REINFORCEMENT LEARNING FINAL PROJECT REPORT

Jiaqi Su
Shanghai Jiao Tong University
Shanghai
sjq2022@sjtu.edu.cn

ABSTRACT

Deep Reinforcement Learning (DRL) has emerged as a powerful paradigm for solving complex sequential decision-making problems. This report presents a comprehensive implementation and evaluation of two prominent families of model-free DRL algorithms: policy-based and value-based methods. For policy-based learning, we implement Proximal Policy Optimization (PPO), a state-of-the-art algorithm renowned for its stability and performance, and test it on high-dimensional continuous control tasks from the MuJoCo suite. For value-based learning, we trace the evolution from the foundational Deep Q-Network (DQN) to its advanced variant, Dueling Double DQN (D3QN), and evaluate its performance on challenging classic Atari games. Our work provides a detailed analysis of the theoretical underpinnings of these algorithms, a robust and modular implementation, and empirical results that highlight their respective strengths and weaknesses in different domains. The code is released at `https://github.com/SuJQ0722/SJTU_CS3316_RL.git`.

***K*eywords** Proximal Policy Optimization · Deep Q-Network · Deep Reinforcement Learning

1 Introduction

Learning to control agents directly from high-dimensional sensory inputs like vision and speech is one of the long-standing challenges of reinforcement learning (RL). Most successful RL applications that operate on these domains have relied on hand-crafted features combined with linear value functions or policy representations. Clearly, the performance of such systems heavily relies on the quality of the feature representation.

With the explosive growth of deep learning, it becoms possible to extract high-level features from raw sensory data, leading to breakthroughs in computer vision and speech recognition. These methods utilise a range of neural network architectures, including convolutional networks, multilayer perceptrons, restricted Boltzmann machines and recurrent neural networks, and have exploited both supervised and unsupervised learning. It seems natural to ask whether similar techniques could also be beneficial for RL with sensory data. Recently, The advent of deep neural networks as powerful function approximators has given rise to Deep Reinforcement Learning (DRL), enabling the application of RL to previously intractable problems with high-dimensional state and action spaces, such as playing Atari games from raw pixels [1] and mastering complex robotic control [2].

DRL algorithms can be broadly categorized into value-based and policy-based methods. Value-based methods, such as Deep Q-Network (DQN) [1], learn a value function that estimates the expected return of taking an action in a given state, and derive a policy implicitly from this function. Policy-based methods, such as Proximal Policy Optimization (PPO) [2], directly learn a parameterized policy that maps states to actions. While value-based methods are often more sample-efficient, policy-based methods can handle continuous action spaces more naturally and often exhibit better convergence properties.

This project aims to provide a hands-on, in-depth exploration of these two DRL paradigms. Our contributions are threefold:

1. We provide a clear exposition of the theoretical progression from DQN to Dueling Double DQN (D3QN), and the principles behind PPO.

2. We develop a modular and reusable codebase for these algorithms, tested on standard benchmarks: Atari 2600 games for D3QN and MuJoCo continuous control tasks for PPO.

3. We conduct a thorough empirical analysis of the algorithms' performance, discussing their stability, sample efficiency, and the effectiveness of their core components.

2 Related Works

The field of reinforcement learning (RL) has progressed from foundational tabular methods to powerful deep learning-based approaches capable of solving complex, high-dimensional problems. This evolution has been primarily driven by two parallel research thrusts: value-based and policy-based methods.

The lineage of policy-based methods began with the REINFORCE algorithm [3], which established the core principle of policy gradient estimation via Monte Carlo rollouts. To mitigate the high variance inherent in this approach, actor-critic architectures were introduced [4], which learn a value function (the critic) to provide a lower-variance estimate of the return, guiding the policy update (the actor). This paradigm was further refined by algorithms like Asynchronous Advantage Actor-Critic (A3C) [5], which leveraged parallelization to stabilize training. However, policy updates remained sensitive to step size. To address this, Trust Region Policy Optimization (TRPO) [6] introduced a theoretically-grounded constraint on the KL divergence between old and new policies. Its successor, Proximal Policy Optimization (PPO) [2], achieved similar stability with a simpler clipped surrogate objective, making it one of the most robust and widely used algorithms for continuous control today.

