Comparison of DQN And Double DQN Reinforcement Learning Algorithms for Stock Market Prediction

Langxi Gao

School of Information Science and Technology, Hainan Normal University, Haikou, Hainan, 571158, China
202024120411@hainnu.edu.cn

Abstract. The financial industry has always considered stock market forecasting to be vital. In recent years, the application of reinforcement learning techniques in stock market prediction has gained attention. This study aims to explore using Deep Q-Networks (DQN) and Double Deep Q-Network (DDQN) for stock market prediction. Historical stock prices and relevant market data are used as inputs to construct a reinforcement learning environment for training the DQN and DDQN models. The objective of these models is to predict the future price trends of stocks by learning optimal policies. Results demonstrate that both DQN and DDQN models exhibit strong performance in stock market prediction tasks. They are able to capture the non-linear characteristics and dynamic changes of the stock market more accurately compared to traditional indicator-based methods. Furthermore, the DDQN model shows slightly superior results in certain metrics, indicating that the use of target networks for stable training can improve prediction performance. The findings hold significance for investors and financial institutions, providing valuable insights for investment strategies and risk management. Additionally, by exploring the application of reinforcement learning methods in stock market prediction, this research offers new perspectives for further studies in the financial domain. However, the complexity and uncertainty of the market may impact prediction performance. Future research can focus on enhancing model architectures, optimizing training algorithms, and considering the incorporation of additional market information to improve the accuracy and robustness of predictions.

Keywords: Reinforcement Learning, Stock Prediction, Deep Q-Network, Double Deep Q-Network.

1 Introduction

WorldCall is a telecommunications company that operates in multiple countries, providing a range of services such as landline, mobile, and internet connectivity. As a publicly traded company, WorldCall's stock performance plays a crucial role in attracting investors and determining its market value [1]. Accurate stock prediction is essential for investors, financial institutions, and stakeholders to make informed decisions regarding buying, selling, or holding WorldCall stocks [2].

B. H. Ahmad (ed.), *Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023)*, Advances in Intelligent Systems Research 180,
https://doi.org/10.2991/978-94-6463-370-2_19

Stock prediction has long been a difficult task. Because the market is complex, which could be strongly influenced by public emergencies. Conventional methods of stock analysis often rely on fundamental analysis, technical analysis, and market sentiment analysis to forecast stock prices. These models, however, face difficulties in achieving the underlying patterns in data related to stock market [3,4].

With the rapid development of machine learning and the accumulation of data, researchers and investors have turned to machine learning and artificial intelligence techniques for stock prediction. Machine learning algorithms, represented by deep and reinforcement learning algorithms, have shown promise in capturing the non-linear relationships and temporal dependencies in stock market data [5].

There are several common methods for stock prediction. Firstly, technical analysis. Technical analysis is a method that predicts stock prices based on price and volume data. It uses tools such as chart patterns, technical indicators, and trend lines to identify price patterns and trends and make corresponding predictions. Secondly, fundamental analysis. Fundamental analysis predicts stock prices by studying a company's fundamental data. This includes analyzing financial statements, industry trends, market prospects, and competitive conditions to evaluate the company's value and potential growth. Thirdly, machine learning. Machine learning methods have gained increasing attention in stock prediction. These methods use a large amount of historical stock data as input and develop prediction models using a variety of automated learning techniques, such as support vector machines, random forests, and decision trees [6].

This research aims to apply machine learning techniques, specifically deep learning models, in order to forecast stock prices of WorldCall. The study will utilize historical stock data, financial indicators, and market sentiment data as inputs to train and evaluate the predictive models. The primary objective is to develop an accurate and robust prediction model that can assist investors in making informed decisions regarding WorldCall stocks. The outcomes of this research have significant implications for investors, financial institutions, and WorldCall itself. Accurate stock prediction can help investors optimize their investment portfolios, reduce risks, and maximize returns. Additionally, WorldCall can leverage the insights gained from stock prediction to make strategic business decisions and enhance its market value.

