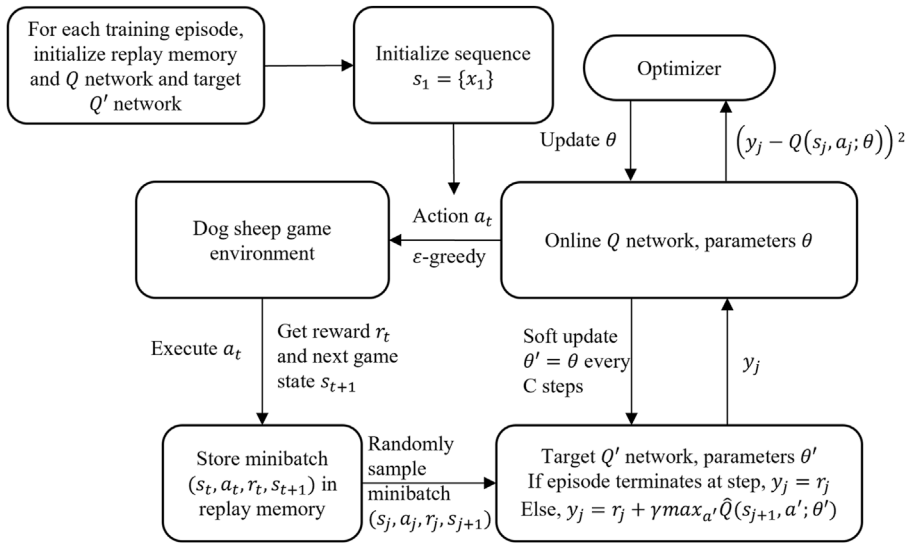

FIGURE 4 | Flowchart of DQN training.

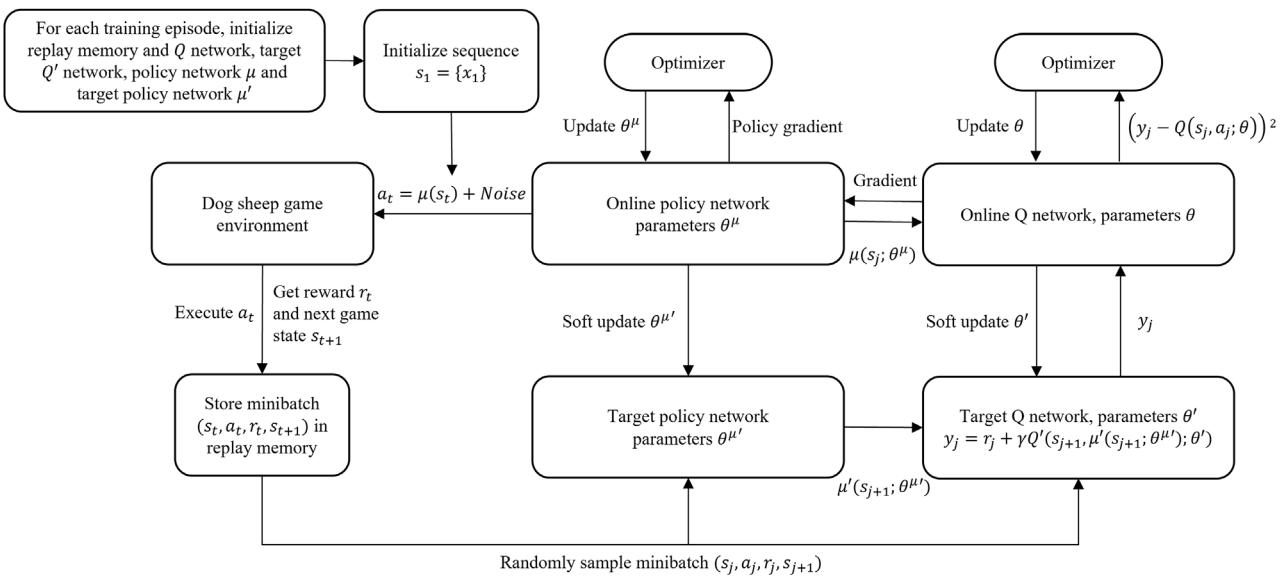

FIGURE 5 | Flowchart of DDPG training.

like π, the angle discount is 0. This means the sheep does not have to change its heading direction at all. The angle discount is 1 when $|\theta_s - \theta_d|$ is 0 or 2π, resulting in a relatively large steering angle.

