JNEST EISSN: 2961-8738 • PISSN: 2961-8916 • DOI: 10.56741/jnest.v1i02.171
Article Received 21 October 2022. Revised 01 November 2022.

Deep Reinforcement Learning for Tehran Stock Trading

Neda Yousefi*

***Abstract*—One of the most interesting topics for research, as well as for making a profit, is stock trading. It is known that artificial intelligence has had a great influence 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.**

***Index Terms*—machine learning, deep learning, reinforcement learning, deep deterministic policy gradient (DDPG), Advantage Actor Critic (A2C), stock trading**

I. Introduction

SEVERAL studies have been devoted to using machine learning in the field of financial markets 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 applications. 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 (DRL) 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 DRL for stock trading. DRL stock trading studies can be categorized under three classes: 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 have been applied to build a stock trading system. Wang, Y., et al. [2] Proposed employing 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]. Moreover, there is no single good paradigm for the trading problem, because the trading environment is too complex to be approximated in discrete space. Besides, the proposed techniques are not good for dynamic online trading applications [4]. In policy-based (actor-only) DRL approaches, the algorithm learns a policy, which directly maps states to actions. These methods learned the policy directly from the continuous data, and are considered a better framework for trading applications compared to the Q-learning approaches [3]. Moody, J., et al. [4] found that systems using direct reinforcement learning (policy based) produce better trading strategies than systems utilizing Q-learning (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, then learn the best from both approaches. Zheng, Z., et al. [5] proposed the extended value-based deep Q-network (DQN) and the asynchronous advantage actor-critic (A3C) model for better adapting to the trading market. Their results show that A3C-extended outperforms other models. In addition, the interesting part of the result is that a simple A3C achieved close results to DQN-extended. It shows 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. DDPG learns the policy by applying the Q-function [6] [7]. Xiong, Z., et al. [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 applications. According to the former studies, these two algorithms produce better results in comparison to the other methods. We built an environment and defined action space, state space, and reward function specifically for the stock 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.

The author is with the Faculty of Mathematics, Statistics, and Computer Science, Allameh Tabataba'i University, Tehran, Iran. (Corresponding Author's email: neda.yousefii@gmail.com).

The rest of this paper is organized as follows. Section II covers preliminaries and background followed by some required definitions and algorithms. Section III contains the research methodology, including formulation of the proposed stock trading problem, environment definition, the architecture of both DDPG and A2C algorithms, performance evaluation, and also the explanation of the stock data in this research. Section IV describes the stock data preprocessing and our experimental setup, followed by the performance evaluation of the proposed strategies. Section V concludes this study.

II. Preliminary Studies and Background

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 other words, 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, $r : S \times A \to R$ is the (immediate) reward function, P is the state transition model, and $\gamma \in (0, 1)$ 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 environment characterizes the exterior world in a programmatic manner. 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 is 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.

Let us define $\pi_\theta(s)$ 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].

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

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.

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

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

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

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.

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

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

Lillicrap, et al. [7] proposed a Deep Deterministic Policy Gradient (DDPG) that is an extension of the Deterministic Policy Gradient (DPG) 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 problems. The DDPG benefits from 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 in learning the Q-values [13].

$$\theta' = \tau\theta + (1 - \tau)\theta', \tag{7}$$

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

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

$$a_t = \mu_\theta(s_t) + \varepsilon. \tag{9}$$

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

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

where $A_\varphi(s, a)$ is the advantage estimate value. To compute $A_\varphi(s, a)$, diverse strategies can be utilized, such as the Monte Carlo advantage estimate (11), Temporal

Difference advantage estimate(12), and n-step advantage estimate (13). [15]

$$A_{\varphi}(s,a) = R(s,a) - V_{\varphi}(s); \tag{11}$$

$$A_{\varphi}(s,a) = r(s,a,s') + \tau V_{\varphi}(s') - V_{\varphi}(s); \tag{12}$$

$$\begin{aligned} A_{\varphi}(s,a) = &\sum_{k=0}^{n-1} \tau^k r_{t+k+1} + \tau^n V_{\varphi}(s_{t+n+1}) \\ &- V_{\varphi}(s_t). \end{aligned} \tag{13}$$

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.

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

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

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

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.

