Autonomous Drone Navigation through Vertical Obstacles with Novel Reward Function and Temporal Shift Architecture

Khuong G. T. Diep[1] and Yong-Guk Kim[2]

***Abstract*— Drones are increasingly utilized in diverse domains such as surveillance, rescue, delivery, and data collection, where autonomous navigation with reliable obstacle avoidance is critical. While deep reinforcement learning (DRL) has shown promise in addressing this challenge, existing models often struggle with enabling effective vertical maneuverability. This paper presents a novel reward function tailored to enhance three-dimensional (3D) navigation in drone systems, addressing this key limitation. Our approach integrates this reward function within a Double Deep Q-Network (D3QN) framework, augmented by a temporal shift technique to model temporal dependencies between observations and a soft update mechanism to improve training stability and convergence speed. The system is evaluated in both simulated and real-world environments using a Parrot Bebop 2.0 drone and an off-board ROS-based computing platform. Real-world tests conducted in indoor corridors with varied static and low-lying obstacles demonstrate that the proposed approach enables safe and efficient 3D navigation, outperforming a baseline reward function in terms of collision avoidance and learning efficiency.**

I. INTRODUCTION

Drones are intelligent flying machines capable of autonomous flight, eliminating the need for onboard human pilots. Initially developed for military purposes, particularly for data collection in hazardous environments. However, operating safely in complex environments requires advanced obstacle avoidance algorithms—intelligent systems that help drones navigate around trees, buildings and people. This study delves into such algorithms to enhance drones' capabilities and ensure mission success, especially in challenging environments with vertical obstacles.

Two common approaches in obstacle avoidance are Simultaneous Localization and Mapping (SLAM) and Structure from Motion (SfM). However, both require the drone to hover frequently to compute depth maps and determine safe paths, making them inefficient for real-time navigation. Moreover, these techniques depend on expensive, heavy, and power-hungry sensors, which are unsuitable for compact drones like quadcopters. In contrast, monocular cameras are a promising alternative—they are lightweight, low-cost, and provide sufficient visual data for navigation. Recent developments in deep learning have enabled depth estimation from RGB images, supplying valuable input for obstacle avoidance systems, including those based on deep reinforcement learning (DRL). These solutions are not only more efficient but also more practical for real-time drone applications. However, many current approaches, while effective at obstacle avoidance, fail to fully exploit the drone's available degrees of freedom. To address this, our study introduces a novel reward function that encourages full 3D navigation—promoting movement across both horizontal and vertical axes.

The primary objective of this study is to enable autonomous navigation for drones, especially in unfamiliar environments filled with numerous unknown obstacles. While the Global Navigation Satellite System (GNSS) has been a common solution, it proves ineffective in low-signal areas like indoor spaces. In such contexts, DRL has emerged as a viable alternative [1] since it allows trained drones to adapt to new environments without needing a lot of extra fine-tuning. A study [2] has shown that deep reinforcement learning can make a drone's performance comparable to that of a skilled human in games. In summary, the proposed algorithm enables drones to navigate autonomously in challenging and unfamiliar environments using only vision-based inputs. The main contributions of this work are summarized below.

- The key innovations of this research lie in the novel Lidar-based reward function combined with appropriate action space. These components are designed to support the drone's navigation capabilities, allowing it to move forward, change its heading, and modify its operating height.
- This study integrates the temporal shift techniques in the D3QN model to enhance the algorithm's temporal awareness, enabling the extraction of intricate temporal information that is crucial for decision-making;
- Finally, the incorporation of the soft update mechanism for the target network plays a key role in improving the performance of the algorithm. This technique not only improves the overall efficiency of the network but also accelerates the convergence process.

II. RELATED WORK

In [3], DroNet with a residual convolutional network was proposed for safe drone navigation in urban and indoor environments. It predicts steering angles and collision probabilities using over 70,000 forward-facing images from driving

*This work was supported by the Information Technology Research Center (ITRC) support program (IITP-2022-RS-2022-00156354) and a grant funded by the Korean government Ministry of Science and Information Technology (MSIT) (No.RS-2019-II190231) from the Institute of Information & communications Technology Planning & Evaluation (IITP), as well as by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2020R1A6A1A03038540).

