Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy

Hongyang Yang[1], Xiao-Yang Liu[2], Shan Zhong[2], and Anwar Walid[3]
[1]Dept. of Statistics, Columbia University
[2]Dept. of Electrical Engineering, Columbia University
[3]Mathematics of Systems Research Department, Nokia-Bell Labs
Email: {HY2500, XL2427, SZ2495}@columbia.edu,
anwar.walid@nokia-bell-labs.com

***Abstract*—Stock trading strategies play a critical role in investment. However, it is challenging to design a profitable strategy in a complex and dynamic stock market. In this paper, we propose an ensemble strategy that employs deep reinforcement schemes to learn a stock trading strategy by maximizing investment return. We train a deep reinforcement learning agent and obtain an ensemble trading strategy using three actor-critic based algorithms: Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Deterministic Policy Gradient (DDPG). The ensemble strategy inherits and integrates the best features of the three algorithms, thereby robustly adjusting to different market situations. In order to avoid the large memory consumption in training networks with continuous action space, we employ a load-on-demand technique for processing very large data. We test our algorithms on the 30 Dow Jones stocks that have adequate liquidity. The performance of the trading agent with different reinforcement learning algorithms is evaluated and compared with both the Dow Jones Industrial Average index and the traditional min-variance portfolio allocation strategy. The proposed deep ensemble strategy is shown to outperform the three individual algorithms and two baselines in terms of the risk-adjusted return measured by the Sharpe ratio.**

***Index Terms*—Deep reinforcement learning, Markov Decision Process, automated stock trading, ensemble strategy, actor-critic framework**

I. Introduction

Profitable automated stock trading strategy is vital to investment companies and hedge funds. It is applied to optimize capital allocation and maximize investment performance, such as expected return. Return maximization can be based on the estimates of potential return and risk. However, it is challenging for analysts to consider all relevant factors in a complex and dynamic stock market [1], [2], [3].

Existing works are not satisfactory. A traditional approach that employed two steps was described in [4]. First, the expected stock return and the covariance matrix of stock prices are computed. Then, the best portfolio allocation strategy can be obtained by either maximizing the return for a given risk ratio or minimizing the risk for a pre-specified return. This approach, however, is complex and costly to implement since the portfolio managers may want to revise the decisions at each time step, and take other factors into account, such as transaction cost. Another approach for stock trading is to model it as a Markov Decision Process (MDP) and use dynamic programming to derive the optimal strategy [5], [6], [7], [8]. However, the scalability of this model is limited due to the large state spaces when dealing with the stock market.

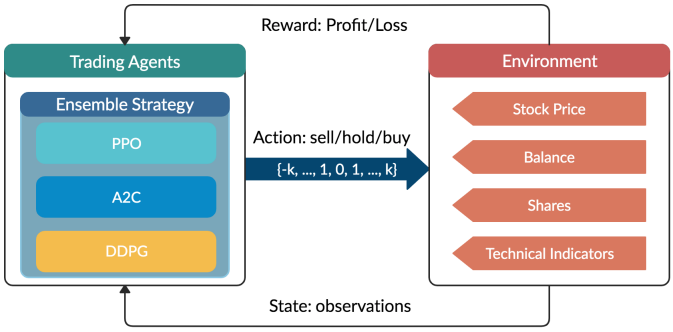

Fig. 1. Overview of reinforcement learning-based stock trading strategy.

In recent years, machine learning and deep learning algorithms have been widely applied to build prediction and classification models for the financial market. Fundamentals data (earnings report) and alternative data (market news, academic graph data, credit card transactions, and GPS traffic, etc.) are combined with machine learning algorithms to extract new investment alphas or predict a company's future performance [9], [10], [11], [12]. Thus, a predictive alpha signal is generated to perform stock selection. However, these approaches are only focused on picking high performance stocks rather than allocating trade positions or shares between the selected stocks. In other words, the machine learning models are not trained to model positions.

In this paper, we propose a novel ensemble strategy that combines three deep reinforcement learning algorithms and finds the optimal trading strategy in a complex and dynamic stock market. The three actor-critic algorithms [13] are Proximal Policy Optimization (PPO) [14], [15], Advantage Actor Critic (A2C) [16], [17], and Deep Deterministic Policy Gradient (DDPG) [18], [15], [19]. Our deep reinforcement learning approach is described in Figure 1. By applying the ensemble strategy, we make the trading strategy more robust and reliable. Our strategy can adjust to different market situations and maximize return subject

to risk constraint. First, we build an environment and define action space, state space, and reward function. Second, we train the three algorithms that take actions in the environment. Third, we ensemble the three agents together using the Sharpe ratio that measures the risk-adjusted return. The effectiveness of the ensemble strategy is verified by its higher Sharpe ratio than both the min-variance portfolio allocation strategy and the Dow Jones Industrial Average [1] (DJIA).

The remainder of this paper is organized as follows. Section 2 introduces related works. Section 3 provides a description of our stock trading problem. In Section 4, we set up our stock trading environment. In Section 5, we drive and specify the three actor-critic based algorithms and our ensemble strategy. Section 6 describes the stock data preprocessing and our experimental setup, and presents the performance evaluation of the proposed ensemble strategy. We conclude this paper in Section 7.

II. Related Works

Recent applications of deep reinforcement learning in financial markets consider discrete or continuous state and action spaces, and employ one of these learning approaches: critic-only approach, actor-only approach, or actor-critic approach [20]. Learning models with continuous action space provide finer control capabilities than those with discrete action space.

The critic-only learning approach, which is the most common, solves a discrete action space problem using, for example, Deep Q-learning (DQN) and its improvements, and trains an agent on a single stock or asset [21], [22], [23]. The idea of the critic-only approach is to use a Q-value function to learn the optimal action-selection policy that maximizes the expected future reward given the current state. Instead of calculating a state-action value table, DQN minimizes the error between estimated Q-value and target Q-value over a transition, and uses a neural network to perform function approximation. The major limitation of the critic-only approach is that it only works with discrete and finite state and action spaces, which is not practical for a large portfolio of stocks, since the prices are of course continuous.

