A Comparative Study of DQN and PPO for Real-Time Decision Making in Financial Trading

生醫四 洪至寬、生醫四 李睿哲

I. INTRODUCTION

Reinforcement Learning (RL) has emerged as a powerful framework for algorithmic trading, offering the ability to learn adaptive strategies from complex and noisy financial data. In this project, we present a comparative study of two prominent RL algorithms — **Dueling Deep Q-Networks (Dueling DQN)** and **Proximal Policy Optimization (PPO)** — representing the value-based and policy-based learning paradigms, respectively.

Both models are implemented independently but trained under a **shared trading environment** and a **unified reward function**, which incorporates profitability, trading cost, RSI-based heuristics, and behavioral penalties. By holding the reward structure constant, we aim to isolate and examine the impact of the algorithmic differences — such as Q-value estimation versus policy optimization — on learning dynamics and trading behavior.

This study evaluates each agent's performance using key financial metrics, including **portfolio value**, and **action stability**. By comparing the behaviors and outcomes of PPO and Dueling DQN under identical reward signals and market conditions, we provide insight into the trade-offs between stability, responsiveness, and risk control in reinforcement learning-based trading systems.

II. Algorithm Background

A. Dueling Deep Q-Networks (Dueling DQN)

Dueling Deep Q-Network (Dueling DQN) is an enhancement of the standard Deep Q-Network (DQN) that introduces a more refined value estimation mechanism. Instead of directly learning the Q-values for each action, Dueling DQN separates the estimation into two streams(Fig 1.): the **state value function** V(s), which estimates how good it is to be in a particular state, and the **advantage function** A(s,a), which measures the relative benefit of a specific action in that state. These two components are then combined to produce the final Q-value:

$$\mathrm{Q(s,a)} = V(s) + (A(s,a) - \frac{1}{|A|}\sum_{a'} A(s,a'))$$

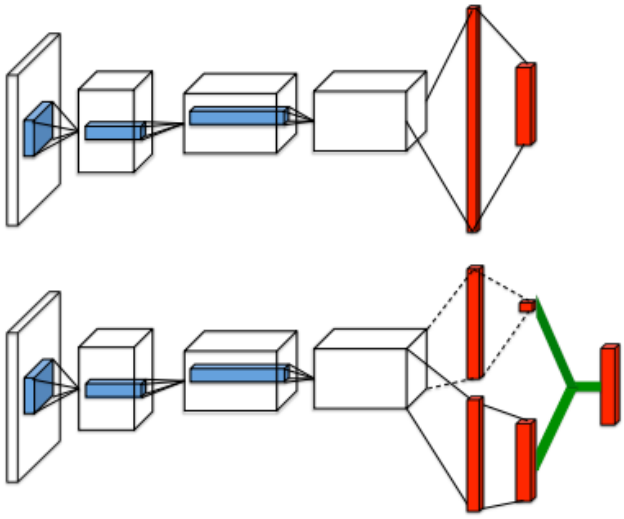

Fig 1. Architectural comparison between standard DQN (**top**) and Dueling DQN (**bottom**)[2].The standard DQN directly estimates Q-values from the shared feature extraction layers. In contrast, the Dueling DQN architecture separates the estimation into two streams: the value stream $V(s)$and the advantage stream $A(s,a)$,which are later combined to produce the final Q-value. This decomposition improves learning stability, especially in environments where the advantage of actions is hard to distinguish.

This decomposition improves learning stability, especially in environments where many actions yield similar outcomes — a common situation in financial trading. It allows the agent to better generalize across actions and state transitions.

The input layer of the Dueling DQN network has 22 neurons, corresponding to the 22-dimensional state vector generated by the trading environment. This vector consists of:

- 4 technical indicators — **price**, **moving average (MA)**, **relative strength index (RSI)**, and **normalized volume** — each computed over a rolling window of size $\omega = 5$, yielding $4 \times 5 = 20$ features.
- 2 additional scalars: **normalized stock holdings** and **normalized cash balance**.

