WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Stock Prediction Using Deep Reinforcement Learning

B Sai Sannidh Rayalu[1], Aman Mahant[2], Thakur Diwakar Suryavanshi[3], Vaibhav Dewangan[4], Prof. Devashish Sharma [5]

Department of Computer Science and Engineering, Shri Shankaracharya Technical Campus, Bhilai, Chhattisgarh, India
b.sai.sannidh@gmail.com[1]; amankm219@gmail.com[2]; balramthakur1973@gmail.com[3]; dewanganvaibhav900@gmail.com[4]; sharma.devashish603@gmail.com[5];

ABSTRACT

Stock trading strategies play a critical role in investment. However, it is challenging to design a profitable strategy in a complex and dynamic stock market. In this paper, we propose an ensemble strategy that employs deep reinforcement schemes to learn a stock trading strategy by maximizing investment return. We train a deep reinforcement learning agent and obtain an ensemble trading strategy using three actor-critic based algorithms: Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C),[6] and Deep Deterministic Policy Gradient (DDPG). The ensemble strategy inherits and integrates the best features of the three algorithms, thereby robustly adjusting to different market situations. The proposed deep ensemble strategy is shown to outperform the three individual algorithms and two baselines in terms of the risk-adjusted return measured by the Sharpe ratio.

Keywords: Deep Reinforcement Learning, Markov Decision Process, automated stock trading, actor-critic framework

1. Introduction

Profitable automated stock trading is key for investment companies and hedge funds. It helps optimize capital allocation and boost investment performance, especially expected return. While return maximization can rely on estimates of potential return and risk, the stock market's complexity makes it hard to consider all factors

Traditional methods aren't fully satisfactory. One two-step approach first computes expected stock returns and the covariance matrix, then finds the best portfolio by maximizing return for a given risk or minimizing risk for a set return. But this method is complex, costly, and needs frequent updates, including things like transaction costs.

Another method models stock trading as a Markov Decision Process (MDP) and uses dynamic programming. However, it struggles to scale due to the large state space of the stock market.

Recently, machine learning and deep learning have been used for prediction and classification in finance. These models combine fundamentals (e.g., earnings reports) and alternative data (e.g., market news, GPS traffic) to extract investment signals [. But they focus on stock selection, not on allocating positions between stocks

2. Methodology

To explore the potential of deep reinforcement learning (DRL) in stock trading, this project uses historical data from Dow 30 companies. The work is structured into three key stages: data preparation, environment design, and agent development.

First, stock data is collected and enhanced using popular technical indicators such as MACD, RSI, CCI, and ADX. These indicators help capture important patterns in the market and provide meaningful inputs for the learning agents.

Next, a custom trading environment is created using the OpenAI Gym framework. This environment simulates a multi-stock portfolio where an agent can decide how to allocate funds across different assets. The agent's performance is measured by changes in net worth over time, while transaction costs and realistic trading constraints are included to reflect real-world conditions.

Several DRL models are trained using the Stable-Baselines3 library. These include A2C, PPO, DDPG, SAC, and TD3 []. To improve reliability and reduce overfitting to specific models, an ensemble agent is created by averaging predictions from all five.

Finally, the trained models are integrated into a user-friendly Streamlit web app. Users can select a date range, and the app runs simulations using the trained agents. The results are shown in a net worth graph, helping users visualize performance over time. The approach makes it possible to compare different models and better understand how DRL techniques perform in the context of financial trading.

3. Stock Market Environment

Before training a deep learning agent (DRL) for financial tasks such as asset management or stock trading, it is necessary to create a simulated environment that accurately reflects real-world market activity. This simulated environment serves as a training environment for the agent, allowing it to interact with historical data, track status changes, and receive performance-based rewards The agent uses this experience to learn strategies that optimize its investment decisions over time. The goal is to replicate real-world business situations so accurately that the trained model can be easily transferred to real-world business environments

To achieve this, we use the OpenAI Gym framework to develop a custom trading environment for multi-stock portfolio management. OpenAI Gym provides a standard interface for RL environments that makes it easy to implement, train, and evaluate modular iteration agents. In our case, the environment simulates trading in a portfolio of 30 individual stocks. The agent may buy, sell or hold shares of these stocks at any time depending on market conditions. [8], [9], [10].