The actor-only approach has been used in [24], [25], [26]. The idea here is that the agent directly learns the optimal policy itself. Instead of having a neural network to learn the Q-value, the neural network learns the policy. The policy is a probability distribution that is essentially a strategy for a given state, namely the likelihood to take an allowed action. Recurrent reinforcement learning is introduced to avoid the curse of dimensionality and improves trading efficiency in [24]. The actor-only approach can handle the continuous action space environments.

The actor-critic approach has been recently applied in finance [27], [28], [17], [19]. The idea is to simultaneously update the actor network that represents the policy, and the critic network that represents the value function. The critic estimates the value function, while the actor updates the policy probability distribution guided by the critic with policy gradients. Over time, the actor learns to take better actions and the critic gets better at evaluating those actions. The actor-critic approach has proven to be able to learn and adapt to large and complex environments, and has been used to play popular video games, such as Doom [29]. Thus, the actor-critic approach is promising in trading with a large stock portfolio.

[1]The Dow Jones Industrial Average is a stock market index that shows how 30 large, publicly owned companies based in the United States have traded during a standard trading session in the stock market.

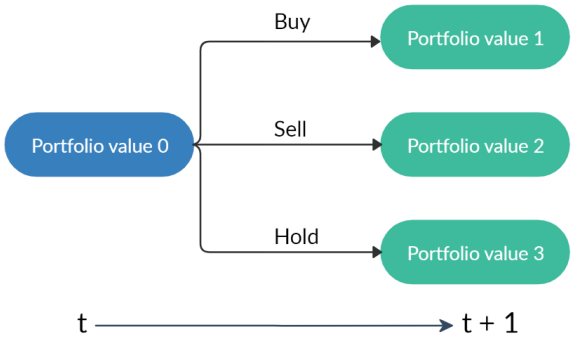

Fig. 2. A starting portfolio value with three actions result in three possible portfolios. Note that "hold" may lead to different portfolio values due to the changing stock prices.

III. Problem Description

We model stock trading as a Markov Decision Process (MDP), and formulate our trading objective as a maximization of expected return [30].

A. MDP Model for Stock Trading

To model the stochastic nature of the dynamic stock market, we employ a Markov Decision Process (MDP) as follows:

- State $\boldsymbol{s} = [\boldsymbol{p}, \boldsymbol{h}, b]$: a vector that includes stock prices $\boldsymbol{p} \in \mathbb{R}_+^D$, the stock shares $\boldsymbol{h} \in \mathbb{Z}_+^D$, and the remaining balance $b \in \mathbb{R}_+$, where D denotes the number of stocks and $\mathbb{Z}_+$ denotes non-negative integers.
- Action $\boldsymbol{a}$: a vector of actions over D stocks. The allowed actions on each stock include *selling*, *buying*, or *holding*, which result in decreasing, increasing, and no change of the stock shares $\boldsymbol{h}$, respectively.
- Reward $r(s, a, s')$: the direct reward of taking action a at state s and arriving at the new state s'.
- Policy $\pi(s)$: the trading strategy at state s, which is the probability distribution of actions at state s.
- Q-value $Q_\pi(s, a)$: the expected reward of taking action a at state s following policy π.

The state transition of a stock trading process is shown in Figure 2. At each state, one of three possible actions is taken on stock d $(d = 1, ..., D)$ in the portfolio.

- Selling $\boldsymbol{k}[d] \in [1, \boldsymbol{h}[d]]$ shares results in $\boldsymbol{h_{t+1}}[d] = \boldsymbol{h_t}[d] - \boldsymbol{k}[d]$, where $\boldsymbol{k}[d] \in \mathbb{Z}_+$ and $d = 1, ..., D$.

- Holding, $\boldsymbol{h_{t+1}}[d] = \boldsymbol{h_t}[d]$.
- Buying $\boldsymbol{k}[d]$ shares results in $\boldsymbol{h_{t+1}}[d] = \boldsymbol{h_t}[d] + \boldsymbol{k}[d]$.

At time t an action is taken and the stock prices update at t+1, accordingly the portfolio values may change from "portfolio value 0" to "portfolio value 1", "portfolio value 2", or "portfolio value 3", respectively, as illustrated in Figure 2. Note that the portfolio value is $\boldsymbol{p^T h} + b$.

B. Incorporating Stock Trading Constraints

The following assumption and constraints reflect concerns for practice: transaction costs, market liquidity, risk-aversion, etc.

- Market liquidity: the orders can be rapidly executed at the close price. We assume that stock market will not be affected by our reinforcement trading agent.
- Nonnegative balance $b \geq 0$: the allowed actions should not result in a negative balance. Based on the action at time t, the stocks are divided into sets for sell $\mathcal{S}$, buying $\mathcal{B}$, and holding $\mathcal{H}$, where $\mathcal{S} \cup \mathcal{B} \cup \mathcal{H} = \{1, \cdots, D\}$ and they are nonoverlapping. Let $\boldsymbol{p}_t^B = [p_t^i : i \in \mathcal{B}]$ and $\boldsymbol{k}_t^B = [k_t^i : i \in \mathcal{B}]$ be the vectors of price and number of buying shares for the stocks in the buying set. We can similarly define $\boldsymbol{p}_t^S$ and $\boldsymbol{k}_t^S$ for the selling stocks, and $\boldsymbol{p}_t^H$ and $\boldsymbol{k}_t^H$ for the holding stocks. Hence, the constraint for non-negative balance can be expressed as

$$b_{t+1} = b_t + (\boldsymbol{p}_t^S)^T \boldsymbol{k}_t^S - (\boldsymbol{p}_t^B)^T \boldsymbol{k}_t^B \geq 0. \quad (1)$$