Concurrently, value-based methods have advanced from early temporal-difference learning. Q-learning [7] laid the groundwork with its off-policy update rule, which learns the optimal action-value function independently of the agent's exploration policy. This off-policy nature proved crucial for sample efficiency and was a key component in the Deep Q-Network (DQN) [1]. DQN successfully scaled Q-learning to high-dimensional visual domains, such as the Atari 2600 suite, by integrating deep convolutional networks with an experience replay buffer and a separate target network. Subsequent work addressed DQN's limitations: Double DQN [8] mitigated value overestimation, Dueling DQN [9] improved value function representation by decoupling state values and action advantages, and Prioritized Experience Replay [10] improved sample efficiency by replaying important transitions more frequently. These enhancements were later unified in the Rainbow algorithm [11].

In recent years, the lines between these two families have blurred. State-of-the-art algorithms for continuous control, such as Soft Actor-Critic (SAC) [12] and TD3 [13], combine off-policy learning from a replay buffer with an explicit actor, leveraging the strengths of both paradigms to achieve remarkable sample efficiency and performance. Our work focuses on implementing and analyzing canonical representatives from both core lineages: PPO for policy-based control and D3QN, a combination of Double and Dueling DQN, for value-based control. A more detailed introduction will be presented in subsection 2.1 and 2.2. These algorithms are evaluated on standard benchmark environments—the high-dimensional continuous control tasks from MuJoCo [14] and the classic pixel-based Atari 2600 games from the Arcade Learning Environment [15], respectively—which have been instrumental in driving and measuring progress in the field.

2.1 Value-Based Reinforcement Learning

Deep Q-Network (DQN) Mnih et al. [1] first successfully combined Q-learning with a deep convolutional neural network, creating the DQN. To stabilize training, they introduced two key innovations: (1) an **experience replay buffer**, which stores past transitions and samples mini-batches from them to break temporal correlations, and (2) a separate **target network**, which is a periodically updated copy of the online network used to compute the TD target, reducing oscillations in the learning process.

Double DQN (DDQN) DQN suffers from a systematic overestimation of Q-values due to the max operator in the TD target. Van Hasselt et al. [8] proposed Double DQN, which decouples the action selection from the action evaluation. The TD target is modified as:

$$y_t = r_t + \gamma Q_{\theta'}(s_{t+1}, \arg\max_{a'} Q_\theta(s_{t+1}, a')) \tag{1}$$

where θ are the parameters of the online network and θ' are the parameters of the target network.

Dueling Network Architecture (D3QN) Wang et al. [9] introduced a network architecture that explicitly separates the representation of state values and action advantages. The Q-value is composed as:

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s,a';\theta,\alpha) \right) \tag{2}$$

where $V(s)$ is the state value and $A(s,a)$ is the action advantage. This architecture allows the network to learn the value of a state without having to learn the effect of each action, leading to better policy evaluation in the presence of many similar-valued actions. When combined with Double DQN, it forms the Dueling Double DQN (D3QN) algorithm.

2.2 Policy-Based Reinforcement Learning

Policy gradient methods directly optimize a parameterized policy $\pi_\theta(a|s)$ by performing gradient ascent on the expected return objective $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]$. The policy gradient theorem provides the gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta} \left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^\pi(s_t,a_t) \right] \tag{3}$$

where $A^\pi(s_t,a_t)$ is the advantage function.

Proximal Policy Optimization (PPO) While effective, vanilla policy gradient methods can suffer from high variance and instability due to large policy updates. Schulman et al. [2] introduced PPO, which constrains the policy update to a "trust region" to prevent destructively large updates. PPO's main innovation is its clipped surrogate objective function:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{4}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio and ϵ is a small hyperparameter. This objective clips the ratio, effectively penalizing policy changes that move $r_t(\theta)$ outside the interval $[1-\epsilon, 1+\epsilon]$. PPO combines this with a value function loss and an entropy bonus to encourage exploration.

3 Algorithm Analysis

3.1 Dueling Double DQN (D3QN)

Our implementation of D3QN integrates the principles of Double DQN and the Dueling Network architecture. The agent learns from experiences stored in a replay buffer. The update rule follows Equation (2), and the network architecture follows Equation (3). A convolutional neural network is used to process the raw pixel inputs from the Atari environments. The full algorithm is detailed in Algorithm 1 and the architecture of D3QN is shown in Fig 1.

