Advanced Strategies in Pong: Reinforcement Learning with DQN, DDQN and PPO

Short Summary:

The project investigates the efficiency of Reinforcement Learning (RL) methods, specifically Deep Q-Networks (DQN), Double Deep Q-Networks (DDQN), and Proximal Policy Optimization (PPO) in mastering Pong, a classic Atari game within the Gymnasium environment. The focus is to develop an autonomous agent that learns optimal gameplay strategies through its own experience, surpassing the built-in AI in Pong. This project seeks to understand the adaptability of RL algorithms in a rapid, reaction-driven game setting and assess whether DDQN or PPO offers substantial improvements over DQN in terms of learning efficiency and strategic depth.

Objectives:

- Develop and train RL agents using DQN, DDQN and PPO algorithms to outplay the standard Pong AI in Gymnasium.
- Conduct a comparative study of DQN, DDQN and PPO in terms of learning speed, strategic efficiency, and overall gameplay performance.
- Investigate and implement algorithmic enhancements to standard models to improve agent performance in high-speed, decision-intensive environments.

Environment:

The project utilizes the Pong environment from Gymnasium, a popular platform for developing and comparing reinforcement learning algorithms. This environment accurately simulates the original Pong game, providing a standardized and controlled setup for training and evaluating the RL agent.

Methodology (DQN):

- **Exploration vs. Exploitation:**
  An epsilon-greedy strategy is used to balance exploration and exploitation. As training progresses, the exploration rate (epsilon) decays exponentially, encouraging the agent to explore diverse actions initially and gradually shift towards exploiting learned policies as it gains more experience.

- **State Representation:**
  The agent's state is represented by stacking several recent frames, allowing the neural network to infer motion and temporal dynamics critical for understanding the environment's dynamics. This approach is particularly useful in environments like pong where the velocity and trajectory of objects (e.g., the ball) are important.

- **Experience Replay:**
  The training leverages an experience replay mechanism where transitions (consisting of state, action, reward, and next state) are stored in a memory bank. This allows the network to learn from a diverse range of past experiences, reducing the correlation between consecutive learning updates and stabilizing the training process.

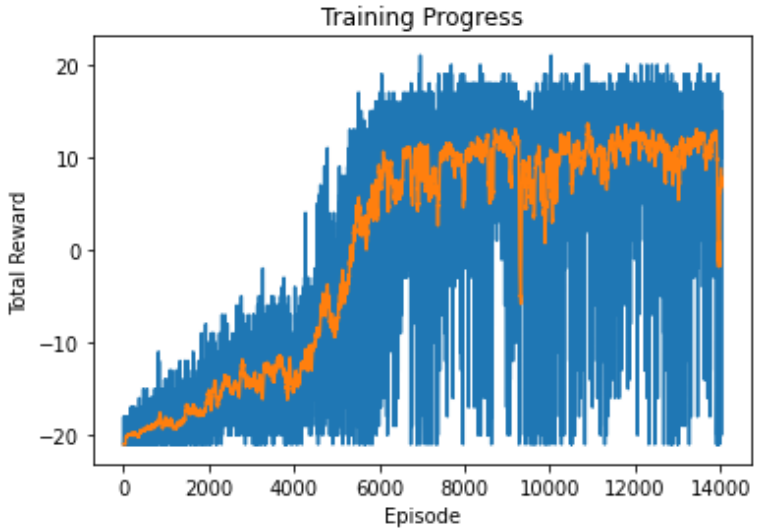
- **Network Architecture:**
  Two networks are used: a strategy network and a target network. The strategy network dictates the policy and is regularly updated, while the target network’s weights are updated less frequently to

provide a stable target for the optimizer. This separation helps mitigate the risk of feedback loops in updates that could lead to divergent or oscillating learning behaviors.

- **Training Dynamics:**
  During training, the agent interacts with the environment to collect experiences, which are then used to refine the strategy network. Regular updates to the target network ensure that the learning targets do not shift too rapidly, promoting stable and effective learning.

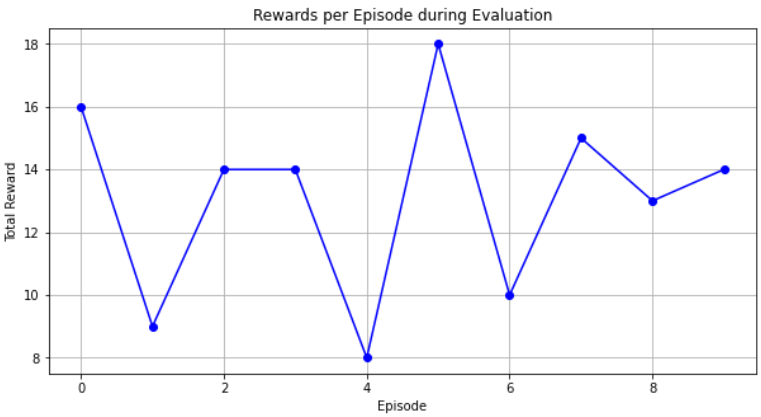
Result/Evaluation:

Training reward:



The general upward trend suggests that the agent is learning and improving its performance as training progresses. However, the high variability, especially in the blue line, indicates that the agent's performance from episode to episode can be quite erratic, a common characteristic in environments with high stochasticity or where the agent is still exploring significantly.

