Indonesian Journal of Data and Science (IJODAS) ISSN: 2715-9930
Vol 3, No 3, December 2022, pp. 97-106

Deep Reinforcement Learning for Tehran Stock Trading

Neda Yousefi [a,1]

[a] *Allameh Tabataba'i University, Varzesh Sq., Dehkadeh Olympic, Tehran dan 1489684511, Iran*
[1] *neda.yousefii@gmail.com;*

ARTICLE INFO

Received : 29 – 09 – 2022
Revised : 20 – 11 – 2022
Published : 31 – 12 – 2022

Keywords:
Machine Learning
Deep learning
Reinforcement learning
Deep Deterministic Policy Gradient
Actor-Critic
Stock trading

ABSTRACT

One of the most interesting topics for research and also for making a profit is stock trading. Artificial intelligence has had a great impact on this path. A lot of research has been done to investigate the application of machine learning, and deep learning methods in stock trading. Despite the large amount of research done in the field of prediction and automation trading, stock trading as a deep reinforcement-learning problem remains an open research area. The progress of reinforcement learning as well as the intrinsic properties of reinforcement learning make it a suitable method for market trading in theory. In this paper, single stock trading models are presented based on the fine-tuned state-of-the-art deep reinforcement learning algorithms (Deep Deterministic Policy Gradient (DDPG) and Advantage Actor Critic (A2C)). These algorithms are able to interact with the trading market and capture the financial market dynamics. The proposed models are compared, evaluated, and verified on historical stock trading data. Annualized return and Sharpe ratio have been used to evaluate the performance of proposed models. The results show that the agent designed based on both algorithms is able to make intelligent decisions on historical data. The DDPG strategy performs better than the A2C and achieves better results in terms of convergence, stability, and evaluation criteria.

I. Introduction

Several studies have been devoted to using machine learning in the field of financial Market and stock prediction and trading.

Among them, Stock trading is a desired topic in the financial market and has been widely discussed in modern artificial intelligence. Exploring autonomous trading algorithms that are adaptable to the dynamic trading market is an essential need for stock trading problems. The trading strategy is a kind of complex sequential decision-making problem, and deep reinforcement learning has achieved remarkable success in solving complex sequential decision-making problems. Reinforcement learning (RL) can directly learn an acting strategy, in the process of interacting with the dynamic environment; therefore, it is a competitive advantage of using deep reinforcement learning (DRL) for stock trading. DRL stock trading studies can be considered under three categories: Value-based DRL (critic-only), Policy-based DRL (actor-only), and Actor-Critic DRL. [1]

In value-based DRL approaches (critic-only), Q-learning and deep Q-Learning are applied to build a stock trading system. [2] Proposed to employ deep Q-learning to build a deep Q-trading system. Their research explains that their proposed deep Q-trading system can have better results than both buy-and-hold strategies and strategies learned by iterative reinforcement learning. These kinds of value-based DRL are always applied to solve the optimization problems defined in discrete space [3], and there is not a good paradigm for the trading problem, because the trading environment is too complex to be approximated in discrete space and also they are not good for dynamic online trading [4]. In Policy based (actor-only) DRL approaches, they learn a policy, which directly maps states to actions. These methods learn the policy directly from the continuous data, and they are a better framework for trading than the Q-learning approaches [3]. [4] Found that direct reinforcement learning (policy based) produces better trading strategies than systems utilizing Q-learning (a value function method). In actor-critic DRL approaches, the reason Actor-Critic might work well for the Stock market is that they consider the Value-based approach as well as the Policy-based approach and learns the best from both worlds. [5] Proposed the extended value-based deep Q-network (DQN) and the asynchronous

advantage actor-critic (A3C) for better adapting to the trading market and their results show that A3C-extended outperforms other models and the interesting part of the result is about the simple A3C that has close results to DQN-extended. It proves the best performance of actor-critic concerning value-based (DQN) approaches.

The Deep Deterministic Policy Gradient (DDPG) algorithm learns the Q function and the policy simultaneously. It can benefit from the use of off-policy data and the Bellman equation to learn the Q-function, and also learn the policy by applying the Q-function [6][7]. [8] explored the potential of DDPG agent training for learning stock trading strategies. Their Results show that the trained agent outperforms the Dow Jones Industrial Average and the portfolio allocation method, with the least variance in cumulative returns. Comparing the Sharpe ratios shows that the DDPG agent is more robust than the others in terms of balancing risk and return.

