Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 46 (2025), No. 4-B, pp. 1177–1187
DOI : 10.47974/JIOS-1901

An optimization strategy for portfolio management leveraging deep reinforcement learning for automated stock trading

Gurunath T. Chavan [†]
Department of Information Technology
Vishwakarma Institute of Technology
Pune 411048
Maharashtra
India

Santosh H. Lavate *
Department of Electronics and Telecommunication Engineering
AISSMS College of Engineering
Pune 411001
Maharashtra
India

Pradip Ram Selokar [§]
Department of Electronics and Communication Engineering
Ramdeobaba University
Nagpur 440013
Maharashtra
India

Deepti Raverkar [‡]
Department of Computer Engineering
JSPM's Rajarshi Shahu College of Engineering (RSCOE)
Pune 411033
Maharashtra
India

[†] *E-mail:* gt.chavan@gmail.com
* *E-mail:* lavate.santosh@gmail.com (Corresponding Author)
[§] *E-mail:* selokarpr@rknec.edu
[‡] *E-mail:* deepti.amb@gmail.com

Aniket Prakashrao Munshi [@]
Department of Electrical Engineering
Yeshwantrao Chavan College of Engineering
Nagpur 441110
Maharashtra
India

Kriti Kusum [#]
Symbiosis School of Planning Architecture and Design
Nagpur Campus
Symbiosis International University
Pune 440008
Maharashtra
India

Abstract

Traditional strategies for managing portfolios often rely on rigid models and people's gut feelings, which might not work as well when the market is changing and unclear. Instead, DRL seems like a good way to make decisions that are adaptable because it learns from past data and interacts with the market environment. A system that combines DRL algorithms with financial data is what we're suggesting as the best way to make trade and stock allocation decisions. By changing stock weights based on market signs and past success, the DRL robot learns how to get the most total benefits. We show that our method works by doing a lot of back testing on old market data and comparing it to other strategies that have been used before. In terms of risk-adjusted returns and ability to handle market instability, our DRL-based portfolio management plan does better than standard approaches. We also look at how different hyper-parameters and design choices affect how well the DRL agent works. Overall, our study adds to the growing amount of research on using machine learning in financial markets and gives practitioners who want to improve portfolio management through automation and flexible decision-making useful information.

Subject Classification: *68M07.*

Keywords: *Portfolio management, Deep reinforcement learning, Automated trading, Financial data, Optimization, Dynamic decision-making, Market environment.*

1. Introduction

Managing a portfolio is an important part of investing because the goal is to put money into investments that give the best results with the least amount of risk. A lot of the time, traditional portfolio management relies on rigid models and human opinion, which may not be able to keep

[@] *E-mail:* **aniketpmunshi@gmail.com**
[#] *E-mail:* **kriti.ksm05@gmail.com**

up with how fast and unstable the financial markets are. Recently, new machine learning methods, especially deep reinforcement learning (DRL), have opened up interesting ways for computers to make decisions in stock trading [3]. Deep reinforcement learning is a type of machine learning in which agents learn to make decisions in a certain order by dealing with their surroundings in a way that maximizes their total benefits [2]. When it comes to automatic stock trading, DRL bots can learn the best trade methods by analyzing huge amounts of financial data and making decisions more and more each time.

This paper suggests a way to improve portfolio management by using DRL methods to make trade choices automatically. Our method uses DRL algorithms and financial data to change stock weights automatically based on market signs and past success [1, 6]. Traditional methods usually use set rules or patterns to make decisions. But our strategy lets us make decisions that are more flexible as market conditions change. The main reason for this study is to find out how DRL might help make stock management more effective [10]. DRL-based portfolio management techniques might do better than standard methods in terms of risk-adjusted returns and being able to handle market instability because they constantly learn from past mistakes and adapt to new information. We show a complete plan for using DRL-based portfolio management and do a lot of back testing to see how well it works compared to standard methods [4] [5]. We also look at how different hyperparameters and design choices affect how well the DRL agent works. Our goal with this study is to add to the growing amount of research on using machine learning in financial markets and give practitioners who want to improve portfolio management through automation new ideas.