5.2.2 Reward Mechanism

Agents in reinforcement learning models interact with environments, striving for high rewards. Therefore, a carefully designed reward mechanism is required when DQN is applied to a new scenario. A poorly designed one may lead to ineffectiveness and incapacity to learn the optimal strategy. The dog sheep game is also a zero-sum game, thus escaping successfully is of utmost significance to the sheep. Whenever the sheep escapes successfully, the terminate reward of 10 is given. On the contrary, once the sheep gets caught, the reward would be −10. This basic reward mechanism encourages the sheep to find a way to escape from the circle without being captured and win the pursuit–evasion game.

In the evasion game theory, the cost function is $J(x; u_1, \alpha_2[u_1]) = \int_0^{T(x;u_1,\alpha_2[u_1])} e^{-s} ds$. As time passes by, the cost increases correspondingly. Apart from the terminate reward, another reward should be given in each step to help minimize the evasion cost. Therefore, we have come up with a time-out reward strategy. If the sheep stays in the circle for too long, it will be given a negative reward. This reward should take the different origin locations into consideration. The sheep that is close to the circle needs less time for evasion than the sheep

located around the center of the circle. So the time-out reward strategy is designed based on the origin location and the current time step.

$$R_{to} = \begin{cases} \rho log_{\frac{v_s}{\nabla_d}}\left(t - \frac{R - r_s}{v_s}\right) & t - \frac{R - r_s}{v_s} > 1, \\ 0 & else. \end{cases} \tag{20}$$

When the time is still within $\frac{R-r_s}{v_s}$, which is the minimum evasion time for the sheep, there will be no reward given. Once the time exceeds the minimum evasion time, the negative reward is given and will increase as time passes by. Adding up this negative reward effectively prevents the sheep from lingering in the circle.

5.2.3 Model Architecture

DQN has shown its capability to train the agent to play video games. It trains the agent to learn through the Q value throughout the training episode. Its structure is displayed in **Figure 4**.

To observe the game state, an MLP works as the state encoder. At each time step, based on the current game state s_t observed, the neural network, whose parameters are randomly initialized, is able to calculate the Q value of every possible action and therefore come out of the best action with the highest potential rewards. Among all the actions, the agent chooses one of them based on the ε-greedy strategy. This means the agent would go for the action with the highest Q value at the possibility of $1 - \varepsilon$, while one of the remaining actions is chosen randomly and is executed at the possibility of ε. Introducing the ε-greedy strategy helps the agent learn from those actions that seem less attractive but may lead to higher rewards in the long run. This is of immense help at the threshold of each training episode. The pseudo code of DQN is shown in Algorithm 1.

Algorithm 1. Deep Q-learning Network.

Initialize replay buffer D of capacity N, function Q with random weights ω, target function Q^* with weights $\omega^* = \omega$

While training not complete

  Initialize state s_1 and preprocessed sequence $\phi_1 = \phi(s_1)$

  For t=1, do

    With probability ε select a random action a_t,

    otherwise choose $a_t = argmax_{a \in A} Q(\phi(s_t), a, \omega)$.

    Decay ε,

    Execute a_t in game and observe r_t and s_{t+1}.