Thus, the full input dimension is calculated as:

$$state\ size(input) = 4 \times \omega + 2 = 22$$

The Dueling DQN network consists of a shared feature extraction layer followed by two separate branches: one for estimating the state value $V(s)$,and another for computing the action advantages $A(s,a)$. The complete architecture is summarized in **Table 1** below:

	Shared Feature Layer	**Value Stream**	**Advantage Stream**
Input Layer	22 dimenions	-	-
Linear-1	22 → 128, ReLU	-	-
Linear-2	-	128 → 64, ReLU	128 → 64, ReLU
Linear-3	-	64 → 1(state value)	64 → 3 (action)
Output Layer	-	-	Aggregated as Q(s,a)

Table 1. Architecture of the Dueling DQN agent

This architecture enables the agent to decouple the value of being in a state from the value of taking a specific action in that state, enhancing its ability to learn under partial observability and noisy signals.

B. Proximal Policy Optimization (PPO)

Proximal Policy Optimization (PPO) is a policy-gradient reinforcement learning algorithm that performs updates through a clipped surrogate objective. This objective helps prevent overly large policy changes during training, improving stability and convergence speed. PPO typically employs an actor-critic architecture(Fig. 2), where:

- The **actor** proposes actions given current states.
- The **critic** estimates the value function $V(s)$ to guide learning.

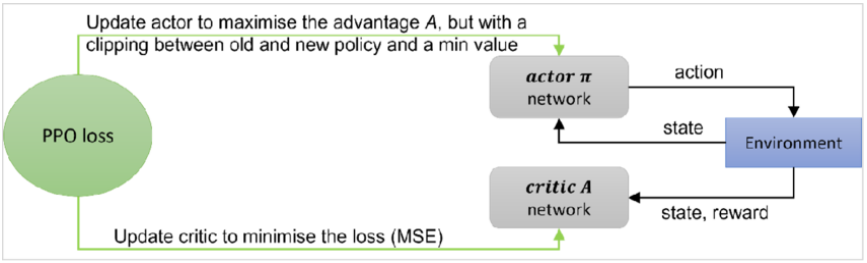

Fig. 2. Basic structure of PPO[1]

In our implementation, PPO is trained in a custom-designed trading environment that mimics real-world market conditions using historical data and realistic constraints like transaction fees and taxes. The environment emits compact yet informative state observations at each step, including:

- Account balance
- Current asset holdings
- Current price
- 5-day and 10-day moving averages
- RSI (Relative Strength Index)

These indicators allow the PPO agent to detect trends, momentum, and overbought/oversold conditions, aligning its policy with human-interpretable trading logic. Aditional, The architectural details of our PPO implementation are summarized in Table 1 below:

	Actor (Policy Network)	**Critic (Value Network)**
Input Layer	6 dimenions	6 dimenions
Linear-1	6 → 256	6 → 256
Linear-2	256 → 128	256 → 128
Linear-3	128 → 3(action)	128 → 1(state value)
Output Layer	Softmax	None

Table 1. Architecture of the PPO Agent

III. Reward Design

To ensure a fair comparison between the two reinforcement learning algorithms, both **Dueling DQN** and **PPO** are trained under a unified reward function specifically crafted for stock trading environments. The reward function is designed to balance short-term profitability with risk control and trading discipline.

The reward at each timestep is defined as:

$$\begin{aligned} \boldsymbol{reward_t} = (\boldsymbol{net\ worth_t - net\ worth_{t-1}}) \\ - \boldsymbol{\alpha \times holding\ penalty} \\ - \boldsymbol{\beta \times transaction\ cost} \\ - \boldsymbol{\gamma \times switch\ penalty} \end{aligned}$$

Components:

- **Net worth change**:
  The primary reward signal encourages profitable trading by measuring the change in total asset value (cash + stock holdings) from the previous step.
- **Holding penalty** ($\boldsymbol{\alpha}$):
  A time-based penalty is applied for excessive holding of positions without action, discouraging passive behavior.
- **Transaction cost penalty** ($\boldsymbol{\beta}$):
  Reflects realistic trading frictions, such as fees and slippage, applied on each buy/sell action.
- **Switch penalty** ($\boldsymbol{\gamma}$):
  Penalizes frequent switching between buy/sell/hold actions to encourage trading stability.
- **RSI-based signal encoding**:
  While not directly part of the reward formula, the reward structure aligns with RSI thresholds (e.g., buying favored when RSI < 30) to reinforce human-interpretable trading behavior.

