Training the Perfect Trader: A Reinforcement Learning Approach

Nguyen Khoa Pham, Hsien-Pang Hsieh, Yigitcan Yildiz
University of Cologne
Albertus-Magnus-Platz, 50923 Cologne, Germany
npham2@smail.uni-koeln.de, hhsieh@smail.uni-koeln.de, yyildiz2@smail.uni-koeln.de

GitHub Repository: click to see our github repo
Video Link: click to see the video

Abstract

The dynamic and complex nature of stock markets presents real challenges in developing optimal trading strategies. Orthodox methods, such as rule-based systems and statistical approaches, often cannot adapt to evolving market conditions, limiting their effectiveness. Reinforcement Learning (RL), particularly Deep Reinforcement Learning (DRL), can introduce a worthwhile solution to this by enabling agents to learn adaptive trading policies through direct interaction with the market. However, RL-based approaches face their-own challenges such as designing effective reward functions, handling market noise, and generalizing their findings across diverse financial assets. In this study, we try to offer a comparative analysis of three RL algorithms—Proximal Policy Optimization (PPO), Recurrent PPO, and Advantage Actor-Critic (A2C)—to identify the most effective strategy for stock trading. We customize a trading environment, incorporate modified reward structures, and evaluate performance across multiple stocks and market conditions. Our experiments suggest that Recurrent PPO outperforms other models, achieving the highest total reward (39.39) and profit (1.38) on our fundamental data, Nvidia (NVDA) stock, while A2C shows strong risk-adjusted returns on other stocks. These findings underline the prospects of RL-based approaches in developing vigorous and adaptive trading strategies.

Introduction

As more people are interested in participating in financial markets to increase their wealth, stock markets interest more and more people every day. However, determining optimal trading strategies remains as a fundamental problem since those markets are dynamic and complex. Traditional approaches rely on rule-based systems, statistical methods, or human expertise, which are sometimes inadequate at reacting to evolving market conditions. Reinforcement Learning (RL), particularly Deep Reinforcement Learning (DRL), offers a better-suited solution by allowing agents to learn optimal trading policies through direct interaction with the environment.

Even today algorithmic trading accounts for a substantial portion of market transactions in financial systems. Developing a robust and adaptive trading strategy by using the capabilities of RL will increase profitability, decrease risks, and save a significant amount of time. However, a fit for purpose RL algorithm for stock trading presents several challenges. These include the necessity of an effective reward function design, handling market noise and non-stationarity, and ensuring model generalization capabilities across different financial assets.

Previous studies have focused on various RL techniques in algorithmic trading. Early works modeled financial trading as a game, using DRL agents to optimize their strategies. Subsequent research introduced advanced policy-based and value-based methods, demonstrating their effectiveness in adaptive trading. Other contributions dealt with key challenges such as multi-objective reward generalization and standardization of DRL-based trading research through benchmarking frameworks.

In this study, we build upon prior research by comparing the performance of three different RL algorithms. Those are Proximal Policy Optimization (PPO), Recurrent PPO, and Advantage Actor-Critic (A2C). Our methodology contains customizing a trading environment, training RL agents with modified reward structures, and evaluating their performance across different market conditions and stocks. By analyzing the strengths and limitations of these algorithms, we aim to identify the most effective approach for real-world stock trading applications.

Related Work

Deep reinforcement learning (DRL) has gained significant attention in financial markets, particularly in the field of algorithmic trading and portfolio management. Various studies have explored different RL approaches to optimize trading strategies, improve decision-making, and maximize profits.

One of the pioneering works in this domain modeled financial trading as a game, utilizing a DRL framework where an agent interacts with the market to optimize its trading policy(Huang 2018). Following this, Practical Deep Reinforcement Learning for Stock Trading introduced an improved framework that leverages policy-based and value-based methods to develop adaptive trading strategies, demonstrat-

ing superior performance over traditional rule-based approaches(Liu et al. 2022a).

