Reinforcement Learning for Adversarial Agents in Combat Boxing

Samyukta Ramesh, Ishan Biswas, Malav Patel, Aparna Shree Govindarajan, Anagha Srinath
Northeastern University, Boston, MA

***Abstract*—This project implements a multi-agent reinforcement learning framework to train independent agents in the Atari Boxing game using the PettingZoo library. Each agent is powered by a Deep Q-Network (DQN) model, employing a convolutional neural network (CNN) to process temporal and spatial features from stacked observations. A custom environment wrapper standardizes inputs by resizing frames, converting them to grayscale, and stacking consecutive observations for context. The agents use experience replay, epsilon-greedy exploration, and target networks to ensure stable and efficient training. Evaluation metrics, such as average rewards over multiple games, demonstrate the agents' ability to develop competitive strategies, showcasing the potential of deep reinforcement learning in multi-agent environments.**

***Index Terms*—Reinforcement Learning, Deep Q-Networks, Multi-Agent, PettingZoo, Combat Boxing, CNN, Adversarial Agents.**

I. Introduction

In recent years, reinforcement learning (RL) has gained significant attention for its ability to train agents that can make decisions and learn from their environment. When multiple agents are involved, the complexity increases, especially in competitive scenarios where each agent's actions can directly affect the other. One such environment is the Combat Boxing game, a part of the PettingZoo Atari suite, where two agents face off in a boxing ring. The goal is to outplay the opponent by moving, punching, and avoiding damage within a set time frame, with points awarded for successful punches and knockouts.

The focus of this project is to design parallel and independent policies for the two agents. Each agent will learn its own strategy without directly influencing the other's learning process. While both agents interact and compete within the same environment, their policies are trained separately, allowing each agent to act independently. This approach is both challenging and interesting because it requires balancing the competitive nature of the game while ensuring that each agent can learn and make decisions based only on its own observations.

In this work, we aim to develop and train these independent policies using reinforcement learning techniques. By doing so, we explore how agents can learn to effectively compete in a dynamic and adversarial environment. The goal is for each agent to become adept at boxing, making intelligent moves and decisions based on its unique policy, while navigating the competitive interactions that arise during the match.

II. Related Work

The field of multi-agent reinforcement learning (MARL) has become an important area of research, particularly when multiple agents need to either cooperate or compete within a shared environment. In competitive settings, agents must adapt not only to the environment but also to the actions and strategies of other agents. MARL has been applied to a variety of domains, from games to robotics, and it continues to evolve as researchers find ways to address the unique challenges posed by multi-agent interactions.

In traditional reinforcement learning, agents typically learn in isolation, where they interact with an environment and optimize their strategies based on rewards or penalties. However, in multi-agent settings, the environment becomes non-stationary because other agents are also learning and adapting along with the environment changing over time. This makes learning much more complex. For example, in competitive environments like Chess or Go, agents need to consider not only the best actions for themselves but also anticipate the opponent's moves. The self-play method has been particularly useful in this context, where agents play against themselves to improve their strategies.

In adversarial games like boxing, agents are in direct competition. These environments, such as the Atari 2600 games, have been widely used to study how agents learn to handle competition. For instance, PettingZoo provides a set of multi-agent environments, including boxing_v2, where two agents face off against each other. The challenge here lies in how the agents learn to compete effectively without either one dominating the learning process. Unlike cooperative games, where agents might share objectives, in competitive games, each agent's actions work directly against the other, making it a more complex problem.

The challenge in developing effective multi-agent policies is ensuring that each agent can learn independently while still interacting with the other. Independent Q-learning and Deep Q-Networks (DQN) have been commonly used to train agents in such environments, where each agent learns based on its own observations and rewards. However, a key issue with independent learning is that the agents may not always converge to stable solutions due to the non-stationary nature of the environment. This project aims to address this challenge by training two independent policies for the boxing environment, where both agents learn separately but compete against each other.

In this work, we draw inspiration from these existing techniques and aim to extend them by creating policies that enable the agents to interact and compete in the boxing environment, while maintaining their independence. By doing so, we contribute to the ongoing research in MARL, specifically focusing on how agents can learn independently yet effectively within adversarial settings.

III. Methods

The Combat Boxing environment, part of the PettingZoo Atari suite, is used for training and evaluating the agents. This environment consists of two agents competing in a boxing ring, with the goal of outplaying the opponent within a set time frame. The agents control pixelated figures that move and interact within the ring.