A. multi-stock environment

Our custom environment simulates an agent's interactions with 30 different stocks using continuous action space. This allows the agent to allocate parts of its portfolio to different assets rather than being limited to separate buy, sell and hold actions. This continuous representation is more realistic and allows for more precise control over investment decisions.

The state space consists of a 181-dimensional vector that contains all relevant information needed by the agent to make an informed decision. This includes:

- Available balance: The cash at the disposal of the agent.

- Adjusted Closing Price: This provides price data that is adjusted for actions such as splits and dividends, thus providing a more accurate picture of market value.

- Shares Owned: Number of shares currently owned.

- Technical Indicators: We aggregate various technical indicators commonly used by traders and analysts, including:

- MACD (Moving Average Convergence Divergence): Analyzes changes in the strength, direction, speed and duration of a trend.

- RSI (Relative Strength Index): Measures movement and changes in price action in identifying overbought or oversold conditions.

- CCI (Commodity Channel Index): Helps identify volatile trends in stock prices.

- ADX (Average Directional Index): Measures the strength of a trend, regardless of its direction.

Together, these features provide a comprehensive set of data points that an agent can use to assess the market and adjust their strategy accordingly This diverse input helps in better generalization of the model and more complex decision making.

B. Memory management

Training reinforcement learners in a multi-stock environment with large datasets and high-dimensional spaces can be extremely memory intensive. To reduce the risk of excessive memory usage, especially with thousands of time steps and many technical indicators, we implement a load demand strategy

Instead of loading all stock data, indicators and results into memory at once (which can easily exceed system limits), we dynamically load only the data needed for the current time step or group This ensures the system remains responsive and scalable even as the dataset grows. This strategy is especially important in long training runs or when using different model architectures and extreme parameters.

By optimizing environment design and memory usage, we build a robust and efficient system for training DRL agents in a realistic financial environment. This forms the basis for the development of smart trading systems that can adapt to complex market conditions.

4. Agent Based on Deep Reinforcement Learning

We use three actor-critic algorithms—A2C, DDPG, and PPO—to implement our trading agent. To create a more robust trading strategy, we combine them using an ensemble approach.

A. Advantage Actor Critic (A2C)

A2C improves policy gradient updates by using an advantage function to reduce variance. Unlike value-only estimation, the critic network estimates how much better an action is compared to average. Multiple agents run in parallel, interact with the same environment, and compute gradients. These are averaged and used to update a global network, improving training diversity and speed—ideal for stock trading due to its stability.

B. Deep Deterministic Policy Gradient (DDPG)

DDPG is designed to maximize returns in continuous action spaces. Unlike DQN, it maps states to actions directly using policy gradients. At each step, the agent performs an action, gets a reward, and stores the transition in a replay buffer. A batch of samples is used to update the Q-value: The critic network minimizes the loss

DDPG is well-suited for stock trading due to its effectiveness with continuous actions.

C. Proximal Policy Optimization (PPO)

PPO controls policy updates to prevent drastic changes using a clipped objective. It simplifies TRPO's method and ensures stable training. PPO is chosen for being stable, fast, and easy to implement and tune.

D. Ensemble Strategy:

To build a strong trading model, we use an ensemble strategy to pick the best agent (PPO, A2C, or DDPG) based on Sharpe ratio performance.

Step 1: Retrain all agents every 3 months using a growing window of data.

Step 2: Validate using a 3-month rolling window and select the agent with the highest Sharpe ratio:

Sharpe ratio = $(\bar{r}p - rf) / \sigma p$ We also adjust risk sensitivity using a turbulence index.