In conclusion, this research aims to apply machine learning techniques to predict the stock prices of WorldCall. By utilizing advanced algorithms and analyzing large-scale financial data, the study seeks to provide valuable insights for investors and stakeholders, contributing to the field of stock prediction and its application in the telecommunications industry.

2 Method

2.1 Deep Q-Network (DQN)

Overview. The DQN is mainly composed of the following modules [7]. Firstly, Q-function. The Q-function is used to estimate the cumulative reward calculated from a state-action pair. $Q(s, a; \theta)$ represents that Q-value is selecting action a in state s, where θ denotes trainable parameters in the model.

Secondly, target Q-value. It is leveraged to update the parameters of the DQN network.

$$Target\ Q - Value\ = r\ + \gamma\ *\ max\ Q(s', a';\ \theta_{target}) \tag{1}$$

Here, r is the right away advantage received by acting in the present situation, s' is the next state after taking action a, γ is the discount percentage that maintains the importance of many benefits both now and in the future, and max $Q(s', a';\ \theta_{target})$ depicts the highest Q-value for selecting action a' in the next state s'. θ_{target} denotes parameters in the target network.

Thirdly, loss function. It measures the Q-value differences between the DQN network and the target Q-value. During training, the mean squared error is widely used as loss function:

$$loss\ =\ (Q(s, a;\ \theta)\ -\ target_q)^2 \tag{2}$$

Here, $Q(s, a;\ \theta)$ is the predicted Q-value by the DQN network for state s and action a, and $target\_q$ is the target Q-value.

Fourthly, Neural network update. The gradient descent algorithm is exploited for the update of neural network.

$$\theta\ =\ \theta\ -\ \alpha\ *\ \nabla loss \tag{3}$$

Here, θ represents trainable parameters, the educational rate is α, and the loss function's gradient with respect to the value of parameter θ is indicated by the letter ∇loss.

The DQN algorithm iteratively executes these steps to gradually optimize the estimation of the Q-function within a reinforcement learning environment, enabling the agent to select the optimal actions given specific states.

The workflow of the DQN. The DQN has the following five steps [8].

(1) Initialize the network: Firstly, a neural network is initialized as the Q-network, which takes as input the state and generates the Q-values of corresponding possible action (the value function for state-action pairs).

(2) Initialize the experience replay buffer: For storing the agent's encounter tuples, create the experience replay buffer. Every interaction tuple contains the reward received, the action taken, the current state, a flag indicating whether the terminal state has been reached, and the next state.

(3) Perform the learning loop: At each time step, execute the following steps:

a. Utilize an ε-greedy policy to choose an action based on the present state. The ε-greedy policy decides to take a chance with ε probability and 1-ε probability for action selection and the greatest Q-value at the moment.

b. Execute the chosen action, then look at the reward you received and the ensuing state.

c. Save the experience tuple in the experience replay buffer.

d. Test a set of experience tuples randomly from the experience replay buffer.

e. Calculate each sample's target Q-values. The target Q-value is calculated using the following formula: Q_target = reward + γ * max(Q(next_state, all_actions)), where γ is called the discount factor. It is leveraged for balancing the significance of future and current rewards.
f. Minimize the mean squared error between the predicted Q-values by the Q-network and the target Q-values.
g. Update the weight parameters of the Q-network.

(4) Repeat the learning loop: Repeat the learning loop until arriving at a predetermined stopping point (e.g., completing all possible iterations or converging to a sufficiently good policy).

(5) Use the trained Q-network for prediction: After training, utilize the trained Q-network for prediction. Based on the current state, select the action which has the largest Q-value as the agent's action.

These are the basic steps of the DQN algorithm. Through continuous learning and training, the DQN method can discover the ideal Q-value function and find the ideal policy for situations involving reinforcement learning.

