Stock Market Prediction Using Deep Reinforcement learning: Ensemble strategy

Bandi Rupendra Reddyy[1], Darukumalli Sai Tharun Reddy[1], Sandeep Preetham M C[1], Amudha J[2]

[1]Computer Science and Engineering, Amrita School of Engineering, Bengaluru, Amrita Vishwa Vidyapeetham, India.
[2]Department of Computer Science and Engineering,Amrita School of Engineering, Bengaluru, Amrita Vishwa Vidyapeetham, India.

[1]bl.en.u4aie19009@bl.students.amrita.edu, [1]bl.en.u4aie190016@bl.students.amrita.edu, [1]bl.en.u4aie19058@bl.students.amrita.edu, [2] j_amudha@blr.amrita.edu

***Abstract*— Stock trading tactics are important in investing. However, designing a successful strategy in a complicated and dynamic stock market is difficult. In our project, we offer an ensemble technique for learning a stock trading strategy by maximizing investment return using deep reinforcement schemes. We use FinRL's fine-tuned state-of-the-art DRL algorithms to train a deep reinforcement learning agent and produce an ensemble trading strategy: Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), and Deep Deterministic Policy Gradient (DDPG). The ensemble approach inherits and combines the greatest qualities of the three algorithms, allowing it to adapt to changing market conditions with ease. We test our algorithms on the stocks data using Yahoo finance stocks API that has adequate liquidity.**

Keywords—PPO, A2C, DDPG, Ensemble strategy, FinRl.

I. INTRODUCTION

Stock trading is the process of purchasing and selling stocks in order to profit from an investment. The key to stock trading is making the proper trading decisions at the right moments, or developing a suitable trading strategy. Many studies in recent years have used machine learning algorithms to forecast stock movements or prices in order to execute stock trading. Furthermore, the stock market is affected by numerous things, including changes in investor psychology and business policies, and so on, and the stock price swings dramatically. A dynamic trading strategy, as opposed to a static trading strategy, can make trading decisions based on changes in the stock market, which offers more advantages. Reinforcement learning solves the sequential decision-making problem and can be used to learn dynamic trading techniques in stock trading. Reinforcement learning, on the other hand, lacks the ability to comprehend the environment. A growing number of researchers are using deep reinforcement learning to build dynamic trading strategies.

When the decision-making capacity of reinforcement learning is combined with the perception ability of deep learning (i.e., deep reinforcement learning), it solves the problem and has more advantages. Deep reinforcement learning (DRL), has been identified as a beneficial strategy for automated stock trading. By learning through contact with an unfamiliar environment, the DRL framework is effective in solving dynamic decision-making challenges. The right analysis of the status of the stock market is one of the obstacles when implementing stock trading based on deep reinforcement learning. However, stock data contains noise, which has an impact on the final analysis conclusions. Technical indicators help lessen the impact of noise by reflecting movements in the stock market from several perspectives. We provide a three-layered FinRL library that simplifies the creation of stock trading methods in this project. FinRL provides common building blocks that enable strategy developers to create virtual stock market environments, train deep neural networks as trading agents, measure trading performance through comprehensive backtesting, and add crucial market frictions.

The environment, which mimics the financial market environment using real historical data and other environmental parameters such as closing price, shares, trading volume, technical indicators, and so on, is the lowest level. The agent layer sits in the center, providing fine-tuned standard DRL algorithms (A2C, PPO, and DDPG), commonly used reward functions, and standard evaluation baselines to reduce debugging time and improve reproducibility. The agent interacts with the environment using state and action space reward functions that are appropriately described. Then, using three actor-critic-based algorithms, we create an ensemble deep reinforcement learning trading strategy. It also integrates the best attributes acquired from the three methods so that the agent can adapt to changing conditions. We looked into the possibilities 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.

II. LITERATURE SURVEY

Recent implementations of deep reinforcement learning in financial markets take into account discrete or continuous state and action spaces and use one of three learning approaches: critic-only, actor-only, or actor-critic [3].The most frequent strategy, critic-only learning, handles a discrete action space problem by training an agent on a single stock or asset using, for example, Deep Q-learning (DQN) and its refinements [5, 6, 9]. The critic-only technique aims to learn the best action-selection strategy that maximizes the predicted future reward given the current state using a Q-value function. Instead of creating a state-action value table,

DQN minimizes the difference between estimated and target Q-values over a transition and does function approximation with a neural network.

