ZEWAIL CITY FOR SCIENCE AND TECHNOLOGY

COMMUNICATIONS AND INFORMATION ENGINEERING

AUTOMATED MULTI-STOCK TRADING USING REINFORCEMENT LEARNING

A Graduation Project
Submitted in Partial Fulfillment of
B.Sc. Degree Requirements in
Communications and Information Engineering

Prepared by

Amr Abo-Elyazid - 201500542
s-amr.aboelyazid@zewailcity.edu.eg

Amr Abd-albadea - 201500358
s-amrmoohamed@zewailcity.edu.eg

Supervised by

Dr. Mohamed Elshenawy

TABLE OF CONTENT

INTRODUCTION

Financial Markets have been playing an important role in the world economy since its foundation. Countries who have strong and steady markets, reflect directly on the country's economy and welfare of its citizens. Market Structure and mechanisms has been developed in many ways such as the execution process and its auction methodology over the years. Governments focus on providing tools and a suitable environment for investors to take part in their markets in order to achieve prosperity and overall benefit to the country. The world of Financial markets is a complex world where many entities interact with each other, also many assets and financial instruments have been developed over the years. Financial markets participants can be classified into two main categories according to the capital used by the entity. First category is the institutions which have a lot of capital under their management such as private or public funds, hedge funds, commercial banks, investment banks and even Sovereign Funds. The second category is the retail investors, in which they manage only their own capital “Much smaller capitals than institutions”. Each type of participant in the market has his own goals and strategies to execute trying to achieve those goals. These are common goals or reasons to participate in the financial markets, such as preserving capital value against inflation, increasing capitals by making profits and participating in accelerating economic growth.

For the entities Participating in the financial markets, they must be aware of the risk associated with investments, due to many factors such as recession and market fluctuations. The science and tools to help investors make such investment decisions has been developed also over the years. Major two historical areas of research in the markets were fundamental analysis and technical analysis, and both have been developed and took many shapes over the years. Fundamentals analysis is associated with evaluating the historical performance of companies to predict future performance, and hence take investment decisions. Technical Analysis on the other hand, has been associated with the idea that the market repeats

itself. It is simply associated with market psychology, and their interaction with the main economical events through basic human emotions such as fear, optimism, panic and many more.

PROBLEM DEFINITION

In the era of massive advancements of computer technology, it played a very important role in the financial markets. Electronic systems have replaced the traditional ones, in all of its functions such as Trading and matching. Also, institutional orders have been more automated, as they follow programmed guidelines and rules to send millions of orders to market. This accompanied a new wave of high frequency trading “HFT” companies that send millions of orders to markets in low-latency networks and exploit small price changes over those small periods. Meanwhile, in another direction the statistics played a major role in that development. With the wave of big data and advanced computational resources, it gave an opportunity to analyze massive amounts of market data to infer statistical meaning behind price changes. A new wave of quantitative hedge-funds and institutions appeared, in which they research applications of machine learning to make data-driven decisions, and they achieved great success.

There has been new research in the reinforcement learning field and how to apply its power to put agents in markets and interact with it maximizing rewards. Through training models and punishing them towards favorable actions through rewarding those actions, reinforcement learning can be a great way to investigate the complex world of financial markets, finding best actions to do.

OBJECTIVES

We are trying to build a trading bot that invests in multiple assets in the market trying to get maximum rewards. There are a few assumptions

about the model and the environment that need to be considered. Firstly, our historical data forms the world that our agent will investigate, based on that actions on this world, will not change it, so it is a static problem. We simply ignore the interaction of the market to the bot actions. Secondly, we ignore the slippage problem in the real market, as we assume that when the model takes an action, it will be executed without delay, at the price of asset at that moment. Thirdly, after training we give unseen data but have the same past assumptions to the trading agent, and it will trade on its own "exploitation phase" . To evaluate our model performance, we compare it to some famous indexes such as the S & P500 and DOW30 industrial average, to see the overall performance of our agent over some period. Indexes is a financial tool formed to measure the economic performance of many companies all together. For example, s&p 500 measures the overall performance of the top 500 companies in the U.S.A. It is also a great way for a safe investment, as by investing in such an index, you diversify your investments and reduce risks associated with single asset performance.