    Preprocess $\phi_{t+1} = \phi(s_{t+1})$

    Store $(\phi_t, a_t, r_t, \phi_{t+1})$ in D

    Sample random minibatch $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

    If game terminate at step $j + 1$ then

      Target value $y_j = r_j$

    Else: $y_j = r_j + \gamma max_a Q^*(\phi_j, a_j; \omega)$

    Perform optimization algorithm Adam on loss $\left(y_j - Q(\phi_j, a_j; \omega)\right)^2$

    Every C step reset $Q^* = Q$

  End

End

Once the action a_t is selected, the reward r_t is given and the next game state s_{t+1} is observed. A four-tuple (s_t, a_t, r_t, s_{t+1}) is stored in the experience replay buffer. The optimizer takes a random sample from the replay buffer and utilizes it to update the parameters of the online Q network. This endows the agent the ability to learn from the valuable experience and helps optimize the learning process. Every several steps, the parameters of target Q network get a soft update from the online Q network.

5.3 Implementation of Deep Deterministic Policy Gradient for Dog Sheep Game

Compared with DQN, the action space of agents in DDPG is continuous. As discussed before, although the steering angle is carefully considered and designed with a distance discount and an angle discount, the action space is discrete after all. The discrete action space with only 2 choices is just a compromise to insufficient computational power. It is always better to have a continuous action space.

DDPG has an actor-critic architecture, where the actor network optimizes its policy and the critic network estimates the Q-value for current policy. The training process is displayed in **Figure 5**. The online Q network estimates the Q-value based on the sampled four-tuple (s_t, a_t, r_t, s_{t+1}). Like DQN, the experience replay and the reward mechanism work similarly in DDPG.

While the Q network updates its parameters, the online policy network, which is an MLP, updates its parameters at the same time. Every couple of steps, the parameters of both target Q and policy networks each gets a soft update from the online Q and policy networks. The details of DDPG are summarized in the Algorithm 2.

Algorithm 2. Deep Deterministic Policy Gradient (DDPG).

Initialize replay buffer D of capacity N, function Q with random weights ω, target function Q^* with weights $\omega^* = \omega$, actor μ with random weights ξ, target function with μ^* weights $\xi^* = \xi$

While training not complete

  Initialize state s_1

  For t=1, do

    Select action $a_t = \mu(s_t, \xi) + N_t$ according to the current policy and exploration noise N_t

    Execute a_t in game and observe r_t and $

    Store (s_t, a_t, r_t, s_{t+1}) in D

    Sample random minibatch (s_j, a_j, r_j, s_{j+1}) from D

    Set $y_j = r_j + \gamma Q^*(s_{j+1}, \mu^*(s_{j+1}; \xi^*); \omega^*)$

    Update critic by minimizing the loss: $\frac{1}{N}\sum_i \left(y_j - Q(s_i, a_i; \omega)\right)^2$

    Update the agent policy using the sampled policy:

$$\nabla_\xi J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\omega)\, \nabla_\xi \mu(s|\xi)$$

    Update the target networks:

$$\omega^* \leftarrow \tau\omega + (1 - \tau)\omega^*$$

$$\xi^* \leftarrow \tau\xi + (1 - \tau)\xi^*$$

  End

End

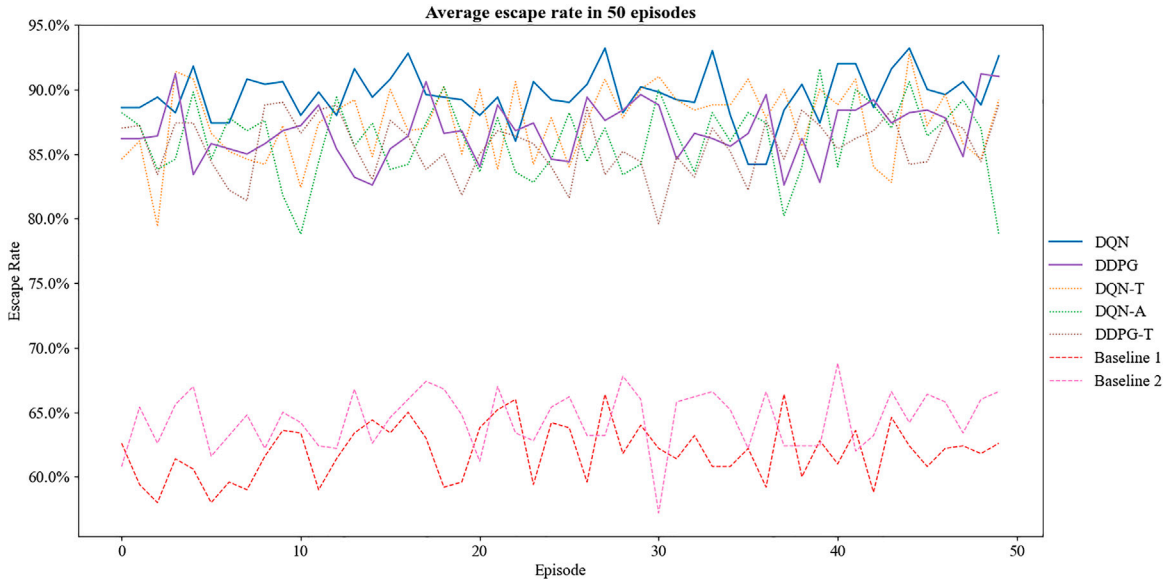

FIGURE 6 | Average escape rate in 50 episodes.

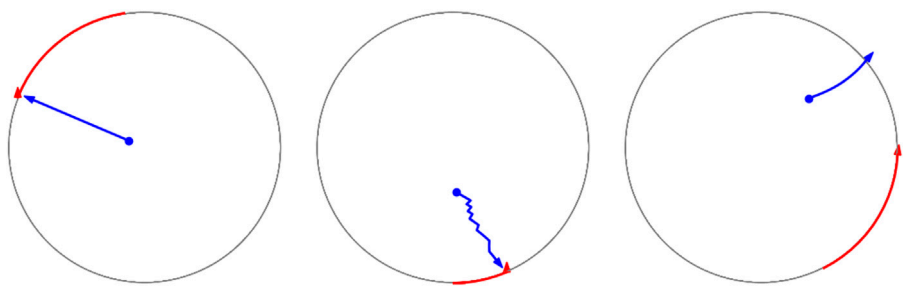

FIGURE 7 | Trajectories of sheep. **(A)** From left to right, each represents the trajectory of sheep from the baseline models 1 and 2 and the DRL model. **(B)** Blue curves are the trajectories of sheep, while the red ones are those of dogs.

5.4 Learning Ability Evaluation

The implementation of the aforementioned methods is done when the speed ratio is 2 and the size of both agents is 10. This section focuses on the learning abilities of DRL methods under different environment difficulties. Among all the parameters, we choose the escape probability of baseline model and the escape rate improvement to quantify the environment difficulty and the learning ability. In this scenario, the size is 10 and the dog speed V_d is 16. Sheep speed v_s fluctuates according to the speed ratio η. If the game is evadable, the speed ratio η has

$$\frac{\pi R}{V_d} > \frac{R - r_0}{v_s}, \tag{21}$$

$$\frac{R}{V_d} = \frac{r_0}{v_s}, \tag{22}$$

$$\eta = \frac{V_d}{v_s} < \pi + 1. \tag{23}$$

This article only focuses on the performance of DRL methods while speed ratio is $\eta \in (1, \pi + 1)$. Each speed ratio η corresponds to an escape probability. A baseline model is introduced to help quantify the environment difficulty. The sheep in this baseline model has no evasion policy and runs straight outward regardless of the condition. After implementing the DRL methods under various environment difficulties, the escape rate improvement is utilized to evaluate their learning ability.

6 DEEP REINFORCEMENT LEARNING TRAINING RESULTS

When implementing the DQN model, an MLP of three layers with game states as input is adopted to evaluate the Q value of each action. ε in the ε-greedy strategy decays linearly from 0.9 to 0.01, which allows the agent to attempt various actions when it is relatively ignorant and gradually prefers the action with the highest potential reward. The experiencing replay buffer is

TABLE 1 | Models adopted in our experiments.