Design Rationale

This reward formulation ensures the agent learns not only to maximize profit, but also to:

- Avoid unnecessary trades that incur costs
- Maintain positions only when justified
- Align actions with technical indicators to promote interpretability

By using this unified reward structure across both PPO and Dueling DQN, we remove the confounding factor of reward-induced bias and enable a more direct evaluation of each algorithm's learning characteristics.

IV. Experiment

1. Experimental Setup

- **Environment**: Custom OpenAI Gym-like trading environment simulating daily stock price movements (NVDA).
- **Observation space**:
  - Current price
  - Moving average
  - RSI
  - Normalized volume
  - Cash balance
  - Inventory holdings
- **Action space**: Discrete — Buy, Sell, Hold
- **Data source**: Nvida(NVDA) Historical daily data from Yahoo Finance, eight years for taining and one years for testing(Fig. 3)

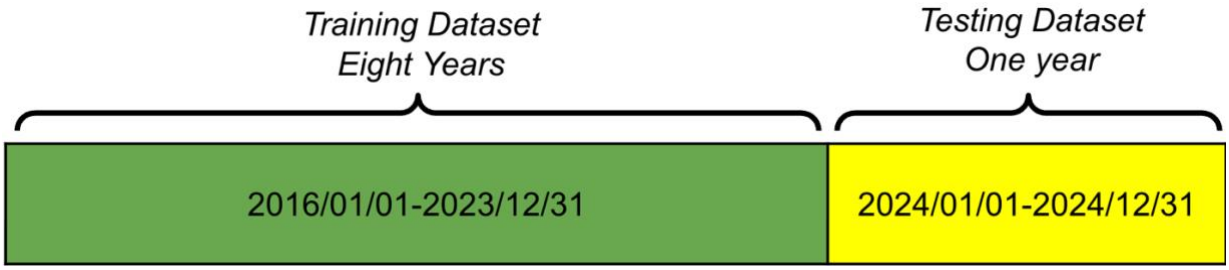

Fig. 3. Training and testing dataset split for Nvidia (ticker: NVDA), with historical daily price data collected from Yahoo Finance. The training period spans from 2016/01/01 to 2023/12/31 (8 years), and the testing period covers 2024/01/01 to 2024/12/31 (1 year).

The same reward function is applied to both agents, incorporating net worth change, transaction cost penalty, holding penalty, and switch penalty (see Section III).

Each model is trained using 10 random seeds to ensure robustness. Evaluation is based on both quantitative metrics and qualitative action behavior.

---

2. Convergence Analysis

Table 2 summarizes the key differences in training behavior between PPO and Dueling DQN. While both agents were trained under identical conditions, PPO demonstrates smoother and more stable learning dynamics, characterized by low variance across different random seeds. In contrast, Dueling DQN tends to learn faster during the early training stages but exhibits more fluctuation in performance, especially under volatile or noisy market conditions. These observations are further visualized in **Figure 3**, which shows the evolution of cumulative policy value across training episodes for both algorithms.

	PPO	Dueling DQN
Convergence Speed	Smooth and gradual	Faster in early stages but volatile
Reward Stability	High (low variance across seeds)	Moderate (more fluctuation)
Training Episodes	200	200