[1]Khuong G. T. Diep was in the Department of Computer Engineering, Sejong University, Seoul, Korea. He is now with Data Design Engineering, Seoul, Korea thuykhuong911@gmail.com

[2]Yong-Guk Kim is in the Department of Computer Engineering, Sejong University, Seoul, Korea ykim@sejong.ac.kr

scenarios and drone-collected obstacle distance data. Experiments showed DroNet's effectiveness in navigating city streets and indoor areas like corridors and parking lots, with the capability to stop when obstacles are detected. On the other hand, DRL has been used as a powerful AI approach, merging deep learning with reinforcement learning. In [4], an obstacle avoidance algorithm used a monocular camera and a DRL. A conditional GAN (Generative Adversarial Network) was trained on 90,000 RGB-D pairs from 22 indoor locations in Gazebo, enabling the generation of depth maps from RGB images. These were input to a recurrent network with temporal attention for navigation. The model demonstrated successful avoidance of both static and dynamic obstacles. In another study, Shin et al. [5] introduced a method combining optical flow, critic networks, and a U-Net within a reward-driven framework. The U-Net output guided an actor network to produce control commands. Among various policy gradient algorithms, ACKTR performed best in continuous action spaces. The system enabled robust navigation in both familiar and unfamiliar environments, with obstacle avoidance achieved through movement along three linear axes.

III. DEEP REINFORCEMENT LEARNING

A. Problem Formulation

It is shown that a vision-enhanced obstacle avoidance drone can be modeled as a Partially Observable Markov Decision Process (POMDP), which is defined as a tuple $\langle S, A, T, R, \Omega, O \rangle$. Here, S, A, T, and R represent states, actions, transition functions, and rewards, respectively. In addition, Ω is the set of observations that partially reveal the true state, and O is the observation function. Our goal is to find the optimal policy π^* that selects actions to maximize the sum of discounted rewards:

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} R_k(o_k, a_k) \tag{1}$$

B. Dueling Double Deep Q Network

This section covers Deep Q Networks (DQN) and the Dueling Double Deep Q Network (D3QN). Unlike Q-learning, which uses a Q-table, DQN uses a deep neural network. DQN calculates Q-values for all actions in the next state, selects the max, and applies temporal difference learning:

$$Q_{s,a} = r(s,a) + \gamma \max_a Q'(s', a') \tag{2}$$

where $Q_{s,a}$ is the estimated Q-value for state s and action a, $r(s,a)$ is the reward received after taking action a in state s, γ is the discount factor, s' is the next state, and $Q'(s', a')$ represents the target Q-values for all possible next actions a' in state s'.

However, since this maximum operation can cause Q-value overestimation, Double DQN (DDQN) addresses this by using two networks: an online network updated every step and a target network updated less frequently. This measure reduces overestimation:

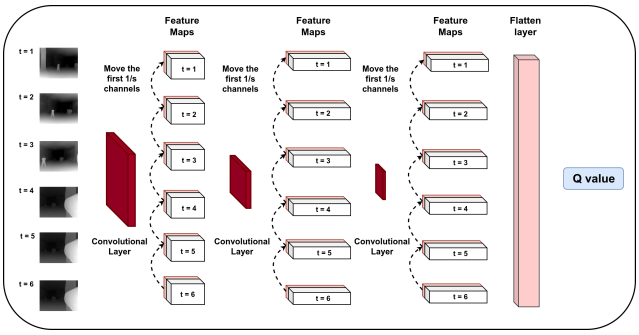


Fig. 1: Flow diagram illustrating the process of shifting channels between successive inputs. In this process, the first $1/s$ channels are shifted and are then fed into the next CNN layer.

$$Q_{s,a}^{Double} = r(s,a) + \gamma Q'(s', \arg\max_a Q(s', a)) \tag{3}$$

Dueling DQN further improves performance by splitting the network into two streams to learn state-value $V(s)$ and advantage $A(s,a)$, allowing the agent to assess state importance independently of actions:

$$A(s,a) = Q(s,a) - V(s) \tag{4}$$

Combining both approaches, Dueling Double DQN (D3QN) provides enhanced performance in discrete action spaces. The Q-value target is the following.

$$Q_{s,a}^{D3QN} = r(s', a') + \gamma Q(s', \arg\max Q(s', a; \theta, \alpha, \beta)) \tag{5}$$

It is known that D3QN outperforms other DQN variants in over 65% of problems [6] as we adopt in this work.