A major challenge in RL-based trading is designing effective reward functions. A study on multi-objective reward generalization proposed a novel approach that balances risk and reward, enhancing the performance of DRL agents in single-asset trading(Cornalba et al. 2023). Similarly, the FinRL library was introduced to standardize and streamline DRL-based stock trading research, integrating various state-of-the-art DRL algorithms and providing an efficient benchmarking environment(Liu et al. 2022b).

Other works have explored different applications of DRL in algorithmic trading, such as using actor-critic models for optimizing trading strategies in volatile markets(Ponomarev, Oseledets, and Cichocki 2019), and applying Deep Q-Networks (DQN) and Proximal Policy Optimization (PPO) to enhance decision-making in high-frequency trading environments(Théate and Ernst 2021). These studies highlight the effectiveness of different RL architectures in trading applications and provide insights into the strengths and limitations of various models.

Our study builds upon these prior works by empirically comparing three RL algorithms—PPO, Recurrent PPO, and A2C—on stock trading tasks, evaluating their performance across different financial assets. By leveraging the insights from previous research, we aim to identify the most effective DRL approach for stock trading under real-world conditions.

Methodology

In this section, we detail the methodology used to develop our reinforcement learning-based stock trading agent. Our approach consists of three main components: (1) environment customization, (2) agent training, and (3) performance evaluation. Figure 1 provides an overview of our methodology pipeline.

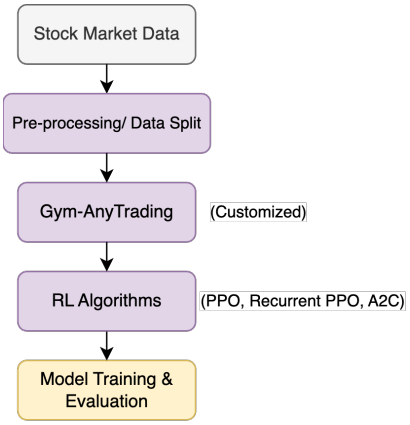

Figure 1: Training Framework

Environment Customization

We built our trading environment based on the Gym-AnyTrading framework(Haghpanah 2019), making specific modifications to apply feature scaling to the original dataset and adjust the transaction cost. The key settings are as follows:

- **State Space:** The observation space includes scaled historical stock prices and other technical indicators. We ensure that the closing price is correctly rescaled to its original value before further processing.
- **Action Space:** The action space consists of only two discrete actions: *Buy* and *Sell*.
- **Transaction Cost:** We introduce a bid-ask spread model where a trade incurs a transaction fee of 0.1% for both buying and selling.
- **Reward Function:** The reward is calculated based on profit changes, incorporating transaction costs.
- **Episode Termination:** An episode ends when the dataset is exhausted.

Core Algorithms

To train the stock trading agent, we employed three reinforcement learning algorithms from the Stable-Baselines3 library: Proximal Policy Optimization (PPO), Recurrent Proximal Policy Optimization (Recurrent PPO), and Advantage Actor-Critic (A2C).

Proximal Policy Optimization (PPO): PPO is a policy gradient method that stabilizes training using a clipped objective function. The policy update is performed by optimizing the following objective:

$$L(\theta) = \mathbb{E}_t \left[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t) \right] \quad (1)$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, A_t is the advantage function, and ϵ is a clipping parameter that controls the update step size (Schulman et al. 2017).

Recurrent Proximal Policy Optimization (Recurrent PPO): Recurrent PPO extends PPO by incorporating a recurrent neural network (LSTM) to handle time-series dependencies. The objective function remains the same, but the policy now considers hidden states h_t:

$$L(\theta) = \mathbb{E}_t \left[\sum_{t=0}^{T} \gamma^t r_t \right] \quad (2)$$

where γ is the discount factor, and r_t represents the reward at time step t. The recurrent structure enables the agent to leverage sequential patterns in stock price movements(Pleines et al. 2022).