Table 2. Comparison of Training Dynamics Between PPO and Dueling DQN

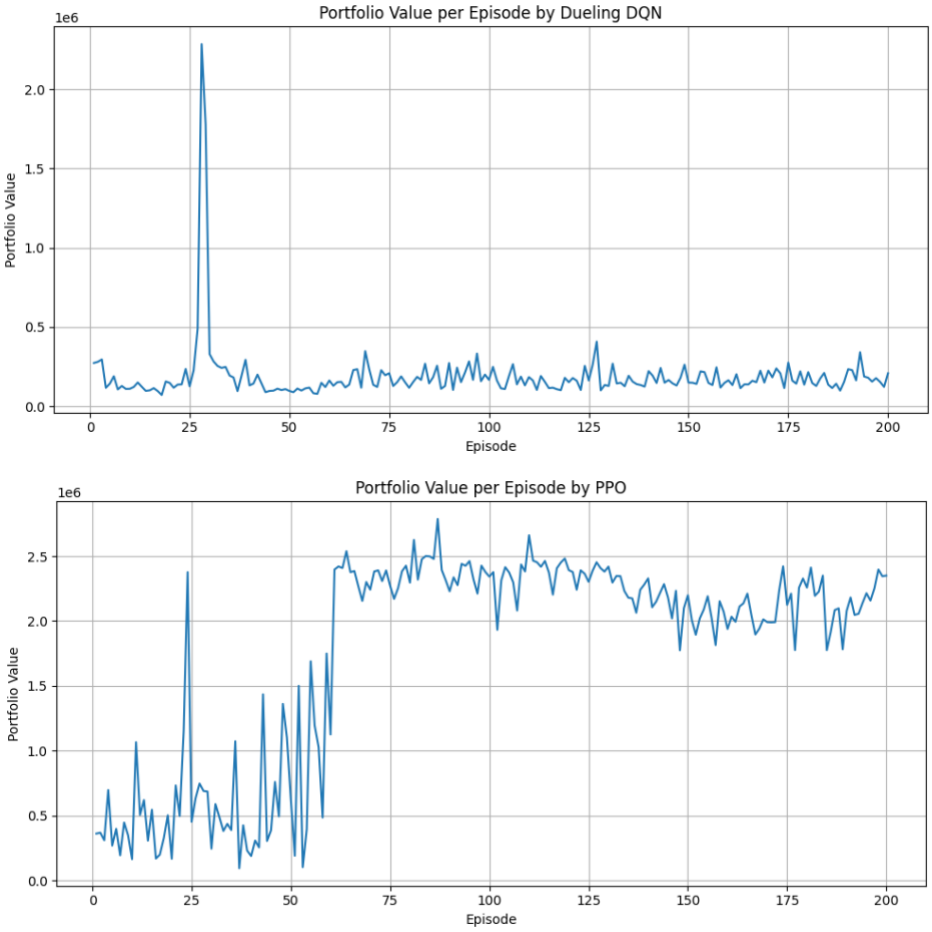

Fig. 3. Learning Curve of Dueling DQN(**top**) and PPO(**bottom**) Over Training Episodes

3. Action Behavior (Same-Day Trading Comparison)

Figure 4 and **Figure 5** illustrate the trading suggestions generated by the Dueling DQN and PPO agents on June 26, 2025, based on the same market data input. The PPO agent recommended a **Buy** action at a share price of 154.31, while the Dueling DQN agent suggested **Sell** at a lower price of 137.37 (on June 2). These contrasting recommendations highlight the behavioral divergence between the two algorithms: PPO tends to adopt a more conservative, trend-following strategy, while Dueling DQN reacts more sensitively to local signals, resulting in higher action variability.

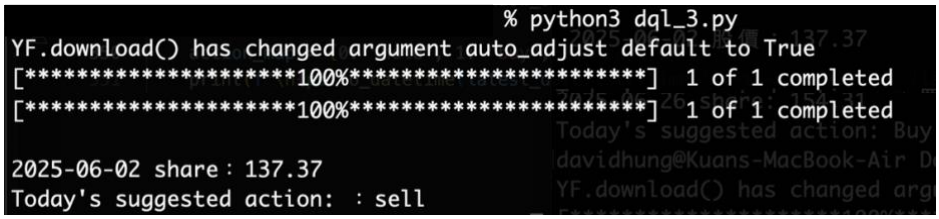

Fig. 4. Suggested trading actions by Dueling DQN.