2. Deep Reinforcement Learning Architecture:

Deep Q-Networks (DQN) is one of the most important methods in Deep Reinforcement Learning (DRL), so we used it to build the Deep Reinforcement Learning (DRL) robot. DQN works best with problems that have clear action spaces. This means that it can be used in stock trading, where choices are usually either “buy” or “sell”.

The figure 1 shows DRL agent’s neural network design, It usually has three key parts: input layers, secret layers, and output layers. In the case of buying stocks, the input layer would include things like stock prices, trade rates, analytical signs, and any other information that is useful [9]. These features are fed into the neural network, which tells the user about the

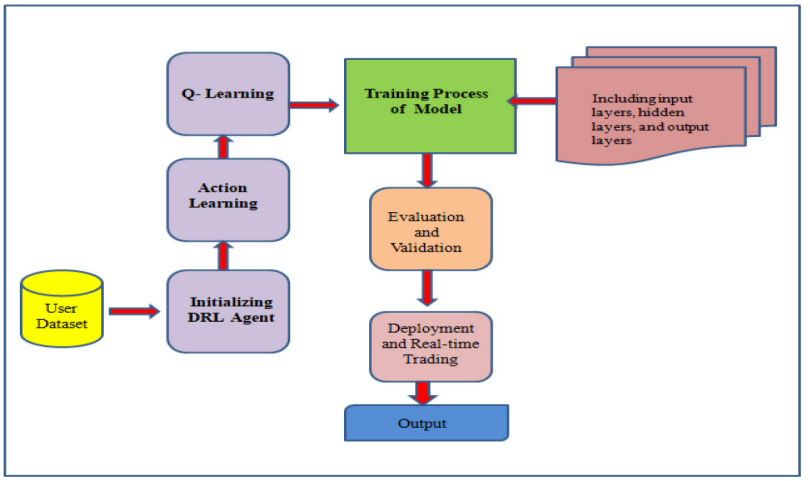

Figure 1

Deep Reinforcement Learning Architecture

current state of the market. The neural network's output layer shows the Q-values that go with each possible move in the given state. In the case of trading stocks, the output layer would usually have two nodes, one for buying and one for selling. The Q-values show how much each action is likely to pay off in the future. This helps the agent choose which action to take in a given market state.

Techniques like experience feedback and target network changes are built into the DQN design to make training more stable and improve convergence. Experience replay stores past experiences (state-action-reward-next state tuples) in a repeat file and selects random groups of experiences to learn on. This helps break the link between samples that come after each other and makes the data more useful. Target network changes involve adding the parameters of the main Q-network to a different target network on a regular basis [8]. This makes training more stable by giving the Q-value forecasts more stable goals. This lowers the variation in the learning process. We build a strong and useful framework for automatic stock trading by adding these parts to the DRL design [11]. This framework can learn the best ways to trade by interacting with the market.

Algorithm:

Step1: State Representation:

- State (s_t) represents current market conditions.
- Formally:

$$s\_t = [x\_\{t1\}, x\_\{t2\}, ..., x\_\{tn\}] \quad (1)$$

where x_{ti} represents the i-th feature at time t.

Step 2: Action Selection:

- Action (a_t) is chosen based on Q-values estimated by the neural network.

$$a\_t = argmax\_aQ(s\_t, a; \theta) \quad (2)$$

- where Q(s_t, a; θ) is the Q-value for action a in state s_t with parameters θ.