2.2 Double DQN (DDQN)

Overview. The main difference between DQN and DDQN lies in the computation of the target Q-value. In DQN, it is obtained by taking the maximum Q-value over all possible actions in the next state. In DDQN, the target Q-value is estimated by using the primary model to choose the action with the largest Q-value in the following state and then using the target network to compute the Q-value for particular activity in the following state [9].

The target Q-value is used to update the parameters of the DDQN. The computation steps of the target Q-value in the DDQN algorithm equals to two:

First, using the primary network (DQN network) to select the action a_max with the maximum Q-value in the next state s':

$$a\_max = argmax\, Q(s', a;\, \theta) \tag{4}$$

Second, the target network (θ_target) is utilized to measure the Q-value for state s' following and the selected action a_{max}:

$$target\_q = r + \gamma * Q(s', a_{max};\, \theta_{target}) \tag{5}$$

Here, r represents the immediate benefit achieved by acting in the present situation, s' is the next state after taking action a, γ is a balancing weight.

The workflow of the DDQN. The DDQN has the following five steps [10].

(1) Initialize the primary and target networks: Firstly, initialize two deep neural networks: the primary and the target network. Both networks have the same architecture and are initialized with the same weights.

(2) Initialize the experience replay buffer: Create a buffer containing n experience replay for saving the agent's experience tuples.

(3) Perform the learning loop: At each time step, execute the following steps:

a. Use the primary network to decide an action according to the present situation. Strike a balance between exploration and exploitation, adopt an ε-greedy policy.

b. Execute the chosen action, then look at the prize you received and the subsequent state.

c. Store the experience tuple (current state, action, reward, next state, and terminal flag) in the experience replay buffer.

d. Test the entire batch of experience tuples randomly from the replay buffer.

e. Utilize the target network to calculate the target Q-values. First, using the primary network to select the best action for the next state. Second, assessing the Q-value of the chosen action using the target network. Third, determining the desired Q-value as the reward plus the discounted target Q-value.

f. Updating trainable weights in primary network via optimizing the mean squared error loss between the predicted Q-values and the target Q-values.

g. Updating the target network iteratively via copying the weights from the primary network.

(4) Repeat the learning loop: Repeat the learning loop until reaching a predefined stopping circumstance (e.g., fulfilling every potential iteration or converging to a sufficiently good policy).

(5) Use the trained primary network for prediction: After training, utilize the trained primary network for making predictions. Given a state, choose the major network's action with the highest Q-value.

The DDQN algorithm is an extension of the DQN algorithm that addresses the overestimation bias issue by decoupling the action selection and Q-value evaluation. By utilizing both the primary and target networks, the DDQN algorithm improves the stability and accuracy of the Q-value estimation during the learning process.

Overall, DDQN improves upon DQN by reducing overestimation bias through the use of double estimation in the target Q-value computation. This can lead to more stable and accurate Q-value estimations, which can result in improved performance in reinforcement learning tasks.

3 Result

Fig 1 and Fig 2 contain both training and testing data, where each candle represents the stock price changes over a period of time. The red, black, and blue candles in the chart represent the actions (sell, buy, hold) taken by the model based on its Q-value function during training and testing, which caused the stock price changes. It could be observed that the total reward and total profit of the model are listed in Table 1. In contrast, DDQN has higher returns and profits, so in terms of stock trading, DDQN's trading strategies are more effective and excellent