LITERATURE REVIEW

In Finance, Reinforcement learning has been applied in many approaches. Many researchers tried to address fundamental financial problems like single- asset trading[1], Multi-stock trading[2], orders placement[3],liquidation strategies[4] and portfolio allocation[5]. This even expanded to more complex financial problems like option pricing[6],risk management[7], Volatility scaling of future contracts[6], deep hedging[7] and high frequency trading[8].

The desire to build smart agents for automated-actions in the market has been a challenge in recent years[9]. The most difficult challenge in Financial world problems is that it has very complex representations. Advances in Machine learning alleviate these problems through novel

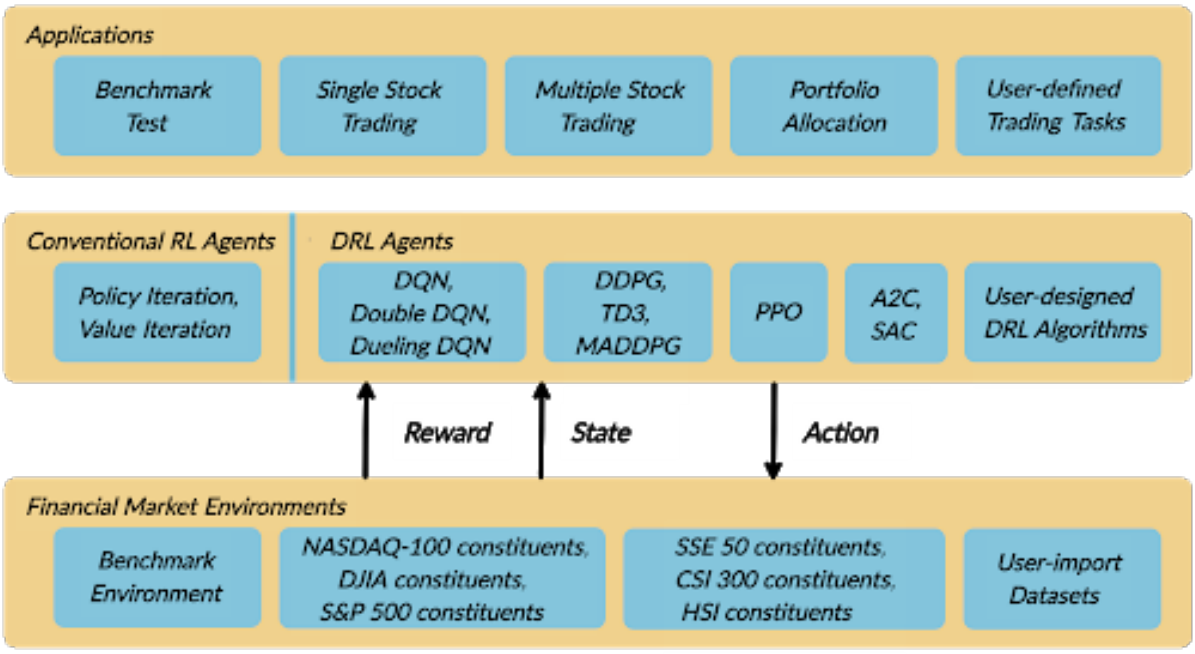
architectures, and have enabled us to build end-to-end pipelines used in quantitative finance and algorithmic trading[10]. This enabled us to deal with such a complex world, and take optimal decisions in a real-time manner.

To be more specific, we will address recent advancements in using reinforcement learning in Trading. According to the nature of the problem, the environment in which agents interact can be classified into discrete or continuous states and action spaces. There are three main approaches for agents to learning which are Critic-only, actor-only and actor-critic. Critic-only agents like Deep Q-learning deal with a discrete actions space, and are used in single stock trading[11][12]. It uses Q-function to learn optimal actions which maximize future rewards, this happens through minimizing error between estimated and target Q-values for a given transition. It cannot be used in large portfolio problems since its nature of dealing with finite state-action spaces. Actor-only agents learn optimal policy rather than Q-values[12]. Policy is the probability distribution "likelihood" of all actions available in a given state. Instead of using the neural network to learn Q-values, it directly learns the policy which is then the strategy of the model[13]. Actor-Critic consider best of them, and have been used widely in Finance lately. It has two networks which are critic network representing Q-value Function, and Actor network representing Policy. First network estimates the function, and the latter one updates the policy according to this function. So, it simultaneously updates weights for both networks, and over time the actor network will take better actions, and the critic network will evaluate them better.