- Transaction cost: transaction costs are incurred for each trade. There are many types of transaction costs such as exchange fees, execution fees, and SEC fees. Different brokers have different commission fees. Despite these variations in fees, we assume our transaction costs to be 0.1% of the value of each trade (either buy or sell) as in [9]:

$$c_t = \boldsymbol{p}^T \boldsymbol{k}_t \times 0.1\%. \quad (2)$$

- Risk-aversion for market crash: there are sudden events that may cause stock market crash, such as wars, collapse of stock market bubbles, sovereign debt default, and financial crisis. To control the risk in a worst-case scenario like 2008 global financial crisis, we employ the financial turbulence index $turbulence_t$ that measures extreme asset price movements [31]:

$$turbulence_t = (\boldsymbol{y_t} - \boldsymbol{\mu})\, \boldsymbol{\Sigma^{-1}} (\boldsymbol{y_t} - \boldsymbol{\mu})' \in \mathbb{R}, \quad (3)$$

where $\boldsymbol{y_t} \in \mathbb{R}^D$ denotes the stock returns for current period t, $\boldsymbol{\mu} \in \mathbb{R}^D$ denotes the average of historical returns, and $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$ denotes the covariance of historical returns. When $turbulence_t$ is higher than a threshold, which indicates extreme market conditions, we simply halt buying and the trading agent sells all shares. We resume trading once the turbulence index returns under the threshold.

C. Return Maximization as Trading Goal

We define our reward function as the change of the portfolio value when action a is taken at state s and arriving at new state s'. The goal is to design a trading strategy that maximizes the change of the portfolio value:

$$r(s_t, a_t, s_{t+1}) = (b_{t+1} + \boldsymbol{p_{t+1}^T h_{t+1}}) - (b_t + \boldsymbol{p_t^T h_t}) - c_t, \quad (4)$$

where the first and second terms denote the portfolio value at $t+1$ and t, respectively. To further decompose the return, we define the transition of the shares $\boldsymbol{h_t}$ is defined as

$$\boldsymbol{h}_{t+1} = \boldsymbol{h}_t - \boldsymbol{k}_t^S + \boldsymbol{k}_t^B, \quad (5)$$

and the transition of the balance b_t is defined in (1). Then (4) can be rewritten as

$$r(s_t, a_t, s_{t+1}) = r_H - r_S + r_B - c_t, \quad (6)$$

where

$$r_H = (\boldsymbol{p}_{t+1}^H - \boldsymbol{p}_t^H)^T \boldsymbol{h}_t^H, \quad (7)$$

$$r_S = (\boldsymbol{p}_{t+1}^S - \boldsymbol{p}_t^S)^T \boldsymbol{h}_t^S, \quad (8)$$

$$r_B = (\boldsymbol{p}_{t+1}^B - \boldsymbol{p}_t^B)^T \boldsymbol{h}_t^B, \quad (9)$$

where r_H, r_S, and r_B denote the change of the portfolio value comes from holding, selling, and buying shares moving from time t to $t+1$, respectively. Equation (6) indicates that we need to maximize the positive change of the portfolio value by buying and holding the stocks whose price will increase at next time step and minimize the negative change of the portfolio value by selling the stocks whose price will decrease at next time step.

Turbulence index $turbulence_t$ is incorporated with the reward function to address our risk-aversion for market crash. When the index in (3) goes above a threshold, Equation (8) becomes

$$r_{sell} = (\boldsymbol{p_{t+1}} - \boldsymbol{p_t})^T \boldsymbol{k_t}, \quad (10)$$

which indicates that we want to minimize the negative change of the portfolio value by selling all held stocks, because all stock prices will fall.

The model is initialized as follows. p_0 is set to the stock prices at time 0 and b_0 is the amount of initial fund. The h and $Q_\pi(s, a)$ are 0, and $\pi(s)$ is uniformly distributed among all actions for each state. Then, $Q_\pi(s_t, a_t)$ is updated through interacting with the stock market environment. The optimal strategy is given by the Bellman Equation, such that the expected reward of taking action a_t at state s_t is the expectation of the summation of the direct reward $r(s_t, a_t, s_{t+1})$ and the future reward in the next state s_{t+1}. Let the future rewards be discounted by a factor of $0 < \gamma < 1$ for convergence purpose, then we have

$$Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}}[r(s_t, a_t, s_{t+1}) + \gamma \mathbb{E}_{a_{t+1} \sim \pi(s_{t+1})}[Q_\pi(s_{t+1}, a_{t+1})]]. \quad (11)$$

The goal is to design a trading strategy that maximizes the positive cumulative change of the portfolio

value $r(s_t, a_t, s_{t+1})$ in the dynamic environment, and we employ the deep reinforcement learning method to solve this problem.

IV. Stock Market Environment

Before training a deep reinforcement trading agent, we carefully build the environment to simulate real world trading which allows the agent to perform interaction and learning. In practical trading, various information needs to be taken into account, for example the historical stock prices, current holding shares, technical indicators, etc. Our trading agent needs to obtain such information through the environment, and take actions defined in the previous section. We employ OpenAI gym to implement our environment and train the agent [32], [33], [34].

A. Environment for Multiple Stocks

We use a continuous action space to model the trading of multiple stocks. We assume that our portfolio has 30 stocks in total.