The environment setup is shown in Figure 1, which illustrates the two agents represented by the white and black figures, the timer, and the score display.

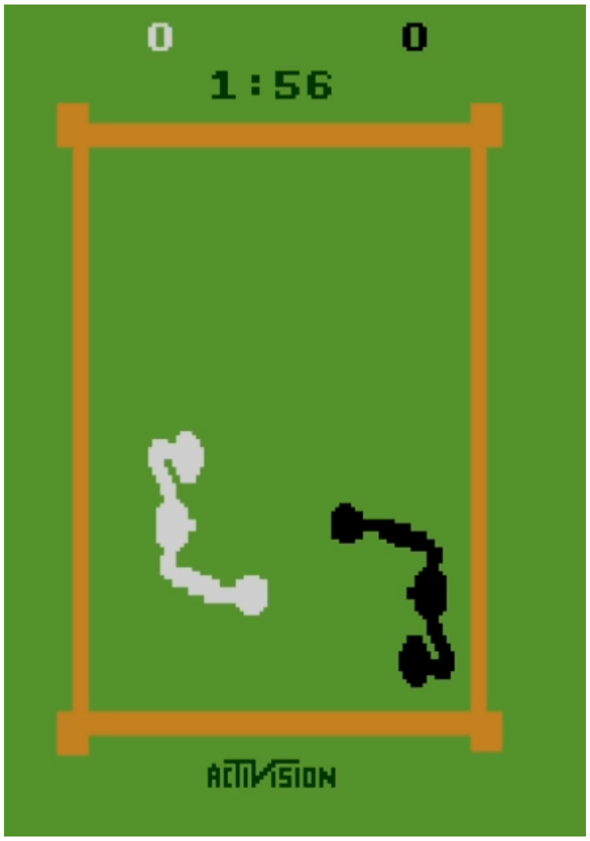

Fig. 1. Combat Boxing game environment.

A. Environment Preprocessing

In reinforcement learning, preprocessing raw observations is critical for ensuring effective model training, particularly in visual domains like Atari games. To this end, the PettingZoo Boxing_v2 environment was wrapped in a custom preprocessing pipeline that tailored observations for input into CNN-based Deep Q-Networks (DQNs). The preprocessing steps are described below:

Grayscale Conversion: The raw RGB observations from the environment were converted to grayscale to simplify input data. This step significantly reduced the dimensionality of the input while retaining the visual features necessary for understanding spatial relationships and game dynamics. Grayscale frames also eliminate redundant color information, focusing the model's attention on motion and object positioning.

Resizing: Grayscale frames were resized to 84x84 pixels, a standard resolution in reinforcement learning for Atari-based environments. This resizing step ensured compatibility with the CNN model architecture while preserving essential spatial features of the original observations.

Frame Stacking: To capture temporal dependencies, four consecutive frames were stacked along the channel dimension to form a single input tensor. By stacking frames, the model could infer motion dynamics, such as the direction and speed of agents' movements. This context is crucial for making decisions in adversarial games, where anticipating the opponent's next move often determines success.

Normalization: The pixel values of the preprocessed frames were normalized to the range [0, 1] to improve numerical stability during training. Normalization helps the model converge faster by reducing the scale of input values, which can otherwise lead to gradient instability.

This preprocessing pipeline ensured that the observations provided to the DQNs were both computationally efficient and rich in the features necessary for learning. The design of this preprocessing was inspired by approaches used in prior work on reinforcement learning in visual domains, emphasizing both performance and simplicity.

B. DQN Agent Architecture

The agents in this project were implemented using Deep Q-Networks (DQNs), a reinforcement learning algorithm designed to handle high-dimensional state spaces by approximating Q-values with neural networks. The architecture of the agents' DQNs was inspired by established methodologies in reinforcement learning for visual inputs.

The network as seen in Figure 2 begins with three convolutional layers that process the stacked frames (input tensor with shape [4, 84, 84]). These layers extract hierarchical spatial features, enabling the agent to understand critical elements such as opponent positions, proximity to boundaries, and punch trajectories.

The fully connected layers process the extracted features and compute the Q-values for each action. A target network is maintained alongside the primary Q-network to stabilize learning, with periodic updates to the target network ensuring that the Q-value targets remain stable.