According to the research background, in this work two of the best-mentioned reinforcement-learning algorithms; Deep Deterministic Policy Gradient (DDPG) [6][7][8], and Advantage Actor Critic (A2C) [9]; are used for stock trading that has better results in comparison to the other methods. An environment is built and action space, state space, and reward function are defined specifically for the problem. In summary, this research studied the application of deep reinforcement learning algorithms for Tehran stock trading. Tehran stock market data is used for comparing, evaluating, and testing the proposed models.

This paper is organized as follows. Section 2 covers preliminaries, background, and some required definitions and algorithms. Section 3 contains the research methodology including formulation of the proposed stock trading problem, defining environment, architecture of both DDPG and A2C algorithms, performance evaluation, and also explanation about the stock data in this research. Section 4 describes the stock data preprocessing and our experimental setup, and presents the performance evaluation of the proposed strategies. Section 5 is conclusion.

The foremost critical highlight of recognizing reinforcement learning from other types of learning is that it employs required data by evaluating the actions taken instead of instructing them by giving correct activities. It has the potential to create the requirement for a dynamic investigation, and an unequivocal trial-and-error exploration for good behavior. Reinforcement learning can be considered an intelligent system where an agent learns from its environment through interaction and evaluates what it learns in real time. In another word, a process that an agent learns to adjust policies by interacting with the unknown environment. The unknown environment is often formalized as Markov Decision Process (MDP) by a tuple (S, A, P, R, γ). Where S is the (discrete or continuous) state space, A is the (discrete or continuous) action space, r: S × A → R is the (immediate) reward function, P is the state transition model, and γ ∈ (0, 1) is the discount factor that balances the short and long-term returns.[10]

Each reinforcement-learning problem includes the following elements: 1- Agent: learning agent. 2- Environment: This characterizes the exterior world programmatically. Everything the agent(s) interacts with is a portion of the environment. 3-Action/s: Agent can take action/s. 4- A reward function: By specifying the reward function, the goal can be defined as a reinforcement-learning problem. An RL agent's ultimate goal is maximizing the total reward it receives in the long run. 5- Policy: A policy determines the learning agent's actions/behavior at a given time. 6- Model: In general, in a model-based algorithm, the agent can potentially predict the dynamics of the environment, because it has an estimate of the transition function (and reward function). [10]

At each time step, the agent performs a mapping from states to probabilities of selecting each possible action. This mapping is called the agent's policy and is shown by π_t, and $\pi_t(a|s)$ is the probability that $A_t = a$ if $S_t = s$.

Nearly all reinforcement-learning algorithms include estimating value functions that estimate how good it is for the agent in a given state.

Based on [11] Studies on reinforcement learning can be divided into three categories: value-based RL, policy-based RL, and actor-critic RL approaches. They are also grouped into off-policy and on-policy approaches.

Q-learning is a model-free, off-policy, and value-based algorithm since it updates the Q values without making any assumptions about the current policy being followed. Rather, the Q-learning algorithm simply states that the Q-value corresponding to a state S_t and action a_t is updated using the Q-value of the next state $S_{t+1}$and the action a_{t+1} that maximizes the Q-value at that state S_{t+1}. Deep Q-learning (DQN) uses a neural network to estimate the Q-value function. In DQN the action space is still discrete but generally, Q-learning (with function approximation) fails on many simple problems and is poorly understood.

$\pi_\theta(s)$ Can be defined as a stochastic policy that assigns probabilities to actions.

Policy Gradient (PG) methods compute the gradient of J($\pi\theta$) and then use gradient descent to update the policy. [10]

$$J(\theta) = \mathbb{E}_{s\sim\rho_\mu}[R(s, \mu_\theta(s))] \quad (1)$$

The Q-value of an action is the expectation of the reward by choosing that action and for the continuous situation; the gradient is equal to the gradient of the Q-values.

$$Q^*(s,a) = \mathbb{E}_\pi[R(s,a)] \quad (2)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho_\mu}[\nabla_\theta Q^\mu(s,a)|_{a=\mu_\theta}(s)] \quad (3)$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho_\mu}[\nabla_{\theta\mu_\theta}(s) \times \nabla_a Q^\mu(s,a)|_{a=\mu_\theta}(s)] \quad (4)$$

Silver et al. [12] Used a function approximation $Q_\varphi(s,a)$ for estimating the Q-value of any action and computing its gradient by minimizing the quadratic error with the true Q-values.