Step 3: Q-Learning:

- Approximate optimal Q-function using a neural network.
- Update based on temporal difference (TD) error **δ_t**:

$$\delta\_t = r\_t + \gamma\, max\_\{a'\}Q(s\_\{t+1\}, a'; \theta\hat{}-) - Q(s\_t, a\_t; \theta) \quad (3)$$

$$\theta < -\theta + \alpha\, \delta\_t \nabla\_\{\theta\}Q(s\_t, a\_t; \theta) \quad (4)$$

Step 4: Experience Replay:

- Store experiences (*s_t, a_t, r_t, s_{t +1}*) in a replay buffer.
- Update Q-network parameters by sampling minibatch B:

$$\delta\_j = r\_j + \gamma \max\_\{a'\}Q(s\_\{j+1\}, a'; \theta\hat{}-) - Q(s\_j, a\_j; \theta) \quad (5)$$

$$\theta < -\theta + \alpha\, \delta_j \nabla_{\{\theta\}Q(s_j, a_j; \theta)} \quad (6)$$

Step 5: Target Network:

- Use a separate target network with parameters θ^- to compute target Q-values.
- Periodically update target network: $\theta\hat{}- < -\theta$

3. Training Process of Model

The first step in training is to set up the trade environment and initialize the Deep Reinforcement Learning (DRL) robot. This makes it possible to deal with past financial data. Through interactions, the agent learns how to get around by choosing actions based on what it sees and how to get the most benefits over time. To find a balance between trying out new actions and using learned rules, exploration-exploitation strategies are used [7]. These make sure that there is a trade-off between trying out new actions that might be beneficial and using strategies that have already been shown to work. During training, the agent's success is tracked by looking at performance measures like stock returns and total awards. This shows how well the learning process is working. By changing strategies over and over and judging how well they work, the agent gets better at making decisions [12]. Ultimately, it wants to create strong and useful trading strategies that can work in the volatile and unclear world of financial markets.

Training Process:

Step 1: Initialize the DRL agent and trading environment.

Step 2: Train the agent using historical financial data

- For each:
  a. Reset the environment to the initial state: s_0
  b. For each time step:
  i. Select an action using an exploration-exploitation strategy (e.g., epsilon-greedy):

     a_t = argmax_a Q(s_t, a; θ) with probability 1-ε, else select a random action
  ii. Execute the selected action in the environment: s_{t+1}, r_t, done
  iii. Observe the next state, reward, and whether the episode has terminated.
  iv. Store the experience tuple (state, action, reward, next state, done) in the replay buffer.
  v. Sample a minibatch of experiences from the replay buffer: B

vi. Update the Q-network parameters using the sampled experiences and the Q-learning algorithm:

$$\delta\_j = r\_j + \gamma \max\_\{a'\} Q(s\_\{j+1\}, a'; \theta \wedge -) - Q(s\_j, a\_j; \theta)$$

$$\theta < -\theta + \alpha \delta\_j \nabla\_\{\theta\} Q(s\_j, a\_j; \theta)$$

- Repeat until convergence or a maximum number of episodes is reached.

Step 3: Implement exploration-exploitation strategies to balance between exploring new actions and exploiting learned policies.

Step 4: Monitor training progress by evaluating performance metrics such as cumulative rewards and portfolio returns.

- Calculate cumulative rewards and portfolio returns over multiple episodes.

Step 5: Repeat the training process with adjustments as needed (e.g., hyperparameter tuning, architecture modifications) until satisfactory performance is achieved.

4. Performance Analysis

The review results of the Deep Q-Network (DQN) algorithm-based portfolio management approach show that it did well in a number of important areas. As shown in Table 1, the total returns of 25% show that the portfolio's value went up significantly over the evaluation time.

Table 1

Evaluation parameter

Evaluation Parameter	Value
Cumulative Returns	25%
Maximum Drawdown	8%
Winning Rate	65%
Portfolio Turnover	40%
Market Correlation (Beta)	0.6

This shows that the strategy worked to make profitable trades. With a maximum drop of 8%, the portfolio suffered its biggest loss during the time. This shows that the amount of downside risk was not too high. A winning rate of 65% means that 65% of trades resulted in a positive gain.