1) State Space: We use a 181-dimensional vector consists of seven parts of information to represent the state space of multiple stocks trading environment: $[b_t, \boldsymbol{p}_t, \boldsymbol{h}_t, \boldsymbol{M}_t, \boldsymbol{R_t}, \boldsymbol{C_t}, \boldsymbol{X_t}]$. Each component is defined as follows:

- $b_t \in \mathbb{R}_+$: available balance at current time step t.
- $\boldsymbol{p}_t \in \mathbb{R}_+^{30}$: adjusted close price of each stock.
- $\boldsymbol{h}_t \in \mathbb{Z}_+^{30}$: shares owned of each stock.
- $\boldsymbol{M}_t \in \mathbb{R}^{30}$: Moving Average Convergence Divergence (MACD) is calculated using close price. MACD is one of the most commonly used momentum indicator that identifies moving averages [35].
- $\boldsymbol{R}_t \in \mathbb{R}_+^{30}$: Relative Strength Index (RSI) is calculated using close price. RSI quantifies the extent of recent price changes. If price moves around the support line, it indicates the stock is oversold, and we can perform the buy action. If price moves around the resistance, it indicates the stock is overbought, and we can perform the selling action. [35].
- $\boldsymbol{C_t} \in \mathbb{R}_+^{30}$: Commodity Channel Index (CCI) is calculated using high, low and close price. CCI compares current price to average price over a time window to indicate a buying or selling action [36].
- $\boldsymbol{X_t} \in \mathbb{R}^{30}$: Average Directional Index (ADX) is calculated using high, low and close price. ADX identifies trend strength by quantifying the amount of price movement [37].

2) Action Space: For a single stock, the action space is defined as $\{-k, ..., -1, 0, 1, ..., k\}$, where k and $-k$ presents the number of shares we can buy and sell, and $k \leq h_{max}$ while h_{max} is a predefined parameter that sets as the maximum amount of shares for each buying action. Therefore the size of the entire action space is $(2k+1)^{30}$. The action space is then normalized to $[-1, 1]$, since the RL algorithms A2C and PPO define the policy directly on a Gaussian distribution, which needs to be normalized and symmetric [34].

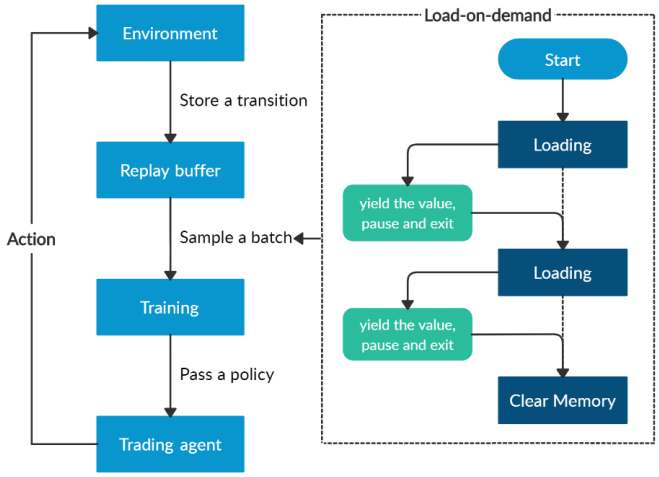

Fig. 3. Overview of the load-on-demand technique.

B. Memory Management

The memory consumption for training could grow exponentially with the number of stocks, data types, features of the state space, number of layers and neurons in the neural networks, and batch size. To tackle the problem of memory requirements, we employ a load-on-demand technique for efficient use of memory. As shown in Figure 3, the load-on-demand technique does not store all results in memory, rather, it generates them on demand. The memory is only used when the result is requested, hence the memory usage is reduced.

V. Trading Agent based on Deep Reinforcement Learning

We use three actor-critic based algorithms to implement our trading agent. The three algorithms are A2C, DDPG, and PPO, respectively. An ensemble strategy is proposed to combine the three agents together to build a robust trading strategy.

A. Advantage Actor Critic (A2C)

A2C [16] is a typical actor-critic algorithm and we use it a component in the ensemble strategy. A2C is introduced to improve the policy gradient updates. A2C utilizes an advantage function to reduce the variance of the policy gradient. Instead of only estimates the value function, the critic network estimates the advantage function. Thus, the evaluation of an action not only depends on how good the action is, but also considers how much better it can be. So that it reduces the high variance of the policy network and makes the model more robust.

A2C uses copies of the same agent to update gradients with different data samples. Each agent works independently to interact with the same environment. In each iteration, after all agents finish calculating their gradients, A2C uses a coordinator to pass the average gradients over all the agents to a global network. So that the global network can update the actor and the critic network. The

presence of a global network increases the diversity of training data. The synchronized gradient update is more cost-effective, faster and works better with large batch sizes. A2C is a great model for stock trading because of its stability.

The objective function for A2C is:

$$\nabla J_\theta(\theta) = \mathbb{E}[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t)], \tag{12}$$

where $\pi_\theta(a_t|s_t)$ is the policy network, $A(s_t, a_t)$ is the Advantage function can be written as:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t), \tag{13}$$

or

$$A(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma V(s_{t+1}) - V(s_t). \tag{14}$$

B. Deep Deterministic Policy Gradient (DDPG)

DDPG [18] is used to encourage maximum investment return. DDPG combines the frameworks of both Q-learning [38] and policy gradient [39], and uses neural networks as function approximators. In contrast with DQN that learns indirectly through Q-values tables and suffers the curse of dimensionality problem [40], DDPG learns directly from the observations through policy gradient. It is proposed to deterministically map states to actions to better fit the continuous action space environment.

At each time step, the DDPG agent performs an action a_t at s_t, receives a reward r_t and arrives at s_{t+1}. The transitions (s_t, a_t, s_{t+1}, r_t) are stored in the replay buffer R. A batch of N transitions are drawn from R and the Q-value y_i is updated as:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'}, \theta^{Q'})), i = 1, \cdots, N. \tag{15}$$

The critic network is then updated by minimizing the loss function $L(\theta^Q)$ which is the expected difference between outputs of the target critic network Q' and the critic network Q, i.e,

$$L(\theta^Q) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1} \sim \text{buffer}}[(y_i - Q(s_t, a_t|\theta^Q))^2]. \tag{16}$$

DDPG is effective at handling continuous action space, and so it is appropriate for stock trading.

C. Proximal Policy Optimization (PPO)

We explore and use PPO as a component in the ensemble method. PPO [14] is introduced to control the policy gradient update and ensure that the new policy will not be too different from the previous one. PPO tries to simplify the objective of Trust Region Policy Optimization (TRPO) by introducing a clipping term to the objective function [41], [14].

