APPLICATION OF DEEP REINFORCEMENT LEARNING FOR INDIAN STOCK TRADING AUTOMATION

A PREPRINT

Supriya Bajpai
IITB-Monash Research Academy, IIT Bombay, India
Monash University, Australia
supriya.bajpai@monash.edu

July 1, 2021

ABSTRACT

In stock trading, feature extraction and trading strategy design are the two important tasks to achieve long-term benefits using machine learning techniques. Several methods have been proposed to design trading strategy by acquiring trading signals to maximize the rewards. In the present paper the theory of deep reinforcement learning is applied for stock trading strategy and investment decisions to Indian markets. The experiments are performed systematically with three classical Deep Reinforcement Learning models Deep Q-Network, Double Deep Q-Network and Dueling Double Deep Q-Netwonrk on ten Indian stock datasets. The performance of the models are evaluated and comparison is made.

***K*eywords** Deep reinforcement learning, Stock trading automation, Deep Q-learning, Double DQN, Dueling Double DQN

1 Introduction

A lot of work has been done to propose methods and algorithms to predict stock prices and optimal decision making in trading. A large number of indicators, machine learning and deep learning techniques [1] such as Moving averages [2, 3], linear regression [4, 5, 6], neural networks [7, 8], Recurrent neural network [9, 10, 11, 12] and Reinforcement learning (RL) have been developed to predict the stock and financial price and strategies [13, 14]. The advanced techniques of artificial neural network have shown better performance as compared to the traditional indicators and methods [15, 16]. The stock price prediction is a very challenging task as the stock market changes rapidly and data availability is also incomplete and not sufficient. Reinforcement learning is one of the methods to solve such complex decision problems. Reinforcement learning can prove to be a better altemative approach for stock price prediction [17] and maximizing expected return. Deep Learning methods have the ability to extract features from high dimentional data. However, it lacks the decision-making capabilities. Deep Reinforcement Learning (DRL) combines the Deep Learning approach with the decision making ability of Reinforcement Learning. Researchers have investigated RL techniques to solve the algorithmic trading problem. Recurrent Reinforcement Learning (RRL) algorithm have been used for discovering new investment policies without the need to build forecasting models [18]. Adaptive Reinforcement Learning (ARL) have been used to trade in foreign exchange markets [19]. Recently, people investigated DRL method to solve the algorithmic trading problem [20, 21, 22, 23, 24, 20, 25].

In the present paper Deep Reinforcement Learning is applied to Indian stock market on ten randomly selected datsets to automate the stock trading and to maximize the profit. Model is trained with historical stock data to predict the stock trading strategy by using Deep Q-Network (DQN), Double Deep Q-Network (DDQNn) and Dueling Double Deep Q-Network (Dueling DDQN) for holding, buying and selling the stocks. The model is validated on unseen data from the later period and performance is evaluated and compared.

2 Methods

Deep Q-Network, Double Deep Q-Network and Dueling Double Deep Q-Network [26] are discussed in the following sections.

2.1 Deep Q-Network

Deep Q-Network is a classical and outstanding algorithm of Deep Reinforcement Learning and it's model architecture is shown in Figure 1. It is a model-free reinforcement learning that can deal with sequential decision tasks. The goal of the learning is to learn an optimal policy $\pi^\star$ that maximizes the long term reward or profit. The agent takes action a_t depending on the current state s_t of the environment and receives reward r_t from the environment. The experience replay is used to learn from the previous experiences and is used to store the previous states, actions, rewards, and next states. The data from the replay memory is sampled randomly and fed to the train network in small batch sizes to avoid overfitting. In deep Q-learning the Convolutional Neural Network (known as Q-Network) is used to learn the expected future reward Q-value function ($Q(s_t, a_t)$). One major difference between the Deep Q-Network and the basic Q-learning algorithm is a new Target-Q-Network, which is given by:

$$Q_{target} = r_{t+1} + \gamma max_{a'}[Q(s'_t, a'_t; \theta)] \tag{1}$$

where, Q_{target} is the target Q value obtained using the Bellman Equation and θ denotes the parameters of the Q-Network. In DQN there are two Q-Networks: main Q-Network and target Q-Network. The target Q-Network is different from the main Q-Network which is being updated at every step. The network values of the target Q-Network are the updated periodically and are the copy of the main network's values. Use of only one Q-Network in the model leads to delayed or sub-optimal convergence when the data incoming frequency is very high and the training data is highly correlated and it may also lead to unstable target function. The use of two different Q-Networks increases the stability of the Q-Network.

Optimal Q-value or the action-value pair is computed to select and measure the actions. DQN takes the max of all the actions that leads to overestimation of the Q-value, as with the number of iterations the errors keeps on accumulating [27]. This problem of overestimation of Q-value is solved by using Double DQN, as it uses another neural network that optimizes the influence of error.

2.2 Double Deep Q-Network

The above problem of overestimation becomes more serious if the actions are taken on the basis of a Target Q-Network as the values of the Target Q-Network are not frequently updated. Double DQN uses two neural networks with same structure as in DQN, the main network and the target network as it provides more stability to the target values for update. In Double DQN the action is selected on the basis of the main Q-Network but uses the target state-action value that corresponds to that particular state-action from the Target Q-Network. Thus, at each step all the action-value pairs for all possible actions in the present state is taken from the main Q-Network which is updated at each time step. Then an argmax is taken over all the state-action values of such possible actions (Equation 2), and the state-action value which maximizes the value, that specific action is selected.

$$Q_{target} = r_{t+1} + \gamma Q(s_t, argmax_{a'} Q(s^{'}_t, a^{'}_t; \theta); \theta^{'}) \tag{2}$$

But to update the main Q-Network the value that corresponds to the selected state-action pair is taken from the target Q-Network. As such we can overcome both the problems of overestimation and instability in Q-values.

2.3 Dueling Double Deep Q-Network

There are two Q-Networks in both DQN as well as in Double DQN, one is the main network and the other is the target network where the network values are the periodic copy of the main network's values. The Dueling Double DQN has non-sequential network architecture where, the convolutional layers get separated into two streams and both the sub-networks have fully connected layer and output layers. The first sub-network corresponds to the value function to estimate the value of the given state and the second sub-network estimates the advantage value of taking a particular action over the base value of being in the current state.

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + (A(s_t, a_t; \theta, \alpha) - \max_{a' \in |A|} A(s_t, a'_t; \theta, \alpha)) \tag{3}$$

here, A is the advantage value. We can get the Q-values or the action-value by combining the output of the first sub-network, that is the base value of state with the advantage values of the actions of the second sub-network. θ is common parameter vector both the sub-networks. α and β are the parameter vectors of the "Advantage" sub-network and State-Value function respectively. The Q value for a given state-action pair is equal to the value of that state which is estimated from the state-value (V) plus the advantage of taking that action in that state. We can write the above Equation 3 as follows.

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s_t; \theta, \beta) + (A(s_t, a_t; \theta, \alpha)) \tag{4}$$

From the above equation we can get the Q-value if we know the value of S and A, but we cannot get the values of S and A if Q-value is known. The last part of the Equation 3 is slightly modified as follows, which also increases the stability of the algorithm.

$$Q(s_t, a_t; \theta, \alpha, \beta) = V(s; \theta, \beta) + (A(s_t, a_t; \theta, \alpha) - \frac{1}{|A|}\sum_{a'} A(s_t, a'_t; \theta, \alpha)) \tag{5}$$