C. Training Framework

The training process for the agents was designed to optimize learning stability and efficiency in a competitive multi-agent environment. The framework incorporates key reinforcement learning techniques, ensuring effective policy learning despite

Algorithm 1 Deep Q-Network (DQN) Training Algorithm

```
1: Initialize environment CombatBoxing
2: Initialize DQN agent with CNN model $A$
3: Initialize target network $T$ as a copy of $A$
4: Initialize replay buffer $B$ with capacity $10^4$
5: Set learning rate $\alpha = 1e-4$, discount factor $\gamma = 0.99$, initial epsilon $\epsilon = 1.0$, epsilon decay rate
6: for episode = 1 to max_episodes do
7:     Reset the environment and get initial state $s_0$
8:     Set $t = 0$ (time step)
9:     while not done do
10:        Select action $a_t$ using epsilon-greedy strategy based on Q-values
11:        if random() ¡ $\epsilon$ then
12:            $a_t \leftarrow$ random action    ▷ Exploration
13:        else
14:            $a_t \leftarrow \arg\max_a Q(s_t, a; \theta)$    ▷ Exploitation
15:        end if
16:        Execute action $a_t$, observe reward $r_t$ and next state $s_{t+1}$
17:        Store transition $(s_t, a_t, r_t, s_{t+1})$ in replay buffer $B$
18:        Sample a mini-batch of transitions $(s, a, r, s', done)$ from $B$
19:        Calculate target Q-value:
               $y = r + \gamma \max_{a'} Q(s', a'; \theta^-) \times (1 - done)$
20:        Update Q-network by minimizing loss:
               $L(\theta) = \mathbb{E}_{(s,a,r,s',done)\sim B}\left[(y - Q(s, a; \theta))^2\right]$
21:        Perform a gradient descent step on $L(\theta)$ using optimizer
22:        Update target network $T$ periodically:
23:        if episode mod target_update_freq == 0 then
24:            $T \leftarrow A$
25:        end if
26:        Decay epsilon: $\epsilon = \max(\epsilon_{\text{min}}, \epsilon \times \text{decay\_factor})$
27:        Increment $t$
28:    end while
29:    Output average reward for episode $e$
30: end for
```

the challenges posed by adversarial dynamics. The main components of the training framework are described below:

Replay Buffer: To break the temporal correlation in consecutive experiences, a replay buffer with a capacity of 10,000 transitions was employed. The buffer stores tuples of (state, action, reward, next_state, done). During each training step, a random batch of transitions is sampled from the buffer, providing diverse experiences for training. This approach enhances learning stability and mitigates overfitting to recent episodes.

Epsilon-Greedy Exploration: Exploration is crucial for ensuring that the agents adequately explore the action space during training. The epsilon-greedy strategy balances exploration and exploitation by selecting a random action with probability epsilon (ϵ) and the best action (based on the Q-network) otherwise.

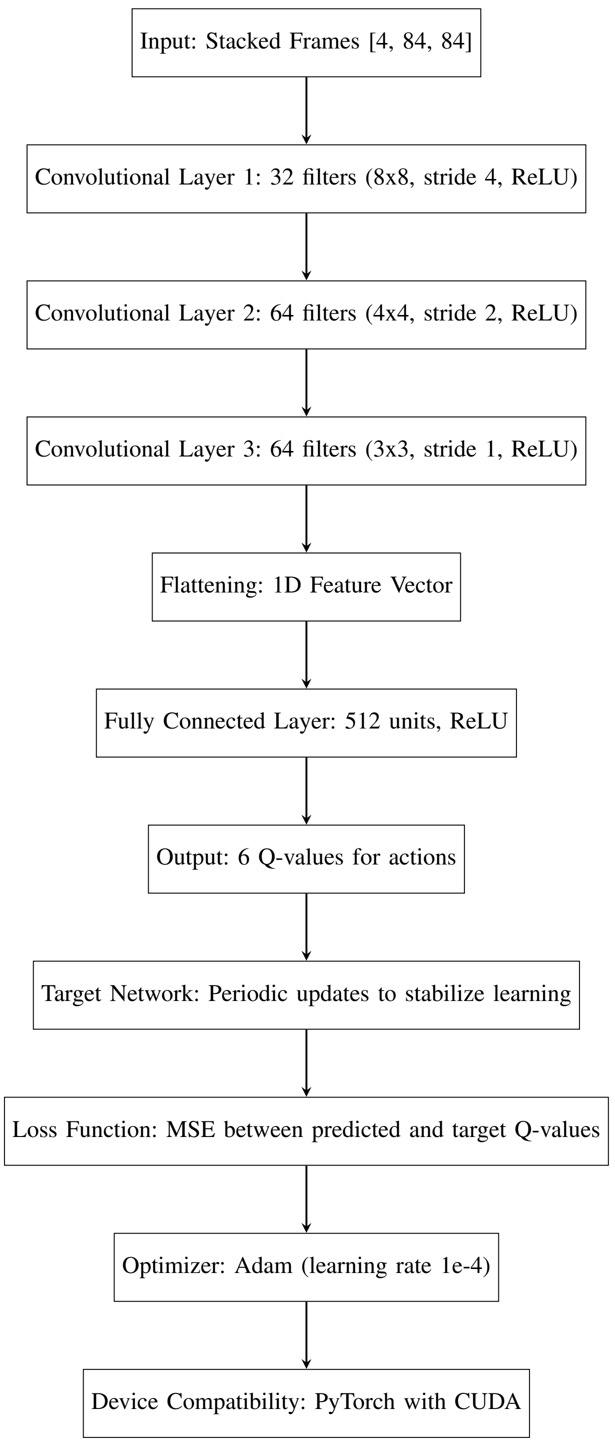
Input: Stacked Frames [4, 84, 84]
Convolutional Layer 1: 32 filters (8x8, stride 4, ReLU)
Convolutional Layer 2: 64 filters (4x4, stride 2, ReLU)
Convolutional Layer 3: 64 filters (3x3, stride 1, ReLU)
Flattening: 1D Feature Vector
Fully Connected Layer: 512 units, ReLU
Output: 6 Q-values for actions
Target Network: Periodic updates to stabilize learning
Loss Function: MSE between predicted and target Q-values
Optimizer: Adam (learning rate 1e-4)
Device Compatibility: PyTorch with CUDA

