AI Boxing Simulation Report

Xuyang Wang, Hongzhen Xu, Ziyi Song, Min Sun

Abstract

Background & Issue: Our study focuses on the behavior of such agents in simulated environments, specifically through the classic "Boxing" using the PettingZoo library and PyTorch. We originally intended to explore a multi-agent system in a battle-royal-like setting. However, our focus shifted towards understanding how agents can refine their learning and decision-making processes specifically within boxing. Training an agent to make strategic decisions from visual cues and maximize its performance against dynamic opponents presents significant challenges. A crucial aspect of this challenge is the agent's ability to balance the exploration of new actions with the exploitation of current strategies for continual improvement. Furthermore, we aim to assess the development of agents under different decision-making policies—namely, the Upper Confidence Bound (UCB) and epsilon-greedy policies—to discern their impacts on the agent's learning efficacy and strategy evolution.

Purpose: This project seeks to implement reinforcement learning, enhanced with a Deep Q-Network (DQN) integrated with Convolutional Neural Networks (CNNs), to train agents under two distinct exploration policies: UCB and epsilon-greedy. Our objective is to gain insights into agent behavior and strategy development within a simulated boxing environment and to evaluate the effectiveness of the UCB policy, in contrast to the epsilon-greedy approach.

Model / Methodology

Environment

Using the Atari Boxing from PettingZoo, two agents compete against each other to simulate a one-on-one boxing match between the agent and an opponent. The agent receives visual observations of the game's state as input, which are represented by 84x84 grayscale images. The agent's action space comprises discrete actions that correspond to various boxing moves and strategies, including movement directions, firing directions, firing, and no operation.

Deep Q-Network

Our agent's decision-making process is modeled within a Deep Q-Network (DQN). This model starts with an input layer designed to accept preprocessed frames from the game environment. These frames are resized to 84x84 pixels and converted to grayscale. Next, the architecture has two convolutional neural networks (CNN), each followed by max-pooling layers. These layers are used for processing the visual inputs and extracting relevant features.

The first convolutional layer applies a kernel size of 8 with a stride of 4, while the second layer uses a kernel size of 4 with a stride of 2. The extracted features are then flattened and passed through two fully connected layers. The initial layer, with an output size of 64 neurons (UCB variant uses 48), serves as a compact representation of these features, whereas the subsequent layer maps these compressed features to the action space, predicting the expected return for each possible action. In our DQN model for UCB, an additional fully connected layer runs in parallel to the standard output layer. This layer is specifically used for computing exploration metrics for the UCB policy. This addition allows our DQN to not only estimate the potential rewards associated with each action but also assess the degree of uncertainty or exploration value of these actions, allowing a more informed and strategic balance between exploring new actions and exploiting known strategies.

Exploration Policies

Epsilon-Greedy: A strategy that selects actions randomly with probability ε. With probability ε, the agent selects a random action for exploration, and with probability (1 - ε), it selects the action with the highest Q-value for exploitation. The value of ε is decayed over time to shift from exploration to exploitation.

Upper Confidence Bound (UCB): A strategy that selects actions by considering both the estimated Q-values and an exploration bonus (confidence interval around each action's Q-value). This bonus is calculated based on the number of times each action has been selected, encouraging the agent to explore less frequently chosen actions.

Training

The agents are trained iteratively through episodes of interaction with the environment. At each timestep, an action is taken according to current policy, after which the agent observes the next state and reward, and updates its model based on the transition observed. The epsilon-greedy strategy is used to populate the replay buffer with a diverse array of experiences, ensuring a balanced mix of exploration and exploitation. The Upper Confidence Bound (UCB) policy not only considers the estimated Q-values but also uses an exploration bonus. The model is refined in mini-batches through experiences randomly sampled from the replay buffer, aiming to minimize the discrepancy between the predicted Q-values and the target Q-values. These target values are determined from observed rewards, using the principle of temporal difference. The model is updated via gradient descent to reduce the mean squared error between its predicted Q-values and these target values. In our experiments, we attempted to train two agents with the same DQN model. However, this resulted in both agents exhibiting identical actions, thereby hindering score accumulation. To resolve this, one agent was programmed to act randomly. Regarding rewards, to incentivize engagement, the reward for scoring was amplified by a factor of 1000, while penalties for

losing points were multiplied by 10. Under the UCB policy, idle states which the agent is neither scoring nor losing, were assigned a minor penalty of -1 to discourage inactivity.

Evaluation Metrics

$total\_reward$: The cumulative rewards gained by the agent per episode, measuring the agent's performance within a single episode.

get_points & $lose\_points$: The points the agent gains for receiving positive rewards and the points it loses for receiving negative rewards, respectively.