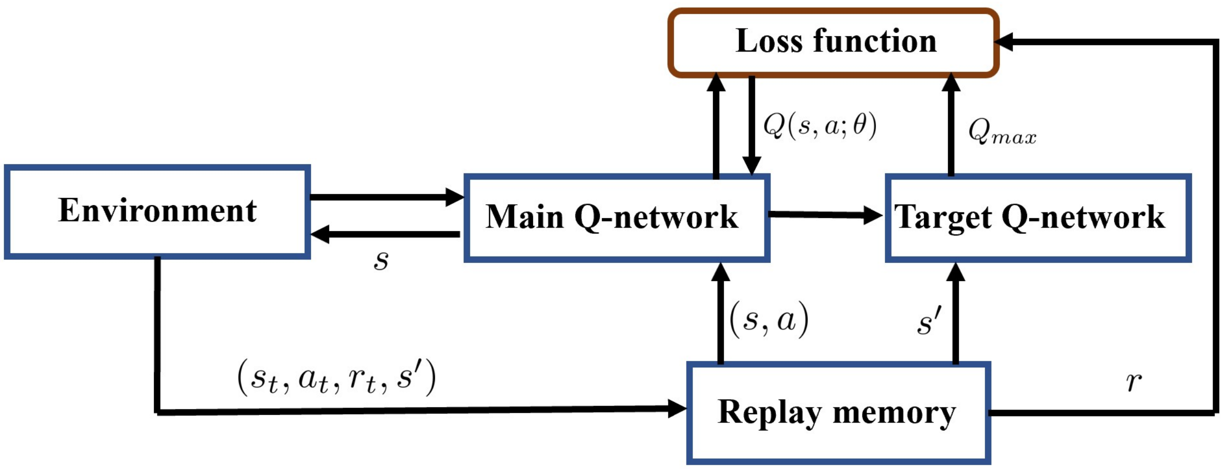


Figure 1: Deep Q-Network model architecture.

3 Experiments

In the present study we evaluate the performance of the deep reinforcement learning algorithms for stock market investment decisions on 10 Indian stock dataset. The dataset is obtained from National Stock Exchange (NSE) India, that consists of the price history and trading volumes of stocks in the index NIFTY 50. We used Deep Q-Network (DQN), Double Deep Q-Network (DDQN), and Dueling Double Deep Q-Network (Dueling DDQN) to automate the stock trading and to maximize the profit. We split the dataset for training and testing purpose in equal proportions. The training and testing dataset is fed to the models and the train and test rewards and profit are estimated and compared.

3.1 Agent Training

The Q-Network has input, hidden and output layers and the hyperparameters are tuned 1 to obtain the optimal weights. Tuning the hyperparameters of the model in time-series problems is very crucial for the long-term reward. The Q-Network is trained by minimizing the loss function as follows:

$$L(\theta) = E[(Q_{target} - Q(s_t, a_t; \theta))^2] \tag{6}$$

The learning rate is 0.00025 and the optimizer is Adam optimizer. The training is done for 50 episodes with batch size of 64 and the agent performs three actions: hold, buy and sell.

3.2 Agent Testing

The testing of the agent is done on the unseen test dataset of later periods of the same time series as the train dataset. The performance of the agent is measured in terms of total profit. The profit is calculated by sale price - purchase price.