III. Research Methodology

A. Reinforcement Learning Formulation for Stock Trading

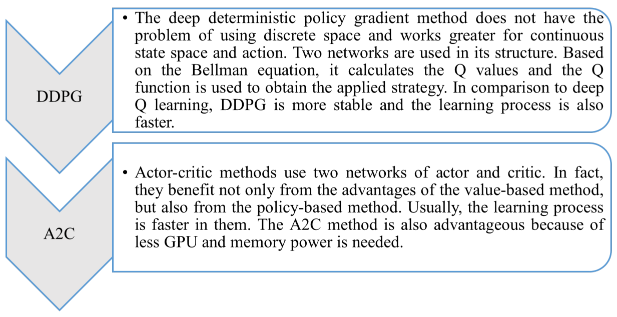
As mentioned before, each reinforcement-learning problem includes the elements [10], which are specified as follows for the proposed stock trading problem:

- 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.
- 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.
- Reward r: taking action will produce an immediate effect on the trading agent; profit or loss.
- Policy: the trading strategy of stocks at state s. It is the probability distribution of choosing an action at state s.
- Action-value function $Q(s, a)$: the expected reward achieved by action a at state s by applying the proposed policy.

At this point, 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 previous studies, have a better performance than the other algorithms for stock trading, are applied to our proposed model. Fig. 1 displays the brief comparison of the DDPG and A2C algorithms. The architecture of the DDPG and A2C algorithms are, respectively, presented in Fig. 2 and Fig. 3. Then, Algorithm 1 and Algorithm 2, depict the pseudo-code for DDPG and A2C algorithms, respectively.


DDPG

- The deep deterministic policy gradient method does not have the problem of using discrete space and works greater for continuous state space and action. Two networks are used in its structure. Based on the Bellman equation, it calculates the Q values and the Q function is used to obtain the applied strategy. In comparison to deep Q learning, DDPG is more stable and the learning process is also faster.

A2C

- Actor-critic methods use two networks of actor and critic. In fact, they benefit not only from the advantages of the value-based method, but also from the policy-based method. Usually, the learning process is faster in them. The A2C method is also advantageous because of less GPU and memory power is needed.

Fig. 1. Brief comparison of DDPG and A2C algorithm.

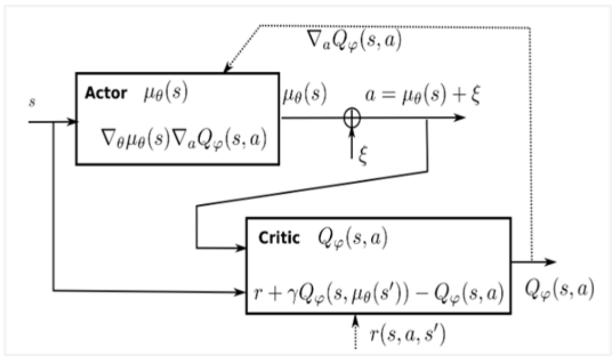

$\nabla_a Q_\varphi(s,a)$
s
Actor $\mu_\theta(s)$
$\nabla_\theta \mu_\theta(s) \nabla_a Q_\varphi(s,a)$
$\mu_\theta(s)$
$a = \mu_\theta(s) + \xi$
ξ
Critic $Q_\varphi(s,a)$
$r + \gamma Q_\varphi(s, \mu_\theta(s')) - Q_\varphi(s,a)$
$Q_\varphi(s,a)$
$r(s,a,s')$

Fig. 2. Deep Deterministic Policy Gradient (DDPG) architecture.

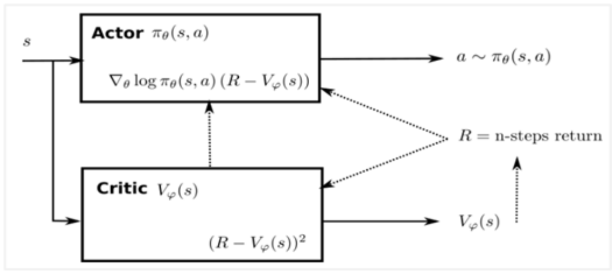


Fig. 3. Advantage Actor Critic (A2C) architecture.