3.2 Proximal Policy Optimization (PPO)

Our PPO implementation uses an Actor-Critic architecture, with separate networks for the policy (Actor) and value function (Critic). We employ Generalized Advantage Estimation (GAE) [16] for stable advantage estimation. The temporal-difference error and GAE are calculated as follows:

$$\delta_t = r_t + \gamma\, V_\phi(s_{t+1}) - V_\phi(s_t) \tag{5}$$

$$\hat{A}_t = \sum_{\ell=0}^{T-t-1} (\gamma\lambda)^\ell\, \delta_{t+\ell}\,, \tag{6}$$

GAE can also be expressed recursively as follows:

$$\hat{A}_t = \delta_t + \gamma\lambda\, \hat{A}_{t+1}\,, \tag{7}$$

which is more frequently used in coding.

The agent collects a batch of trajectories and then performs multiple epochs of stochastic gradient ascent on the clipped surrogate objective. The total loss function is:

$$L(\theta) = L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \tag{8}$$

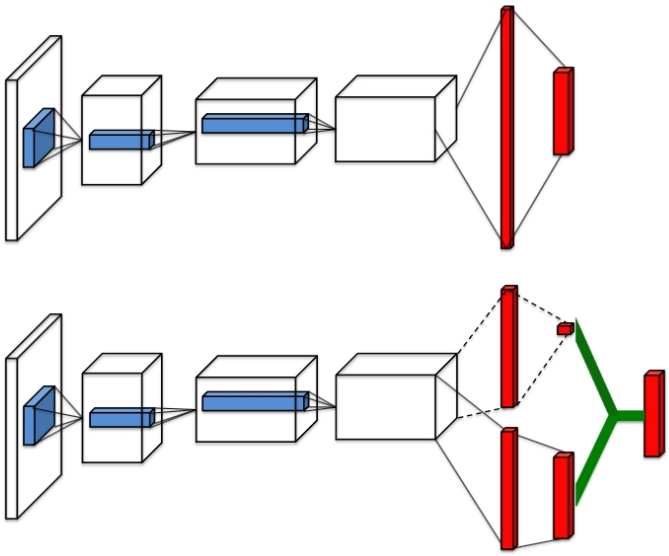


Figure 1: A popular single stream Q-network (top) and the dueling Q-network (bottom). The dueling network has two streams to separately estimate (scalar) state-value and the advantages for each action; the green output module is used to combine them. Both networks output Q-values for each action.

Algorithm 1 Dueling Double DQN (D3QN)

Initialize replay buffer $\mathcal{D}$ to capacity N
Initialize action-value network Q with random weights θ
Initialize target network $\hat{Q}$ with weights $\theta' \leftarrow \theta$
for episode = 1, M **do**
    Reset environment and get initial state s_1
    for t = 1, T **do**
        With probability ϵ select a random action a_t, otherwise select $a_t = \arg\max_a Q(s_t, a; \theta)$
        Execute action a_t and observe reward r_t and next state s_{t+1}
        Store transition $(s_t, a_t, r_t, s_{t+1}, \text{done})$ in $\mathcal{D}$
        Sample a random mini-batch of transitions $(s_j, a_j, r_j, s_{j+1}, \text{done}_j)$ from $\mathcal{D}$
        Set $y_j = r_j$ if done$_j$ is true, else
            $a'_j = \arg\max_{a'} Q(s_{j+1}, a'; \theta)$
            $y_j = r_j + \gamma \hat{Q}(s_{j+1}, a'_j; \theta')$
        Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ with respect to θ
        Every C steps, reset $\theta' \leftarrow \theta$
    end for
end for

where L^{VF} is the squared-error loss for the value function and S is an entropy bonus.