State of the art models do address those approaches like OpenAI Gym[14], OpenAI Baselines[15], Stable Baselines[16], Google Dopamine[17], RLlib[18] and Horizon[19]. For example, OpenAI Gym provides environments for the reinforcement learning agents[14]. OpenAI baseline used those environments to implement modulus for deep-reinforcement learning agents[15], and Stable Baselines forked this into a

more user-friendly environment[16]. Google Dopamine, RLlib and Horizon provided algorithms that are scalable, reusable and pluggable for fast prototyping[17][18][19].

Since Reinforcement learning has become an effective approach, many researchers and practitioners have developed and explored a lot of its potential in Finance. Stock trading is all about making real-time dynamic decisions like placement of orders, deciding amounts of shares to trade off, and deciding entry and exit points in a very complex and stochastic environment[20]. To develop more robust models and strategies, it needed a framework to accelerate this process. This will open the field more for non-practitioners and beginners, as It will save unnecessary and time-headache of developing such a workflow from scratch. Frameworks contain necessary and reproducible modulus such as organization of data, trading environments and managing trading states, results and evaluation processes.I will introduce an open-source library along with two real-world examples using it, pointing out how developing such a framework helped developing robust and realistic strategies. FinRl library is a three-layer architecture library that consists of environments, agents and applications[21]. Each component can be customized and reproduced to serve a specific goal or problem formulation. The library architecture is illustrated in Figure 1.