Advantage Actor-Critic (A2C): A2C is a synchronous version of the Actor-Critic algorithm that estimates the value function while optimizing policy updates(Mnih et al. 2016). The advantage function is computed as:

$$A_t = R_t - V(s_t) \quad (3)$$

where R_t is the discounted return and $V(s_t)$ is the estimated value function. The policy gradient is updated using:

$$L_{\text{policy}} = -\mathbb{E}_t \left[\log \pi_\theta(a_t|s_t) A_t \right] \tag{4}$$

while the value function is optimized using:

$$L_{\text{value}} = \mathbb{E}_t \left[(V_\phi(s_t) - R_t)^2 \right] \tag{5}$$

Additionally, an entropy term is often added to encourage exploration:

$$L_{\text{entropy}} = -\mathbb{E}_t \left[\pi_\theta(a_t|s_t) \log \pi_\theta(a_t|s_t) \right] \tag{6}$$

These algorithms enable the agent to learn optimal trading strategies through policy optimization and advantage estimation, ensuring robustness in dynamic market environments.

Performance Evaluation

We computed the following key performance metrics for all models:

- **Total Profit:** The final portfolio value relative to the initial capital.
- **Sharpe Ratio:** Measures risk-adjusted returns, defined as the ratio of mean returns to standard deviation.
- **Maximum Drawdown:** The maximum observed loss from a peak to a trough.

Baseline Strategies

Buy and Hold: The strategy involves holding the asset from the beginning to the end of the test period, with profit calculated as the difference between the final and initial prices:

$$\text{Total Profit} = P_{\text{final}} - P_{\text{initial}} \tag{7}$$

where P_{final} is the closing price at the end of the test period, and P_{initial} is the opening price.

Moving Average Crossover: A technical strategy that buys when a short-term moving average crosses above a long-term moving average and sells when the opposite occurs. The strategy return is computed as:

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}} \tag{8}$$

where P_t is the asset price at time step t. The cumulative profit is given by:

$$\text{Cumulative Profit} = \prod_{t=1}^{T} (1 + R_t) - 1 \tag{9}$$

We evaluated three reinforcement learning models by simulating trades in a test environment and comparing them against baseline strategies. The detailed experimental results, along with graphical analysis, will be presented in the experiment chapter.

Experiment

Our dataset consists of daily historical stock prices for Nvidia (NVDA), retrieved using the Yahoo Finance API through the `yfinance` library. The dataset spans from January 1, 2020, to the present day, encompassing multiple market conditions, including bullish, bearish, and volatile phases. The dataset includes the following key attributes:

- Date: The trading date.
- Open, High, Low, Close Prices: The stock's price movement for the day.
- Volume: The number of shares traded.
- Technical Indicators: Computed indicators such as SMA, EMA, MACD, RSI, ATR, Bollinger Bands, CCI, MFI, and ROC, which provide insights into market trends and momentum.

Temporal splits were applied to avoid lookahead bias: training (Jan 2020–Sep 2023), validation (Oct 2023–Sep 2024), and testing (Oct 2024 onward).We applied various preprocessing steps to enhance data quality and suitability for reinforcement learning:

- Feature Engineering: We computed technical indicators using the `ta` library and added them as features.
- Handling Missing Data: Only essential cleaning steps were necessary due to the relative cleanliness of the raw data. Specifically, any infinite values produced during the computation of technical indicators were replaced with NaN, and rows containing any missing values were subsequently removed.
- Scaling: Recognizing the importance of feature scaling in deep reinforcement learning, all numerical columns were standardized using `scikit-learn`'s StandardScaler.
- Sorting and Cleaning: The data was sorted chronologically, and unnecessary columns such as "Dividends" and "Stock Splits" were removed.