The actor-only approach was applied In [10, 13, 14]. The idea here is that the agent learns the best policy on its own. Instead of learning the Q-value, the neural network is trained to learn the policy. The policy is a probability distribution that serves as a strategy for a particular situation, namely the likelihood of taking a permitted action. In [14], recurrent reinforcement learning is used to overcome the dimensionality curse and increase trade efficiency. Continuous action space settings can be handled using an actor-only method.The performer improves his or her acts with time, while the critic improves his or her ability to evaluate those actions. The actor-critic approach has been utilized to play popular video games like Doom [15] and has demonstrated be the ability to learn and adapt to big and complicated landscapes. As a result, the actor-critic method of trading with a large stock portfolio appears to be promising.

STOCK MARKET PREREQUISITE:-

The stock market is a venue where people may purchase and sell equity shares in firms (buyers and sellers of stocks). Investors and traders looking for short-term or long-term profits are among the participants. The majority of investors have a long-term view and profit from capital growth over time. Traders, on the other hand, seek quick profits by concentrating on modest price fluctuations in equities shares that often last a few minutes or the entire trading session. In order to analyze the stock trend and its statistics, we use various mathematical functions for a better understanding of the stock trend.

RL Formulation:
Considering the stochastic and interactive nature of the automated stock trading tasks, a financial task is modeled as a Markov Decision Process (MDP) problem. The training process involves observing stock price change, taking an action, and reward calculations to have the agent adjust its strategy accordingly. By interacting with the environment, the trading agent will derive a trading strategy with maximized rewards as time proceeds. Our trading environments, based on the OpenAI Gym framework, simulate livestock markets with real market data according to the principle of time-driven simulation. FinRL library strives to provide trading environments constructed by six datasets across five stock exchanges.

Now we give definitions of the state space, action space, and reward function.

State-space S:
The state-space describes the observations that the agent receives from the environment. Just as a human trader needs to analyze various information before executing a trade, so our trading agent observes many different features to better learn in an interactive environment. We provide various features for users:

Balance bt ∈ R+: the amount of money left in the account at the current time step t.
Shares own ht ∈ Z n +: current shares for each stock n represents the number of stocks.
Closing price pt ∈ R n +: one of the most commonly used features.
Opening/high/low prices ot, ht,lt ∈ R n +: used to track stock price changes.
Trading volume vt ∈ R n +: the total quantity of shares traded during a trading slot.
Technical indicators: Moving Average Convergence Divergence (MACD) Mt ∈ R n and Relative Strength Index (RSI) Rt ∈ R n +, etc.
Multiple-level of granularity: we allow data frequency of the above features to be daily, hourly, or on a minute basis

Action space A.

The action space specifies the possible interactions between the agent and the environment. Typically our actions involve three actions: a ∈ {−1, 0, 1}, where -1, 0, 1 represents selling, holding, and purchasing one stock. An action can also be performed on multiple shares. We employ the action space {-n, ..., −1, 0, 1, ..., n}, where n is the number of shares.

The likely mechanism for an agent to learn a better action is the reward function. Reward functions come in a variety of shapes and sizes. The following are some of the most regularly used:

• When action is executed at state s and you arrive at the new state, the portfolio value changes.

• When action is taken at state s and new state s, the portfolio log returns.

• The Sharpe ratio for the time intervals t = {1,..., T}.

• User-defined reward functions, such as risk factors or transaction cost terms, are also supported by FinRL.

III. BACKGROUND

TECHNICAL INDICATORS

Technical Indicators are nothing but mathematical patterns formed from the historical data of a stock which is used by the traders in order to predict the future trends of the stock. The historical data here refers to the stock price, opening-closing price, or even the volume of that stock, where volume refers to the total quantity of the stock.

The indicator is weighted based on the historically adjusted returns, the objective behind the trader's investment, and also based on identifying the right trading opportunity.

There are two types of technical indicators:-

- Oscillators- They are a subset of technical indicators which oscillate between the local minimum and the

maximum of the stock market prices. They are also referred to as momentum indicators.

- Overlays- They are a special kind of technical indicator that is used by the traders to talk about the strength of the stock which is nothing but to identify overbought and oversold levels. They serve as a good insight into the supply and demand of the stock.