Fig. 2. DQN Agent Architecture Flowchart

Decay Schedule: The value of ϵ starts at 1.0 and decays exponentially to a minimum of 0.01 over 500 steps. This ensures robust exploration in the initial stages of training and more deterministic behavior as learning progresses.

Q-value updates follow the Bellman equation:

$$Q(s,a) = r + \gamma \max_{a'} Q'(s',a')$$

Where $Q'(s',a')$ is the target Q-value from the target network, γ is the discount factor (set to 0.99), and r is the reward. The loss is computed as the mean squared error (MSE) between predicted and target Q-values.

Batch Training: A mini-batch of size 32 is sampled from the replay buffer at each training step. The loss, computed as the mean squared error (MSE) between the predicted and target Q-values, is minimized using the Adam optimizer.

Target Network Updates: To further stabilize training, the weights of the target network are synchronized with the Q-network every 10 episodes. This periodic update reduces oscillations in Q-value estimation, leading to smoother convergence.

Episode Rewards: Each training episode tracks cumulative rewards for both agents to monitor learning progress. Rewards are used to evaluate performance trends and identify potential bottlenecks in learning.

Checkpointing: Model weights for both agents are periodically saved to ensure recovery in case of interruptions and to allow evaluation of intermediate models. Checkpoints are saved every 50 episodes, capturing the state of the agents throughout training.

The training framework was implemented with modularity in mind, enabling seamless integration of enhancements such as advanced exploration strategies or new reward structures. This design not only facilitated effective learning but also provided a robust foundation for future experiments.

IV. Evaluation Metrics

The performance of the agents was evaluated over 10 independent matches in the Boxing_v2 environment. The evaluation focused on metrics that quantify the agents' ability to compete effectively, including average rewards and qualitative observations of strategy development.

Evaluation Setup: During evaluation, the agents acted deterministically, exploiting their learned policies without exploration. The epsilon value (ϵ) was set to 0 to ensure that actions were solely based on the Q-values computed by the trained networks. Each match was played to completion, with scores recorded for both agents.

The following metrics were used to assess performance:

- **Average Reward:** The cumulative reward averaged over all evaluation matches for each agent.
- **Win Rate:** The proportion of matches won by each agent.
- **Action Diversity:** An analysis of the distribution of actions chosen by each agent.

V. Results

The results from the evaluation of the agents are summarized below, showcasing the performance and strategies developed by each agent.

Average Rewards:

- Agent 1: *-1.7*
- Agent 2: *1.7*

Win Rate:

- Agent 1: *0.3251*
- Agent 2: *0.6749*

Both agents demonstrated diverse action selection patterns, reflecting adaptive strategies during gameplay.

Observations:

- **Agent 1:** Exhibited an aggressive strategy, prioritizing offensive moves and maintaining proximity to the opponent for scoring punches.
- **Agent 2:** Adopted a defensive approach, focusing on evasion and counter-attacks, effectively balancing offensive opportunities with risk minimization.