Type	Model	Characteristic
Models with discrete action space	DQN-T	DQN model without the time-out reward strategy
	DQN-A	DQN model without the attenuation mechanism of the deflection angle
	Refined DQN	Refined deep Q network model
	Baseline 1	Sheep run straight outwards
Models with continuous action space	DDPG	Deep deterministic policy gradient
	DDPG-T	DDPG model without the time-out reward strategy
	Baseline 2	Sheep take random deflection angle

TABLE 2 | Average reward of different models.

Model	Average reward
Refined DQN	7.3601
DDPG	7.2507
DQN-T	6.6804
DQN-A	5.7600
DDPG-T	6.8300
Baseline 1	1.8237
Baseline 2	2.9427

set to be 10,000. The discount factor in Algorithm 1 is set to be 0.95.

In our experiment, we introduce several baseline models whose key characteristics are displayed in the following **Table 1**. The DQN-T, DDPG-T, and DQN-A model are tested to evaluate the effectiveness of the time-out reward strategy and the attenuation mechanism.

For the DQN models and the DDPG models, the number of training episodes is set to be 1,500 and that of simulation episodes is 500. The dog speed is 16, and the sheep speed is 8. Experiments show that the DDPG and DQN models both outperform the baseline models. Compared to the two baseline models, all DRL methods improve the escape rate by a large margin of about 50%. The escape rate of each model in 50 rounds of simulation with various random speed is plotted in **Figure 6**.

It can be seen that the sheep trained by the refined DQN model has a greater chance of escaping than the DQN-T and DQN-A models. The DDPG model also achieves a higher escape rate than the DDPG-T model. This indicates the effectiveness of the time-out reward strategy. It is illustrated that the modifications and adjustments effectively increase the success rate of evasion and the performance of the DQN model. To better illustrate the differences in the DRL methods and the baseline models, the trajectories of sheep during their evasion process are shown in **Figure 7**. From left to right, each represents the trajectory of sheep from the baseline models 1 and 2 and the DRL model.

In addition, the performance difference between the refined DQN model and the DDPG model is quite narrow. Yet generally speaking, the refined DQN model outperforms the DDPG model by a small margin.

To better evaluate the performance of the aforementioned models, the average reward in 50 rounds of simulation is displayed in **Table 2**. Unsurprisingly, the average reward of each model corresponds to the average escape rate. Compared to the time-out reward strategy, the attenuation mechanism of the deflection angle causes greater enhancement for the DQN and DDPG models. This attenuation mechanism helps the DQN model achieve even better performance than the DDPG model.

Furthermore, we manage to evaluate the learning abilities of the aforementioned DRL methods. For speed ratio η taken at the interval of 0.5 from 1.5 to 4, the average improvement of escape probabilities in 50 rounds of simulation is shown in **Figure 8**.