Evaluation:



The zigzag pattern might indicate episodes where the agent either performs very well or very poorly. This could be due to the agent encountering various states of the environment that it is still learning to handle. High

variance in performance like this is often observed in reinforcement learning settings, especially in complex environments or early in training.

Methodology (DDQN):

- **Enhanced Exploration vs. Exploitation:**
  The DDQN method continues to use an epsilon-greedy strategy to balance exploration and exploitation. Epsilon decays exponentially over time, promoting an initial phase of exploration that transitions into a focus on exploiting the learned strategies as the agent gains experience.

- **State Representation:**
  Similar to DQN, the state in DDQN is represented by a sequence of recent frames, providing the agent with information about motion and temporal changes. This multi-frame approach allows the neural network to effectively capture dynamics that are essential for predicting future states and making informed decisions.

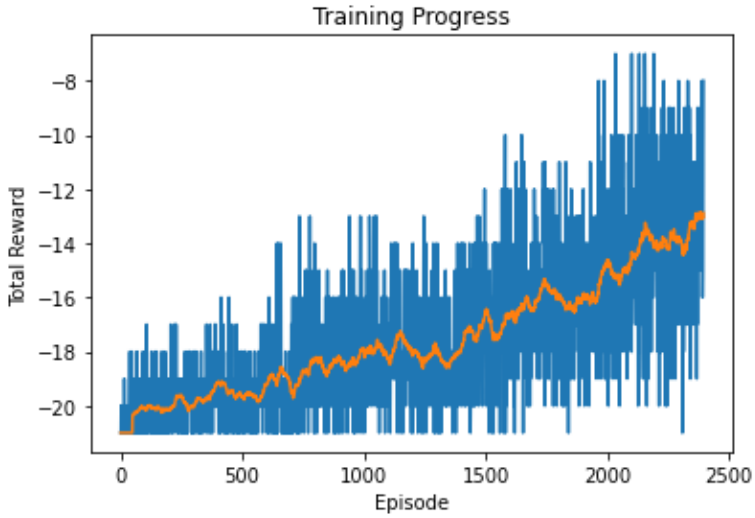
- **Experience Replay:**
  DDQN also utilizes an experience replay mechanism. By storing past transitions and randomly sampling from them to update the model, the network benefits from learning a more diverse set of experiences, which helps to break the correlation between sequential updates and improves learning stability.

- **Double Q-Learning:**
  A key feature of DDQN is the use of two separate networks with identical architectures: a strategy (policy) network and a target network. Unlike DQN, where both networks are used interchangeably to select and evaluate actions, DDQN decouples the action selection from the target Q-value generation. The strategy network is used to determine the greedy action, and the target network is used to evaluate the Q-value of taking that action at the next state. This reduces overestimations of Q-values seen in traditional DQN, leading to more stable and reliable learning outcomes.

- **Training Dynamics:**
  The training loop involves interacting with the environment to gather experiences, which are then used to update the strategy network regularly. The target network's weights are updated less frequently to provide a stable comparison for the strategy network's updates. This periodic updating helps prevent the rapid propagation of errors that could occur if the target values were continuously shifting.

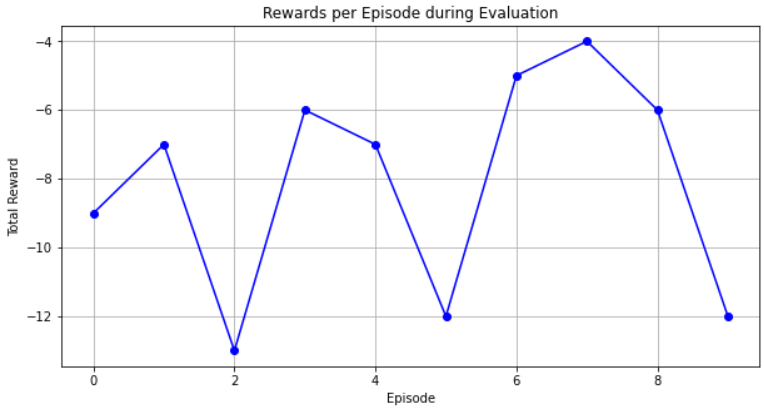
Result/Evaluation:

Training reward:



The upward trend in both the orange and blue lines indicates that the agent is improving its performance over time, suggesting effective learning. The high variance depicted by the blue bars shows episodes of varied success, which is typical in reinforcement learning due to the agent encountering diverse states or experimenting with different strategies. The agent wasn't trained for enough episodes, but observing the trend, it would easily outperform DQN or achieve similar results much sooner if trained for more episodes.