Now, let's talk about the technical indicators used in our model. FinRL provides various inbuilt technical indicators. We shall be extracting the most effective indicators from the open-source library that serve us with a high cumulative return.

- Moving average convergence divergence
- Relative strength index
- Bollinger bands
- Commodity channel index
- Volatile index
- Simple Moving Average

MACD – It's a momentum indicator that analyses the difference between two moving averages to indicate the link between them. The MACD indicator reflects the trend's intensity as well as its direction. The MACD is generated by subtracting the 12-period EMA from the 26-period EMA (exponential moving average). When the technical signal passes above the signal line, it activates a buy signal, and when it crosses below the zero line, it triggers a sell signal.

RSI – It is also a momentum indicator that measures that analyses the strength and weakness of the trend of the asset to check whether it is overbought or oversold. It is displayed as an oscillator (a line graph that moves between 2 extreme points) and is allowed to have a reading from 0-100 in such a way that the stock is said to be oversold if the oscillator moves less than 30 and it is overbought if it is more than 70.

BB - Bollinger Band is another technical analysis tool characterized by a series of trendlines plotting two standard deviations (positively and negatively) away from a security's price's simple moving average (SMA), but which can be altered to the user's preferences. If the closer prices get to the upper band, the more overbought the market becomes, and the closer prices get to the lower band, the more oversold it becomes.

VIX – Cboe volatility indexing is a mathematical representation of how volatile the market is expected to be in the coming 30 days.

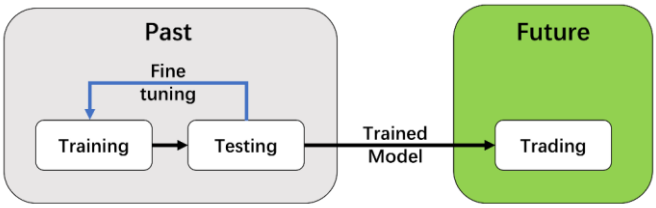
SMA – It is another kind of technical indicator that calculates the average of a selected range of prices (closing prices) by the total number of periods in that particular range. It can help in determining if the stock price will continue or if it will reverse a bull.

FINRL:

FinRL is an open-source library built by AI4Finance, whose main purpose was to serve quantitative finance. FinRL is a combination of Value-based, Policy-based, and actor critic-based algorithms. It has multiple built-in agents and mathematical-statistical models which makes the work handy for the user.

FInRL is built over a three-layered architecture, which consists of Stock Market environments (FinRL-Meta), DRL trading agents, and stock trading applications. It is built in such a way that the lower layers provide APIs for the upper layer which makes the upper layers transparent to the lower layers. The agent here interacts in an exploration-exploitation manner hoping to converge the epsilon value by the end of the training phase.

- The trading environment is based on OpenAI Gym which tries to simulate the real-world market data using time-driven simulations. It provides trading environments that are constructed by the datasets across many stock exchanges. We will be following the DataOps paradigm in the data layer, where we follow the Training-Testing-Trading pipeline. Here the agent is allowed to first learn during the training environment and it is further validated (here the hyper parameters are tuned if the user chooses to use a customs agent). Then the validated agent is tested with the historical datasets. Then the agent will be allowed for trading.



- FinRL has an inbuilt finetuned standard DEL algorithms. The agents are built by fine-tuned standard DRL algorithms which are dependent on three famous DRL libraries that are ElegantRL, Stable Baseline 3, and Rlib. The above algorithms include DQN, DDPG, Multiagent DDPG, PPO, and many more. The below picture depicts the comparison of DRL algorithms which is taken from the official FInRL document.

This gives the flexibility for the users to choose their agent according to their requirements and train their model.

- FinRL provides various applications such as stock trading, high-frequency trading, cryptocurrency trading, etc. In our model, we have used Multi-Agent stock trading as our application.