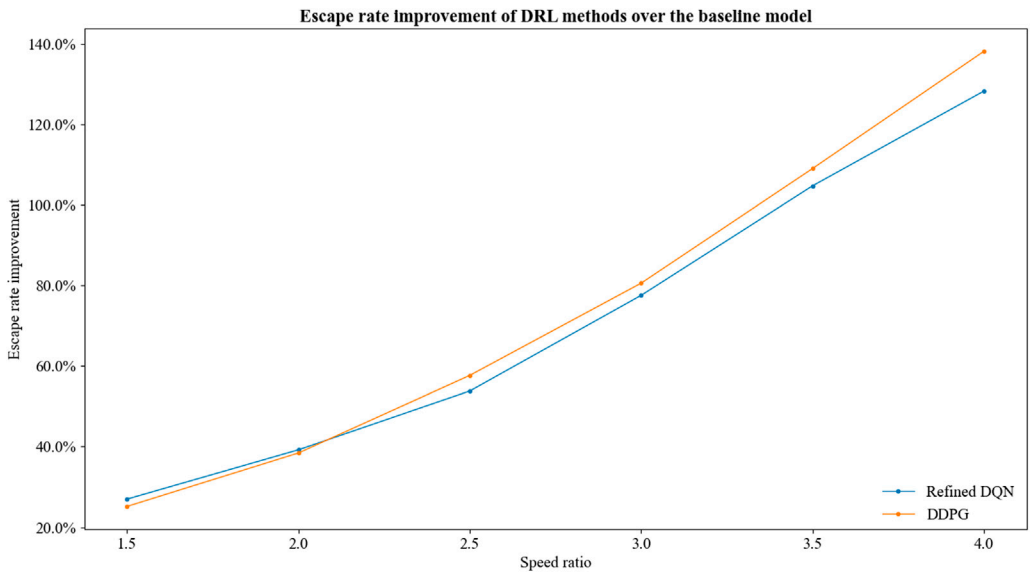

FIGURE 8 | Escape rate improvement of DRL methods over the baseline model.

As the speed ratio increases, the game environment difficulty increases. Correspondingly, the average improvement of escape probability increases. The DRL models have stronger learning ability in harsh game environments.

7 DISCUSSION

In this article, we first introduce a bio-inspired differential pursuit–evasion game: the dog sheep game. Our research explores the implementation of DRL methods for training the ignorant sheep to escape. Compared to other methods applied to differential pursuit–evasion games, DRL methods adopted are model-free and require less optimization.

Based on the traditional differential pursuit–evasion game theory, we come up with the kinematic model for the dog and the sheep. The dog's strategy can be summarized as reducing the inferior angle formed by the radii where the dog and the sheep are located. The sheep's strategy, which depends on its original location, is a bit more complicated. For a sheep that is theoretically evadable, it should run straight outward once the critical evasion conditions are satisfied.

Subsequently, we manage to adopt deep reinforcement learning methods to learn evasion strategy. In terms of the game environment settings, an attenuation mechanism of the deflection angle is applied due to the discrete action space of the DQN model. According to the idea of the cost function in differential evasion games, a time-out strategy is added to the reward mechanism. The aforementioned modifications show great improvement compared to the original DQN model and the baseline model. We also adopt the DDPG model, which allows the action space to be continuous. It has excellent performance as well. The refined DQN model outperforms the DDPG model by a small margin. The learning abilities of the DRL methods under different environment difficulties are assessed based on the improvement of the escape probabilities to the baseline model. Simulations indicate that they both have excellent learning ability in harsh environments.

This research shows that DRL methods are of great significance to differential pursuit–evasion games. Implementing these methods requires no manually designed features and less optimization for different game scenarios. The limitation of this research is the dog sheep game itself. This particular differential pursuit–evasion game scenario is simple. The DRL methods adopted by us are able to cope with more complex game scenarios, and we are looking forward to applying them to other games in the future.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are available at https://github.com/LEOXC1571/DSG. Supplementary Material, further inquiries can be directed to the corresponding author.

AUTHOR CONTRIBUTIONS

CX and YZ contributed to conception and design of the study and have equal contributions. CX preformed the kinematic analysis and optimization of DRL methods and wrote the manuscript. YZ built up the DRL training environment, implemented the DRL methods, and revised the manuscript. WW and LD helped the design of this study and offered writing guidance and assistance. All authors contributed to manuscript revision and read and approved the submitted version.

FUNDING

This project is supported by the Natural Science Foundation of Zhejiang Province (LY19A010004), the National Natural Science Foundation of China (11701509, 11971432), and the First Class Discipline of Zhejiang-A (Zhejiang Gongshang University-Statistics), the characteristic and preponderant discipline of key construction universities in Zhejiang Province (Zhejiang Gongshang University-Statistics), Collaborative Innovation Center of Statistical Data Engineering Technology and Application.