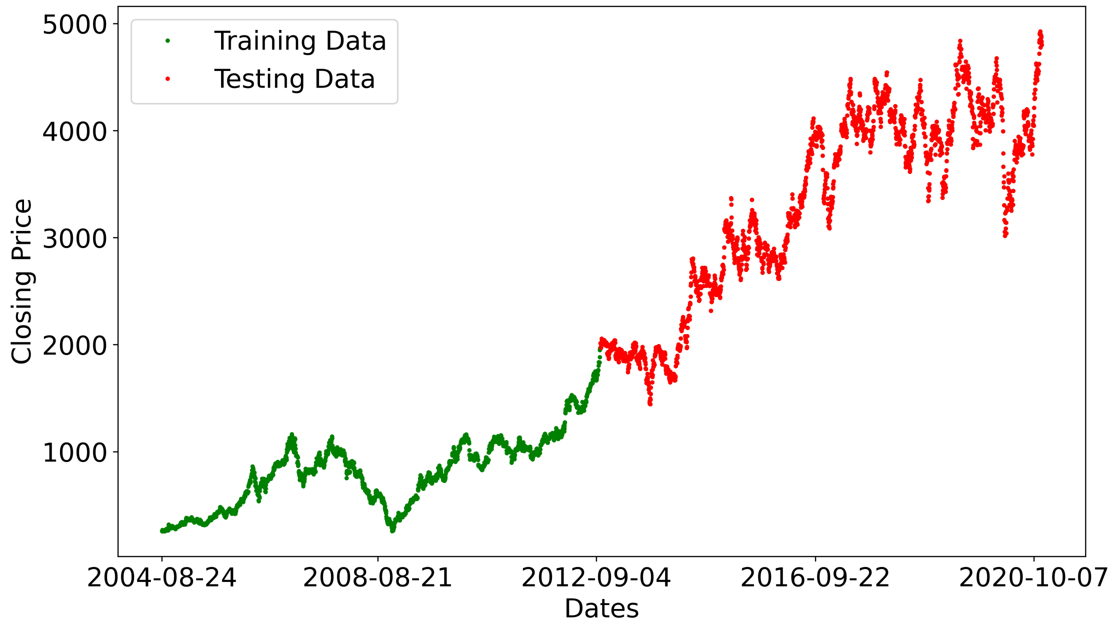


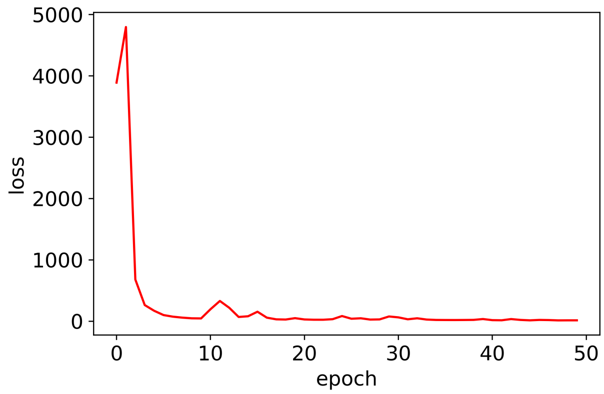
Figure 2: Plot showing train and test dataset of ULTRACEMCO stock price.

Table 1: Model hyperparameters

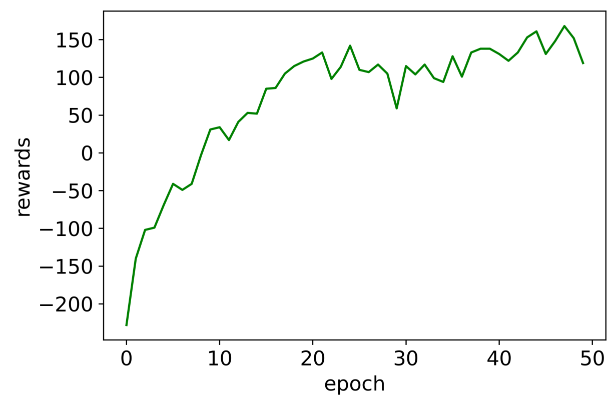
Hyperparameters	Values
Window size	90
Batch size	64
Episodes	50
Gamma	0.95
Epsilon	1
Learning rate	0.00025
Epsilon minimum	0.1
Epsilon decay	0.995
Optimizer	Adam
Loss function	Mean square error