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho_\mu}[\nabla_{\theta\mu_\theta}(s) \times \nabla_a Q_\varphi(s,a)|_{a=\mu_\theta}(s)] \quad (5)$$

$$J(\varphi) = \mathbb{E}_{s\sim\rho_\mu}[Q^\mu(s,\mu_\theta(s)) - Q_\varphi(s,\mu_\theta(s)))^2] \quad (6)$$

Lillicrap et al. (2015) [7] proposed a Deep Deterministic Policy Gradient (DDPG) that is an extension of the Deterministic Policy Gradient approach to work with non-linear function approximators. In fact, they combined ideas from DQN and DPG to create an algorithm to solve continuous, off-policy, and (DDPG) algorithms. The benefits of DDPG are using experience replay memory to store past transitions and learn off-policy, and using target networks to stabilize learning. In the DDPG algorithm, the target network tracks the trained network much more slowly than the DQN (update parameters states in (7) and $\tau << 1$), and as a result, it creates more stability to learn the Q-values.[13]

$$\theta' = \tau\theta + (1-\tau)\theta' \quad (7)$$

$$J(\varphi) = \mathbb{E}_{s\sim\rho_\mu}[(r(s,a,s') + \gamma Q_{\varphi'}(s', \mu_\theta(s')) - Q_\varphi(s,a))^2] \quad (8)$$

For doing exploration, DDPG uses an additive noise added to the deterministic action to explore the environment.

$$a_t = \mu_\theta(s_t) + \varepsilon \quad (9)$$

In addition to the DDPG method, Also Advantage actor-critic methods could be better than DQN and approximate the advantage of the action.[14]

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(s,a) A_\varphi(s,a)] \quad (10)$$

$A_\varphi(s,a)$ is called the advantage estimate and diverse strategies may be utilized to compute it, including a Monte Carlo advantage estimate (11), Temporal Difference advantage estimate(12), and an n-step advantage estimate (13) . [15]

$$A_\varphi(s,a) = R(s,a) - V_\varphi(s) \quad (11)$$

$$A_\varphi(s,a) = r(s,a,s') + \tau V_\varphi(s') - V_\varphi(s) \quad (12)$$

$$A_\varphi(s,a) = \sum_{k=0}^{n-1} \tau^k r_{t+k+1} + \tau^n V_\varphi(s_{t+n+1}) - V_\varphi(s_t) \quad (13)$$

Advantage actor critic (A2C) has an actor-critic architecture and benefits from an n-step advantage estimate. It creates a batch of transitions (s,a,r,s') by applying policy π_θ and computing the discounted sum of the next n rewards.

$$R_t = \sum_{k=0}^{n-1} \tau^k r_{t+k+1} + \tau^n V_\varphi(s_{t+n+1}) \quad (14)$$

$$\nabla_\theta J(\theta) = \sum_t \nabla_\theta \log \pi_\theta (s_t, a_t) (R_t - V_\varphi (s_t))] \quad (15)$$

$$\mathcal{L}(\varphi) = \sum_t (R_t - V_\varphi(s_t))^2 \quad (16)$$

Concerning the above-mentioned reinforcement learning algorithms and previous research background, DDPG and actor-critic (especially A2C) have better performance in comparison to other reinforcement learning techniques.

II. Methodology

A. *Reinforcement learning formulation for stock trading*

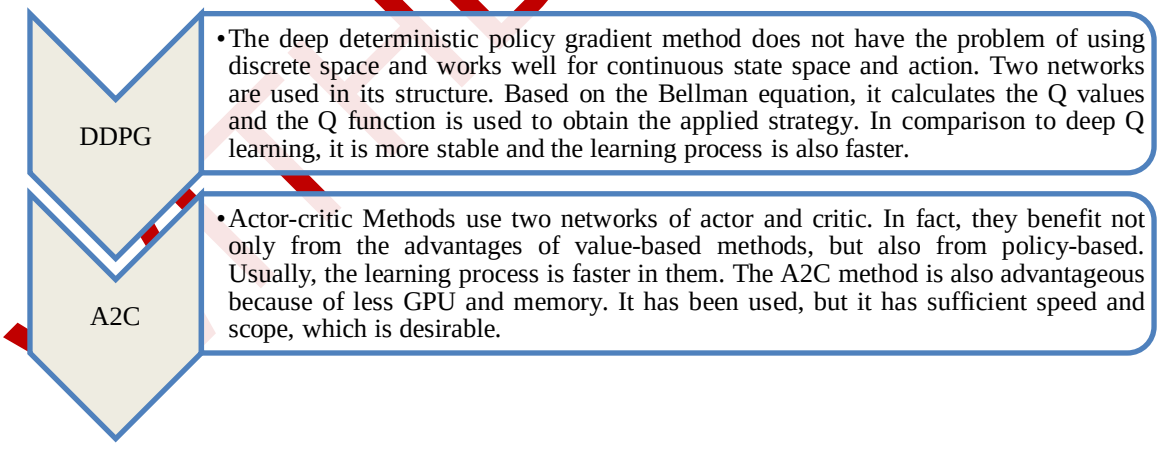
As mentioned before, each reinforcement-learning problem includes the elements [10] which are specified as follows for the proposed stock trading problem:

1) *State s = [open price, close price, high price, low price, adjusted close price, volume, the number of holdings of stocks, remaining balance]; Each state is a vector that describes the current situation: current Balance, market variables (market variables are released from the Tehran stock market exchanges, which include open, close, high, low price, adjusted close and volume), owned shares.*
2) *Action A: a set of actions on all stocks. The available actions for each stock include selling, buying, and holding, which result in decreasing, increasing, and no change of the holdings stocks, respectively.*
3) *Reward r: taking action will produce an immediate effect on the trading agent; profit or loss.*
4) *Policy: the trading strategy of stocks at state s. It is the probability distribution of choosing action at state s.*
5) *Action-value function Q (s, a): the expected reward achieved by action an at state s by applying the proposed policy.*

The goal of the trading agent is to maximize the cumulative profit.

B. *Architecture of Algorithms*

In this research, both DDPG and A2C algorithms that, based on the background, have a better performance than the other algorithms are applied to our proposed model. The architecture and pseudo code for both algorithms are provided as follows.



1) Architectures of Deep Deterministic Policy Gradient (DDPG)

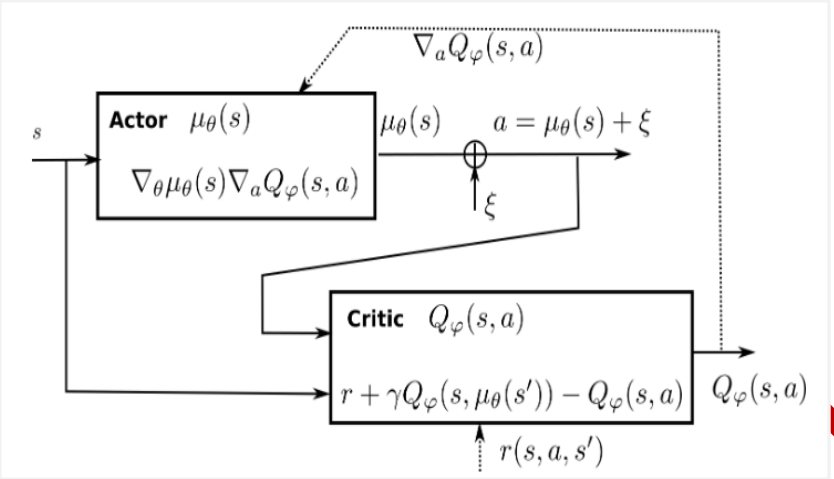


Figure 1. Deep Deterministic Policy Gradient (DDPG) architecture

Algorithm DDPG algorithm

Randomly initialize critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$
Initialize replay buffer R
for episode = 1, M **do**
  Initialize a random process $\mathcal{N}$ for action exploration
  Receive initial observation state s_1
  for t = 1, T **do**
    Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $L = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$
    Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s,a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$$

  end for
end for

Figure 2. Deep Deterministic Policy Gradient (DDPG) Algorithm

2) Architectures of Advantage Actor Critic (A2C)