C. Performance Evaluations

Evaluation criteria are 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

Algorithm 1 Deep Deterministic Policy Gradient (DDPG) Algorithm [15].

```
 1: **Randomly initialize** critic network $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$.
 2: **Initialize** target network $Q'$ and $\mu'$ with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$.
 3: **Initialize** replay buffer $R$.
 4: **For** episode=1, $M$ do
 5:     **Initialize** a random process $\nu$ for action exploration.
 6:     **Receive** initial observation state $s_1$
 7:     **For** $t = 1$, $T$ **do**
 8:         Select action $a_t = \mu\left(s_t|\theta^\mu\right) + \nu_t$ according to the current policy and exploration noise.
 9:         Execute action $a_t$ and observe reward $r_t$ and observe new state $s_{t+1}$.
10:         Store transition $(s_t, a_t, r_t, s_{t+1})$ in $R$.
11:         Sample a random minibatch of $N$ transitions $(s_i, a_i, r_i, s_{i+1})$ from $R$.
12:         Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'\left(s_{i+1}|\theta^{\mu'}\right)\Big|\,\theta^{Q'})$.
13:         Update critic by minimizing the Loss: $L = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2$.
14:         Update the actor policy using the sampled policy gradient:
15:         $\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q\left(s, a|\theta^Q\right)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)|_{s_i}$
16:         Update the target networks:
17:         $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\,\theta^{Q'}; \theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\,\theta^{\mu'}$
18:     **End** for
19: **End** for
```

Algorithm 2 Advantage Actor Critic (A2C) Algorithm [15].

```
1: **Randomly initialize** critic network $V_\varphi$ and actor $\pi_\theta$ with weights.
2: **Acquire** a batch of transitions $(s, a, r, s')$ using the actor $\pi_\theta$.
3: **For** each state encountered, compute the discounted sum of the next $n$ rewards $\sum_{k=0}^{n}\gamma^k r_{t+k+1}$ and use the critic to
   estimate the value of the state encountered $n$ steps later $V_\varphi\left(s_{t+n+1}\right)$
   $R_t = \sum_{k=0}^{n-1}\gamma^k r_{t+k+1} + \gamma^n V_\varphi(s_{t+n+1})$.
4: **Update** the actor by using:
       $\nabla_\theta J\left(\theta\right) = \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}[\nabla_\theta \log\pi_\theta\left(s_t, a_t\right) A_\varphi\left(s_t, a_t\right)]$
       $A_\varphi\left(s, a\right) = \sum_{k=0}^{n-1}\gamma^k r_{t+k+1} + \gamma^n V_\varphi\left(s_{t+n+1}\right) - V_\varphi(s_t)$
       $A_\varphi\left(s, a\right) = \sum_{k=0}^{n-1}\gamma^k r_{t+k+1} + \gamma^n V_\varphi\left(s_{t+n+1}\right) - V_\varphi(s_t)$
       $\nabla_\theta J\left(\theta\right) = \sum_t \nabla_\theta \log\pi_\theta\left(s_t, a_t\right)\left(R_t - V_\varphi\left(s_t\right)\right)$
5: **Update** the critic by using:
       $L\left(\varphi\right) = \sum_t (R_t - V_\varphi(s_t))^2$
6: **Repeat**
```

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

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}. \tag{17}$$

where R_p is the return of the portfolio, R_f is the risk-free rate, and σ_p is the standard deviation of the portfolio's excess return.

D. Stock Data

The dataset of the three biggest companies of the Tehran stock market is used for evaluating and testing the proposed model: Iran Telecommunication Company (MKBT1), Isfahan's Mobarakeh Steel Company (FOLD1), and Isfahan Oil Refinery Company (PNES1). Stock data from the New York Stock Exchange has been obtained from the Yahoo Finance Site, and the data of the Tehran Stock Exchange is downloaded from the Tehran Securities Exchange Technology Management Company. All data is used on a daily basis. The data 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 used. All data is considered from the first month of 2009 up to April 2021. The dataset is divided into training, testing, and trading. The data from 2009 up to 2019 is used for training, whereas the data from 2019 up to 2021 is used as test data.

In the preprocessing steps, we perform data cleaning and data transformation for the stock data. In the data cleaning phase, eliminating and/or filling in missing data and null data for all different stocks has been done. In the data transformation phase, different stock data have been normalized.