4 Results

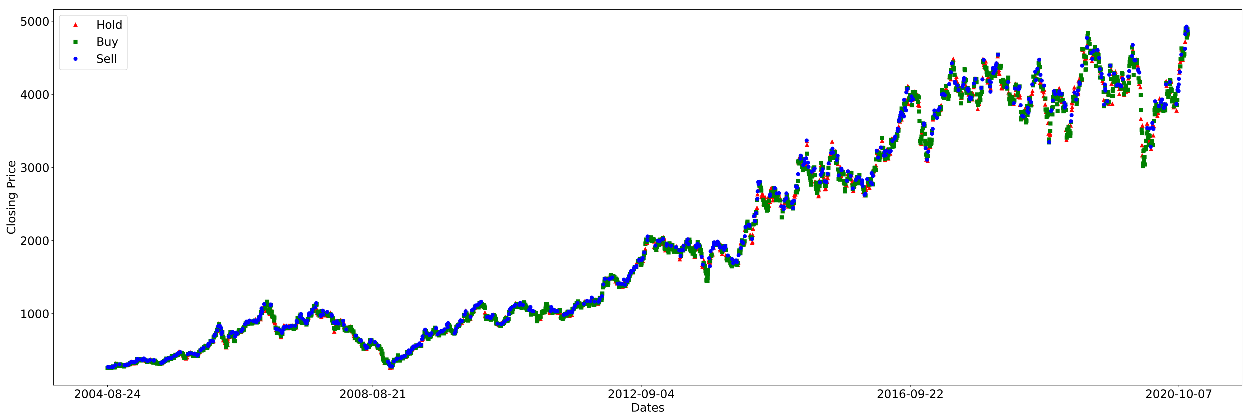
Ten Indian stock datasets and three deep Q-networds are used to perform the experiments. Each dataset is trained on train data and tested on the unseen test data. Total rewards and profit of training data and test data is calculated for ten Indian stocks using three deep reinforcement learning models (DQN, Double DQN and Dueling DDQN) are shown in Table 2,3,4 respectively. Figure 2 shows the train and test data used for each dataset. We randomly choose one stock dataset (ULTRACEMCO dataset) and plot the train and test data and also the training loss and training rewards with respect to number of epochs for DQN (Figure 3a,b). Mean square error is used to calculate the loss that estimates the difference between the actual and predicted values. Figure 3c shows the time-market value of the DQN model corresponding to the ULTRACEMCO dataset. Red, green and blue points corresponds to hold, buy and sell the stock respectively. Similarly, Figure 4a,b,c shows the training loss, training rewards and time-market value for the



(a)



(b)



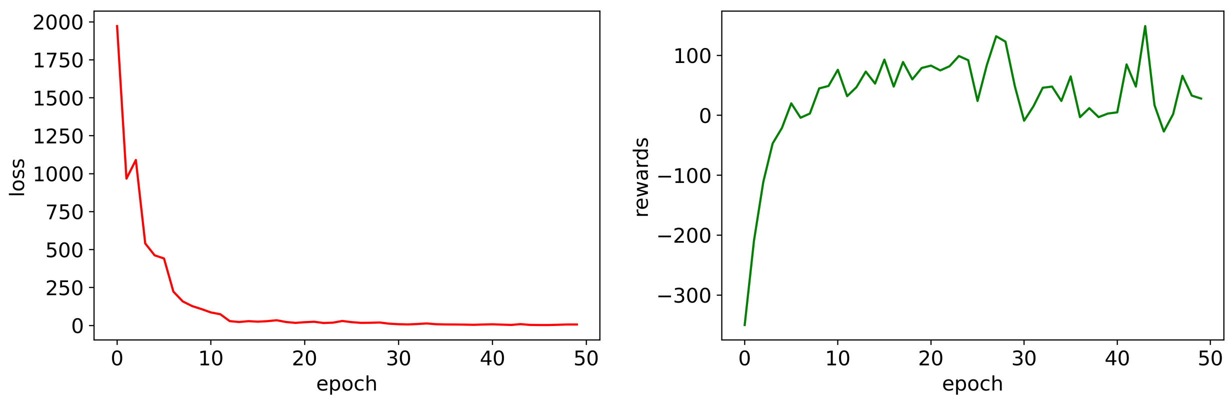
(c)

Figure 3: Plots showing (a) train loss (b) train rewards (c) time-market profile of ULTRACEMCO stock using DQN

Table 2: Rewards and profit obtained during training and testing of the Indian stock datasets using DQN.