Descriptive statistics of the dataset, comprising 1,251 observations, provide key insights into the market behavior of NVIDIA (NVDA) can be found in Table 1. The closing price has a mean of approximately \$41.27 with a standard deviation of \$40.30, ranging from a low of \$4.89 to a high of \$149.43. Trading volume averages around 445.98 million shares, indicating significant day-to-day fluctuations. Technical indicators reflect these trends: the Simple Moving Average (SMA) and Exponential Moving Average (EMA) closely follow the closing price, as expected, given their direct derivation from it. Meanwhile, the Moving Average Convergence Divergence (MACD), with a mean of 0.71 and a standard deviation of 1.83, exhibits notable variability, highlighting the dynamic momentum shifts in the stock's price movements.

Fine Tuning Process and Training Process

To ensure that our reinforcement learning models perform optimally, we employed Bayesian optimization using Optuna to fine-tune the hyperparameters for the PPO, RecurrentPPO, and A2C models. This systematic approach allowed us to efficiently search the hyperparameter space and

Table 1: Descriptive Statistics of Key Technical Indicators

	Close	Volume (10^8)	SMA	EMA	MACD	RSI
count	1251	1251	1251	1251	1251	1251
mean	41.3	44.6	40.8	40.8	0.71	55.9
std	40.3	18.8	39.9	39.9	1.83	14.2
min	4.89	9.79	5.48	5.52	-5.17	20.0
25%	14.0	30.7	13.9	13.9	-0.13	45.7
50%	22.2	42.1	22.0	22.1	0.34	56.1
75%	47.1	54.6	46.9	46.9	1.05	65.6
max	149.4	154.4	145.8	144.8	9.77	93.2

identify configurations that maximize the evaluation reward on a validation environment.

Fine-Tuning with Bayesian Optimization

For the PPO model, we defined an objective function in which key hyperparameters—such as the learning rate, number of steps per update (`n_steps`), discount factor (γ), entropy coefficient (`ent_coef`), clip range, batch size, generalized advantage estimation parameter (`gae_lambda`), and the number of epochs (`n_epochs`)—were sampled from specified ranges. Each trial involved training the model for 100,000 timesteps, during which an evaluation callback monitored performance every 500 timesteps. The best-performing hyperparameter set was then selected based on the mean reward observed over 20 evaluation episodes. This process was repeated for 200 trials, ensuring a comprehensive search of the hyperparameter space.

In the case of the RecurrentPPO model, we leveraged the best hyperparameters obtained from the PPO optimization whenever available. In such cases, the PPO hyperparameters were reused, and only the recurrent-specific parameters—namely, the number of LSTM layers, the hidden size, and the dropout rate—were fine-tuned. This approach capitalizes on the robustness of the PPO settings while allowing the recurrent architecture to adapt to the temporal dependencies in the data. If the PPO parameters were not available, the entire hyperparameter set for RecurrentPPO was optimized from scratch. The training setup for RecurrentPPO mirrored that of PPO, with the model trained for 100,000 timesteps and evaluated using an identical callback strategy.

For the A2C model, the fine-tuning process also followed a Bayesian optimization framework. The objective function for A2C sampled hyperparameters such as the learning rate, `n_steps` (with a smaller range compared to PPO due to its synchronous update nature), γ, `ent_coef`, `gae_lambda`, as well as A2C-specific parameters like the value function coefficient (`vf_coef`) and the maximum gradient norm (`max_grad_norm`). As with the other models, the training was conducted for 100,000 timesteps and the model's performance was evaluated on the validation environment at regular intervals. This study was also conducted over 200 trials to ensure that the selected hyperparameters provided the best trade-off between exploration and stable learning.

Training Process

The training process for all three models—PPO, RecurrentPPO, and A2C—followed a similar structure to ensure consistency and comparability. After hyperparameter tuning via Bayesian optimization, the best parameters were used to build the final model. A trading environment was created using historical stock data for training, while a separate validation environment, wrapped with a monitor, was used to evaluate performance during training. An evaluation callback was configured to run every 500 timesteps, saving the best-performing model based on validation rewards.