Let us assume the probability ratio between old and new policies is expressed as:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}. \tag{17}$$

The clipped surrogate objective function of PPO is:

$$\begin{aligned} J^{\text{CLIP}}(\theta) = &\hat{\mathbb{E}}_t[\min(r_t(\theta)\hat{A}(s_t, a_t), \\ &\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}(s_t, a_t))], \end{aligned} \tag{18}$$

where $r_t(\theta)\hat{A}(s_t, a_t)$ is the normal policy gradient objective, and $\hat{A}(s_t, a_t)$ is the estimated advantage function. The function $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$ clips the ratio $r_t(\theta)$ to be within $[1-\epsilon, 1+\epsilon]$. The objective function of PPO takes the minimum of the clipped and normal objective. PPO discourages large policy change move outside of the clipped interval. Therefore, PPO improves the stability of the policy networks training by restricting the policy update at each training step. We select PPO for stock trading because it is stable, fast, and simpler to implement and tune.

D. Ensemble Strategy

Our purpose is to create a highly robust trading strategy. So we use an ensemble strategy to automatically select the best performing agent among PPO, A2C, and DDPG to trade based on the Sharpe ratio. The ensemble process is described as follows:

Step 1. We use a growing window of n months to retrain our three agents concurrently. In this paper we retrain our three agents at every three months.

Step 2. We validate all three agents by using a 3-month validation rolling window after training window to pick the best performing agent with the highest Sharpe ratio [42]. The Sharpe ratio is calculated as:

$$Sharpe\ ratio = \frac{\bar{r}_p - r_f}{\sigma_p}, \tag{19}$$

where $\bar{r}_p$ is the expected portfolio return, r_f is the risk free rate, and σ_p is the portfolio standard deviation. We also adjust risk-aversion by using turbulence index in our validation stage.

Step 3. After the best agent is picked, we use it to predict and trade for the next quarter.

The reason behind this choice is that each trading agent is sensitive to different type of trends. One agent performs well in a bullish trend but acts bad in a bearish trend. Another agent is more adjusted to a volatile market. The higher an agent's Sharpe ratio, the better its returns have been relative to the amount of investment risk it has taken. Therefore, we pick the trading agent that can maximize the returns adjusted to the increasing risk.

VI. Performance Evaluations

In this section, we present the performance evaluation of our proposed scheme. We perform backtesting for the three individual agents and our ensemble strategy. The result in Table 2 demonstrates that our ensemble strategy achieves higher Sharpe ratio than the three agents, Dow Jones Industrial Average and the traditional min-variance portfolio allocation strategy.

Our codes are available on Github [2].

[2]Link: https://github.com/AI4Finance-LLC/Deep-Reinforcement-Learning-for-Automated-Stock-Trading-Ensemble-Strategy-ICAIF-2020

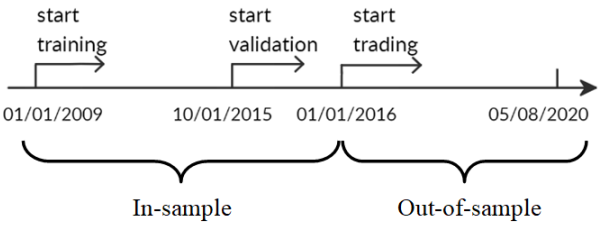

Fig. 4. Stock data splitting.

A. Stock Data Preprocessing

We select the Dow Jones 30 constituent stocks (at 01/01/2016) as our trading stock pool. Our backtestings use historical daily data from 01/01/2009 to 05/08/2020 for performance evaluation. The stock data can be downloaded from the Compustat database through the Wharton Research Data Services (WRDS) [43]. Our dataset consists of two periods: in-sample period and out-of-sample period. In-sample period contains data for training and validation stages. Out-of-sample period contains data for trading stage. In the training stage, we train three agents using PPO, A2C, and DDPG, respectively. Then, a validation stage is then carried out for validating the 3 agents by Sharpe ratio, and adjusting key parameters, such as learning rate, number of episodes, etc. Finally, in the trading stage, we evaluate the profitability of each of the algorithms.

The whole dataset is split as shown in Figure 4. Data from 01/01/2009 to 09/30/2015 is used for training, and the data from 10/01/2015 to 12/31/2015 is used for validation and tuning of parameters. Finally, we test our agent's performance on trading data, which is the unseen out-of-sample data from 01/01/2016 to 05/08/2020. To better exploit the trading data, we continue training our agent while in the trading stage, since this will help the agent to better adapt to the market dynamics.

B. Performance Comparisons

1) Agent Selection: From Table 1, we can see that PPO has the best validation Sharpe ratio of 0.06 from 2015/10 to 2015/12, so we use PPO to trade for the next quarter from 2016/01 to 2016/03. DDPG has the best validation Sharpe ratio of 0.61 from 2016/01 to 2016/03, so we use DDPG to trade for the next quarter from 2016/04 to 2016/06. A2C has the best validation Sharpe ratio of -0.15 from 2020/01 to 2020/03, so we use A2C to trade for the next quarter from 2020/04 to 2020/05. Five metrics are used to evaluate our results:

- Cumulative return: is calculated by subtracting the portfolio's final value from its initial value, and then dividing by the initial value.
- Annualized return: is the geometric average amount of money earned by the agent each year over the time period.
- Annualized volatility: is the annualized standard deviation of portfolio return.
- Sharpe ratio: is calculated by subtracting the annualized risk free rate from the annualized return, and the dividing by the annualized volatility.
- Max drawdown: is the maximum percentage loss during the trading period.

TABLE I
SHARPE RATIOS OVER TIME.

