TEMPLE UNIVERSITY

Value-Based Reinforcement Learning in Sandbox Environment

Yihan Zhang

1 INTRODUCTION

A fundamental concept in machine learning involves sequential decision-making, where the objective is to determine, based on past experiences, the series of actions to take within an uncertain environment to accomplish specific goals. This type of task has broad applications across various domains, including but not limited to robotics, healthcare, smart grids, finance, self-driving cars, and numerous other fields. The potential impact of mastering sequential decision-making is substantial, offering valuable insights and advancements in diverse areas of technology and industry.

Drawing inspiration from behavioral psychology, particularly as discussed in [Sut84], reinforcement learning (RL) introduces a structured framework for addressing this challenge. The central concept involves an artificial agent learning through dynamic interactions with its environment, mirroring the learning process observed in biological agents. Through the accumulation of experiences, the artificial agent aims to optimize predefined objectives, represented in the form of cumulative rewards.

In this project, I've crafted a sandbox environment and devised a range of actions for agents to execute. The primary aim is to train diverse models with specific focuses and subsequently assess their performance by observing how the agents respond to varying situations and evaluating their final scores. The central methodology involves the utilization of the Deep Q Network ([Gu+16]) to train the online model, leveraging its capacity for approximating optimal action-value functions. Additionally, sophisticated techniques such as experience replay [Foe+17] are employed to enhance model performance. Experience replay involves systematically storing and randomizing past encounters, providing the models with a diverse training set.

2 Related Works

2.1 Deep Reinforcement Learning

In recent years, Reinforcement Learning (RL) has witnessed a surge in popularity, attributed to its efficacy in tackling intricate sequential decision-making challenges. Notably, the integration of RL with deep learning techniques, as highlighted by influential works such as [LBH15], [Sch15], has played a pivotal role in these achievements. Termed deep RL, this amalgamation proves particularly advantageous in scenarios characterized by high-dimensional state-spaces.

Unlike previous RL approaches that grappled with feature selection challenges ([Bel+13]), deep RL excels in tasks with limited prior knowledge. Its success lies in the ability to autonomously learn diverse levels of abstractions directly from raw data. A notable example is the capacity of a deep RL agent to proficiently learn from visual perceptual inputs, composed of thousands of pixels, as demonstrated by [Mni+15]. This ability to extract meaningful insights from complex and unstructured data sets deep RL apart, showcasing its potential for addressing real-world problems with minimal pre-defined features.

2.2 Deep Q Network

The DQN (Deep Q Network [Mni+15]) shares conceptual roots with the model introduced by [LRV12], yet it stands out as the pioneering RL algorithm capable of directly operating on raw visual inputs across diverse environments. Specifically crafted to process raw visual data, the DQN's design involves a final fully connected layer that produces Q values for all actions within a discrete set. This set encompasses various actions. This design not only facilitates the selection of the best action with a single forward pass but also allows the network to effectively encode action-independent knowledge in its lower convolutional layers. Tasked with maximizing its score in a video game, the DQN autonomously learns to discern crucial visual features, encompassing objects, their movements, and, notably, their interactions. Leveraging techniques initially developed for interpreting the behavior of Convolutional Neural Networks (CNNs) in object recognition tasks, we can scrutinize the agent's visual focus, shedding light on the aspects it deems important.

2.3 Experience Replay

In the realm of online learning, the agent can leverage a replay memory, as introduced by [Lin92]. This mechanism enhances data efficiency by storing the agent's past experiences, allowing for their reprocessing at a later stage. The replay memory serves a dual purpose, not only preserving a history of the agent's interactions but also facilitating stability in mini-batch updates. This is achieved by ensuring that updates draw from a relatively stable data distribution stored in the replay memory, particularly when the size of the replay memory is sufficiently large, contributing to convergence and overall stability.

This approach proves particularly advantageous in the context of off-policy learning, where utilizing experiences from different policies poses no bias issues and can even enhance exploration. Methods employing algorithms like DQN or model-based learning can effectively and safely incorporate a replay memory in their training processes. In an online setting, the replay

memory retains information for the last N time steps, with the specific value of N determined by the available memory resources. This strategic use of a replay memory stands as a key element in optimizing the efficiency, stability, and performance of online learning algorithms.

3 Environment Design

To gain a deeper understanding of the nature of deep reinforcement learning, I opted not to use an existing game environment in this project. Instead, I designed my own environment along with corresponding rewards. The general idea of this environment is as follows:

Firstly, the agent starts with a certain initial life and has two actions. One action is named "sleep," which carries no risk but yields a small reward. The second action is named "attack," which has a random chance of success or failure. If successful, the agent receives a large reward; if unsuccessful, there is no reward, and a substantial amount of life is deducted.

In each iteration, the agent experiences a slight reduction in health, and the success probability of the "attack" action periodically increases and resets. If the agent chooses "sleep" every round, it is certain to lose the game. Therefore, I expect the agent to predominantly choose "sleep" in most iterations and to opt for "attack" when the success probability is high. The specific design of the environment is outlined in Algorithm 1.