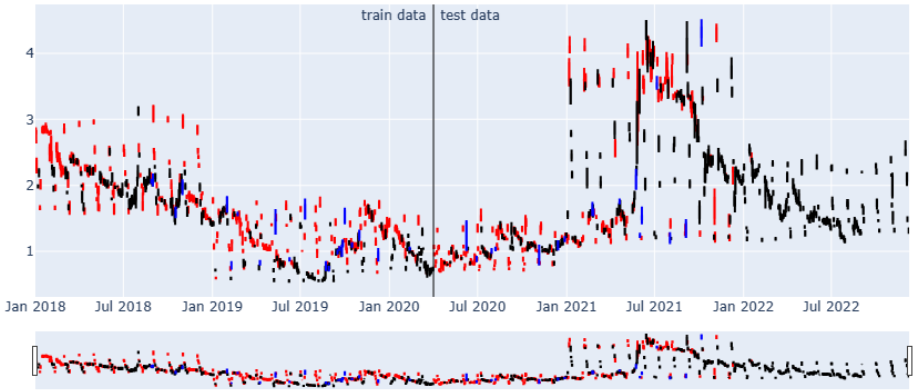

Fig. 1. Performance of DQN (Figure credit: Original).

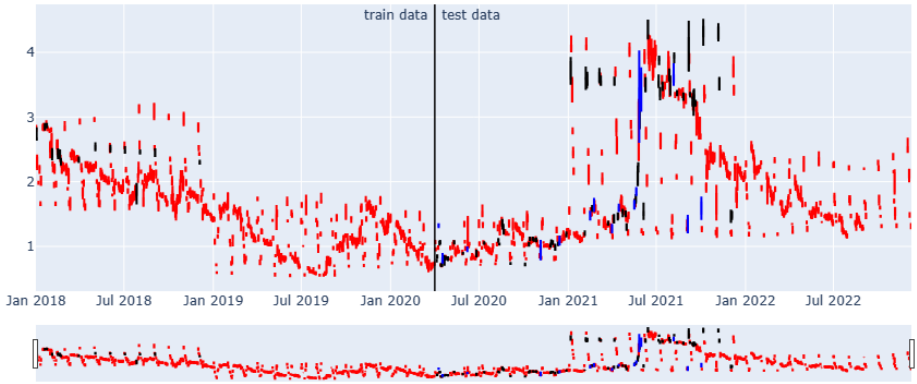

Fig. 2. Performance of Double DQN (Figure credit: Original).

Table 1. Reward comparison of DQN and Double DQN.

	Training		Testing	
	S-Reward	Profit	S-Reward	Profit
DQN	22	13	14	6
Double DQN	0	0	16	8

4 Discussion

Here is the translation of the discussion on the advantages and disadvantages of the DQN and DDQN models:

There are many advantages of the DQN model. Firstly, simple and effective. DQN is a classic algorithm in deep reinforcement learning that is easy to understand and implement. Secondly, mitigates data correlation. DQN utilizes an experience replay buffer to store and reuse the agent's experiences, reducing data correlation and

improving sample efficiency. Thirdly, handles high-dimensional state spaces. DQN uses deep neural networks to approximate the state-action value function, making it effective in dealing with problems with high-dimensional state spaces.

Still, there are many disadvantages of the DQN model. Firstly, overestimation. DQN may suffer from overestimation of Q-values, leading to learning instability and performance degradation. Secondly, time delay. DQN uses the same network to select and evaluate actions when updating target Q-values, which can introduce time delay and slow down the learning process.

As for advantages of the DDQN model. Firstly, mitigates overestimation. DDQN addresses the overestimation issue by decoupling action selection and Q-value evaluation using two separate networks, namely the primary network and the target network. Secondly, improves learning stability. DDQN uses the target network to evaluate action Q-values when updating target Q-values, reducing time delay and improving learning stability. Thirdly, enhanced performance. DDQN can achieve better performance than DQN, especially in scenarios where overestimation is a concern.

Moreover, there are disadvantages of the DDQN model. Firstly, increased computational complexity: DDQN requires maintaining two networks, the primary network and the target network, which increases computational and memory overhead. Secondly, parameter tuning. DDQN involves more hyperparameters to tune, such as the frequency of target network updates and the size of the experience replay buffer.

In summary, both the DQN and DDQN models have their own advantages and disadvantages.