These results indicate that the agents successfully learned distinct and competitive strategies. While the aggressive approach of Agent 1 yielded higher average rewards in certain scenarios, the defensive strategy of Agent 2 proved effective in achieving a balanced performance across matches as seen in Figure 3.

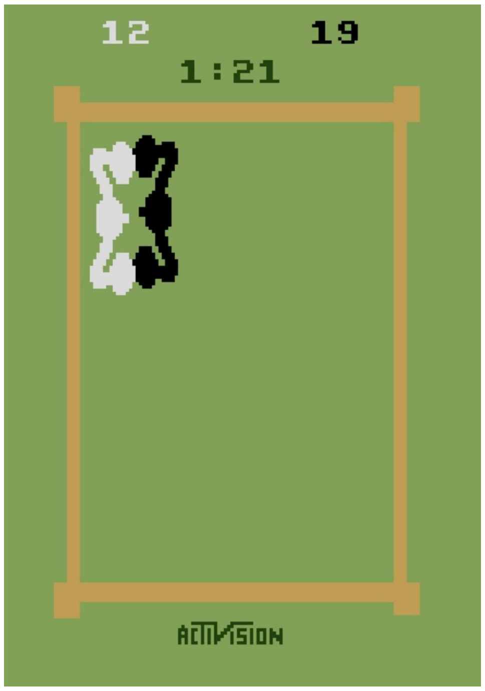

Fig. 3. Combat Boxing game environment with final scores

VI. Discussion

The results from training and evaluation highlight the effectiveness of using independent Deep Q-Networks (DQNs) for adversarial multi-agent reinforcement learning. Several key insights and challenges emerged from this project, which are discussed below:

A. Strengths of the Approach

- **Independent Learning:** By training each agent independently, the framework avoided the complexity of centralized training while allowing each agent to adapt to the opponent's strategies dynamically. This modular approach is scalable to environments with more agents or additional complexity.
- **Efficient Feature Extraction:** The CNN architecture effectively processed spatio-temporal features from stacked frames, enabling the agents to understand game dynamics such as movement patterns and opponent positioning.
- **Stable Training:** Techniques like experience replay and target network updates reduced Q-value oscillations, contributing to smooth convergence during training.
- **Emergence of Strategies:** The agents developed distinct playstyles, with Agent 1 adopting a more offensive approach and Agent 2 demonstrating defensive capabilities. This divergence in strategies underscores the ability of independent DQNs to learn tailored policies based on individual rewards and observations.

B. Challenges and Limitations

- **Non-stationary Environment:** As both agents were learning simultaneously, the environment's dynamics continually shifted, introducing instability in policy optimization. While the use of target networks mitigated this to some extent, further techniques such as opponent modeling could enhance stability.
- **Hyperparameter Sensitivity:** The performance of the agents was highly dependent on hyperparameter choices such as learning rate, epsilon decay, and batch size. Optimizing these parameters required extensive experimentation.
- **Sparse Rewards:** In some episodes, the agents struggled with long periods of sparse or delayed rewards, which slowed convergence. Reward shaping or auxiliary tasks could address this issue in future iterations.

C. Comparison with Related Work

Compared to other approaches in multi-agent reinforcement learning, the independent DQN framework demonstrated competitive results. However, more sophisticated strategies such as self-play or cooperative-adversarial models could further enhance the agents' adaptability and performance.

Overall, the project successfully demonstrated the feasibility of independent reinforcement learning in an adversarial multi-agent setting. While effective strategies emerged, addressing the limitations discussed above could lead to further improvements in agent performance and learning efficiency.

VII. Conclusion and Future Work

In this project, we successfully demonstrated the application of independent Deep Q-Networks (DQNs) for adversarial multi-agent reinforcement learning in the Combat Boxing environment. Through careful training and evaluation, we highlighted the effectiveness of independent learning, where each agent learns its own strategy in a competitive setting. The agents developed distinct strategies, with one adopting an aggressive offensive playstyle and the other focusing on defense. This divergence in approaches demonstrated the ability of DQNs to learn adaptive and competitive strategies in complex environments.

Despite the successful learning of the agents, several challenges were encountered, such as the non-stationary nature of the environment, sensitivity to hyperparameters, and the issue of sparse rewards. These limitations provide valuable insights and open several avenues for future research to further enhance the performance, scalability, and generalizability of the framework.