The standout advantage of using FinRL for deep reinforcement stock trading problems is it provides

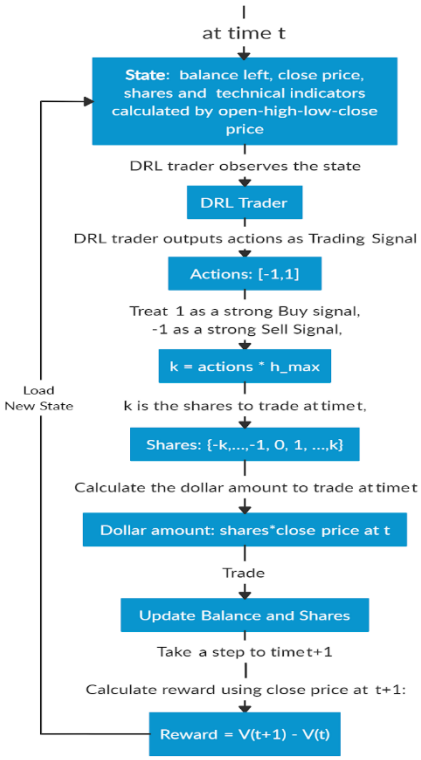
Modularity, very applicable and user-friendly, and Easier, and better market environment modelling.

OVERVIEW OF THE DEEP REINFORCEMENT LEARNING FOR MULTIPLE STOCK TRADING

For our project, we have used the most popular US stocks which are the Dow 30 constituents. Generally, Let's consider a trained DRL agent " DRL Trader" which we would be using to trade the multiple stocks in our portfolio.

- As we have earlier spoken about our state space, let's assume that we are at time t(which is at the end of the day) As the day ends we will know about the closing, opening, and high low prices of the stock in that particular day.
- Alongside we can also use a few of the technical indicators which help our agent in learning the stock statistics.
- Initially, the portfolio value of our stock at time t is V(t) = the current balance + the amount of the stock at time t
- This data of the states in our environment will be sent as an input to our agent which further decides on the action to take. The agent has the flexibility to take 3 actions which are (buy, hold and sell) which are stochastically represented within a range from [-1,1] where -1 represents a strong sign to sell the stock and likewise to buy if the action represents a positive 1.
- Initially to decide on the maximum shares to trade a parameter is being initialized which is represented by h_max and set to 100.
- Now we calculate K which is the dot product of actions * h/-max, just so we have the list of the shares that we would like to trade
- We now have got to update the balance and shares in our portfolio which is calculated as Updated balance = balance(t) – the amount of money spent to buy the stocks + amount of money received to sell the stocks. Likewise for the updated shares = shares held – shares to be sold + shares to buy.
- We take actions to trade based on the agent at the end of the day (at time t). We should by now be aware that the time t's close price equals time t+1's open price.
- Now a step is being taken to time t+1, at the end of the day. As we will be aware of the closing price at t+1, we shall calculate the dollar amount of the stocks(t+1) = sum(updated shares * closing price (t+1)). The portfolio value V(t+1) = balance(t+1)+dollar amount of the stocks (t+1).
- Now that we have the portfolio value at t+1, the step reward received by the agent for performing a specific action is going to be the difference between the new portfolio value at time t+1 and the previous portfolio value at time t, r =v(t+1) – v(t).
- During the training phase of the agent, we give It the flexibility to have a positive or a negative reward but during the testing phase, we expect the agent to yield us positive rewards only.
- This process is repeated until the agent finishes its reinforcement learning.

Now we shall represent a logic chart for our multiple stock trading and an example to demonstrate:-



IV. METHODS

ADVANTAGE ACTOR-CRITIC ALGORITHM

The A2C combines two types of reinforcement learning which are policy-based (where the agent learns from the policy) and value-based (where the agent learns to select a particular action based on the input state or action.

The Algorithm consists of 3 main components which are the actor-network, the critic network, and the advantage function which is used to calculate the agent's temporal difference error. In simple words, the actor-networks perform a particular action and the critic network evaluates the action taken by the actor.

- Advantage function
  In order to know the advantage function in A2C, we first need to be aware of the temporal difference. In temporal difference, the agent tends to learn the trend by making predictions about future rewards and adjusting their action based on the prediction error. The advantage function captures how better an action is compared to the others at a given state. The standout role of the advantage function is to present to us if a state is better or worse than an expected, for example, if the advantage function is less than 0 then the action is said to be worse ad we would

suggest the agent to not perform that action. The advantage function can be defined as

$$A(s_t, a_t) = Q_w(s_t, a_t) - V_v(s_t)$$

- Actor-Network
  The network mas the input states to a corresponding action where it outputs a probabilistic distribution corresponding to each action.