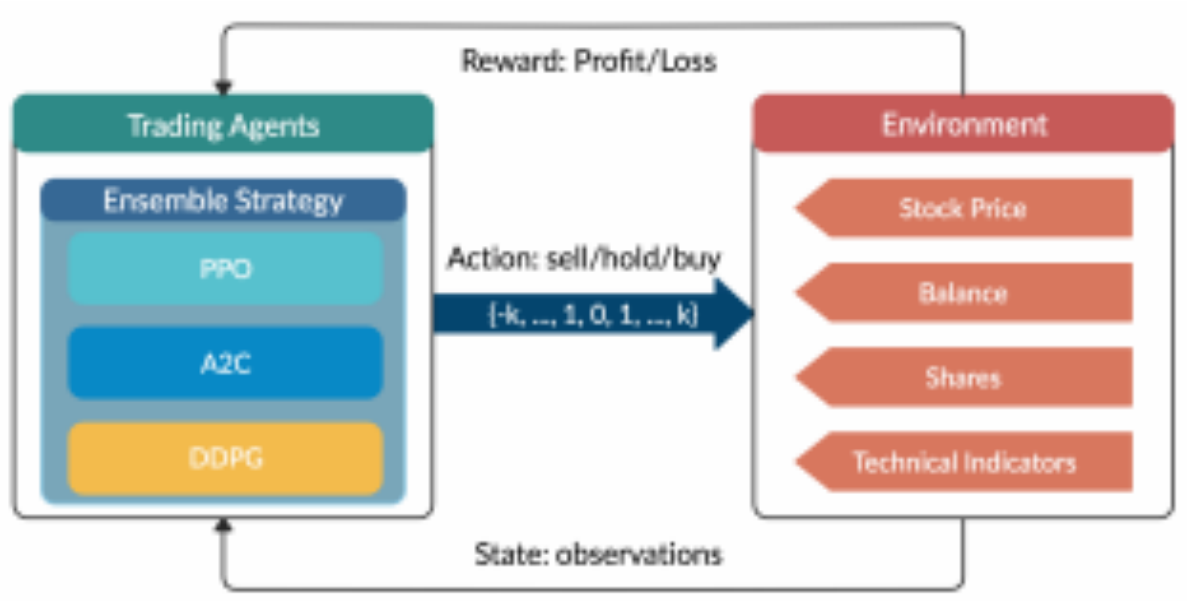


All of them interact with one another through API's, as down layers provide API for upper ones. It provides simplicity, Modularity, Extendibility, Applicability, and also Better Market Environment Modeling[21]. Modeling the Stock trading Environment as a Markov Decision process (MDP) problem , makes the environment so realistic and very close to the stochastic stock market world[21]. They provide realistic implementation of state spaces, action spaces and Reward functions along with a time-driven Trading simulator[21]. They also provide fine-tuned DRL algorithms which are adaptive, along with Evaluation performance metrics and baselines to measure their performance[21]. Different Algorithms are illustrated in Figure 2.

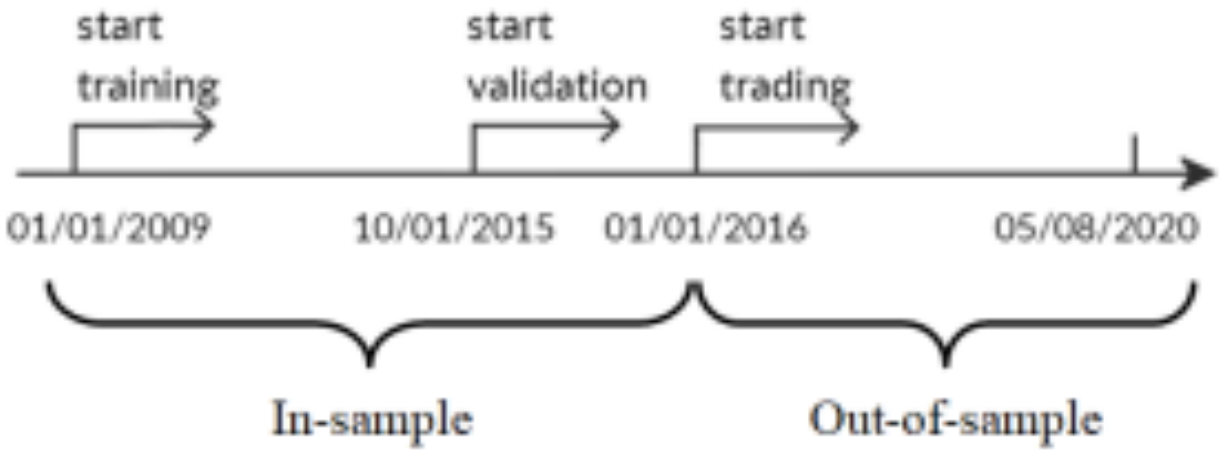
Algorithms	Input	Output	Type	State-action spaces support	Finance use cases support	Features and Improvements	Advantages
DQN	States	Q-value	Value based	Discrete only	Single stock trading	Target network, experience replay	Simple and easy to use
Double DQN	States	Q-value	Value based	Discrete only	Single stock trading	Use two identical neural network models to learn	Reduce overestimations
Dueling DQN	States	Q-value	Value based	Discrete only	Single stock trading	Add a specialized dueling Q head	Better differentiate actions, improves the learning
DDPG	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Being deep Q-learning for continuous action spaces	Better at handling high-dimensional continuous action spaces
A2C	State action pair	Q-value	Actor-critic based	Discrete and continuous	All use cases	Advantage function, parallel gradients updating	Stable, cost-effective, faster and works better with large batch sizes
PPO	State action pair	Q-value	Actor-critic based	Discrete and continuous	All use cases	Clipped surrogate objective function	Improve stability, less variance, simply to implement
SAC	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Entropy regularization, exploration-exploitation trade-off	Improve stability
TD3	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Clipped double Q-Learning, delayed policy update, target policy smoothing.	Improve DDPG performance
MADDPG	State action pair	Q-value	Actor-critic based	Continuous only	Multiple stock trading, portfolio allocation	Handle multi-agent RL problem	Improve stability and performance

I'll go through a practical example using FinRL to Build Multi-stock trading agent. This example is a summary of An ensemble strategy for Automated Stock Trading[22]. They 've used three actor-critic based algorithms which are Advantage Actor Critic (A2C), Proximal Policy Optimization(PPO) and Deep Deterministic Policy Gradient(DDPG) integrating best features of

them into one trading strategy[22]. The main scheme of strategy is illustrated in Figure 3.