The full algotithm of PPO is detailed in Algorithm 2

Algorithm 2 Proximal Policy Optimization (PPO) with GAE

Input: initial policy parameters θ_0, initial value function parameters ϕ_0
Hyperparameters: clipping threshold ϵ, number of epochs K, minibatch size M, GAE lambda λ
Initialize policy network π_θ with θ_0 and value network V_ϕ with ϕ_0
for iteration = $1, 2, \ldots$ **do**
  Set $\theta_{\text{old}} \leftarrow \theta$
  Initialize a rollout buffer $\mathcal{D}$
  Let s be the initial state from environment reset
  ▷ Phase 1: Collect a batch of trajectories
  for t = $1, \ldots, T$ **do** ▷ T is the number of steps per rollout
    Sample action $a_t \sim \pi_{\theta_{\text{old}}}(\cdot|s_t)$
    Get value estimate $v_t = V_\phi(s_t)$
    Execute a_t in the environment, observe reward r_t and next state s_{t+1}
    Store transition $(s_t, a_t, r_t, \log \pi_{\theta_{\text{old}}}(a_t|s_t), v_t)$ in $\mathcal{D}$
    $s_t \leftarrow s_{t+1}$
  end for
  ▷ Phase 2: Compute advantage estimates using GAE
  Let rewards be $R = \{r_1, \ldots, r_T\}$ and values be $V = \{v_1, \ldots, v_T\}$
  Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$ using GAE:
    $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$
    $\hat{A}_t = \sum_{l=0}^{T-t-1} (\gamma\lambda)^l \delta_{t+l}$
  Compute returns-to-go $\hat{R}_t = \hat{A}_t + v_t$
  ▷ Phase 3: Optimize policy and value networks
  for epoch = $1, \ldots, K$ **do**
    for minibatch $(s, a, \log \pi_{\text{old}}, \hat{A}, \hat{R})$ from $\mathcal{D}$ **do**
      Let $r(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} = \exp(\log \pi_\theta(a|s) - \log \pi_{\text{old}})$
      $L^{\text{CLIP}}(\theta) = \min(r(\theta)\hat{A}, \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)\hat{A})$
      $L^{\text{VF}}(\phi) = (V_\phi(s) - \hat{R})^2$
      $L^{\text{S}}(\theta) = S[\pi_\theta(\cdot|s)]$ ▷ Entropy bonus
      Update parameters θ, ϕ by minimizing $L = -L^{\text{CLIP}} + c_1 L^{\text{VF}} - c_2 L^{\text{S}}$
    end for
  end for
end for

4 Experiments

4.1 Experimental Setup

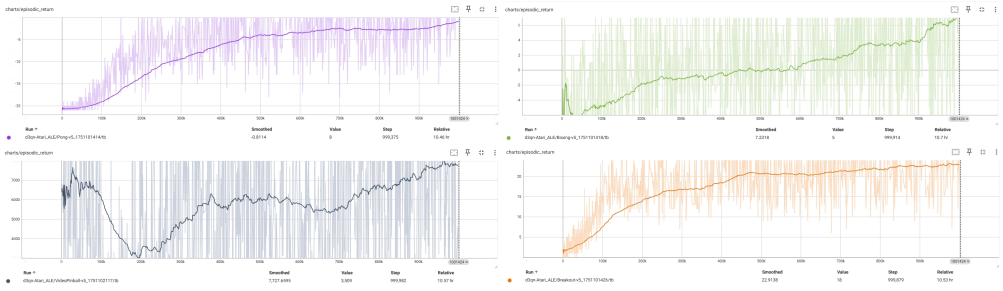
Environments For D3QN, we selected four game environments from the Atari 2600 suite, namely `Breakout-v5`, `Boxing-v5`, `Pong-v5`, `VideoPinball-v5`, using standard pre-processing wrappers from 'gymnasium' (grayscale, frame-stacking of 4, etc.). For PPO, we chose the `Hopper-v4`, `Ant-v4`, `Humanoid-v4` and `HalfCheetah-v4` environments from the MuJoCo physics simulator, which feature high-dimensional continuous state and action spaces.

Hyperparameters Key hyperparameters were chosen based on values commonly reported in the literature. For D3QN, we used a learning rate of 10^{-4}, a buffer size of 10^5, and an ϵ greedily decayed from 1.0 to 0.01. For PPO, we used a learning rate of $3 \cdot 10^{-4}$, a clip range $\epsilon = 0.2$, and collected rollouts of 2048 steps. All experiments were run on a NVIDIA 3090 GPU using PyTorch.