Looking ahead, there are several expectations for the future of DQN, DDQN models, and the development of deep reinforcement learning. Algorithmic improvements: Further enhance the DQN and DDQN algorithms to address their limitations and drawbacks. For example, research on mitigating overestimation issues, reducing time delays, and improving learning efficiency to boost model performance. Sample efficiency enhancement: Current deep reinforcement learning algorithms have low sample efficiency when dealing with continuous action spaces and high-dimensional state spaces. Future research can focus on improving sample efficiency to make the algorithms more applicable to complex real-world tasks. Multi-objective and hierarchical learning: Extend DQN and DDQN to multi-objective reinforcement learning and hierarchical learning problems. This will enable the models to handle environments with multiple interrelated objectives and possess higher-level decision-making capabilities. Integration with other learning methods: Combine deep reinforcement learning with other learning methods such as Generative Adversarial Networks (GANs), Inverse Reinforcement Learning (IRL), etc., to enhance the models' learning capabilities and generalization. Expansion of practical applications: Apply DQN, DDQN, and other deep reinforcement learning models to a wider range of real-world scenarios such as autonomous driving, robot control, financial trading, etc. Continuously improve and optimize the models through validation and iteration in practical environments.

In summary, the expectations for DQN and DDQN models involve research in algorithmic improvements, sample efficiency enhancement, multi-objective and hierarchical learning, integration with other learning methods, and expansion of practical applications. These efforts aim to drive the progress of deep reinforcement learning,

enabling it to achieve better performance and effectiveness in more complex tasks and practical applications.

5 Conclusion

DQN is a simple and effective algorithm suitable for many reinforcement learning problems but may suffer from overestimation and time delay issues. DDQN mitigates overestimation and improves learning stability, potentially leading to better performance in certain cases, but it comes with increased computational complexity and parameter tuning challenges. DQN and DDQN are reinforcement learning algorithms that have been successfully applied in various domains, including stock market prediction. Here are the advantages of using DQN and DDQN for stock market prediction: Sequential decision-making, Handling high-dimensional data, Overcoming exploration-exploitation trade-off and Nonlinearity modeling. It's important to note that while DQN and DDQN offer advantages for stock market prediction, predicting stock prices accurately remains a challenging task due to the inherent volatility and unpredictability of financial markets. For the future development of DQN and DDQN, one can expect algorithmic improvements, integration with other reinforcement learning techniques, expansion to multi-agent reinforcement learning, incorporation of domain knowledge, and broader practical applications. These efforts aim to drive advancements in deep reinforcement learning and enable better performance and effectiveness in more complex tasks and real-world applications.

References

1. Goto, M. Financial performance analysis of US and world telecommunications companies: Importance of Information Technology in the telecommunications industry after the AT&T breakup and the NTT divestiture. Decision Support Systems, 48(3), 447-456 (2010).
2. Gandhmal, D. P., and Kumar K. Systematic analysis and review of stock market prediction techniques. Computer Science Review, 34, 100190 (2019).
3. Shah, D., Haruna I., and Farhana Z. Stock market analysis: A review and taxonomy of prediction techniques. International Journal of Financial Studies, 7(2), 26 (2019).
4. Nabipour, M., et al. Deep learning for stock market prediction. Entropy, 22(8), 840 (2020).
5. Jordan, M. I., and Tom M. Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260 (2015).
6. Agrawal, J. G., Chourasia V., and Mittra A. State-of-the-art in stock prediction techniques. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2(4), 1360-1366 (2013).
7. Roderick, M., James M., and Stefanie T. Implementing the deep q-network. arXiv preprint arXiv:1711.07478 (2017).
8. Fan, J., et al. A theoretical analysis of deep Q-learning. Learning for dynamics and control. 486-489 (2020).
9. Sewak, M., and Mohit S. Deep Q Network (DQN), Double DQN, and Dueling DQN: A Step Towards General Artificial Intelligence. Deep Reinforcement Learning: Frontiers of Artificial Intelligence, 95-108 (2019).

10. Van H., Hado, A. G., and David S. Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI conference on artificial intelligence, pp. 2094-2100, AAAI Press, USA (2016).