To sum up their methodology, they incorporated Markov Decision Process(MDP) along with some constraints like Market liquidity and Risk-aversion[22]. They tested their strategy on the 30 stocks found in the 30 Dow Jones Industrial Index[22]. Feeding basic Stocks data like closing price, volume .. etc along with some Technical Indicator Like Relative Strength Index(RSI), and Moving Average Convergence Divergence(MACD)[22]. They also designed a Memory Management system to employ a load-on demand technique for better memory usage[22]. Main strategy was to train the three models on a given period (3 months) and then select the best one to trade in the next quartile based on its sharpe ratio[22]. Figure 4 shows the period of data they choose, along with training, validation and test set Periods [22] .



The summary of their strategy performance, showing Algorithms sharpe ratio, cumulative return and other metrics such as Annual return over different trading sessions is illustrated then in Table 1 [22].

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

They Found that A2C agent was very good at the bearish market, and more adaptive to risk , as it has the lowest annual volatility and minimum score of Max Drawdown[22]. Also, PPO agent were very good at the Bullish market following upward trends, as it has the highest annual and cumulative return respectively. Summary of Agents performance metrics are shown in Table 2.

(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%

Through combining the three agents through the method illustrated above, they yielded a very good strategy which has higher values of performance metrics than each agent has individually. Figure 5 shows the cumulative return of the algorithms and ensemble strategy backtesting all of them against DOW30 Benchmark[22].

METHODOLOGY

Our project is using the Reinforcement Learning approach to best utilize the powers of learning in a stock market. The idea is to train the model for a given state space (stock market) with certain actions and rewards. The model tries to learn how to trade (actions) to maximize the balance with these actions (rewards). The model is trained with a trial-and-error scheme given a feed of actions and rewards for every state the model lies in so that it has a sense of how to approach the stock market with its varying features and specifications [22].

The model uses set of features that will be described later in the methodology in order to understand the stock market and have enough descriptors for each state and each action given a reward scheme that will be applied to reward and penalize the actions of the model.

ENVIRONMENT SETUP

Our project uses FinRL library for simulating the trading environment, training the model, comparing results with backtesting, configuring and trying models with plenty of schemes.

We begin by installing all the required libraries such as

1. FinRL
2. YFinance for retrieving daily stock market data from yahoo database
3. Stockstate for specific analysis and statistics for the stock market
4. Stable-baselines, GYM, Tensorflow for Reinforcement Learning (RL) algorithms

DATA RETRIEVAL

We retrieve the data from yfinance library which fetches daily data from the yahoo database with the open, low, high, closing prices for each day. It can retrieve historical data for a given number of years which is really useful for training.

STOCK SCREENER

We use a **Stock Screener** for filtering the stocks to have the most suitable stocks for training and developing a trading strategy as the market is very volatile and it is hard to find good quality data to use for training.

The screener uses 8 criteria for selecting the most suitable stocks

1. Choose a current price of the security that is greater than the simple moving averages of 150 and 200-day.
2. Choose the 150-day simple moving average that is greater than the 200-day simple moving average.
3. Choose a stock that has the 200-day simple moving average trending up for at least 1 month.
4. Choose a stock that has the 50-day simple moving average greater than the 150 simple moving average and the 200 simple moving average.
5. Choose the current price that is greater than the 50-day simple moving average.
6. Choose the current price that is at least 30 percent above the 52 week's low.
7. Choose the current price that is within 25 percent of the 52 week's high.
8. Choose a stock that has the IBD Relative-strength Rating — gives an insight of last year's performance in comparison to the overall market — greater than 70.