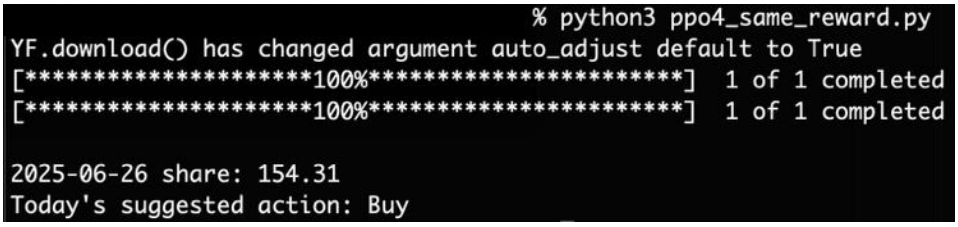

Fig. 5. Suggested trading actions by PPO.

On evaluation days, we compared the **buy/sell/hold decisions** made by both agents:

- **PPO**:
  - Tends to hold longer positions
  - Makes fewer but more deliberate trades
  - More conservative near RSI threshold zones
- **Dueling DQN**:
  - Reacts more frequently to RSI changes
  - More aggressive in switching positions
  - Higher action variance across episodes

This behavioral divergence illustrates the architectural difference: PPO optimizes expected return across a smoother policy space, whereas DQN selects actions with maximum Q-value at each step, resulting in more reactive strategies.

---

4. Summary of Findings

- **Under the same reward design**, the choice of algorithm significantly influences both training dynamics and trading behavior.
- PPO favors **stable, conservative** strategies, while Dueling DQN produces **more aggressive, signal-sensitive** responses.
- The unified reward design enables a clean comparison, revealing the **behavioral inductive bias** of each architecture.

VI. Discussion

Through controlled experiments using a shared reward structure, several key observations emerged regarding the differences between PPO and Dueling DQN in financial trading environments:

- **Learning Dynamics**: PPO demonstrated smoother and more stable convergence, while Dueling DQN exhibited faster early learning but greater reward variance over time.
- **Action Strategy**: PPO favored longer holding periods and more cautious decisions, particularly around ambiguous RSI signals. In contrast, Dueling DQN responded more quickly and frequently to market changes, especially when technical signals were strong.
- **Market Adaptability**: PPO showed stronger generalization in noisy or trendless market segments, likely due to its policy-based averaging over actions. Dueling DQN performed better in high-signal clarity conditions, where Q-value distinctions were more pronounced.
- **Behavioral Inductive Bias**: Despite operating under the same reward function, each algorithm exhibited distinct behavior patterns, underscoring the impact of learning architecture on strategy formation.

These findings affirm that even with identical external incentives, the internal mechanisms of RL algorithms shape how agents interpret and act upon market signals.

VII. Conclusion

This study presents a direct comparison between two reinforcement learning approaches — PPO and Dueling DQN — in the context of algorithmic stock trading, using a unified reward design to isolate algorithmic effects.

By standardizing the reward structure, we highlight how each algorithm's intrinsic learning dynamics influence convergence stability, responsiveness, and trading style. PPO tends to learn stable and conservative strategies, while Dueling DQN is more aggressive and reactive.

The results provide practical guidance for selecting RL architectures based on desired trading behavior and market characteristics. In particular, policy-based methods like PPO may be more suitable for environments where long-term stability and noise robustness are prioritized, while value-based methods like DQN may excel in short-term, signal-rich conditions.

Code Availability

All source code used for training, testing, and evaluation of PPO and Dueling DQN agents is available on GitHub at:

PPO: https://github.com/DavidHung0501/2025-spring-Modern-AI.git
DQN: https://github.com/harrylee1971/2025-spring-Modern-AI.git

The repository contains environment setup, model architecture definitions, and scripts for training and daily inference.

Reference

[1] Chadi, Mohamed-Amine, and Hajar Mousannif. "Understanding reinforcement learning algorithms: The progress from basic Q-learning to proximal policy optimization." arXiv preprint arXiv:2304.00026 (2023).
[2] Wang, Ziyu, et al. "Dueling network architectures for deep reinforcement learning." International conference on machine learning. PMLR, 2016