IV. METHOD

A. Network Structures

CNN is typically used to extract spatial information from the input and yet overlooks the temporal relationships between successive observations. Inspired by the TSM module in [7], the temporal shift technique was incorporated into a shallow CNN to address this problem. A sequence of six observations is input into the network. Initially, these observations pass through the first convolutional layer. Subsequently, the first $1/s$ channels are shifted (s is set to 5 in this paper) among a total of t feature maps (in this case, t = 6). After this, the shifted feature maps undergo the next convolutional layer. This operation is repeated once more, as illustrated in Fig. 1. Finally, all the features are flattened and fed into the Multilayer Perceptron (MLP) layer to obtain the Q-value.

B. Soft Update

Contrast to the original D3QN where the target network is updated after a fixed number of steps, we adopt a soft update strategy [8]. In this scheme, updates occur at every time step. For example, τ is set to 0.005 in the training

of this model. The new target network is calculated as the summation of 0.5% of the prediction network weights and 99.5% of the old target network weights. This soft update does not contradict the purpose of fixed target networks since it can still be considered as a fixed one by retaining 99.5% of the former target network weights. The update scheme is expressed as following:

$$\theta^{target} = \tau \times \theta^{prediction} + \big(1 - \tau\big) \times \theta^{target} \tag{6}$$

C. Lidar-based reward functions

Designing an effective reward function is crucial for the success of reinforcement learning. We proposes a robust reward function that utilizes five Lidar sensors mounted on the drone during training by covering a specific zone with each sensor as illustrated in Fig. 2. The reward function is organized into four distinct categories to promote a comprehensive understanding of the environment by the drone: high-speed reward, turning reward, up-and-down reward, and collision reward.

1) High-speed reward function: Lidar provided by AirSim is used to measure the object distance by generating a point cloud wherein each point provides x, y, and z coordinates relative to the drone. The shortest distance is identified by determining if the nearest obstacle from the Lidar is either in the High-speed or Turning zone: Obstacles beyond 1.7 meters fall in the High-speed zone whereas those closer are categorized as the Turning zone. Since forward motion is more energy-efficient, the reward function in equation 7 is designed to promote straight, high-speed flight. Here, v_x represents linear speed along the x-axis, ω is the angular velocity, and v_z is the linear speed along the z-axis, respectively. Within the High-speed zone, the drone is encouraged to move forward quickly to maximize its reward, whereas any yawing or vertical movement leads to a penalty. To discourage unnecessary vertical motion, a coefficient λ is applied, reducing the reward when the drone moves up or down without need.

$$R = v_x - |\omega| - \lambda \times |v_z| \tag{7}$$

2) Turning reward function: When obstacles are within the Turning zone, the drone should prioritize yawing for safe navigation. This decision is based on two conditions: (1) the nearest object or wall is less than 1.7 meters away, and (2) the shortest x-coordinate from Lidar 2 ($min_{x_{li_2}}$) equals that from Lidar 3 ($min_{x_{li_3}}$). The second condition indicates no clear path above or below, suggesting that changing direction is the best option. This scenario is illustrated in Fig. 2b. Equation 8 defines the drone's behavior in the Turning zone. α is a temporal discount factor, and n_{yaw} is the number of consecutive yawing actions. The term $|\omega| \times (1 - \alpha \times (n_{yaw} - 1))$ discourages the drone from repeatedly turning in place, preventing it from hovering or circling near obstacles. The yawing reward decreases over time and can turn negative if excessive yawing is detected.

$$R = |\omega| \times (1 - \alpha \times (n_{yaw} - 1)) - v_x - \lambda \times |v_z| \tag{8}$$