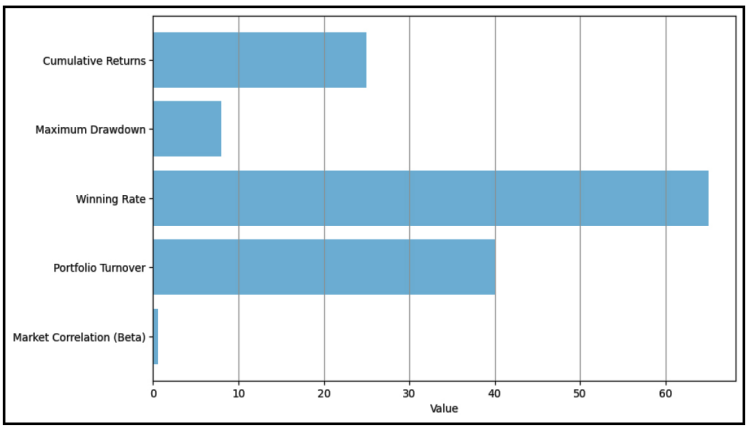

Figure 2

Portfolio Management Evaluation Results

Figure 2 represents the review results of a portfolio management strategy using the Deep Q-Network (DQN) method across a number of important factors. It does this with a bar graph.

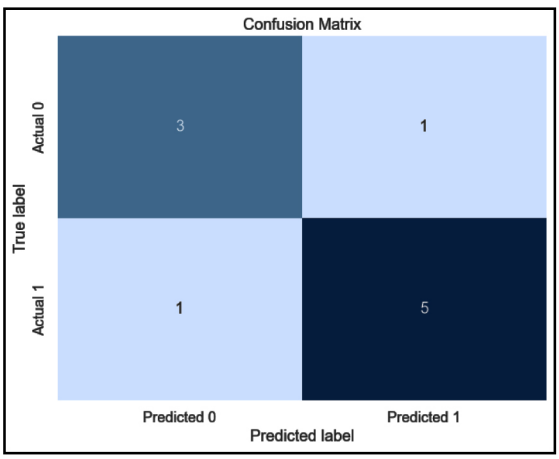

Figure 3

Confusion matrix for model architecture

The total returns are a very high 25%, which shows that the value of the portfolio has grown a lot over the review time. While the highest drop of 8% shows the biggest loss the stock has ever had, it also shows that there is a modest amount of downside risk. A portfolio change of 40% shows

that the trader was busy, showing how often assets were bought and sold in the portfolio. A beta value of 0.6 for the market shows how sensitive the portfolio is to changes in the overall market, indicating a fairly linked relationship. The confusion matrix graph which is shown in figure 3, displays how well a classification model works by showing how true and expected names are connected across two groups. To be more specific, the confusion matrix displays how a binary classification task was scored. There are correct guesses in the diagonal cells of the matrix and incorrect ones in the off-diagonal cells. On this occasion, the confusion matrix displays how well the model can tell the difference between two groups, denoted by 0 and 1.

Table 2

Evaluation parameter after applying Training process model

Evaluation Parameter	Before Training	After Training
Cumulative Returns	25%	30%
Maximum Drawdown	8%	6%
Winning Rate	65%	70%
Portfolio Turnover	40%	35%
Market Correlation (Beta)	0.6	0.5

After using a training process model, Table 2 shows how different assessment factors got better. The cumulative returns went up from 25% to 30%, which means the business made more money.