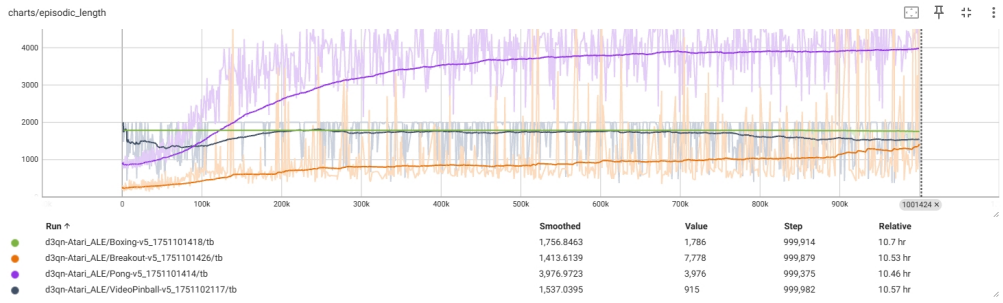
4.2 Results

D3QN on Atari We trained the D3QN agent on four environments respectively for 1 million timesteps. Figure 2 shows the learning curve, plotting the episodic return averaged over training. The agent demonstrates stable learning,

progressively improving its score. The Dueling architecture, combined with Double DQN, effectively mitigates value overestimation and leads to a robust policy.



(a) Episode Return



(b) Episode Length

Figure 2: Learning curve of D3QN on the four environments. The y-axis in (a) represents the episodic return and (b) represents the episodic length, and the x-axis represents the training timesteps.

An intriguing learning dynamic was observed during the training of D3QN on the `VideoPinball-v5` environment. As depicted in the learning curve, the agent initially achieved a high episodic return. However, this performance degraded significantly during the early stages of training (approximately the first 200k timesteps), before gradually recovering and ultimately surpassing the initial performance level. We hypothesize that this phenomenon is attributable to the unique reward structure of `VideoPinball-v5`, where a purely random policy can incidentally accumulate high scores by keeping the ball in play. The initial high return reflects this effect. As the agent begins to learn, its exploratory but non-optimal policy disrupts the effective random behavior, leading to a temporary performance drop. True, directed learning only emerges after this initial phase, as the agent starts to understand the causal relationship between its actions and long-term rewards.

This hypothesis is strongly supported by the Q-network's loss trajectory, shown in Figure 3. The Q-loss initially decreases as the network learns to fit the value of the preliminary, high-return random trajectories. Subsequently, as the agent's policy improves and it explores more valuable but complex state-action spaces, the Bellman error temporarily increases. This rise in loss corresponds to the period where the agent's return begins its steady ascent, indicating that the network is actively correcting its initial, simplistic value estimates to accommodate a more sophisticated and ultimately superior policy.

PPO on MuJoCo We trained the PPO agent for 1 million timesteps respectively on four environments(mentioned in section 4.1). The learning curves are shown in Figure 4. PPO exhibits remarkable stability and sample efficiency in these continuous control tasks. The policy steadily improves, achieving high scores, which demonstrates the effectiveness of the clipped surrogate objective in preventing catastrophic performance drops.

4.3 Analysis

Our experiments confirm the strengths of both algorithmic families. D3QN, with its experience replay and sophisticated network architecture, is well-suited for discrete action spaces with complex visual input. The separation of value and advantage streams appears crucial for tasks where the value of a state is not strongly dependent on every action.

PPO's performance in MuJoCo highlights its suitability for continuous control. The trust region-based updates ensure that the learned policy does not drastically change between iterations, leading to smooth and stable learning. The