Dataset	DQN			
	Train Rewards	Train Profit	Test Rewards	Test Profit
TCS	246	12382	22	4770
RELIANCE	117	17103	-77	1246
ZEEL	295	6639	124	2923
TATAMOTORS	210	10506	-1	1670
TECHM	-426	66	-409	-678
UPL	179	3671	82	4828
ULTRACEMCO	199	8818	16	25188
TATASTEEL	225	3481	36	48
NESTLEIND	-120	11774	-180	16389
POWERGRID	199	1145	51	807

ULTRACEMCO dataset using Double DQN. Figure 5a,b,c shows the training loss, training rewards and time-market value for the ULTRACEMCO dataset using Dueling Double DQN. From Table 2,3,4 we observe that on an average the Dueling DDQN performs better than rest two models and the performance of DDQN is better than DQN.



(a) (b)

(c) NESTLEIND train and test data

Figure 4: Plots showing (a) train loss (b) train rewards (c) time-market profile of ULTRACEMCO stock using Double DQN

Table 3: Rewards and profit obtained during training and testing of the Indian stock datasets using Double DQN.

Dataset	Double DQN			
	Train Rewards	Train Profit	Test Rewards	Test Profit
TCS	225	14946	276	38095
RELIANCE	-175	0	-211	48
ZEEL	-1	17	3	12
TATAMOTORS	52	718	85	1067
TECHM	-15	52	3	117
UPL	6	409	6	658
ULTRACEMCO	23	655	319	57626
TATASTEEL	36	1158	-8	8
NESTLEIND	7	8589	8	22016
POWERGRID	169	-174	167	814

5 Conclusion

We implemented deep reinforcement learning to automate trade execution and generate profit. We also showed how well DRL performs in solving stock market strategy problems and compared three DRL networks: DQN, DDQL and Dueling DDQN for 10 Indian sock datasets. The experiments showed that all these three deep learning algorithms perform well in solving the decision-making problems of stock market strategies. Since, the stock markets are highly stochastic and changes very fast, these algorithms respond to these changes quickly and perform better than traditional