$total\_points$: The total rewards or scores obtained by the agent across all episodes, measuring the agent's performance throughout the training period.

Hyperparameters

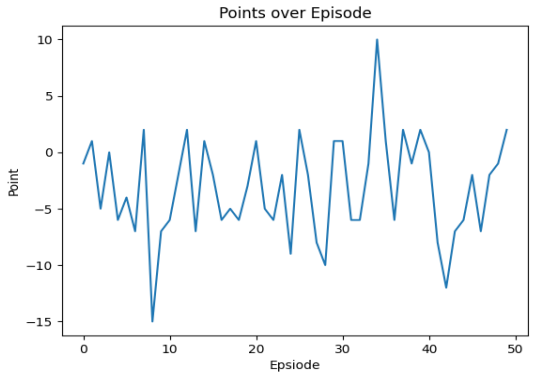
- $state\_size$: (84, 84), representing the dimensions of the preprocessed input frames.
- $learning\_rate$: 0.00025, controlling the step size of the model's weight updates.
- $discount\_rate$ ($gamma$): 0.99, determining the importance of future rewards.
- $memory\_size$: 100000, defining the capacity of the replay buffer.
- $batch\_size$: 64, specifying the number of experiences used in each training iteration.
- $epsilon$: 1.0, initial exploration rate for epsilon-greedy policy.
- $epsilon\_decay$: 0.99, decay rate for epsilon; $epsilon\_min$: 0.01, min value for epsilon.
- $num\_episodes$: 500, defines the total number of training cycles.
- UCB_C: 1.0, balancing exploration and exploitation in UCB policy.

Result Analysis

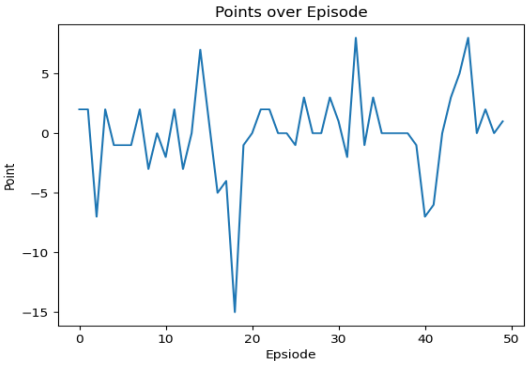
DQN Agent VS. DQN Agent

When first employed two agents with identical DQN architectures, we observed that after 100 episodes of training, neither agent scored. This lack of scoring can be due to the agents mirroring each other's strategies too closely. Such symmetry in behavior suggests that when agents share the same decision-making model, they may fail to develop the diverse strategies. Therefore, it may be necessary to introduce diversity in training or agent architectures.

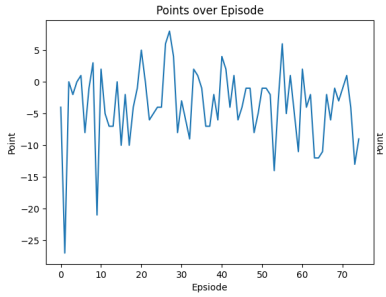
UCB DQN Agent VS. Random Opponent



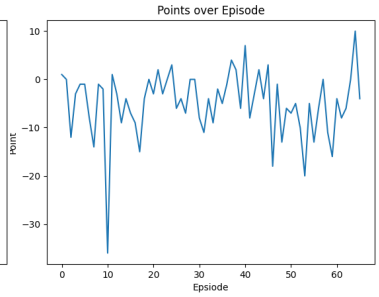
1-50



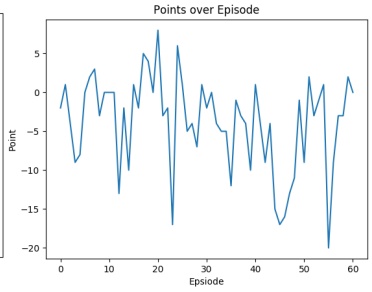
51-100



101-180



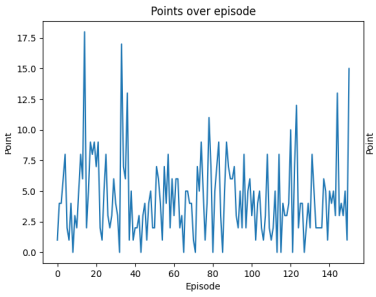
181-250



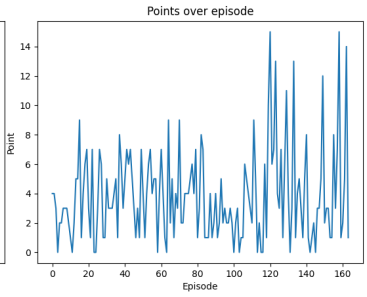
251-310

Overall, the results indicate that the UCB-based DQN is learning and improving its performance over the course of 310 episodes. The agent initially explores widely, then gradually shifts towards exploiting its learned knowledge while still maintaining some exploration. The performance oscillates but shows some improvement, particularly in episodes 181-250. The downward in the later episodes suggests that further training and refinements to the model architecture and hyperparameters is needed.