Several directions for future work are proposed to address these challenges and push the boundaries of independent multi-agent reinforcement learning:

- **Advanced Exploration Strategies:** While the epsilon-greedy exploration strategy provided a solid foundation, more sophisticated methods such as noisy networks or parameterized action space exploration could improve the balance between exploration and exploitation. These methods would allow the agents to discover more diverse strategies and adapt more effectively to complex adversarial dynamics.
- **Opponent Modeling:** Incorporating an opponent modeling mechanism could enable the agents to anticipate and counter their adversary's strategies more effectively. Techniques such as recurrent neural networks (RNNs) or transformer-based architectures could be employed to analyze and predict opponent behaviors based on historical actions, leading to more intelligent decision-making.
- **Reward Shaping and Auxiliary Tasks:** Sparse rewards presented a challenge during training. Designing auxiliary tasks or shaping rewards (such as providing intermediate rewards for specific actions like successful punches or avoiding knockouts) could accelerate learning and improve the robustness of the learned policies.
- **Scalability to Multi-Agent Scenarios:** Extending the framework to environments with more than two agents would test the scalability and adaptability of the approach. Cooperative-competitive dynamics could be explored, where agents must balance collaboration and rivalry to achieve success, further increasing the complexity of the problem.
- **Transfer Learning Across Environments:** To evaluate the generalizability of the framework, it could be tested on other PettingZoo environments or similar multi-agent platforms. Transfer learning techniques could be employed to adapt the policies learned in the Boxing_v2 environment to new scenarios with minimal retraining.
- **Evaluation Against Human Players:** To further validate the effectiveness of the learned policies, the agents could be evaluated against human players. Such experiments could provide insights into the agents' ability to generalize beyond simulated environments and offer opportuni-

ties for improvement based on real-world performance.

- **Incorporating Advanced Architectures:** Recent advancements in reinforcement learning, such as attention mechanisms and graph neural networks (GNNs), could be integrated into the framework to enhance policy learning and decision-making. GNNs, for instance, could enable the agents to reason more effectively about spatial relationships and interactions within the environment.

By addressing these areas, future work can enhance the capabilities of independent multi-agent reinforcement learning, enabling agents to not only compete effectively but also adapt to a wide range of complex environments and scenarios. This work lays a strong foundation for further advancements, and as techniques evolve, so too will the ability of these agents to perform in increasingly sophisticated and dynamic settings. Ultimately, this approach could extend beyond gaming applications to real-world scenarios where multiple agents need to interact, cooperate, or compete in dynamic environments, opening up new possibilities in robotics, autonomous systems, and artificial intelligence.

VIII. Team Contributions

All team members contributed equally to designing and implementing the Deep Q-Network and the multi-agent reinforcement learning framework for the boxing environment. Collaborative efforts were made in developing the agents, training them, and refining their performance. Additionally, the creation of the project presentation and the drafting of the final report were shared responsibilities, ensuring equal participation and input from all members throughout the project.

IX. GitHub Repository

Link to the GitHub Repo.

References

[1] K. Lee, S. Subramanian, and M. Crowley, ”Investigation of independent reinforcement learning algorithms in multi-agent environments,” *Frontiers in Artificial Intelligence*, vol. 5, 2022.

[2] R. Pina, et al., ”Fully Independent Communication in Multi-Agent Reinforcement Learning,” *Adaptive Agents and Multi-Agent Systems*, 2024.

[3] W. Li, Z. Ding, S. Karten, and C. Jin, ”FightLadder: A Benchmark for Competitive Multi-Agent Reinforcement Learning,” *2024*.

[4] L. Busoniu, R. Babuska, and B. De Schutter, ”A Comprehensive Survey of Multiagent Reinforcement Learning,” IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 38, no. 2, pp. 156-172, 2008.

[5] M. Lanctot, V. Zambaldi, A. Gruslys, et al., ”A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning,” Advances in Neural Information Processing Systems, vol. 30, pp. 4190-4203, 2017.

[6] H. van Hasselt, A. Guez, and D. Silver, ”Deep Reinforcement Learning with Double Q-Learning,” AAAI Conference on Artificial Intelligence, pp. 2094-2100, 2016.

[7] T. Tampuu, M. Matiisen, D. Kodelja, et al., ”Multiagent cooperation and competition with deep reinforcement learning,” PLoS ONE, vol. 12, no. 4, 2017.

[8] L. Peng, B. Liu, and Q. Wei, ”A Survey of Multi-Agent Deep Reinforcement Learning: Current Status and Future Directions,” IEEE Access, vol. 9, pp. 56755-56782, 2021.