REFERENCES

Chen, T., Peng, L., Yang, J., and Cong, G. (2021). Analysis of User Needs on Downloading Behavior of English Vocabulary APPs Based on Data Mining for Online Comments. *Mathematics* 9 (12), 1341. doi:10.3390/math9121341

Chen, T., Rong, J., Yang, J., Cong, G., and Li, G. (2021). Combining Public Opinion Dissemination with Polarization Process Considering Individual Heterogeneity. *Healthcare* 9 (2), 176. doi:10.3390/healthcare9020176

Chen, T., Yin, X., Peng, L., Rong, J., Yang, J., and Cong, G. (2021). Monitoring and Recognizing enterprise Public Opinion from High-Risk Users Based on User Portrait and Random forest Algorithm. *Axioms* 10 (2), 106. doi:10.3390/axioms10020106

Hao, Z., Wang, Z., Bai, D., Tao, B., Tong, X., and Chen, B. (2021). Intelligent Detection of Steel Defects Based on Improved Split Attention Networks. *Front. Bioeng. Biotechnol.* 9, 810876. doi:10.3389/fbioe.2021.810876

Isaacs, R. (1965). *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. New York: John Wiley and Sons.

Jiang, F., Guo, X., Zhang, X., Zhang, Z., and Dong, D. (2020). "Approximate Soft Policy Iteration Based Reinforcement Learning for Differential Games with Two Pursuers versus One Evader," in 2020 5th International Conference on Advanced Robotics and Mechatronics (ICARM), Shenzhen, China, 18-21 Dec. 2020, 471–476. doi:10.1109/icarm49381.2020.9195328

Jiang, D., Li, G., Sun, Y., Hu, J., Yun, J., and Liu, Y. (2021a). Manipulator Grabbing Position Detection with Information Fusion of Color Image and Depth Image Using Deep Learning. *J. Ambient Intell. Hum. Comput* 12 (12), 10809–10822. doi:10.1007/s12652-020-02843-w

Jiang, D., Li, G., Tan, C., Huang, L., Sun, Y., and Kong, J. (2021b). Semantic Segmentation for Multiscale Target Based on Object Recognition Using the Improved Faster-RCNN Model. *Future Generation Comput. Syst.* 123, 94–104. doi:10.1016/j.future.2021.04.019

Lewin, J., and Breakwell, J. V. (1975). The Surveillance-Evasion Game of Degree. *J. Optimization Theor. Appl.* 16 (3-4), 339–353. doi:10.1007/bf01262940

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous Control with Deep Reinforcement Learning. arXiv:1509.02971 [Preprint]. Available at: http://arxiv.org/abs/1509.02971 (Accessed January 28, 2022).

Lin, L. J. (1992). *Reinforcement Learning for Robots Using Neural Networks*. Pittsburgh: Carnegie Mellon University.

Liu, X., Jiang, D., Tao, B., Jiang, G., Sun, Y., Kong, J., et al. (2021). Genetic Algorithm-Based Trajectory Optimization for Digital Twin Robots. *Front. Bioeng. Biotechnol.* 9, 793782. doi:10.3389/fbioe.2021.793782

Liu, Y., Jiang, D., Yun, J., Sun, Y., Li, C., Jiang, G., et al. (2021). Self-tuning Control of Manipulator Positioning Based on Fuzzy PID and PSO Algorithm. *Front. Bioeng. Biotechnol.* 9, 817723. doi:10.3389/fbioe.2021.817723

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P., and Mordatch, I. (2020). Multi-agent Actor-Critic for Mixed Cooperative-Competitive Environments [Preprint]. Available at: https://arxiv.org/abs/1706.02275 (Accessed January 28, 2022).

Merz, A. W. (1971). *The Homicidal Chauffeur-Aa Differential Game*. Stanford: Stanford University.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing Atari with Deep Reinforcement Learning [Preprint]. Available at: https://arxiv.org/abs/1312.5602 (Accessed January 28, 2022).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-Level Control through Deep Reinforcement Learning. *Nature* 518, 529–533. doi:10.1038/nature14236

Perot, E., Jaritz, M., Toromanoff, M., and de Charette, R. (2017). "End-To-End Driving in a Realistic Racing Game with Deep Reinforcement Learning," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Honolulu, HI, USA, 21-26 July 2017, 3–4. doi:10.1109/cvprw.2017.64

Shao, K., Tang, Z., Zhu, Y., Li, N., and Zhao, D. (2019). A Survey of Deep Reinforcement Learning in Video Games [Preprint]. Available at: http://arxiv.org/abs/1912.10944 (Accessed January 28, 2022).