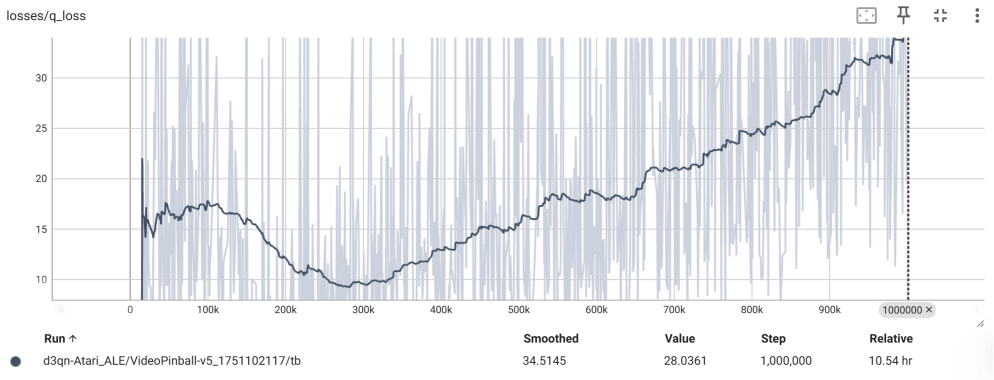
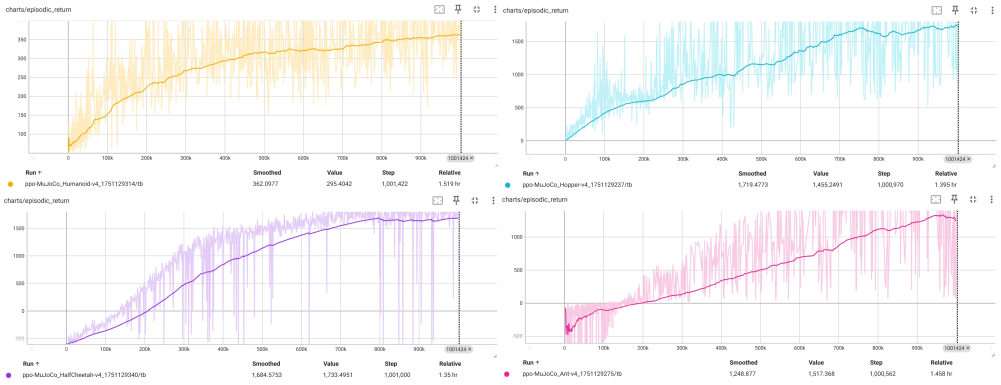
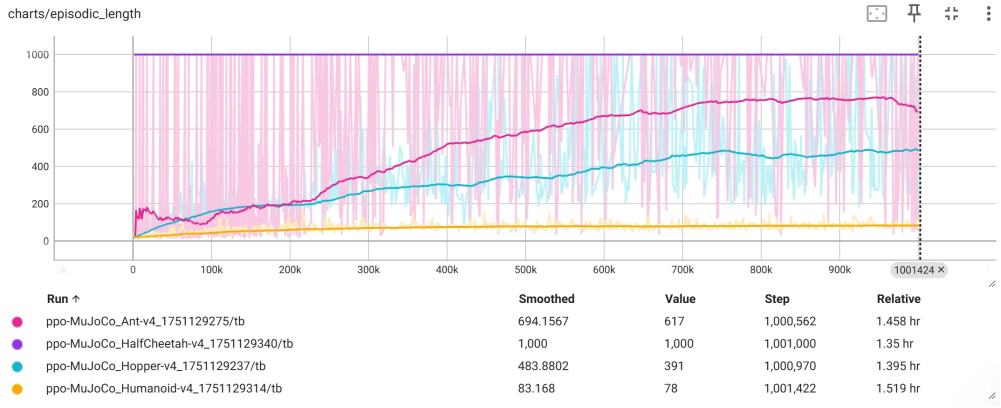


Figure 3: The training loss of the D3QN on `VideoPinball-v5`. The initial decrease corresponds to fitting the value of a high-scoring random policy, while the subsequent rise aligns with the period of meaningful policy improvement, where the network grapples with larger, more informative Bellman errors.