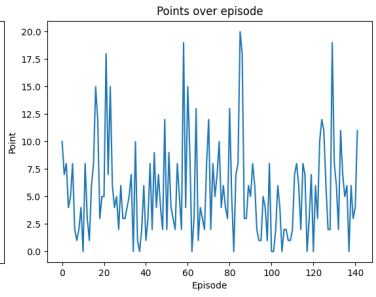
Epsilon-Greedy DQN Agent VS. Random Opponent



1-160 Episodes



161-340 Episodes



341-500 Episodes

Based on the results of the Epsilon-Greedy policy results, we can see that the DQN agent is learning and improving its performance over 500 episodes. The agent initially explores extensively, resulting in high variability of points. As the training progresses, the agent starts to balance exploration and exploitation, leading to some improvement in performance. In the later episodes, the agent's performance becomes a bit more consistent, the points are mostly concentrated in the range of 5 to 15.

We observed that during one of our training sessions, the agent appeared to learn its most effective strategy: cornering its opponent could significantly increase its points. As a result, the agent began to corner its opponent at the start of each game, effectively maximizing its scoring potential by exploiting this tactical advantage.

The highest score reached by the agent under the UCB policy is approximately 10. We had anticipated a better performance given the UCB policy's approach to balancing exploration and exploitation. Conversely, the agent trained with the Epsilon-Greedy policy reached a peak score of about 20. Throughout all the training sessions, the highest score achieved by the Epsilon-Greedy policy was 38. The UCB policy's learning graph, which doesn't show stabilized performance over 300 episodes, points to the need for more extensive training and tuning. In contrast, the Epsilon-Greedy policy's learning trajectory over 500 episodes displays a gradual upward trend, signifying more effective learning. Neither policy has led to stabilized performance within the given training duration, with continued fluctuations in points, indicating that a longer training timeframe is required.

In theory, the UCB policy offers a more structured exploration-exploitation balance, which could lead to a steadier improvement overtime. On the other hand, the Epsilon-Greedy method relies on a decreasing probability of random action selection, which might allow for higher peaks in the training as the agent explores aggressively but could also introduce more volatility. For the current implementation, the Epsilon-Greedy policy appears to be the better performer in terms of peak scores achieved and the overall learning curve.

Discussion

Limitations

Hardware: Our computational resources restricted the scope, affecting depth of neural network we could explore and the number of agents we could use.

Library: PettingZoo has constraints in terms of customization and scalability.

Frame Processing: Our model currently processes one frame at a time. Using multiple

consecutive frames could provide context that may lead to more informed decision-making.

Training Duration: With each episode taking four minutes to complete, the lengthy training time limits the amount of feedback and learning for the agents within a reasonable timeframe.

Exploration Strategies: Our strategies haven't been diverse enough to allow agents escaping local optima in the strategy space, which might lead to suboptimal long-term performance.

Future Improvements

We could enhance our model by adjusting hyperparameters, like the exploration and learning rates, to achieve a better mix of exploration and exploitation. Also, using more sophisticated exploration techniques could lead the agents to find better strategies. Trying out various network designs, ways to shape rewards, and ways for agents to work together would be useful too. Another improvement involves using sequences of images for input instead of just one. This change means the model would learn from a series of actions and situations, not just one moment, making training much more effective.

Conclusion

This project demonstrates the applicability of reinforcement learning, particularly the Deep Q-Network integrated with convolutional neural networks, in training agents within the Atari Boxing environment by PettingZoo. Our findings show the differences in performance between the Upper Confidence Bound and epsilon-greedy exploration policies, with the latter showing a more consistent and higher peak performance over extended training episodes. Moving forward, we'll refine by incorporating more complex neural architectures and multi-frame input could further improve the strategic depth and consistency of the agents.

Contributions

Xuyang: In charge of Proposal and this Report, analyzed performance data.

Min: Set the project's scope, co-authored the final report, analyzed codes.

Hongzhen: Created the abstract, game dynamics, coded and trained epsilon-greedy agent.

Ziyi: Led the training and implementation of the UCB policy.

GitHub: **https://github.com/hx394/FoundationOfAIGroupProject**

Reference: **https://pettingzoo.farama.org/environments/atari/boxing/**