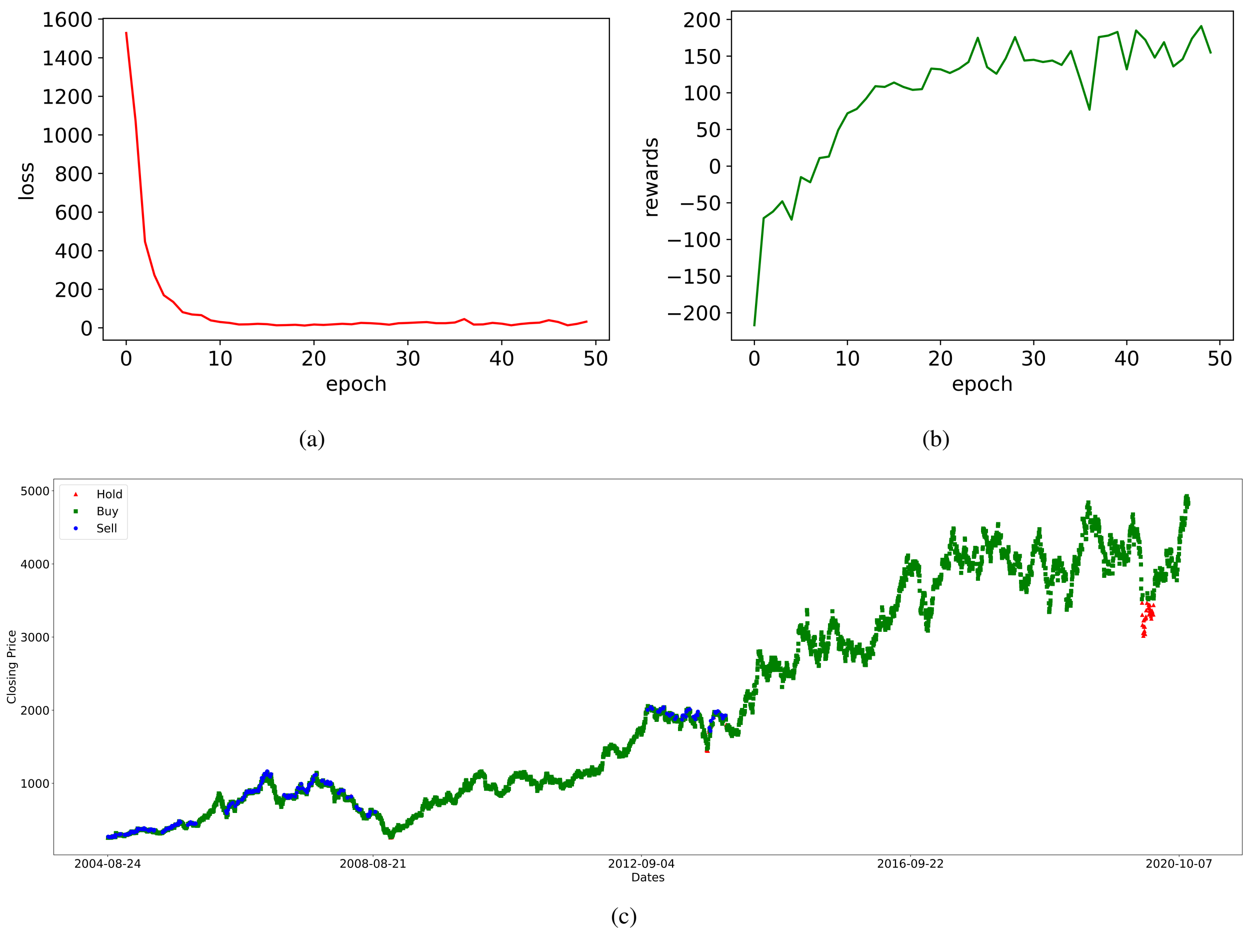


Figure 5: Plots showing (a) train loss (b) train rewards (c) time-market profile of ULTRACEMCO stock using Dueling DDQN

Table 4: Rewards and profit obtained during training and testing of the Indian stock datasets using Dueling DDQN.

Dataset	Dueling DDQN			
	Train Rewards	Train Profit	Test Rewards	Test Profit
TCS	47	3497	114	17278
RELIANCE	361	29392	347	29769
ZEEL	28	1701	151	2836
TATAMOTORS	250	16592	188	8312
TECHM	64	26024	86	14831
UPL	104	7972	176	10284
ULTRACEMCO	123	7113	35	6257
TATASTEEL	1	17	3	57
NESTLEIND	139	43900	79	101731
POWERGRID	59	560	102	1252

methods. We observe that on an average the Dueling DDQN network performed better than DDQN and DQN and Double DQN performed better than DQN.

References

[1] M Hiransha, E Ab Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Nse stock market prediction using deep-learning models. *Procedia computer science*, 132:1351–1362, 2018.

[2] SGM Fifield, DM Power, and DGS Knipe. The performance of moving average rules in emerging stock markets. *Applied Financial Economics*, 18(19):1515–1532, 2008.

[3] Adebiyi A Ariyo, Adewumi O Adewumi, and Charles K Ayo. Stock price prediction using the arima model. In *2014 UKSim-AMSS 16th International Conference on Computer Modelling and Simulation*, pages 106–112. IEEE, 2014.

[4] Dinesh Bhuriya, Girish Kaushal, Ashish Sharma, and Upendra Singh. Stock market predication using a linear regression. In *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, volume 2, pages 510–513. IEEE, 2017.