4 Experiment

4.1 Experiment Setting

In this experiment, I use a batch size of 128, set the replay memory length to 10^5, and configure the reward decay rate as 0.99. For action selection, I follow an ϵ-greedy policy, where ϵ starts at 0.9 and ends at 0.05, with a decay rate of 1,000. As for the optimizer, I employ the Adam optimizer with a learning rate of 0.0001.

For the model, I employed a three-layer fully connected neural network as the evaluation for the action value function. The input to this network comprises the current life and score of the agent, while the output provides the values for the two available actions. During each selection, the action with the higher output is chosen as the action to be executed at the current iteration.

4.2 Results for Different Action Selection Mode

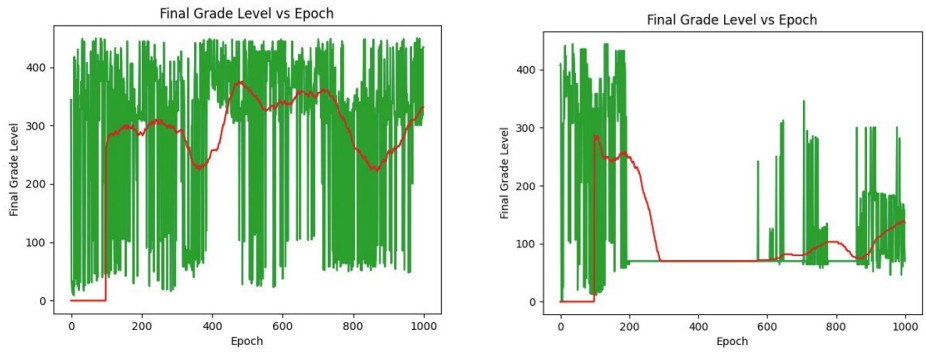
Regarding the impact of random action selection, I kept other parameters unchanged, removed the ϵ-greedy policy, and compared the results with those obtained using the ϵ-greedy policy, as shown in the Fig.4.1.

From the Fig.4.1, it can be observed that without random action selection, the agent's final score remains around 70, indicating that the agent consistently chooses "sleep" in each iteration. On the other hand, with random action selection, the model gradually learns to choose "attack" at appropriate time.

The reason behind the result without random action selection, in my view, lies in the fact that the success probability of the "attack" action is initially low at the beginning of each

Algorithm 1 Environment Definition and Reward Definition

Initialization $life = 70$ $score = 0$
Goal $score = 300$
for `each iteration` **do**
  $life = life - 1$
  $reward = 0$
  Agent chooses an $action$ among 2 different actions: sleep and attack
  if $action$ is sleep **then**
    $score = score + 1$
    $reward = 1$
  else if $action$ is attack **then**
    compute the passing rate $p = 0.015 * (grade - 100 * level)$ where $level = grade//100$
    Draw a random number $rand$ uniformly from 0-1
    if $rand \leq p$ **then**
      $score+ = 0.5 * score$
      if $life \leq (100 - (score\%100))$ and $p > 0.6$ **then**
        $reward = 10000$
      else if then
        $reward = 0.5 * score$
      end if
    else if $rand > p$ **then**
      $life = life - 5$
      $reward = -5$
    end if
  end if
  At the end of iteration,
  if $life \leq 0$ **then**
    Terminate the current epoch
  else if $score \geq 300$ **then**
    $reward = 10000$
    Terminate the current epoch
  else if $oldscore < 100(200)$ and $newscore \geq 100(200)$ **then**
    $life = life + 70$
  end if
end for



(a) Results with Random Action Selection (b) Results without Random Action Selection

Figure 4.1: Final Grade vs Epoch for Different Action Selection Mode

epoch. Consequently, the action-value function for "attack" will be much lower than that for "sleep." In such a scenario, the model tends to favor "sleep" over time. Once the random action selection mechanism is removed, without the randomness, "sleep" quickly dominates the experience replay list. Even if occasional "attack" actions are sampled for updating the model, since these "attack" instances occur early in the training when the expected reward is negative, it exacerbates the model's bias, ultimately preventing effective updates.

The underlying reason for this phenomenon, in my opinion, is that deep Q-learning is essentially a form of conventional evaluation algorithm. However, in this case, the data is not sourced externally but is derived from the decisions made by the model during training and the rewards from the environment. Without random action selection, deep Q-learning generates severely imbalanced and homogenized data, making it challenging to train a meaningful evaluation model.

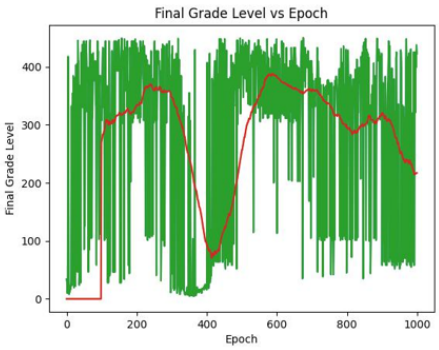
4.3 Results for Different Reward Definition

To investigate the impact of reward definition, I removed all additional reward definitions, specifically those associated with particular actions at specific times. I retained only the score changes upon winning and the score changes resulting from the increase or decrease in grade per iteration. The following Fig.4.2 is a comparison between the results with additional rewards and without additional rewards.