Shinar, J., and Gutman, S. (1978). "Recent Advances in Optimal Pursuit and Evasion," in 1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes, San Diego, CA, USA, 10-12 Jan. 1979, 960–965. doi:10.1109/cdc.1978.268074

Shinar, J. (1981). Solution Techniques for Realistic Pursuit-Evasion Games. *Control. Dynamic Syst.* 17, 63–124. doi:10.1016/b978-0-12-012717-7.50009-7

Singh, G., Lofaro, D., and Sofge, D. (2020). "Pursuit-Evasion with Decentralized Robotic Swarm in Continuous State Space and Action Space via Deep Reinforcement Learning," in 12th International Conference on Agents and Artificial Intelligence (ICAART), Valletta, Malta, 22-24, Feb. 2020, 226–233. doi:10.5220/0008971502260233

Sundaram, S., Kalyanam, K., and Casbeer, D. W. (2017). "Pursuit on a Graph under Partial Information from Sensors," in 2017 American Control Conference (ACC), Seattle, WA, USA, 24-26 May 2017, 4279–4284. doi:10.23919/acc.2017.7963613

Wan, K., Wu, D., Zhai, Y., Li, B., Gao, X., and Hu, Z. (2021). An Improved Approach towards Multi-Agent Pursuit-Evasion Game Decision-Making Using Deep Reinforcement Learning. *Entropy* 23 (11), 1433. doi:10.3390/e23111433

Wang, L., Wang, M., and Yue, T. (2019). A Fuzzy Deterministic Policy Gradient Algorithm for Pursuit-Evasion Differential Games. *Neurocomputing* 362, 106–117. doi:10.1016/j.neucom.2019.07.038

Wang, M., Wang, L., and Yue, T. (2019). "An Application of Continuous Deep Reinforcement Learning Approach to Pursuit-Evasion Differential Game," in 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 15-17 March 2019, 1150–1156. doi:10.1109/itnec.2019.8729310

Watkins, C. J. C. H., and Dayan, P. (1992). Q-Learning. *Machine Learn.* 8 (3-4), 279–292. doi:10.1023/a:1022676722315

Wei, Z., Wang, D., Zhang, M., Tan, A.-H., Miao, C., and Zhou, Y. (2018). "Autonomous Agents in Snake Game via Deep Reinforcement Learning," in 2018 IEEE International Conference on Agents (ICA), Singapore, 28-31 July 2018, 20–25. doi:10.1109/agents.2018.8460004

Xiao, F., Li, G., Jiang, D., Xie, Y., Yun, J., Liu, Y., et al. (2021). An Effective and Unified Method to Derive the Inverse Kinematics Formulas of General Six-DOF Manipulator with Simple Geometry. *Mechanism Machine Theor.* 159, 104265. doi:10.1016/j.mechmachtheory.2021.104265

Yang, Z., Jiang, D., Sun, Y., Tao, B., Tong, X., Jiang, G., et al. (2021). Dynamic Gesture Recognition Using Surface EMG Signals Based on Multi-Stream Residual Network. *Front. Bioeng. Biotechnol.* 9, 779353. doi:10.3389/fbioe.2021.779353

Yong, J. (1986). *On Differential Games of Evasion and Pursuit (Capturability)*. Purdue University.

Yong, J. (2014). *Differential Games: A Concise Introduction*. New Jersey: World Scientific.

Zhao, G., Jiang, D., Liu, X., et al. (2022). A Tandem Robotic Arm Inverse Kinematic Solution Based on an Improved Particle Swarm Algorithm. *Front. Bioeng. Biotechnol.* 10, 832829. doi:10.3389/fbioe.2021.832829

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors, and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.