3) Up and Down reward function: The above equations are designed to make the drones to avoid obstacles during its flight. Addressing the unique vertical maneuverability of drones, which differentiates them from conventional automobiles, this paper presents a novel reward function. This function is specifically engineered to dynamically alter the drone's height when encountering obstacles in the upper or lower airspace. Figure 2c illustrates the obstacle detection capabilities of Lidar sensors 2 and 3 within this specific scenario. Upon detection of an obstruction within the lower zone, characterized by a $min_{x_{li_3}}$ value less than $min_{x_{li_2}}$, the drone is prompted to initiate an ascent. This upward vertical displacement serves to avoid the detected obstacle and facilitate the identification of a safer trajectory. Conversely, the detection of an obstacle within the upper zone, where $min_{x_{li_3}}$ exceeds $min_{x_{li_2}}$, triggers a descent maneuver. This controlled vertical movement enables the drone to achieve optimal maneuverability in response to obstacles present in both the lower and upper zones. A new parameter, β, which is set to 2.0. is introduced to encourage the drone to ascend or descend when encountering obstacles in the Lower or Upper zones.

$$R = \begin{cases} \beta \times v_z - v_x - |\omega| & min_{x_{li_2}} > min_{x_{li_3}} \\ \beta \times v_z - v_x - |\omega| & min_{x_{li_2}} < min_{x_{li_3}} \end{cases} \tag{9}$$

4) Collision reward function: However, a negative reward is assigned to the drone upon contact with obstacles or walls. When the minimum distance (min_{dis}) is less than 0.8m, the drone is considered to have collided with an obstacle, and a reward of -1 is given. In a similar vein, leveraging the drone's capacity for vertical displacement, Lidar sensors 4 and 5 are implemented to determine the vertical distance to the ground or ceiling. Additionally, these sensors serve to measure the distance to obstacles located above or below the drone during navigational operations. When this distance is less than 0.4m, the same collision penalty of -1 is applied.

D. Baseline Algorithm

In order to benchmark our algorithm's efficacy, a basic reward function, excluding Lidar sensor input during training, is additionally proposed in this study. The drone's action space remains congruent with our proposed algorithm. When the decision is made to move forward, a reward equivalent to the v_x value is given, with a maximum reward capped at 0.8. Conversely, the drone receives a reward equal to the value of the angular velocity ω or vertical velocity v_z. The inclusion of the 1.2 factor limits the maximum achievable reward in this instance to 0.6, which is inferior to the maximum reward for forward progression.

$$R = \begin{cases} v_x & v_x > 0 \\ 1.2 \times (|\omega| + 1.25 \times |v_z|) & v_x = 0 \\ -10 & collision \end{cases} \tag{10}$$

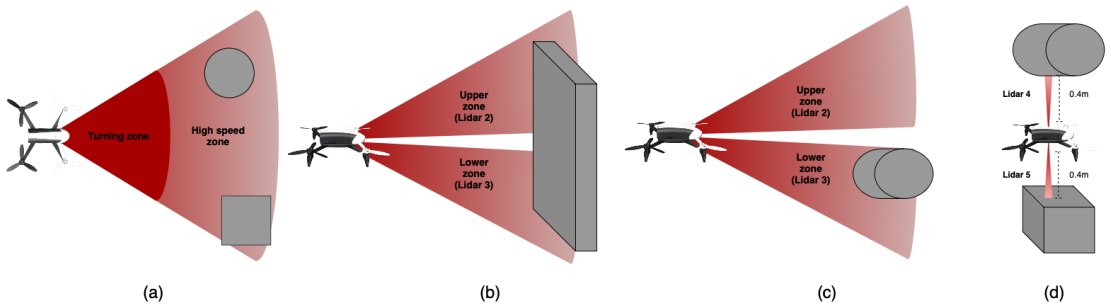


Fig. 2: Illustration of different Lidar zones in simulation: (a) Zones for turning and high-speed movement, (b) Upper and lower detection zones, (c) Upper and lower zones with obstacles present, and (d) Collision-checking LiDAR measuring distance from the ground or ceiling to the drone.

Fig. 3: The training environment is a maze, consisting of various horizontal and vertical obstacles, including humans. The starting points are indicated as 10 red dots, superposed on the top view of the environment.