[5] Yahya Eru Cakra and Bayu Distiawan Trisedya. Stock price prediction using linear regression based on sentiment analysis. In *2015 international conference on advanced computer science and information systems (ICACSIS)*, pages 147–154. IEEE, 2015.

[6] J Gregory Trafton, Erik M Altmann, and Raj M Ratwani. A memory for goals model of sequence errors. *Cognitive Systems Research*, 12(2):134–143, 2011.

[7] Goutam Dutta, Pankaj Jha, Arnab Kumar Laha, and Neeraj Mohan. Artificial neural network models for forecasting stock price index in the bombay stock exchange. *Journal of Emerging Market Finance*, 5(3):283–295, 2006.

[8] Reza Gharoie Ahangar, Mahmood Yahyazadehfar, and Hassan Pournaghshband. The comparison of methods artificial neural network with linear regression using specific variables for prediction stock price in tehran stock exchange. *arXiv preprint arXiv:1003.1457*, 2010.

[9] Zahra Berradi and Mohamed Lazaar. Integration of principal component analysis and recurrent neural network to forecast the stock price of casablanca stock exchange. *Procedia computer science*, 148:55–61, 2019.

[10] Taewook Kim and Ha Young Kim. Forecasting stock prices with a feature fusion lstm-cnn model using different representations of the same data. *PloS one*, 14(2):e0212320, 2019.

[11] David MQ Nelson, Adriano CM Pereira, and Renato A de Oliveira. Stock market's price movement prediction with lstm neural networks. In *2017 International joint conference on neural networks (IJCNN)*, pages 1419–1426. IEEE, 2017.

[12] Adil Moghar and Mhamed Hamiche. Stock market prediction using lstm recurrent neural network. *Procedia Computer Science*, 170:1168–1173, 2020.

[13] Parag C Pendharkar and Patrick Cusatis. Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, 103:1–13, 2018.

[14] Terry Lingze Meng and Matloob Khushi. Reinforcement learning in financial markets. *Data*, 4(3):110, 2019.

[15] Ren Jie Kuo. A decision support system for the stock market through integration of fuzzy neural networks and fuzzy delphi. *Applied Artificial Intelligence*, 12(6):501–520, 1998.

[16] Norio Baba and Motokazu Kozaki. An intelligent forecasting system of stock price using neural networks. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pages 371–377. IEEE, 1992.

[17] Jae Won Lee. Stock price prediction using reinforcement learning. In *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, volume 1, pages 690–695. IEEE, 2001.

[18] John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, 12(4):875–889, 2001.

[19] Michael AH Dempster and Vasco Leemans. An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30(3):543–552, 2006.

[20] Yuming Li, Pin Ni, and Victor Chang. Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing*, pages 1–18, 2019.

[21] Zhuoran Xiong, Xiao-Yang Liu, Shan Zhong, Hongyang Yang, and Anwar Walid. Practical deep reinforcement learning approach for stock trading. *arXiv preprint arXiv:1811.07522*, 2018.

[22] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

[23] João Carapuço, Rui Neves, and Nuno Horta. Reinforcement learning applied to forex trading. *Applied Soft Computing*, 73:783–794, 2018.

[24] Ioannis Boukas, Damien Ernst, Thibaut Théate, Adrien Bolland, Alexandre Huynen, Martin Buchwald, Christelle Wynants, and Bertrand Cornélusse. A deep reinforcement learning framework for continuous intraday market bidding. *arXiv preprint arXiv:2004.05940*, 2020.

[25] Jinho Lee, Raehyun Kim, Yookyung Koh, and Jaewoo Kang. Global stock market prediction based on stock chart images using deep q-network. *IEEE Access*, 7:167260–167277, 2019.

[26] Mohit Sewak. Deep q network (dqn), double dqn, and dueling dqn. In *Deep Reinforcement Learning*, pages 95–108. Springer, 2019.

[27] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.