Step 3: The best agent is used to trade in the next quarter.

Each algorithm performs differently across market trends. Some handle bullish or bearish trends better, while others manage volatility well. The ensemble picks the agent that offers the best return for the level of risk taken.

5. User Interface Development

To enhance user accessibility and interaction, a web-based interface was developed using Streamlit, an open-source Python library designed for building data apps with minimal effort. The interface provides an intuitive and interactive platform where users can simulate stock trading scenarios using pre-trained deep reinforcement learning (DRL) agents.

The core functionality of the interface allows users to select a custom date range, which defines the historical period for simulation. Once the start and end dates are chosen, the application loads the relevant CSV files containing historical stock data for the Dow 30 companies. The data is then preprocessed by computing several technical indicators such as MACD, RSI, CCI, and ADX to enrich the input features used by the models.

A key feature of the interface is its ability to load and run a selected DRL model—such as A2C, PPO, or an ensemble model—and execute it within the custom-built trading environment. As the simulation runs, it tracks the net worth of the portfolio at each timestep.

For visualization, the interface generates a line chart that displays changes in the portfolio's net worth over time, allowing users to observe the agent's trading performance. This interactive visualization provides immediate feedback and helps users evaluate how well the agent performs under various market conditions.

Overall, the Streamlit interface serves as a user-friendly bridge between complex AI models and end users, making it easier to test, observe, and understand the dynamics of DRL-based stock trading systems.

6. Performance Evaluation

To test how well our strategy works, we ran backtests using the three individual reinforcement learning agents (PPO, A2C, and DDPG) and our combined ensemble strategy. The results showed that the ensemble method performed better than any single agent, as well as traditional methods like the Dow Jones Index (DJIA) and the minimum-variance portfolio. This was measured using the Sharpe ratio, which balances return with risk.

A. Stock Data Preparation

We used historical daily stock data from the Dow Jones 30 companies. The data covers the period from January 1, 2009, to May 8, 2020. For training, we used data from 2009 to September 2015. Data from October to December 2015 was used to validate the models. Finally, testing was done on data from January 2016 to May 2020. To help the models adjust to market changes, we continued training even during the trading phase. [11]

B. Comparing Agent Performance

Here's how the agents were selected:

- PPO performed best from Oct–Dec 2015 and was used from Jan–Mar 2016.
- DDPG performed best from Jan–Mar 2016 and was used from Apr–Jun 2016.
- A2C was best in early 2020 and used from Apr–May 2020.

We compared the agents using five metrics:

- **Cumulative return**: how much the portfolio grew overall
- **Annual return**: average return each year
- **Volatility**: how much returns fluctuate
- **Sharpe ratio**: return relative to risk
- **Max drawdown**: biggest loss during trading

C. Key Findings

- A2C had the lowest risk and losses, making it good for down markets.
- PPO gave the highest returns, performing well in rising markets.
- DDPG was similar to PPO but slightly less effective.
- All three agents did better than the DJIA and the traditional portfolio method.

D. Market Crash Test

During the 2020 market crash, all models (including the ensemble) reacted well. They sold off assets when market turbulence was high and resumed trading afterward. This helped reduce losses. The model can be adjusted to take even less risk if needed.

7. Results

Test set performance shows the ensemble agent delivering robust performance by combining the strengths of PPO, A2C, and DDPG. It consistently outperformed individual agents across the testing period. The ensemble achieved superior stability and returns with moderate volatility.

The following chart illustrates net worth over time.