- Critic network
  Unlike the actor-network, the critic network outputs a probability distribution of actions, it chooses to output the temporal difference target of the input state as a floating-point number.

In our model the actor-network is going to take the states which are the stock prices and the technical indicators and perform a particular action as to sell or buy or hold the stock, it's the role of the critic to evaluate the action performed by the actor and assign its Q values. Then the advantage function helps in letting the agent know if the performed action is better than compared to other actions (buy, sell, hold) in that particular state at time t. simultaneously the agent learns.

DEEP DETERMINISTIC POLICY GRADIENT (DDPG)

DDPG mainly consists of four different neural networks (parameters)

- Q network
- - Deterministic policy function
- Target Q-network
- Target policy network

The difference between DDPG and that with A2C is that in DDPG the Actor directly maps states to actions instead of outputting the probability distribution across the discrete action space. The target networks are nothing but the time-delayed copies of their original networks hat slowly track, the learned networks. On using the target networks helps the model to not be pruned towards divergence.

The pseudo code for DDPG is given as :-

Algorithm 1 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

DDPG also uses Replay buffer just like Deep Q learning, it is used to sample experience to update neural network parameters, so during each trajectory roll out, we save our experience tuples which are state, action, next state, and the reward, and store them in a finite-sized cache called as Replay buffer. Further random mini-batches of experience get sampled from the buffer when the value and the policy networks get updated.

The Value network here gets updated very similar to Q-learning. However, in DDPG, the next state's Q-values are calculated with the help of the target value network and target policy network and further minimize the squared loss between the updated Q value and original Q value.

PROXIMAL POLICY OPTIMISATION

PPO aims to achieve a balance between important factors like ease of implementation, ease of tuning, sample complexity, etc. PPO is an on-policy algorithm that is flexible to use for environments that are either discrete or continuous action spaces. It ensures that the newly updated policy isn't too different from the previously used policy. It can also be used for online data. The PPO algorithm is given as:-

$$L^{CLIP}(\theta) = \hat{E}_t[min(r_t(\theta)\hat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)]$$

where,
θ *is the policy parameter*
$\hat{E}_t$ *denotes the empirical expectation over timesteps*
r_t*denotestherationoftheprobabilitiesunderthenewandoldpolicies*
$\hat{A}_t$ *is the estimated advantage at time t*
ϵ *is a hyperparameter, usually* 0.1 *or* 0.2.

At Each step, there is an update to the existing policy to acknowledge improvement on certain parameters and to ensure that the updated value is not too large from the previous policy. Where the epsilon hyper parameter talks about the limit of range within which the update is allowed.

In PPO, data is being observed in a small set of batches which once after use is thrown away in order to incorporate a new batch of observations called minibatch. Further, the updated policy will be clipped just to ensure that there is no huge change in the policy as in PPO very large updates are strictly avoided.

Algorithms	Input	Output	Type	State-action Spaces support	Finance use cases
DDPG	State- action Pair	Q-value	Actor-critic Based	Continuous only	Multiple stock trading, portfolio allocation
A2C	State- action Pair	Q-value	Actor-critic Based	Continuous and Discrete	Single stock trading, Multiple stock trading, portfolio allocation
PPO	State- action Pair	Q-value	Actor-critic Based	Continuous and Discrete	Single stock trading, Multiple stock trading, portfolio allocation

V. IMPLEMENTATION

Dataset:

First, we have to install all the packages needed for our implementation like Yahoo Finance API, pandas, numpy, matplotlib, stockstats, OpenAI gym, stable-baselines, tensorflow, and pyfolio. And then we have to download the data.

Pre-processing:

And now, we have done the pre-processing which is a crucial step for training a high-quality machine learning model. In order to convert the data into a model-ready condition, we must check for missing data and do feature engineering. We've also included technical indicators. Various information, such as historical stock prices, current holding shares, technical indicators, and so on, must be considered in real trading.

We will show two trend-following technical indicators in this project: MACD and RSI. A turbulence index should be added. Risk aversion refers to whether or not an investor prefers to keep his or her money safe. It also has an impact on one's trading technique while dealing with varying levels of market volatility. FinRL uses the financial turbulence index, which monitors extreme asset price fluctuation, to manage risk in the worst-case scenario, such as the financial crisis of 2007–2008.