Trading Quarter	PPO	A2C	DDPG	Picked Model
2016/01-2016/03	**0.06**	0.03	0.05	PPO
2016/04-2016/06	0.31	0.53	**0.61**	DDPG
2016/07-2016/09	-0.02	0.01	**0.05**	DDPG
2016/10-2016/12	**0.11**	0.01	0.09	PPO
2017/01-2017/03	**0.53**	0.44	0.13	PPO
2017/04-2017/06	0.29	**0.44**	0.12	A2C
2017/07-2017/09	**0.4**	0.32	0.15	PPO
2017/10-2017/12	-0.05	-0.04	**0.12**	DDPG
2018/01-2018/03	**0.71**	0.63	0.62	PPO
2018/04-2018/06	-0.08	-0.02	**-0.01**	DDPG
2018/07-2018/09	-0.17	**0.21**	-0.03	A2C
2018/10-2018/12	0.30	**0.48**	0.39	A2C
2019/01-2019/03	-0.26	-0.25	**-0.18**	DDPG
2019/04-2019/06	**0.38**	0.29	0.25	PPO
2019/07-2019/09	**0.53**	0.47	0.52	PPO
2019/10-2019/12	-0.22	**0.11**	-0.22	A2C
2020/01-2020/03	-0.36	**-0.13**	-0.22	A2C
2020/04-2020/05	-0.42	**-0.15**	-0.58	A2C

Cumulative return reflects returns at the end of trading stage. Annualized return is the return of the portfolio at the end of each year. Annualized volatility and max drawdown measure the robustness of a model. The Sharpe ratio is a widely used metric that combines the return and risk together.

2) Analysis of Agent Performance: From both Table 2 and Figure 5, we can observe that the A2C agent is more adaptive to risk. It has the lowest annual volatility 10.4% and max drawdown -10.2% among the three agents. So A2C is good at handling a bearish market. PPO agent is good at following trend and acts well in generating more returns, it has the highest annual return 15.0% and cumulative return 83.0% among the three agents. So PPO is preferred when facing a bullish market. DDPG performs similar but not as good as PPO, it can be used as a complementary strategy to PPO in a bullish market. All three agents' performance outperform the two benchmarks, Dow Jones Industrial Average and min-variance portfolio allocation of DJIA, respectively.

3) Performance under Market Crash: In Figure 6, we can see that our ensemble strategy and the three agents perform well in the 2020 stock market crash event. When the turbulence index reaches a threshold, it indicates an extreme market situation. Then our agents will sell off all currently held shares and wait for the market to return to normal to resume trading. By incorporating the turbulence index, the agents are able to cut losses and successfully survive the stock market crash in March 2020. We can tune the turbulence index threshold lower for higher risk aversion.

4) Benchmark Comparison: Figure 5 demonstrates that our ensemble strategy significantly outperforms the DJIA and the min-variance portfolio allocation [9]. As can be

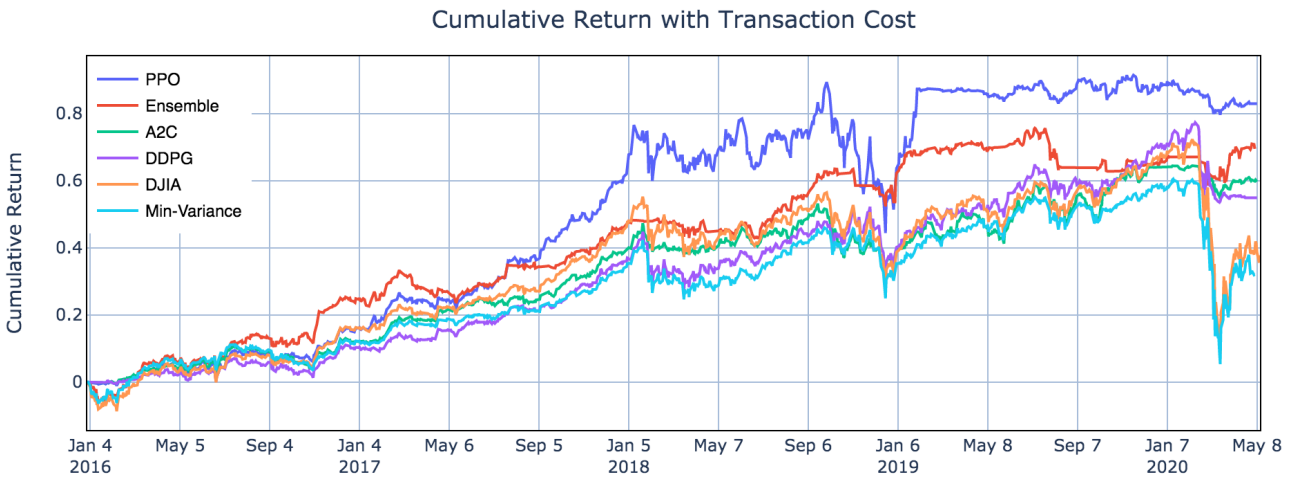

Fig. 5. Cumulative return curves of our ensemble strategy and three actor-critic based algorithms, the min-variance portfolio allocation strategy, and the Dow Jones Industrial Average. (Initial portfolio value $\$1,000,000$, from 2016/01/04 to 2020/05/08).

TABLE II
PERFORMANCE EVALUATION COMPARISON.