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

IV. Results and Discussion

In the first step, data preprocessing was done on all datasets. Then, in the training phase, by using data between

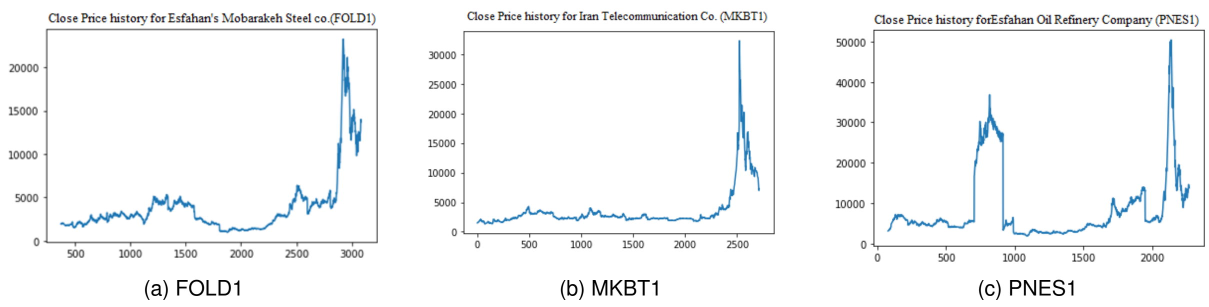


Fig. 4. Stock price history as per close price.

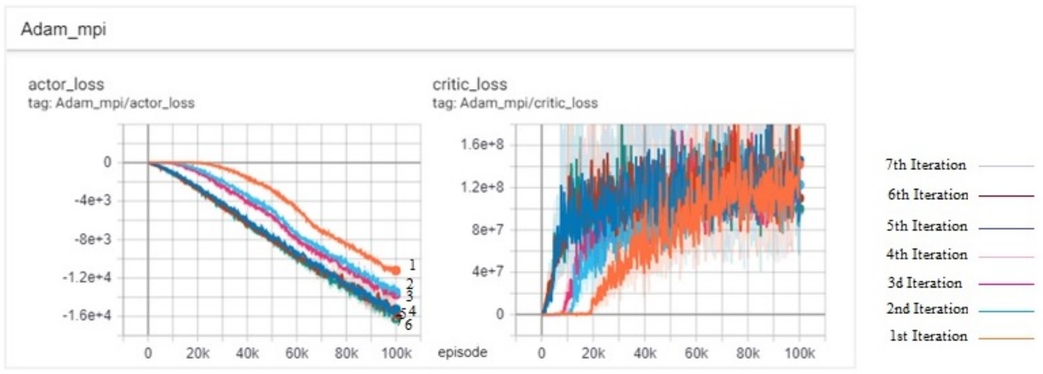


Fig. 5. DDPG loss during the learning process in various iterations.

2009 and 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 uses data from 2019 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 Fig. 4, we can see a rapid decrease in the selected stocks trends. One of the important reasons to choose these specific stocks is their decreasing procedure. Since clearly by applying trading agents to the increasing data stocks, both proposed models will work well. However, decreasing data stocks induce a different level of challenge.

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 (see Figs. 6-7).

It is clear from Fig. 6 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 figures (i.e., Fig. 6 and Fig. 7), it is obvious that, unlike the DDPG trading agent where almost all the iterations had the same behavior and all converged to a specific value, with 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.

The results show that the DDPG trading agent outperforms the A2C performance in terms of 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 I.

TABLE I
RESULTS OBTAINED FROM 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	1.84	83.88	1.68	78.53	1.40
A2C	80.40	1.31	82.43	1.66	73.32	1.20

V. CONCLUSION

In this research, in addition to theory, the application of the deep reinforcement learning algorithms is examined by using the Deep Deterministic Policy Gradient (DDPG) agent and Advantage Actor Critic (A2C) agent for the Tehran stock data. The results show that these algorithms can be applied to stock trading. In addition, it is revealed that 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 the DDPG algorithm that uses the experience replay memory to store past transitions and learn off-policy and uses target networks to stabilize learning and convergence. Moreover, an additive noise added to deterministic action creates more exploration and provides a better return for the DDPG agent.

In the future, it will be desirable to study a larger number of stocks and also consider the choice of stock among many other stocks. It is also desirable to explore

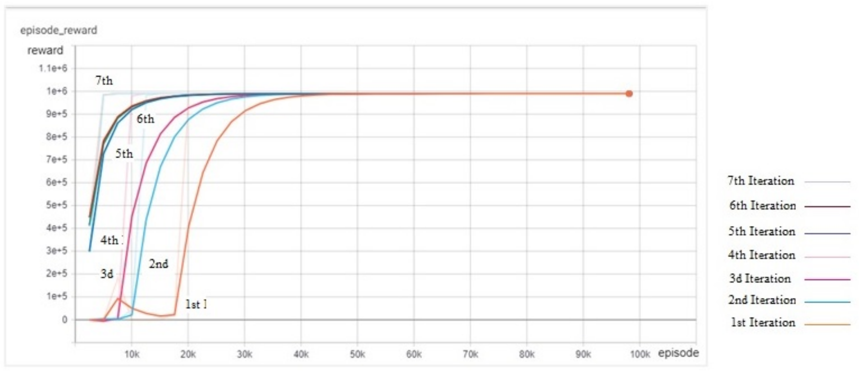


Fig. 6. Convergence of the DDPG trading agent in various iterations.