(a) Episode Return



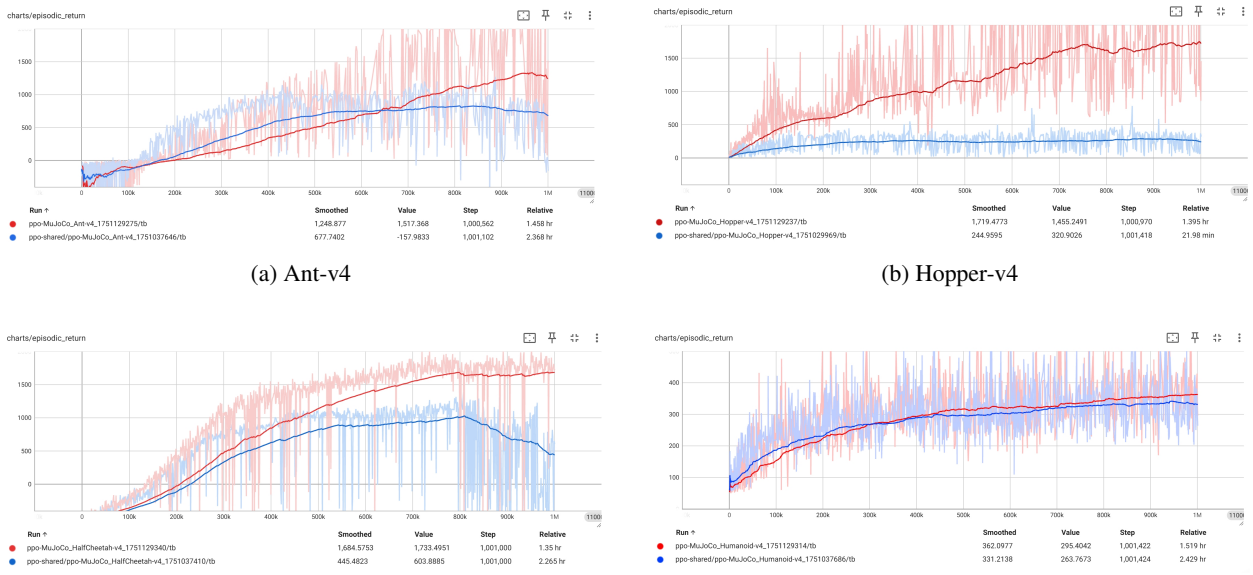
(b) Episode Length

Figure 4: Learning curves of PPO on MuJoCo environments. PPO consistently learns effective gaits for both tasks, showcasing its robustness in continuous action spaces.

Actor-Critic framework, augmented with GAE, provides a low-variance gradient estimate that is critical for learning complex locomotion skills.

Furthermore, to analyze the factors influencing training efficacy, we conducted an ablation study on the network architecture of our PPO agent. Following the implementation details of the original PPO paper [2], our primary setup utilizes two independent networks for the policy and the value function, the results of which is shown in Fig 4. We compared this against an alternative, commonly used architecture where the actor and critic share a common

feature-extraction body. To ensure a fair comparison and maintain training stability, both configurations employed fully-connected MLPs with two hidden layers of 64 units and Tanh activation functions. Furthermore, we adopted orthogonal weight initialization, a best practice for stabilizing training in modern actor-critic implementations [17]. The comparison and analysis are shown in Fig 5.



(a) Ant-v4

(b) Hopper-v4

(c) HalfCheetah-v4

(d) Humanoid-v4

Figure 5: Ablation study comparing PPO with separate vs. shared actor-critic networks across four MuJoCo environments. Each plot displays the performance of the separate-network architecture (rad) against the shared-network architecture (blue). The results consistently demonstrate that employing separate networks, as in the original PPO paper, yields superior performance and stability. This suggests that decoupling the policy and value function optimization prevents conflicting gradients, leading to more stable and effective learning in these complex continuous control tasks.

5 Conclusion

In this project, we successfully implemented and analyzed two state-of-the-art deep reinforcement learning algorithms, Dueling Double DQN and Proximal Policy Optimization. We demonstrated their effectiveness on challenging benchmark tasks from the Atari and MuJoCo domains, respectively. Our work provides a practical guide to the implementation details of these algorithms and empirically validates their well-documented strengths.

Acknowledgments

This project was completed for the Reinforcement Learning course at Shanghai Jiao Tong University. Special thanks to Prof. Zou and TA Yuyang Huang for their guidance and support,and my advisor Prof siheng Chen for his computing resources support.

References

[1] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[2] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[3] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[4] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[5] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.

[6] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[7] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

[8] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[9] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[10] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[11] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[12] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[13] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[14] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[15] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of artificial intelligence research*, 47:253–279, 2013.

[16] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[17] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.