(2016/01/04-2020/05/08)	Ensemble (Ours)	PPO	A2C	DDPG	Min-Variance	DJIA
Cumulative Return	70.4%	83.0%	60.0%	54.8%	31.7%	38.6%
Annual Return	13.0%	15.0%	11.4%	10.5%	6.5%	7.8%
Annual Volatility	9.7%	13.6%	10.4%	12.3%	17.8%	20.1%
Sharpe Ratio	1.30	1.10	1.12	0.87	0.45	0.47
Max Drawdown	-9.7%	-23.7%	-10.2%	-14.8%	-34.3%	-37.1%

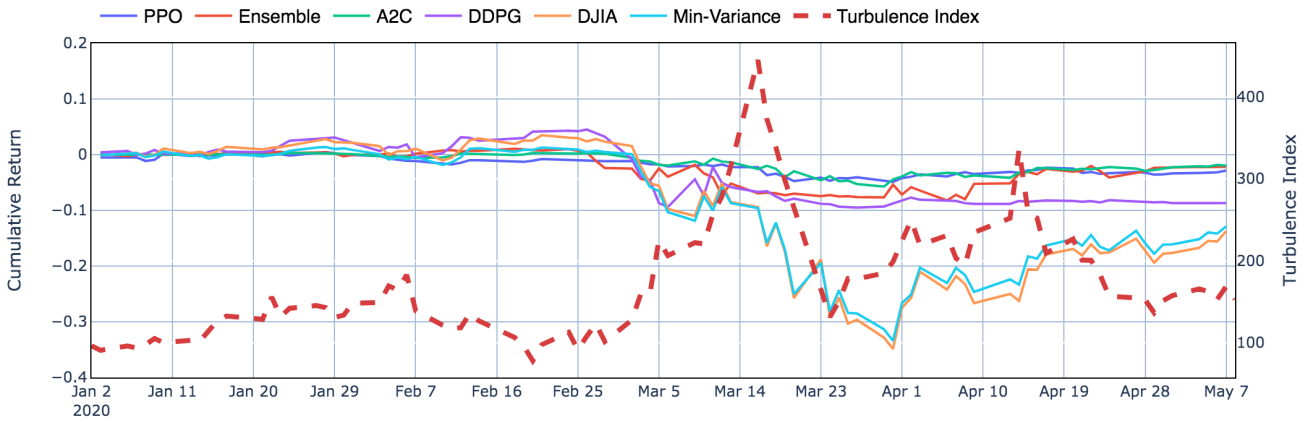

Fig. 6. Performance during the stock market crash in the first quarter of 2020.

seen from Table 2, the ensemble strategy achieves a Sharpe ratio 1.30, which is much higher than the Sharpe ratio of 0.47 for DJIA, and 0.45 for the min-variance portfolio allocation. The annualized return of the ensemble strategy is also much higher, the annual volatility is much lower, indicating that the ensemble strategy beats both the DJIA and min-variance portfolio allocation in balancing risk and return. The ensemble strategy also outperforms A2C with a Sharpe ratio of 1.12, PPO with a Sharpe ratio of 1.10, and DDPG with a Sharpe ratio of 0.87, respectively. Therefore, our findings demonstrate that the proposed ensemble strategy can effectively develop a trading strategy that outperforms the three individual algorithms and the two baselines.

VII. CONCLUSION

In this paper, we have explored the potential of using actor-critic based algorithms which are Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Deterministic Policy Gradient (DDPG) agents to learn stock trading strategy. In order to adjust to different market situations, we use an ensemble strategy to automatically select the best performing agent to trade based on the Sharpe ratio. Results show that our ensemble strategy outperforms the three individual algorithms, the Dow Jones Industrial Average and min-variance portfolio allocation method in terms of Sharpe ratio by balancing risk and return under transaction costs.

For future work, it will be interesting to explore more sophisticated model [44], solve empirical challenges [45], deal with large-scale data [46] such as S&P 500 constituent stocks. We can also explore more features for the state space such as adding advanced transaction cost and liquidity model [47], incorporating fundamental analysis indicators [9], natural language processing analysis of financial market news [48], and ESG features [12] to our observations. We are interested in directly using Sharpe ratio as the reward function, but the agents need to observe a lot more historical data, the state space will increase exponentially.

REFERENCES

[1] Stelios D. Bekiros, "Fuzzy adaptive decision-making for boundedly rational traders in speculative stock markets," *European Journal of Operational Research*, vol. 202, no. 1, pp. 285–293, April 2010.

[2] Yong Zhang and Xingyu Yang, "Online portfolio selection strategy based on combining experts' advice," *Computational Economics*, vol. 50, 05 2016.

[3] Youngmin Kim, Wonbin Ahn, Kyong Joo Oh, and David Enke, "An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms," *Applied Soft Computing*, vol. 55, pp. 127–140, 02 2017.

[4] Harry Markowitz, "Portfolio selection," *Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[5] Dimitri Bertsekas, *Dynamic programming and optimal control*, vol. 1, 01 1995.

[6] Francesco Bertoluzzo and Marco Corazza, "Testing different reinforcement learning configurations for financial trading: introduction and applications," *Procedia Economics and Finance*, vol. 3, pp. 68–77, 12 2012.

[7] Ralph Neuneier, "Optimal asset allocation using adaptive dynamic programming," *Conference on Neural Information Processing Systems, 1995*, 05 1996.

[8] Ralph Neuneier, "Enhancing q-learning for optimal asset allocation.," 01 1997.

[9] Hongyang Yang, Xiao-Yang Liu, and Qingwei Wu, "A practical machine learning approach for dynamic stock recommendation," in *IEEE TrustCom/BiDataSE, 2018.*, 08 2018, pp. 1693–1697.

[10] Yunzhe Fang, Xiao-Yang Liu, and Hongyang Yang, "Practical machine learning approach to capture the scholar data driven alpha in ai industry," in *2019 IEEE International Conference on Big Data (Big Data) Special Session on Intelligent Data Mining*, 12 2019, pp. 2230–2239.

[11] Wenbin Zhang and Steven Skiena, "Trading strategies to exploit blog and news sentiment.," in *Fourth International AAAI Conference on Weblogs and Social Media, 2010*, 01 2010.

[12] Qian Chen and Xiao-Yang Liu, "Quantifying esg alpha using scholar big data: An automated machine learning approach," *ACM International Conference on AI in Finance, ICAIF 2020*, 2020.

[13] Vijay Konda and John Tsitsiklis, "Actor-critic algorithms," *Society for Industrial and Applied Mathematics*, vol. 42, 04 2001.

[14] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 07 2017.

[15] Zhipeng Liang, Kangkang Jiang, Hao Chen, Junhao Zhu, and Yanran Li, "Adversarial deep reinforcement learning in portfolio management," *arXiv: Portfolio Management*, 2018.