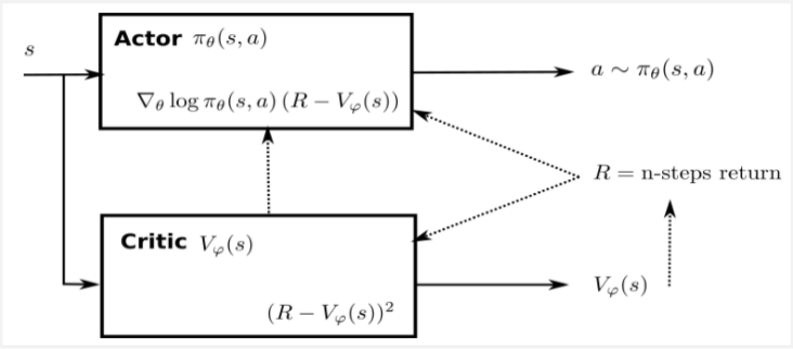


Figure 3. Advantage Actor Critic (A2C) architecture

Algorithm A2C algorithm

- Initialize the actor π_θ and the critic V_φ with random weights.
- Observe the initial state s_0.
- for $t \in [0, T_{\text{total}}]$:
  - Initialize empty episode minibatch.
  - for $k \in [0, n]$: # Sample episode
    - Select a action a_k using the actor π_θ.
    - Perform the action a_k and observe the next state s_{k+1} and the reward r_{k+1}.
    - Store (s_k, a_k, r_{k+1}) in the episode minibatch.
  - if s_n is not terminal: set $R = V_\varphi(s_n)$ with the critic, else $R = 0$.
  - Reset gradient $d\theta$ and $d\varphi$ to 0.
  - for $k \in [n-1, 0]$: # Backwards iteration over the episode
    - Update the discounted sum of rewards $R = r_k + \gamma R$
    - Accumulate the policy gradient using the critic:

$$d\theta \leftarrow d\theta + \nabla_\theta \log \pi_\theta(s_k, a_k)\,(R - V_\varphi(s_k))$$

    - Accumulate the critic gradient:

$$d\varphi \leftarrow d\varphi + \nabla_\varphi (R - V_\varphi(s_k))^2$$

  - Update the actor and the critic with the accumulated gradients using gradient descent or similar:

$$\theta \leftarrow \theta + \eta\, d\theta \qquad \varphi \leftarrow \varphi + \eta\, d\varphi$$

Figure 4. Advantage Actor Critic (A2C) algorithm

C. Performance Evaluations

Evaluation criteria should be used to measure the efficiency of proposed deep reinforcement learning algorithms for stock trading and what the results of such methods will be in practice. The main goal of stock trading is to maximize long-term profit. Usually and the same here, criteria such as annualized return (AR) and sharp ratio (SR) are used to measure and evaluate financial strategies and stock trading performance.

The annualized return is the geometric average amount of money earned by an investment each year over a given time period. [16]

$$\text{sharpe Ratio} = \frac{R_p - R_f}{\sigma_p} \tag{17}$$

$Rp = Return\ of\ portfolio,$

$Rf = Risk\ free\ rate,$

σp = Standard deviation of the portfolio's excess return

D. Stock data

The dataset of three biggest companies of the Tehran stock market is used for evaluating and testing the implementation; Iran Telecommunication Company (MKBT1), Isfahan's Mobarakeh Steel Company (FOLD1), Isfahan Oil Refinery Company (PNES1).

Stock data of the New York Stock Exchange has been obtained from the Yahoo Finance[1] site, and the data of the Tehran Stock Exchange is downloaded from the Tehran Securities Exchange Technology

[1] https://finance.yahoo.com/

Management company[2]. All data is used on a daily basis, and it has six main columns: date, open price, close price, the lowest stock price of the day, the highest stock price of the day, value, volume (the number of shares traded per day), and adjusted close price. In addition, Moving Average Convergence Divergence (MACD) has also been calculated and used. All data is considered from the first of 2009 up to April 2021. The dataset is divided into training, test and trading. Data from 2009 up to 2019 is used for training, and data from 2019 up to 2021 is used as test data.

The implementation has been done by using Python within the TensorFlow [17], OpenAI gym [18], and by using stable baselines [19]. Stable baselines is free and open source and it contains a valuable collection of improved implementations of reinforcement learning and has been used in many types of research, including [8] and [20]. It also has an integrated structure for algorithms, which provides a better comparison for the implementation of different algorithms and their results.

III. Results and Discussion

In the first step, data preprocessing has been done on all datasets. Then, in the training phase, by using data from 2009 up to 2019, a trading agent is generated. The next phase is performed to achieve the best parameters including learning rate, number of episodes, gamma, discount factor, etc. The final phase and actually the test/trading phase using data from 2019 up to 2021 to evaluate the performance of the proposed trading agent. All those steps have been done for both the DDPG trading agent and the A2C trading agent.