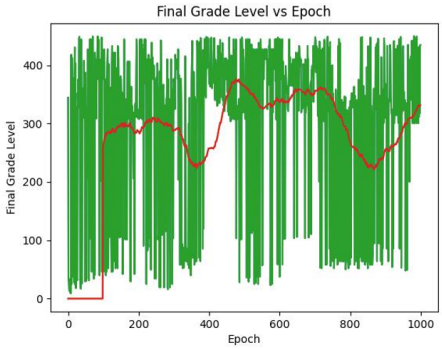
From the Fig.4.2, it is evident that without defining additional rewards, the model's update process becomes more unstable, and the convergence speed is slower. This indicates that the rewards defined in my environment are reasonable.

4.4 Results for Different Difficulty Settings

To better understand the impact of varying levels of difficulty in winning, I conducted experiments with different difficulty settings. In the more challenging setting, I increased the penalty for each unsuccessful attack from a life deduction of 5 to 20. The results are depicted in the following Fig.4.3.

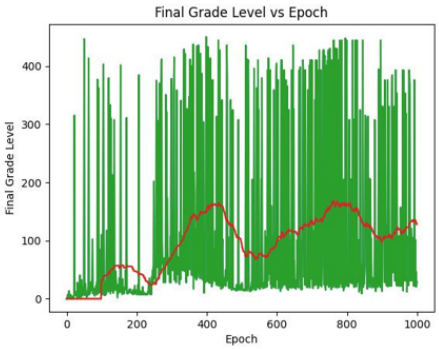


(a) Results without Extra Reward Definition

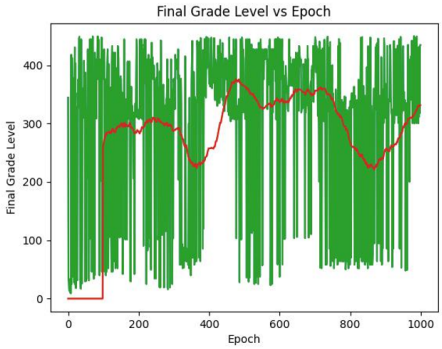


(b) Results with Extra Reward Definition

Figure 4.2: Final Grade vs Epoch for Different Reward Definition



(a) Results with High Difficulty



(b) Results with Low Difficulty

Figure 4.3: Final Grade vs Epoch for Different Difficulty Settings

From the Fig.4.3, it is evident that under the high difficulty setting, it is challenging to train a model that consistently makes correct choices. This is because when the life penalty is adjusted to 20, treating the initial model as a randomly choosing model, it struggles to generate data points for successful attacks.

The specific reason is that, in the early stages, a random choice of "attack" is likely to fail, and due to the substantial life penalty, the agent may not have enough life to progress the game to the stage with a high success rate for attacks. Once this situation occurs, the imbalance in the data distribution makes it difficult for the model to update accurately.

5 Conclusion

In this project, I delved into the foundational paradigm and processes of deep reinforcement learning, gaining a comprehensive understanding of the concepts and details of deep Q-learning. I also explored and applied the experience replay technique. In specific experiments, I successfully designed a sandbox-like environment, defining sensible formulas for

environment transitions and rewards. Additionally, I compared and analyzed the model's performance under different difficulty settings, reward definitions, and action selection modes. This project provided me with a thorough insight into the practical application of deep reinforcement learning and how adjusting the environment and parameters can impact the model's performance.

6 Acknowledgements

Thank you to Prof. Wang for introducing us to numerous technologies in the broad field of artificial intelligence in this course. This has provided me with a general understanding of artificial intelligence technologies beyond deep learning. I also appreciate the professor's exploration of more fundamental and philosophical questions in artificial intelligence during the course. This has allowed me to broaden my knowledge of the entire field of artificial intelligence while focusing on my specific area.

In addition, I would like to express my gratitude to Bowen Xu and Lei Wang. I consulted them for many questions during the course.

References

[Sut84] Richard Stuart Sutton. "Temporal Credit Assignment in Reinforcement Learning". AAI8410337. PhD thesis. 1984.

[Lin92] Long-Ji Lin. "Self-improving reactive agents based on reinforcement learning, planning and teaching". In: *Machine learning* 8 (1992), pp. 293–321.

[LRV12] Sascha Lange, Martin Riedmiller, and Arne Voigtländer. "Autonomous reinforcement learning on raw visual input data in a real world application". In: *The 2012 international joint conference on neural networks (IJCNN)*. IEEE. 2012, pp. 1–8.

[Bel+13] Marc G Bellemare et al. "The arcade learning environment: An evaluation platform for general agents". In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 253–279.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *nature* 521.7553 (2015), pp. 436–444.

[Mni+15] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[Sch15] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[Gu+16] Shixiang Gu et al. "Continuous deep q-learning with model-based acceleration". In: *International conference on machine learning*. PMLR. 2016, pp. 2829–2838.

[Foe+17] Jakob Foerster et al. "Stabilising experience replay for deep multi-agent reinforcement learning". In: *International conference on machine learning*. PMLR. 2017, pp. 1146–1155.