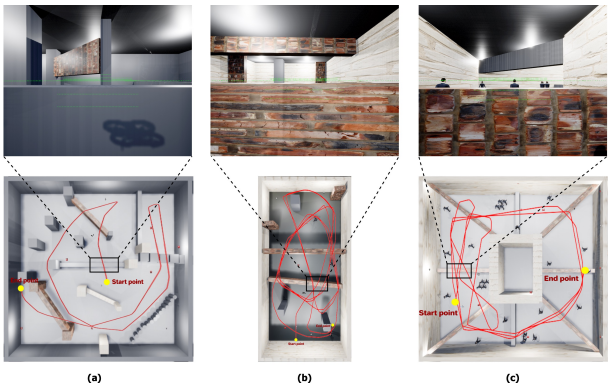


Fig. 4: Three testing environments. Each First-Person-View (FPV) is seen by a flying drone at a position indicated by a black rectangle in the environment (A, B, C), respectively. At that position, the drone is required to climb over the horizontal obstacle to navigate further within the environment.

V. EXPERIMENT

A. Environment

1) Tranining Environment: A simulated training environment, depicted in Fig. 3, incorporates a diverse array of obstacles, including cylindrical structures, planar walls, columnar pillars, and human figures. To facilitate the learning of vertical maneuvering skills, horizontal bar structures are strategically positioned to encourage the drone to modulate its altitude during translational movement. These vertical obstacles serve to emulate terrain features characterized by varying elevations. Lidar sensors are employed during the training phase to quantify the spatial separation between the drone and proximal environmental elements, while these sensors are not utilized during the evaluation phase. Instead, a monocular camera acquires RGB imagery for autonomous navigation in test scenarios.

The drone is initialized at one of the designated positions, represented by red dots in Fig. 3. In these initial positions, the drone is sometimes restarted near the horizontal bars to encourage learning to change its height, not just its direction when facing obstacles. Training continues until the drone either collides with an obstacle or exceeds 500 steps.

2) Testing Environment: To evaluate the performance of our algorithm, we created three test maps, each featuring distinctively different scenarios. Test Map A includes pillars and horizontal bars. Test Map B shares similarities with Test Map A but introduces additional elements, such as humans. Finally, the complexity is increased by adding more horizontal bars and human figures in Test Map C. Fig. 4 illustrates 3 testing maps and, in each cases, a FPV seen by the flying drone is illustrated.

B. Results

1) Simulation Testing: Three methods were evaluated for comparison: Random movement, a baseline algorithm, and the proposed approach. Each of the four test maps underwent 35 trials, with consistent parameters across all runs, including action space, velocity, orientation, and collision criteria. Collisions were detected using the AirSim API, with a return of True indicating a failed attempt. Performance was assessed using two key metrics: average travel distance (TD) and flight time (T), which provide a reliable measure of the navigation effectiveness of each method. As shown in Table I, the results highlight clear differences between the methods. Random motion exhibited the poorest performance, characterized by limited travel distances and curtailed flight durations, thereby underscoring its inefficiency for autonomous navigation tasks. The baseline

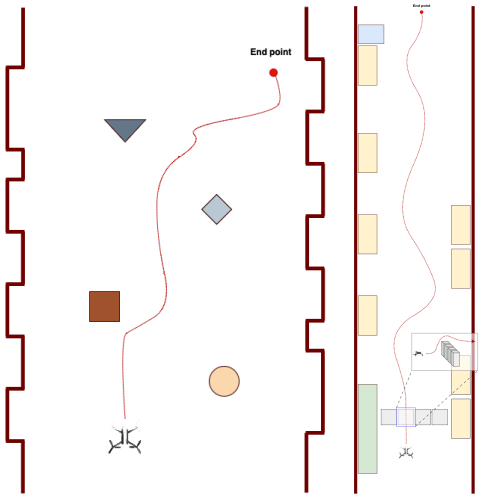
methodology, trained with a rudimentary reward function, achieved a moderate level of success, demonstrating a limited capacity for obstacle avoidance but exhibiting suboptimal overall efficiency. However, the proposed algorithm significantly outperformed both baselines. Its strength lies in the ability to distinguish between scenarios requiring vertical or rotational maneuvers, enabling more precise and adaptive navigation. This improvement underscores the effectiveness of the proposed reward function in guiding drones through complex environments and making more informed movement decisions.