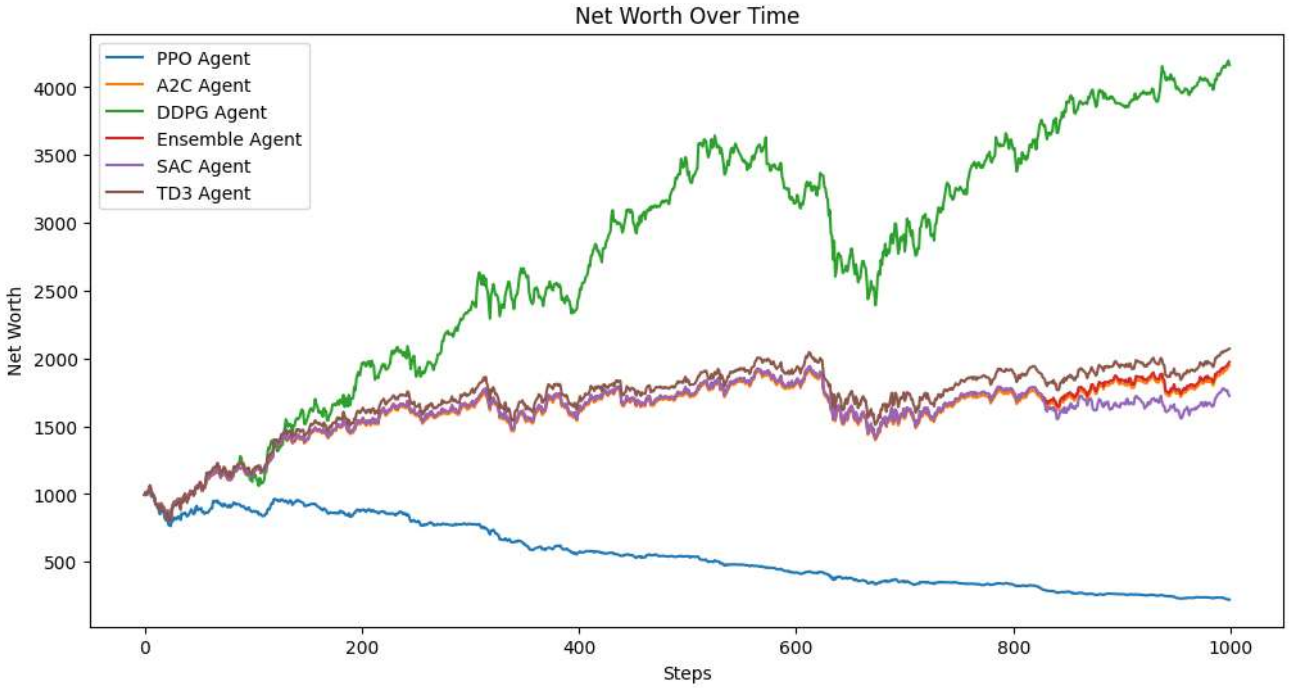

Figure 1: Displaying Net Worth Over Time for PPO, A2C, DDPG, and Ensemble agents.