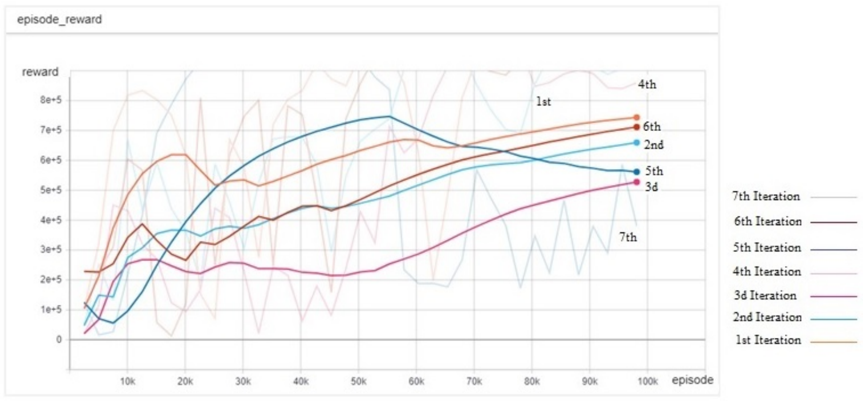


Fig. 7. Convergence of the A2C trading agent in various iterations.

the possible application of proposed frameworks to other real-world scenarios in a real-time manner.

REFERENCES

[1] T. G. Fischer, "Reinforcement learning in financial markets-a survey," FAU Discussion Papers in Economics, Tech. Rep., 2018.
[2] Y. Wang, D. Wang, S. Zhang, Y. Feng, S. Li, and Q. Zhou, "Deep Q-trading," *cslt. riit. tsinghua. edu. cn*, 2017.
[3] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, 2016.
[4] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 875–889, 2001.
[5] Y. Li, W. Zheng, and Z. Zheng, "Deep robust reinforcement learning for practical algorithmic trading," *IEEE Access*, vol. 7, pp. 108 014–108 022, 2019.
[6] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International Conference on Machine Learning*. PMLR, 2014, pp. 387–395.
[7] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.
[8] X.-Y. Liu, Z. Xiong, S. Zhong, H. Yang, and A. Walid, "Practical deep reinforcement learning approach for stock trading," *arXiv preprint arXiv:1811.07522*, 2018.
[9] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1928–1937.
[10] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction. 2018," *Google Scholar Digital Library*, 2011.
[11] T. G. Fischer, "Reinforcement learning in financial markets-a survey," FAU Discussion Papers in Economics, Tech. Rep., 2018.
[12] D. Silver, H. Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto *et al.*, "The predictron: End-to-end learning and planning," in *International Conference on Machine Learning*. PMLR, 2017, pp. 3191–3199.
[13] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[14] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in Neural Information Processing Systems*, vol. 12, 1999.
[15] J. Vitay, "Deep reinforcement learning," 2020.
[16] I. Cooper, "Arithmetic versus geometric mean estimators: Setting discount rates for capital budgeting," *European Financial Management*, vol. 2, no. 2, pp. 157–167, 1996.
[17] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "{TensorFlow}: a system for {Large-Scale} machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.
[18] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
[19] "Welcome to stable baselines docs! - RL baselines made easy," https://stable-baselines.readthedocs.io/, accessed: 2022-11-15.
[20] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, "Deep reinforcement learning for automated stock trading: An ensemble strategy," in *Proceedings of the First ACM International Conference on AI in Finance*, 2020, pp. 1–8.

Neda Yousefi received her bachelor's degree in Applied Mathematics and her first master's degree in Industrial Engineering (Economic and Social Systems Engineering) from the Amirkabir University of Technology, Iran (Tehran Polytechnic). In 2021, she received her second master's degree in Computer Science (Soft Computing and Artificial intelligence) from Allameh Tabataba'i University Tehran, Iran. Her research interests are in Machine Learning, Deep Learning, Computer Vision, and Image Processing.