As can be seen in Figure 5, the selected stocks are rapidly decreasing. In addition, one of the important reasons to choose these specific stocks is their decreasing procedure. Since clearly by applying trading agents to increasing data stocks, both proposed models work well, but here there is a challenge for proposed algorithms

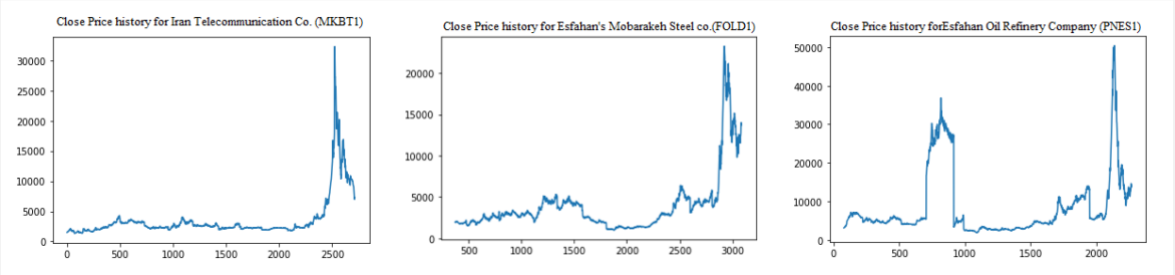


Figure 5. Stock price history as per close price

First, the Deep Deterministic Policy Gradient (DDPG) algorithm is implemented on the stocks. This algorithm is repeated and learned several times. The training, learning, convergence, and error for all the steps are examined and analyzed in Figure 6, Figure 7.