TABLE I: The obstacle avoidance performance is evaluated across four test maps in AirSim (`https://www.youtube.com/watch?v=Y8A3pt9yxP8`).

Map	A		B		C	
Measurement	TD (m)	T (s)	TD (m)	T (s)	TD (m)	T (s)
Random	5.71	68.22	10.87	136.6	6.45	88.19
Baseline	102.02	397.48	139.87	670.91	105.72	450.46
Proposed model	**178.21**	426.3	**383.38**	992.19	**183.34**	576.33

2) Real-world Testing: For real-world testing, an off-board laptop with an Intel Core i7-10750H CPU, NVIDIA RTX 2070 Super GPU, and 16 GiB RAM runs ROS Noetic and the bebop-autonomy driver. The Parrot Bebop 2.0 serves as the drone platform, communicating with the laptop via WiFi. For safety, the velocity is reduced to [0.2, 0.1] for x velocity, [$\pm$0.3, $\pm$0.15, 0] for angular velocity, and [$\pm$0.2, $\pm$0.1] for z velocity.

The experiments were conducted within the corridors of the AI Center at Sejong University. Scenario 1 was implemented in a corridor of moderate dimensions, featuring four distinct static obstacles. Scenario 2 involved the strategic placement of low-lying obstacles at the corridor's commencement, necessitating an upward vertical displacement of the drone for avoidance. Subsequently, additional obstructions, such as cabinets, were introduced at various spatial locations along the corridor. Fig. 5 provides the overall estimated trajectories of two real test maps. As depicted in the figure, the drone navigated through various scenarios, avoiding collisions with obstacles successfully.



(a) (b)

Fig. 5: Indoor testing environments and trajectories (`https://www.youtube.com/watch?v=yC2c9BgMw1Q`)

VI. Discussion and Conclusion

This paper details the development of an obstacle avoidance algorithm for a drone equipped with a monocular camera, leveraging deep reinforcement learning methodologies. Unlike previous methods, such as the drones in [3], [4], and [5], which have limitations in dynamic path adjustment, altitude control, or movement axes, the proposed method enables drones to change direction and perform vertical movements, enhancing mobility and navigation in diverse environments.

This methodology utilizes a depth estimation model to transform RGB images into estimated depth maps, thereby mitigating the discrepancy between simulated and real-world environments. The proposed reward function, in conjunction with Lidar sensor data, enables comprehensive drone navigation in both horizontal and vertical axes. Furthermore, temporal shift techniques and a soft update scheme are integrated to enhance training efficacy by leveraging temporal dependencies within the data. The algorithm's performance is validated through rigorous simulations, wherein it is benchmarked against a baseline reward function across four distinct testing environments. The experimental outcomes affirm the efficacy of the proposed reward function for achieving robust three-dimensional navigation and autonomous obstacle avoidance in varied environments. Subsequent work will address to enhance the reward function and expand the sensor suite to achieve superior outdoor performance.

References

[1] G. Muñoz, C. Barrado, E. Çetin, and E. Salami, "Deep reinforcement learning for drone delivery," *Drones*, vol. 3, no. 3, 2019.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb 2015.

[3] A. Loquercio, A. I. Maqueda, C. R. del Blanco, and D. Scaramuzza, "Dronet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[4] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in uav with limited environment knowledge," *Trans. Intell. Transport. Syst.*, vol. 22, p. 107–118, dec 2020.

[5] S.-Y. Shin, Y.-W. Kang, and Y.-G. Kim, "Reward-driven u-net training for obstacle avoidance drone," *Expert Systems with Applications*, vol. 143, p. 113064, 2020.

[6] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, p. 1995–2003, JMLR.org, 2016.

[7] J. Lin, C. Gan, and S. Han, "Temporal shift module for efficient video understanding," *CoRR*, vol. abs/1811.08383, 2018.

[8] Y. Li and J. Li, "Application of d3qn algorithm based on behavior exploration strategy," *Scientific Journal of Intelligent Systems Research*, vol. 4, 2022.