For example, the PPO model was trained for 300,000 timesteps with its optimized parameters. We chose 300,000 timesteps because this number struck a balance between providing the models with sufficient experience in the trading environment and managing computational resources effectively. In our preliminary experiments, using fewer timesteps resulted in inadequate convergence, where the models struggled to capture the complex market dynamics. On the other hand, significantly more timesteps offered only marginal improvements while increasing the risk of overfitting and substantially lengthening training times. It seems that around 300,000 timesteps allow our models to stabilize and generalize well across different market conditions, making it an effective choice for our final training runs. Finally, the best checkpoint was reloaded to compute the final evaluation metrics over 20 episodes. The same approach was applied to the RecurrentPPO and A2C models.

Results

We tested our reinforcement learning algorithms on Nvidia (NVDA) stock and observed varying performances. A2C achieved a total reward of **14.70** and a total profit of **0.97**, indicating moderate success. PPO, however, struggled with a total reward of **-2.49** while maintaining a total profit of **0.89**, suggesting instability in its policy learning. RecurrentPPO outperformed the other models, achieving the highest total reward of **39.39** and a total profit of **1.38**, demonstrating its ability to effectively capture temporal dependencies and optimize trading decisions for Nvidia stock.

To evaluate the generalizability of our RL algorithms, we also tested them on three different stocks: **AMD**, **ARM**, and **TEM**. We compared their performance against two common baseline strategies: **Buy & Hold** and **Moving Average Crossover**. The results are summarized in Table 2,3,and 4. Those include metrics such as **Sharpe Ratio**, **Maximum Drawdown**, and **Total Profit**.

Key Observations

- **RecurrentPPO** showed strong performance on **ARM**, achieving a Sharpe ratio of **2.45** and a total profit of **2.28**, outperforming both the baselines and other RL models. However, on **AMD**, its performance was weaker, with a Sharpe ratio of **-1.21** and a total profit of **0.72**. Lastly, on **TEM**, it achieved a decent performance with Sharpe ratio of **1.50** and a total profit of **1.78**, indicating adaptability across different market conditions.

- **A2C** showed competitive performance, particularly on **TEM**, where it achieved the highest Sharpe ratio (**1.63**) and a total profit of **1.80**. This indicates that A2C is effective in managing risk while delivering consistent returns.

- **PPO** showed varied performance across the three stocks. On **AMD**, it had the best performance compared to others with a Sharpe ratio of **0.03** and a total profit of **0.99**. On **ARM**, PPO performed better compared to its previous performance, with a Sharpe ratio of **0.52** and a total profit of **1.16**, however this wasn't enough for it to suppress RecurrentPPO . Lastly, on **TEM**, PPO underperformed, with a negative Sharpe ratio (**-0.78**) and a total profit of **0.92**, struggling in the volatile market. Overall, it can be said that PPO works well in stable markets but struggles with high volatility, suggesting a need for further improvements in such conditions.

- **Baseline Strategies**:

  - **Buy & Hold** performed well on **ARM** and **TEM**, with high total profits (**1.35** and **2.68**, respectively) and competitive Sharpe ratios. However, it performed poorly on **AMD**, with a negative Sharpe ratio (**-0.98**) and a total profit of **0.63**.
  - **Moving Average Crossover** consistently performed the worst across all stocks, with negative Sharpe ratios and low total profits. For example, on **TEM**, it had a Sharpe ratio of **-1.76** and a total profit of only **0.03**, highlighting its limitations in volatile markets.

Results on Different Metrics

- **Sharpe Ratio**: The Sharpe ratio measures risk-adjusted returns, with higher values indicating better performance relative to risk. RecurrentPPO achieved the highest Sharpe ratio for ARM and second highest for TEM stock, indicating its ability to deliver high returns with relatively low risk. In contrast, Moving Average Crossover had mostly the lowest Sharpe ratios, making it the least attractive strategy for risk-averse investors.

- **Total Profit**: The best performer in terms of total profit varied by stock. It was **PPO** for **AMD**, **RecurrentPPO** for **ARM**, and **Buy & Hold** for **TEM**. This suggests that the most effective strategy depends on the specific characteristics of the stock, highlighting the importance of adapting the strategy to the stock's behavior.