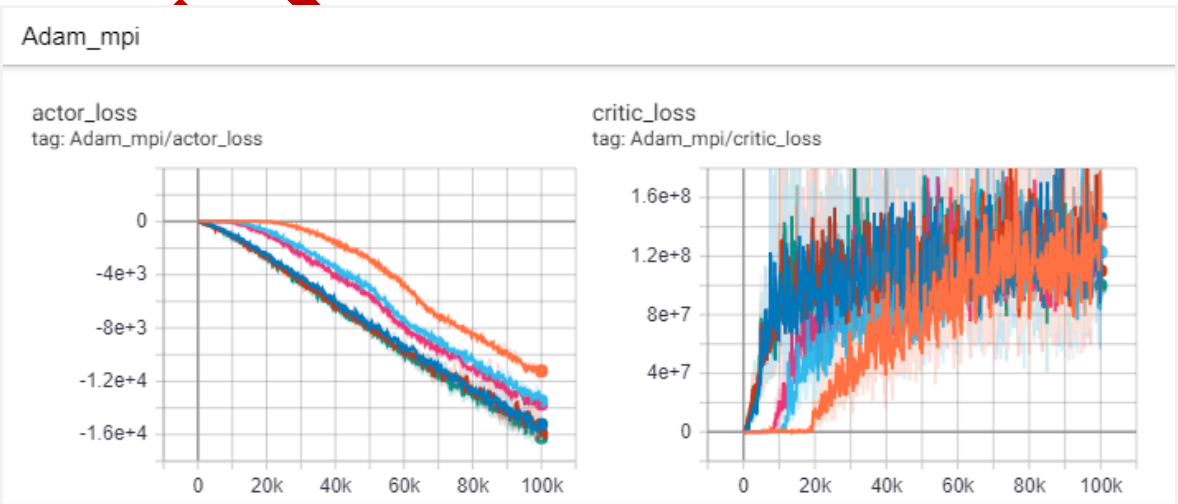


Figure 6. DDPG loss during the learning process in various iterations

[2] http://en.tsetmc.com/

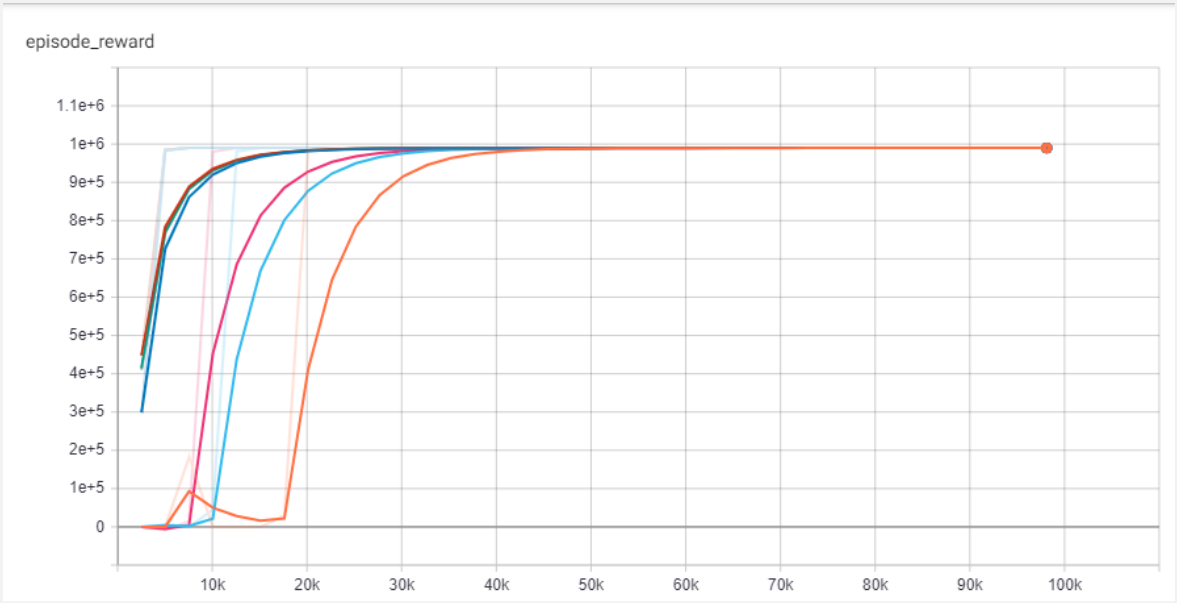


Figure 7. Convergence of the DDPG trading agent in various iterations

It is clear from Figure 7 that convergence of the model is achieved in all iterations, which is almost a similar and integrated process for all of them. In addition, all iterations finally converged to a certain value and the only difference is in the episode of the convergence.

By comparing two Figure 7, Figure 8, it is obvious that, unlike the DDPG trading agent, almost all the iterations had the same behavior and all converged to a specific value, but about the A2C trading agent, different iterations lead to different results. In addition, the average results from the A2C trading agent are lower than the DDPG trading agent.

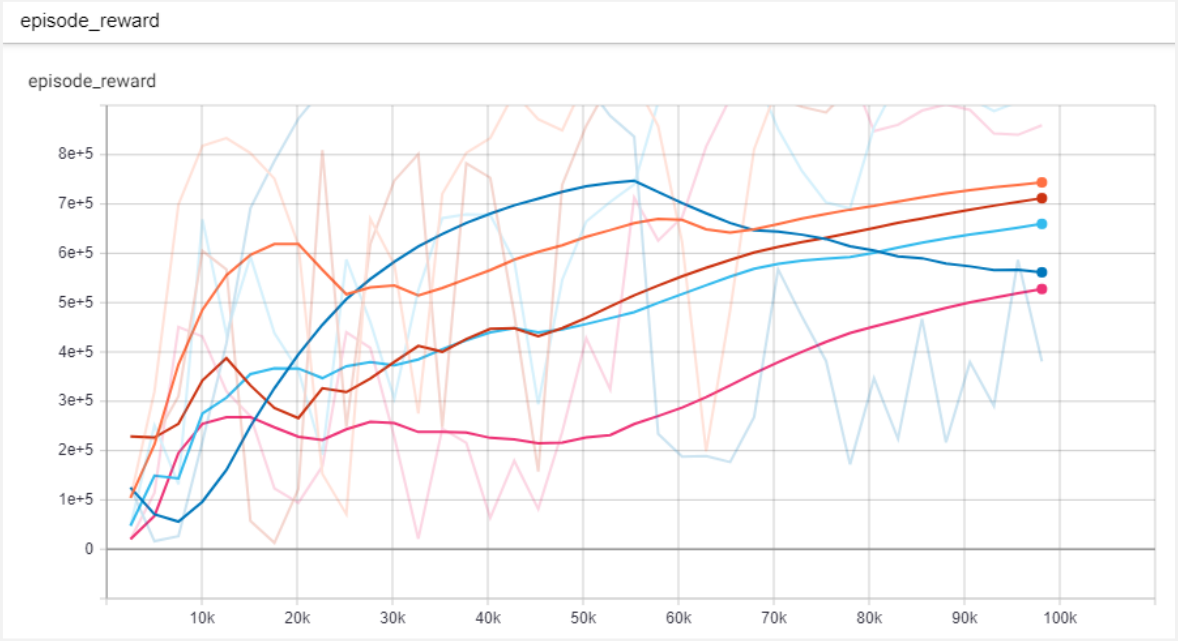


Figure 8. Convergence of the A2C trading agent in various iterations

The results show that the DDPG trading agent outperforms the A2C, concerning the speed of convergence, stability of convergence and also the value of profit. The annualized return and Sharpe ratio are calculated for both proposed algorithms and all stocks, and are shown in Table 1.