Design Environment:

Then, we have created an environment Considering the stochastic and interactive nature of the automated stock trading tasks, a financial task is modelled as a **Markov Decision Process (MDP)** problem. The training process involves observing stock price change, taking an action, and reward calculations to have the agent adjust its strategy accordingly. By interacting with the environment, the trading agent will derive a trading strategy with maximized rewards as time proceeds. Our trading environments, based on the OpenAI Gym framework, simulate livestock markets with real market data according to the principle of time-driven simulation.

The action space specifies the possible interactions between the agent and the environment. Typically our actions involve three actions: a $\in \{-1, 0, 1\}$, where -1, 0, 1 represents selling, holding, and purchasing one stock.

An action can also be performed on multiple shares. We employ the action space $\{-n, ..., -1, 0, 1, ..., n\}$, where n is the number of shares and -n denotes the number of shares to sell. The continuous action space needs to be normalized to [-1, 1] since the policy is defined on a Gaussian distribution, which needs to be normalized and symmetric, and then we have divided our dataset for train and test purposes.

Models:

- Now, we have implemented DRL algorithms, the implementation of the DRL algorithms is based on **OpenAI Baselines** and **Stable Baselines**. Stable Baselines is a fork of OpenAI Baselines, with major structural refactoring, and code cleanups.
- FinRL library includes fine-tuned standard DRL algorithms, such as DDPG, PPO, and A2C. We also allow users to design their own DRL algorithms by adapting this DRL algorithm and then we train our models with the three models

Trading:

Then, for testing, we used a DDPG model to trade Dow jones 30 stocks. If the current turbulence index is greater than the threshold, we assume that the current market is volatile. After that, we updated our DRL model, which needs to be updated on a regular basis in order to fully utilize the data; ideally, we should retrain our model once a year, quarterly, or monthly. We also need to tweak the parameters along the way; in this notebook, I only tune the parameters once using in-sample data from 2009-01 to 2020-07, so there is some alpha decay as the trade date lengthens.

Backtest Our Strategy:

Backtesting is an important part of analysing a trading strategy's performance. Backtesting software that is automated is preferred because it reduces human error. Backtesting our trading techniques is commonly done with the Quantopian pyfolio program. It is simple to use and consists of several distinct charts that provide a full picture of a trading strategy's performance.

Ensemble strategy:

Then, in order to construct a highly robust trading strategy for our model, we create our own ensemble strategy. Based on the Sharpe ratio, this ensemble strategy method will automatically select the best performing agent from PPO, A2C, and DDPG to trade. The following is a description of our ensemble process:

1).To begin, we employed an n-month growth window to simultaneously train our three agents. In our project, we again train our three agents every three months.

2). After that, we employed a three-month validation rolling window to validate all three agents we used in our project, followed by training to select the best performing agent with the highest Sharpe ratio. In our validation stage, we also use the turbulence index to adjust risk aversion.

3). Finally, following the validation stage, we only select and employ the optimal model with the highest Sharpe ratio to forecast and trade for the next quarter using the models described above.

Start date	End date	Model used	A2C Sharpe	PPO Sharpe	DDPG Sharpe
2019-01-02	2019-04-01	DDPG	-0.230842	-0.0926179	0.320516
2019-04-01	2019-09-24	PPO	-0.0330591	-0.0154111	-0.0335808
2019-06-27	2019-12-20	A2C	0.0826873	-0.316546	-0.203569
2019-09-24	2019-03-24	A2C	0.409842	-0.078961	0.409062
2019-12-20	2019-07-01	DDPG	-0.914022	-0.897626	-0.654031
2019-03-24	2019-10-02	PPO	0.373885	0.408824	0.369866
2019-07-01	2019-10-02	DDPG	-0.16808	-0.154769	0.325383

Table 1: Sharpe Ratios over time

From Table 1, we can see that DDPG has the best validation Sharpe ratio of 0.320 from 2019/1 to 2019/4, so we use DDPG for the next quarter from 2019/4 to 2019/9. PPO has the best validation Sharpe ratio of -0.09 from 2019/4 to 2019/9, so we use PPO to trade for the next quarter from2019/6 to 2019/12. A2C has the best validation Sharpe ratio of 0.08 so we use A2C for the next quarter.