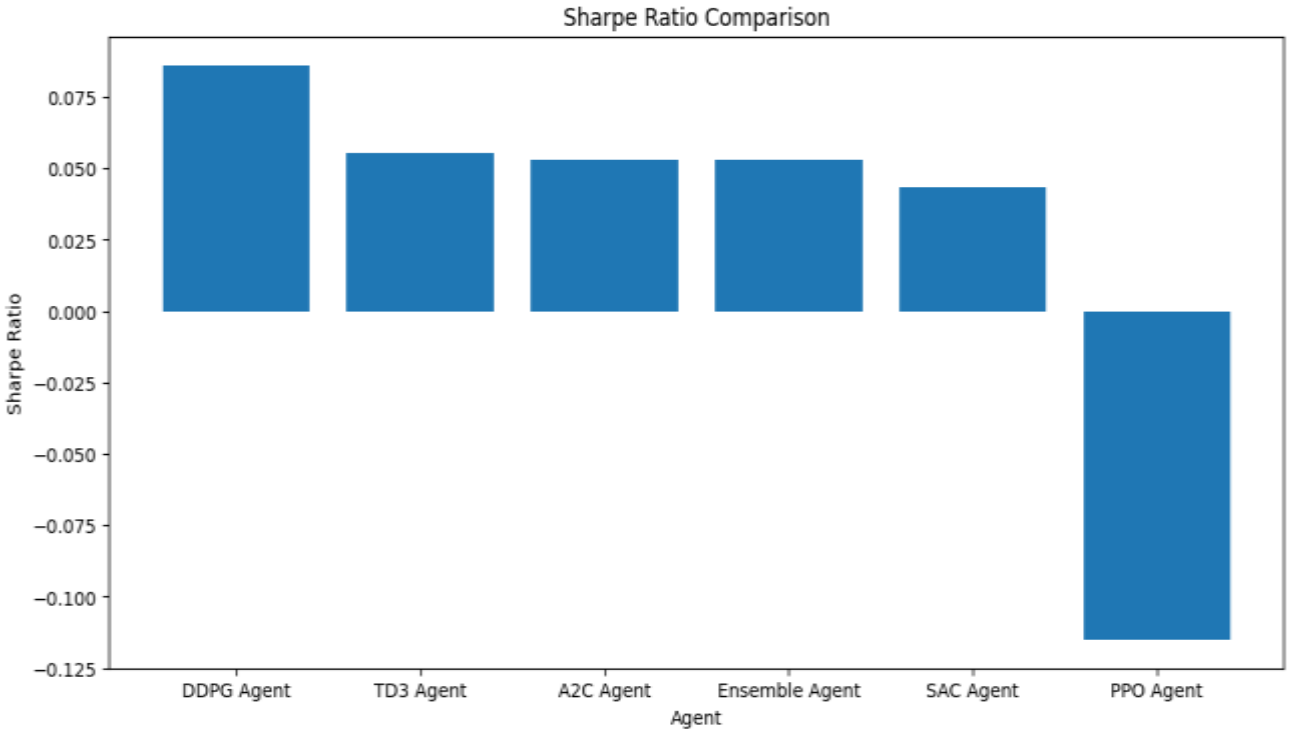

Figure 2:Sharpe ratio comparison.

8. Conclusion

In this paper, we explored how actor-critic based algorithms—Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Deterministic Policy Gradient (DDPG)—can be used to learn a stock trading strategy. To adapt to different market conditions, we implemented an ensemble strategy that automatically selects the best-performing agent using the Sharpe ratio.

We can also enrich the state space by incorporating advanced transaction cost and liquidity models [47], adding fundamental analysis indicators [9], analyzing financial news using natural language processing [48], and including ESG features [12]. One direction of interest is to directly use the Sharpe ratio as the reward function. However, doing so would require access to much more historical data, and the state space would grow significantly as a result.

9. References

[1] *Vijay Konda and John Tsitsiklis, "Actor-critic algorithms," Society for Industrial and Applied Mathematics, vol. 42, 04 2001.*

[2] *John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, "Proximal policy optimization algorithms," arXiv:1707.06347, 07 2017.*

[3] *Zhipeng Liang, Kangkang Jiang, Hao Chen, Junhao Zhu, and Yanran Li, "Adversarial deep reinforcement learning in portfolio management," arXiv: Portfolio Management, 2018.*

[4] *Volodymyr Mnih, Adria Badia, Mehdi Mirza, Alex Graves, Timo- ` thy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," The 33rd International Conference on Machine Learning, 02 2016.*

[5] *Zihao Zhang, "Deep reinforcement learning for trading," ArXiv 2019, 11 2019.*

[6] *Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, "Continuous control with deep reinforcement learning," International Conference on Learning Representations (ICLR) 2016, 09 2015.*

[7] *W.F. Sharpe, "The sharpe ratio," Journal of Portfolio Management, 01 1994.*

[8] *Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, "Openai gym," 2016.*

[9] *Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.*

[10] *Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018*

[11] *Wharton Research Data Service, "Standard & poor's compustat," 2015, Data retrieved from Wharton Research Data Service*