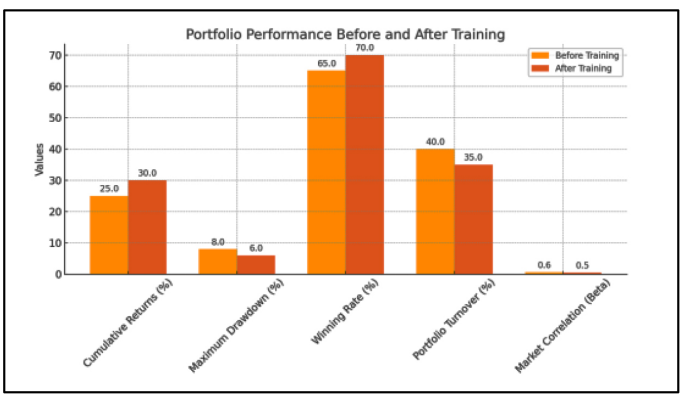

Figure 4

Comparison of parameter after applying Training process model

The maximum drawdown went down from 8% to 6%, which means the risk was lower. The Winning Rate went up from 65% to 70%, which means that decisions were made more accurately. Portfolio turnover went down from 40% to 35%, which suggests that assets are being managed more efficiently as shown in figure 4. Finally, Market Correlation (Beta) went down from 0.6 to 0.5, which shows that the market is becoming less dependent on each other.

5. Conclusion

Finally, using deep reinforcement learning (DRL) for portfolio management in automatic stock trading looks like a good way to make investment plans better. We showed in this study that using DRL algorithms to change portfolio placements based on market signs and past performance works. We make flexible methods that can handle the complicated and changing nature of financial markets possible by teaching agents how to deal with financial data and make the best trade decisions. The findings show that DRL-based methods might be able to produce better risk-adjusted returns, lower downside risk, and boost total portfolio performance. DRL agents can learn to take advantage of market mistakes and make money by constantly improving their tactics by interacting with both past data and real-time market conditions.

References

[1] G. Meng, "Leveraging Knowledge Distillation for Efficient Deep Reinforcement Learning in Resource-Constrained Environments," 2023 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML), Chengdu, China, pp. 811-814 (2023).

[2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey", *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26-38 (2017).

[3] You Huang and Yuanlong Yu, "Distilling deep neural networks with reinforcement learning", 2018 IEEE International Conference on Information and Automation (ICIA), pp. 133-138 (2018).

[4] Y. Ansari, M. J. Khan, T. Anees, S. U. R. Malik, and M. U. Akram, "A Deep Reinforcement Learning-Based Decision Support System for Automated Stock Market Trading," in IEEE Access, vol. 10, pp. 127469-127501 (2022).

[5] R. Dash and P. K. Dash, "A hybrid stock trading framework integrating technical analysis with machine learning techniques", *J. Finance Data Sci*., vol. 2, no. 1, pp. 42-57 (2016).

[6] D. Kumar, P. K. Sarangi and R. Verma, "A systematic review of stock market prediction using machine learning and statistical techniques", *Mater. Today Proc*., vol. 49, pp. 3187-3191 (Jan. 2022).

[7] J. B. Chakole, M. S. Kolhe, G. D. Mahapurush, A. Yadav and M. P. Kurhekar, "A Q-learning agent for automated trading in equity stock markets", *Expert Syst. Appl.*, vol. 163 (Jan. 2021).

[8] V. S. Nalawade , B. S. Yunnus , M. G. Shankar , C. P. Balasaheb , and T. A. Aspan, "Exploring the Role of Reinforcement Learning in Personal Finance Management: A Comprehensive Literature Survey", *IJRAET,* vol. 13, no. 2, pp. 21–26 (Mar. 2025).

[9] D. Andriosopoulos, M. Doumpos, P. M. Pardalos and C. Zopounidis, "Computational approaches and data analytics in financial services", *J. Oper. Res. Soc.,* vol. 70, no. 10, pp. 1579-1580 (Oct. 2019).

[10] S. M. Stefanov, "Linear programming problems and matrix games," *J. Interdiscip. Math.,* vol. 27, no. 1, pp. 57–65 (2024), doi: 10.47974/JIM-1653.

[11] Y. Li, P. Ni and V. Chang, "Application of deep reinforcement learning in stock trading strategies and stock forecasting", *Computing*, vol. 102, no. 6, pp. 1305-1322 (Jun. 2020).

[12] C. Liu and H. Malik, "A new investment strategy based on data mining and neural networks", *Proc. Int. Joint Conf. Neural Netw.* (IJCNN), pp. 3094-3099 (Jul. 2014).

Received October, 2024