Table 1. Results by applying the test dataset (AR=Annualized Return and SR= Sharpe Ratio)

	Esfahan's Mobarakeh Steel Company (FOLD1)		Iran Telecommunication Company (MKBT1)		Esfahan Oil Refinery Company (PNES1)	
	AR	SR	AR	SR	AR	SR
DDPG	85.20	1.8	83.88	1.68	78.53	1.40
A2C	80.40	1.31	82.43	1.66	73.32	1.20

IV. Conclusion

In this research, in addition to theory, the application of deep reinforcement learning algorithms is examined by using Deep Deterministic Policy Gradient (DDPG) agent and Advantage Actor-Critic (A2C) agent for Tehran stock data. The results show that these algorithms can be applied to stock trading and also the DDPG trading agent shows a better performance in regard to convergence, stability, return and also relatively other metrics. These results prove the benefits of DDPG by using experience replay memory to store past transitions and learn off-policy, and using target networks to stabilize learning.

In the future, it will be desirable to study a larger number of stocks and also more considering the choice of a stock among many other stocks

References

[1] Fischer, T.G., 2018. Reinforcement learning in financial markets-a survey (No. 12/2018). FAU Discussion Papers in Economics.

[2] Wang, Y., Wang, D., Zhang, S., Feng, Y., Li, S. and Zhou, Q., 2017. Deep Q-trading. cslt. riit. tsinghua. edu. cn.

[3] Deng, Y., Bao, F., Kong, Y., Ren, Z. and Dai, Q., 2016. Deep direct reinforcement learning for financial signal representation and trading. IEEE transactions on neural networks and learning systems, 28(3), pp.653-664.

[4] Moody, J. and Saffell, M., 2001. Learning to trade via direct reinforcement. IEEE transactions on neural Networks, 12(4), pp.875-889.

[5] Li, Y., Zheng, W. and Zheng, Z., 2019. Deep robust reinforcement learning for practical algorithmic trading. IEEE Access, 7, pp.108014-108022.

[6] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D. and Riedmiller, M., 2014, January. Deterministic policy gradient algorithms. In International conference on machine learning (pp. 387-395). PMLR.

[7] Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[8] Xiong, Z., Liu, X.Y., Zhong, S., Yang, H. and Walid, A., 2018. Practical deep reinforcement learning approach for stock trading. arXiv preprint arXiv:1811.07522.

[9] Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K., 2016, June. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928-1937). PMLR.

[10] Sutton, R.S. and Barto, A.G., 2011. Reinforcement learning: An introduction.

[11] Fischer, T.G., 2018. Reinforcement learning in financial markets-a survey (No. 12/2018). FAU Discussion Papers in Economics.

[12] Silver, D., Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., Dulac-Arnold, G., Reichert, D., Rabinowitz, N., Barreto, A. and Degris, T., 2017, July. The predictron: End-to-end learning and planning. In International Conference on Machine Learning (pp. 3191-3199). PMLR.

[13] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. nature, 518(7540), pp.529-533.

[14] Konda, V. and Tsitsiklis, J., 1999. Actor-critic algorithms. Advances in neural information processing systems, 12.

[15] Vitay, J., 2020. Deep Reinforcement Learning.

[16] Cooper, I., 1996. Arithmetic versus geometric mean estimators: Setting discount rates for capital budgeting. European Financial Management, 2(2), pp.157-167.

[17] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16) (pp. 265-283).

[18] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W., 2016. Openai gym. arXiv preprint arXiv:1606.01540.

[19] https://stable-baselines.readthedocs.io/

[20] Yang, H., Liu, X.Y., Zhong, S. and Walid, A., 2020, October. Deep reinforcement learning for automated stock trading: An ensemble strategy. In Proceedings of the First ACM International Conference on AI in Finance (pp. 1-8).