Evaluation:



The sharp fluctuations represent the agent achieving high rewards in some episodes and lower in others. This can be indicative of an environment where the agent might be exploiting well-learned strategies successfully in some states but struggling in others, or it may reflect the stochastic nature of the environment itself. Regardless, the agent needs to be trained for more episodes for better results

Methodology (PPO):

- **Actor-Critic Architecture:**
  PPO uses an actor-critic architecture where the policy (actor) and the value function (critic) are often represented by a single neural network with shared layers. This network outputs both action probabilities and state value estimates, facilitating simultaneous updates to both policy and value functions during training.

- **Multiple Environment Parallelism:**
  The code leverages parallel environments (**AsyncVectorEnv**) to collect diverse experiences simultaneously. This approach significantly speeds up data collection and provides a more diverse sample of experiences for each update, helping to stabilize training by reducing variance.

- **Policy Update Mechanism:**
  PPO introduces a clipping mechanism in the policy objective function to prevent large updates that could destabilize the policy. This is achieved by using a surrogate objective, where the ratio of new policy probabilities to old probabilities (from the policy before the update) is clipped. This ensures that the updates are kept within a reasonable range, promoting smooth policy evolution.

- **Generalized Advantage Estimation (GAE):**
  For estimating the advantages, PPO employs GAE, which is a technique to reduce variance in policy gradient estimates. GAE combines information from multiple time steps to form more stable and reliable advantage estimates, balancing the bias-variance tradeoff more effectively.

- **Optimization and Epochs:**
  After collecting a batch of data, the policy is optimized over multiple epochs, where the collected data is reused. This multiple epoch training on the same batch of data makes PPO sample-efficient, as it extracts more learning signal from the same amount of data.

- **Entropy Bonus:**
  To encourage exploration, an entropy bonus is added to the loss function. This bonus rewards the policy for maintaining diversity in the actions taken, preventing premature convergence to suboptimal deterministic policies.

- **Value Function Update:**
  Alongside the policy update, the value function is also updated using a squared-error loss. This critic update helps in providing more accurate value estimates, which in turn guide the policy updates more effectively.

Result/Evaluation:

Training reward

:

Initially, the rewards are significantly negative, indicating that the agent frequently underperformed or failed to complete tasks as per the environment's rewards structure. However, as episodes progress, there's a visible upward trend in the reward values, suggesting that the agent is increasingly learning from its interactions and making decisions that yield higher rewards.

Conclusion:

The Deep Q-Network (DQN) demonstrated respectable performance, indicating effective learning and adaptability within the training environment. However, while the Double Deep Q-Network (DDQN) showed potential to exceed the performance of DQN, it became apparent that achieving this would require extended training over more episodes. This suggests that DDQN, with its capability to reduce the overestimation of Q-values inherent in standard DQN, needs additional episodes to fully stabilize and leverage its refined learning mechanics. On the other hand, Proximal Policy Optimization (PPO) outperformed both DQN and DDQN, underscoring its efficiency and robustness as a policy gradient method. Nevertheless, to realize its full potential, PPO would also benefit from a longer training period, allowing it to consistently refine its policy estimations and adapt more comprehensively to the complexities of the environment. These observations highlight the necessity of balancing training duration with algorithmic sophistication to maximize performance in reinforcement learning setups.

References:

1. Mnih, V., et al. "Human-level control through deep reinforcement learning." Nature, 2015.
2. Van Hasselt, H., Guez, A., & Silver, D. "Deep Reinforcement Learning with Double Q-learning." AAAI, 2016.
3. Brockman, G., et al. "OpenAI Gym." arXiv preprint arXiv:1606.01540, 2016.
4. https://karpathy.github.io/2016/05/31/rl/
5. https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html
6. https://towardsdatascience.com/intro-to-reinforcement-learning-pong-92a94aa0f84d
7. https://huggingface.co/sb3/dqn-PongNoFrameskip-v4
8. https://stable-baselines.readthedocs.io/en/master/guide/examples.html
9. https://towardsdatascience.com/getting-an-ai-to-play-atari-pong-with-deep-reinforcement-learning-47b0c56e78ae