DDPG vs PPO: A Comparative Study on Reinforcement Learning for ETF Portfolio Optimization

# Abstract

This paper explores and compares two prominent reinforcement learning algorithms — Deep Deterministic Policy Gradient (DDPG) and Proximal Policy Optimization (PPO) — within the context of an ETF portfolio optimization problem. We investigate the suitability, advantages, and limitations of both methods with a real-world environment built using financial data from multiple asset classes. PPO is ultimately chosen for its robustness and stability in non-stationary environments like financial markets.

# 1. Introduction

Reinforcement Learning (RL) is gaining traction in the field of financial modeling, especially in portfolio optimization where the agent must learn to balance risk and return. This study is part of a broader project involving multiple asset types including stocks, ETFs, and cryptocurrencies. In this section of the project, we focused on ETFs and employed PPO to train the agent.

# 2. Methodology

## 2.1 Data and Environment

We used a set of 10 major ETFs and downloaded their adjusted closing prices from Yahoo Finance from 2018 to 2025. Our custom environment `ETFPortfolioEnv` simulates real-world trading by incorporating technical indicators, volatility regimes (classified from SPY), transaction costs, and risk-adjusted reward.

## 2.2 PPO Algorithm Implementation

PPO was selected as the learning algorithm due to its ability to handle continuous action spaces, stable training with clipped policy updates, and compatibility with on-policy data. The agent learns to allocate weights to each ETF based on market conditions and indicators.

# 3. Why PPO Was Preferred Over DDPG

The key reasons PPO was chosen over DDPG are:  
- On-policy learning suitable for non-stationary environments  
- Clipped objective avoids large, destabilizing updates  
- Stochastic policy ensures effective exploration in high-dimensional action space  
- Simpler to tune and more robust against reward noise and market regime shifts  
- No dependency on outdated experiences like in DDPG’s replay buffer

# 4. Limitations of DDPG in Our Case

If DDPG were used, we would face major and minor issues including:  
- Poor adaptation to market regime shifts due to replay buffer  
- Sensitivity to noise and hyperparameters  
- Inefficient exploration in a 10-dimensional action space  
- Risk of instability and Q-value overestimation without additional mechanisms  
- Increased training complexity and time

# 5. Ideal Scenario for DDPG

DDPG is better suited for environments where:  
- The environment is stationary or slowly changing  
- Continuous actions are required with high precision (e.g., robotic control)  
- Stable, noise-free reward signals are available  
- Off-policy learning is beneficial due to costly environment interactions

# 6. PPO vs DDPG: Key Differences

|  |  |  |
| --- | --- | --- |
| Aspect | PPO | DDPG |
| Learning Type | On-policy | Off-policy |
| Exploration | Stochastic policy (distribution-based) | Deterministic policy + noise |
| Stability | High (clipped updates) | Low (sensitive to noise) |
| Replay Buffer | Not used | Required |
| Ease of Tuning | Relatively easy | Difficult |
| Handling Noise | Good | Poor without tuning |
| Best Use Case | Financial markets, high-dim exploration | Robotics, control, simulation |

# 7. Conclusion

Through our ETF agent implementation and experimental insights, we conclude that PPO is a better fit for portfolio optimization in dynamic environments. Its ability to handle continuous actions, adapt quickly to market changes, and remain stable during training make it a strong choice over DDPG, which would have introduced instability due to its replay buffer and sensitivity to hyperparameters.

# References

- OpenAI Spinning Up Documentation: https://spinningup.openai.com/en/latest/algorithms/ddpg.html  
- DeepMind PPO Paper: https://arxiv.org/abs/1707.06347  
- Number Analytics DDPG Guide: https://www.numberanalytics.com/blog/deep-deterministic-policy-gradient-for-beginners  
- MathWorks DDPG Reference: https://www.mathworks.com/help/reinforcement-learning/ug/ddpg-agents.html