VI. RESULTS

FinRL is used to replicate both stock trading. To boost robustness, the ensemble technique integrates three DRL algorithms (PPO, A2C, and DDPG). FinRL makes implementation simple. In the agent layer, we choose three algorithms (PPO, A2C, DDPG) and an environment with start and end dates in the environment layer. DRL algorithms and data pre-treatment implementations are transparent to users, reducing programming and debugging workloads. As a result, FinRL makes strategy formulation easier, allowing users to concentrate on improving trading performance.

2020/07/01 – 2021/07/01	A2C	DDPG	PPO	Ensemble(A2C,DDPG,PPO)
Initial value	1,000,000$	1,000,000$	1,000,000$	1,000,000$
Annual return	11.4%	10.5%	15.0%	13.162%
Cumulative returns	60.0%	54.8%	73.0%	71.305%
Sharpe ratio	0.655	0.730	0.571	1.61
Max Drawdown	-10.2%	-14.8%	-23.7%	-5.7%

Table 2: Performance of stock trading over the DJIA constituent's stocks using FinRL. The Sharpe ratios of the ensemble strategy and the individual DRL agents exceed those of the DJIA index

fig: 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.

Table 1 and above fig present the backtesting results for Dow 30 member equities, as of July 1, 2020. On a daily basis, the training period runs from 2009/01/01 to 2020/06/30, while the testing period runs from 2020/07/01 to 2021/06/30. In terms of numerous parameters, the performance is consistent with the findings presented in this section, where we show statistics from a recent trading period. The DJIA index shows that the trading period has been bullish, with an annual return of 32.84 percent. The ensemble strategy has a Sharpe ratio of 2.81 and a 52.61 percent yearly return. With a Sharpe ratio of 2.24, PPO with a Sharpe ratio of 2.23, DDPG with a Sharpe ratio of 2.05, and DJIA with a Sharpe ratio of 2.02, it outperforms A2C, PPO, DDPG, and DJIA, respectively.

VII. CONCLUSION

We investigated the possibilities of applying three actor-critic-based algorithms to develop a stock trading strategy in our project: Proximal Policy Optimization (PPO), Advantage Actor-Critic (A2C), and Deep Deterministic Policy Gradient (DDPG) agents. We employ an ensemble technique to automatically select the best-performing agent to trade based on the Sharpe ratio in order to respond to different market scenarios. By balancing risk and return under transaction costs, our ensemble technique outperforms the three individual algorithms, the Dow Jones Industrial Average, and the min-variance portfolio allocation method in terms of the Sharpe ratio.

For future work, It will be interesting to research more complicated models, overcome empirical issues, and deal with largescale data such as S&P 500 constituent stocks in future studies. More features for the state space can be added to our observations, such as an enhanced transaction cost and liquidity model, fundamental analysis indicators, and natural language processing analysis of financial market news. We'd like to use the Sharpe ratio as the reward function directly, however, the agents will need to monitor a lot more historical data, and the state space will grow exponentially.

REFERENCES

[1] Stelios D. Bekiros. 2010. Fuzzy adaptive decision-making for boundedly rational traders in speculative stock markets.

European Journal of Operational Research 202, 1 (April 2010), 285–293.

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

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

[4] Ikhlaas Gurrib. 2018. Performance of the average directional index as a market timing tool for the most actively traded USD based currency pairs. Banks and Bank Systems 13 (08 2018), 58–70.

[5] Lin Chen and Qiang Gao. 2019. Application of deep reinforcement learning on automated stock trading. In the 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS). 29–33

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

[7] Dimitri Bertsekas. 1995. Dynamic programming and optimal control. Vol. 1.

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

[9] Gyeeun Jeong and Ha Kim. 2018. Improving financial trading decisions using deep Q-learning: predicting the number of shares, action strategies, and transfer learning. Expert Systems with Applications 117 (09 2018).

[10] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. 2016. Deep direct reinforcement learning for financial signal representation and trading. IEEE Transactions on Neural Networks and Learning Systems 28 (02 2016), 1–12.

[11] W.F. Sharpe. 1994. The Sharpe ratio. Journal of Portfolio Management (01 1994).

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

[13] Zhengyao Jiang and Jinjun Liang. 2017. Cryptocurrency portfolio management with deep reinforcement learning. In 2017 Intelligent Systems Conference.

[14] John Moody and Matthew Saffell. 2001. Learning to trade via direct reinforcement. IEEE Transactions on Neural Networks 12 (07 2001), 875–89.

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