- **Maximum Drawdown**: This metric measures the largest peak-to-trough loss, with lower values indicating better risk management. A2C showed a really good performance on this metrics maximum drawdowns, suggesting it is the most robust in managing losses during downturns. For example, on **TEM**, A2C had a maximum drawdown of only **-0.02**, compared to **-0.79** for Buy & Hold.

Table 2: Performance of AMD stock

Model	Sharpe R.	Max Drawdown	Profit
Buy & Hold	-0.98	-0.47	0.63
Moving Avg Crossover	-0.48	-0.51	0.73
PPO	0.03	-0.16	0.99
RecurrentPPO	-1.21	-0.28	0.72
A2C	-0.19	-0.13	0.96

Table 3: Performance of ARM stock

Model	Sharpe R.	Max Drawdown	Profit
Buy & Hold	0.68	-0.48	1.35
Moving Avg Crossover	-0.58	-0.61	0.54
PPO	0.52	-0.32	1.16
RecurrentPPO	2.45	-0.09	2.28
A2C	0.16	-0.12	1.03

Table 4: Performance of TEM stock

Model	Sharpe R.	Max Drawdown	Profit
Buy & Hold	1.46	-0.79	2.68
Moving Avg Crossover	-1.76	-0.99	0.03
PPO	-0.78	-0.17	0.92
RecurrentPPO	1.50	-0.22	1.78
A2C	1.63	-0.02	1.80

Conclusion

This study highlights the importance of customized reinforcement learning (RL) algorithms tailored to different stocks and market conditions. Our experiments demonstrate that RL-based approaches, particularly Recurrent Proximal Policy Optimization (Recurrent PPO), generally outperform traditional methods like Buy & Hold and Moving Average Crossover by achieving higher rewards and profits. Recurrent PPO excels in capturing temporal dependencies, delivering superior performance on stocks such as Nvidia (NVDA), while A2C consistently shows strong risk management across various assets. Although PPO performs well in stable conditions, its struggles in volatile markets underscore the need for further refinements. Overall, these results underscore the adaptability of RL algorithms in dealing with various market dynamics, from stable to volatile conditions. The experimental superiority of RL-based strategies over traditional methods is promising, suggesting that customized, data-driven approaches can significantly improve trading performance. Future research should focus on further refining these algorithms to improve their robustness and generalizability across even broader financial contexts.

References

[Cornalba et al. 2023] Cornalba, F.; Disselkamp, C.; Scassola, D.; and Helf, C. 2023. Multi-objective reward generalization: Improving performance of deep reinforcement learning for applications in single-asset trading.

[Haghpanah 2019] Haghpanah, M. A. 2019. Gym-anytrading: Openai gym environments for reinforcement learning-based trading algorithms. https://github.com/AminHP/gym-anytrading. MIT License.

[Huang 2018] Huang, C. Y. 2018. Financial trading as a game: A deep reinforcement learning approach.

[Liu et al. 2022a] Liu, X.-Y.; Xiong, Z.; Zhong, S.; Yang, H.; and Walid, A. 2022a. Practical deep reinforcement learning approach for stock trading.

[Liu et al. 2022b] Liu, X.-Y.; Yang, H.; Chen, Q.; Zhang, R.; Yang, L.; Xiao, B.; and Wang, C. D. 2022b. Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance.

[Mnih et al. 2016] Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning.

[Pleines et al. 2022] Pleines, M.; Pallasch, M.; Zimmer, F.; and Preuss, M. 2022. Generalization, mayhems and limits in recurrent proximal policy optimization.

[Ponomarev, Oseledets, and Cichocki 2019] Ponomarev, E. S.; Oseledets, I. V.; and Cichocki, A. S. 2019. Using reinforcement learning in the algorithmic trading problem. *Journal of Communications Technology and Electronics* 64(12):1450–1457.

[Schulman et al. 2017] Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms.

[Théate and Ernst 2021] Théate, T., and Ernst, D. 2021. An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications* 173:114632.