[16] Volodymyr Mnih, Adrià Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," *The 33rd International Conference on Machine Learning*, 02 2016.

[17] Zihao Zhang, "Deep reinforcement learning for trading," *ArXiv 2019*, 11 2019.

[18] Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, "Continuous control with deep reinforcement learning," *International Conference on Learning Representations (ICLR) 2016*, 09 2015.

[19] Zhuoran Xiong, Xiao-Yang Liu, Shan Zhong, Hongyang Yang, and A. Elwalid, "Practical deep reinforcement learning approach for stock trading," *NeurIPS Workshop on Challenges and Opportunities for AI in Financial Services: the Impact of Fairness, Explainability, Accuracy, and Privacy, 2018.*, 2018.

[20] Thomas G. Fischer, "Reinforcement learning in financial markets - a survey," FAU Discussion Papers in Economics 12/2018, Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics, 2018.

[21] Lin Chen and Qiang Gao, "Application of deep reinforcement learning on automated stock trading," in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, 2019, pp. 29–33.

[22] Quang-Vinh Dang, "Reinforcement learning in stock trading," in *Advanced Computational Methods for Knowledge Engineering. ICCSAMA 2019. Advances in Intelligent Systems and Computing, vol 1121. Springer, Cham*, 01 2020.

[23] Gyeeun Jeong and Ha Kim, "Improving financial trading decisions using deep q-learning: predicting the number of shares, action strategies, and transfer learning," *Expert Systems with Applications*, vol. 117, 09 2018.

[24] John Moody and Matthew Saffell, "Learning to trade via direct reinforcement," *IEEE Transactions on Neural Networks*, vol. 12, pp. 875–89, 07 2001.

[25] 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*, vol. 28, pp. 1–12, 02 2016.

[26] Zhengyao Jiang and Jinjun Liang, "Cryptocurrency portfolio management with deep reinforcement learning," in *2017 Intelligent Systems Conference*, 09 2017.

[27] Stelios Bekiros, "Heterogeneous trading strategies with adaptive fuzzy actor-critic reinforcement learning: A behavioral approach," *Journal of Economic Dynamics and Control*, vol. 34, pp. 1153–1170, 06 2010.

[28] Jinke Li, Ruonan Rao, and Jun Shi, "Learning to trade with deep actor critic methods," *2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, vol. 02, pp. 66–71, 2018.

[29] Yuxin Wu and Yuandong Tian, "Training agent for first-person shooter game with actor-critic curriculum learning," in *International Conference on Learning Representations (ICLR), 2017*, 2017.

[30] A. Ilmanen, "Expected returns: An investor's guide to harvesting market rewards," 05 2012.

[31] Mark Kritzman and Yuanzhen Li, "Skulls, financial turbulence, and risk management," *Financial Analysts Journal*, vol. 66, 10 2010.

[32] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, "Openai gym," 2016.

[33] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.

[34] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[35] Terence Chong, Wing-Kam Ng, and Venus Liew, "Revisiting the performance of macd and rsi oscillators," *Journal of Risk and Financial Management*, vol. 7, pp. 1–12, 03 2014.

[36] Mansoor Maitah, Petr Procházka, Michal Čermák, and Karel Šrédl, "Commodity channel index: evaluation of trading rule of agricultural commodities," *International Journal of Economics and Financial Issues*, vol. 6, pp. 176–178, 03 2016.

[37] Ikhlaas Gurrib, "Performance of the average directional index as a market timing tool for the most actively traded usd based currency pairs," *Banks and Bank Systems*, vol. 13, pp. 58–70, 08 2018.

[38] Richard Sutton and Andrew Barto, "Reinforcement learning: an introduction," *IEEE Transactions on Neural Networks*, vol. 9, pp. 1054, 02 1998.

[39] Richard Sutton, David Mcallester, Satinder Singh, and Yishay Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Conference on Neural Information Processing Systems (NeurIPS), 1999*, 02 2000.

[40] Lucian Busoniu, Tim de Bruin, Domagoj Tolić, Jens Kober, and Ivana Palunko, "Reinforcement learning for control: Performance,

stability, and deep approximators," *Annual Reviews in Control*, 10 2018.

[41] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel, "Trust region policy optimization," in *The 31st International Conference on Machine Learning*, 02 2015.

[42] W.F. Sharpe, "The sharpe ratio," *Journal of Portfolio Management*, 01 1994.

[43] Wharton Research Data Service, "Standard & poor's compustat," 2015, Data retrieved from Wharton Research Data Service,.

[44] Lu Wang, Wei Zhang, Xiaofeng He, and Hongyuan Zha, "Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation," in *Conference on Knowledge Discovery and Data Mining (KDD), 2018*, 07 2018, pp. 2447–2456.

[45] Gabriel Dulac-Arnold, N. Levine, Daniel J. Mankowitz, J. Li, Cosmin Paduraru, Sven Gowal, and T. Hester, "An empirical investigation of the challenges of real-world reinforcement learning," *ArXiv*, vol. abs/2003.11881, 2020.

[46] Yuri Burda, Harrison Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei Efros, "Large-scale study of curiosity-driven learning," in *2019 Seventh International Conference on Learning Representations (ICLR) Poster*, 08 2018.

[47] Wenhang Bao and Xiao-Yang Liu, "Multi-agent deep reinforcement learning for liquidation strategy analysis," *ICML Workshop on Applications and Infrastructure for Multi-Agent Learning, 2019*, 06 2019.

[48] Xinyi Li, Yinchuan Li, Hongyang Yang, Liuqing Yang, and Xiao-Yang Liu, "Dp-lstm: Differential privacy-inspired lstm for stock prediction using financial news," *33rd Conference on Neural Information Processing Systems (NeurIPS 2019) Workshop on Robust AI in Financial Services: Data, Fairness, Explainability, Trustworthiness, and Privacy, December 2019*, 12 2019.