DATA PREPARATION

After the stock list have been filtered by the stock screener, we retrieve all the historical data for those stocks from yahoo. For example, the S&P 500 has the index of the top 500 companies in the united states. However, after filtering the most suitable stocks, the result can be 50 or 60 indices of companies. We get the historical data of 8-10 years for each company from yahoo to get prepared for the feature engineering to extract and calculate all the relevant features that will be used for training as shown below.

	date	open	high	low	close	volume	tic	day
0	2013-07-11	32.682404	32.889843	32.496422	30.396236	2979837	A	3
1	2013-07-11	66.330002	66.389999	64.769997	57.120152	599200	ALB	3
2	2013-07-11	39.790001	39.810001	39.029999	39.400002	503500	ALGN	3
3	2013-07-11	101.190002	102.370003	100.500000	101.440002	1347600	ALXN	3
4	2013-07-11	45.582565	45.943001	45.297569	42.279068	6745103	APTV	3

FEATURE ENGINEERING

We use some technical indicators as features to help the model understand the data well. The technical indicators are some analysis and statistics in the stock market that gives an indication and a description of the performance of a certain stock market.

We use those list of technical indicators as initial

1. Moving average convergence divergence (MACD), the relationship between two moving averages.

2. Relative Strength Index (RSI), measures the magnitude of recent price changes to evaluate overbought or oversold conditions in the price of a stock.

3. Directional Movement Index (DX), measure the strength of a trend.

4. Displaced moving average (DMA), a moving average that is displaced to better match with high or low prices.

5. Triple Exponential Average (TRIX), the percentage change in a triple exponentially smoothed moving average.

6. Simple moving average (SMA), the average of a selected range of prices, by the number of periods in that range, which aids in determining if an asset price will continue or not.

7. Turbulence, which measures sudden changes in a stock due to an unexpected event, such as market crash.

8. Covariance matrix, which correlates the prices of each stock in a look-back period (1 year for example) with all of the other assets or stocks.

	date	open	high	low	close	volume	tic	day	macd	rsi_30	dx_30	dma	trix	open_30_sma	open_60_sma	turbulence	cov_list
0	2014-07-11	40.293278	40.586552	40.164520	37.981606	1654253	A	4	-0.045162	48.609869	22.544819	0.396257	0.000186	41.428708	40.380901	0.0	[[0.0001909236206161 3792, 3.841703213798981e-0...
1	2014-07-11	71.580002	72.050003	71.389999	64.183411	311900	ALB	4	0.525586	59.345296	14.618137	1.775689	0.123977	71.221666	69.488167	0.0	[[0.0001909236206161 3792, 3.841703213798981e-0...
2	2014-07-11	54.570000	54.799999	53.459999	54.090000	779900	ALGN	4	0.740161	51.674022	0.650143	3.144600	0.291583	53.795000	52.365833	0.0	[[0.0001909236206161 3792, 3.841703213798981e-0...
3	2014-07-11	162.750000	163.750000	160.889999	163.210007	724300	ALXN	4	0.037353	51.733991	4.895405	-0.243198	-0.051408	163.588999	160.261833	0.0	[[0.0001909236206161 3792, 3.841703213798981e-0...
4	2014-07-11	58.122379	58.600186	58.122379	54.438877	1323634	APTV	4	0.301047	55.024341	0.044609	0.879698	0.070940	57.688740	57.141799	0.0	[[0.0001909236206161 3792, 3.841703213798981e-0...

The features are then calculated and processed for our stock list data for each day for each stock in the given time period.

REINFORCEMENT LEARNING

Our approach is called actor-critic approach which simultaneously update the actor scheme — the policy —, and the critic scheme — Q-function—. The critic is used for estimating the Q-function and the actor updates the policy probability distribution and they iterate in a feedback mechanism to update the policy and the Q-function respectively. This approach encourages the actor to learn taking better actions and helps the critic to criticize those actions [22].

The reinforcement learning approach used is modeled with the Markov Decision Process scheme for learning and it is implemented as shown below.

1. State $s = [p, h, b]$ where p is the stock price, h is the shares, b is the balance.

2. Action a which is the set of allowable actions for given number of stocks. The actions are Buy, Hold, and Sell.

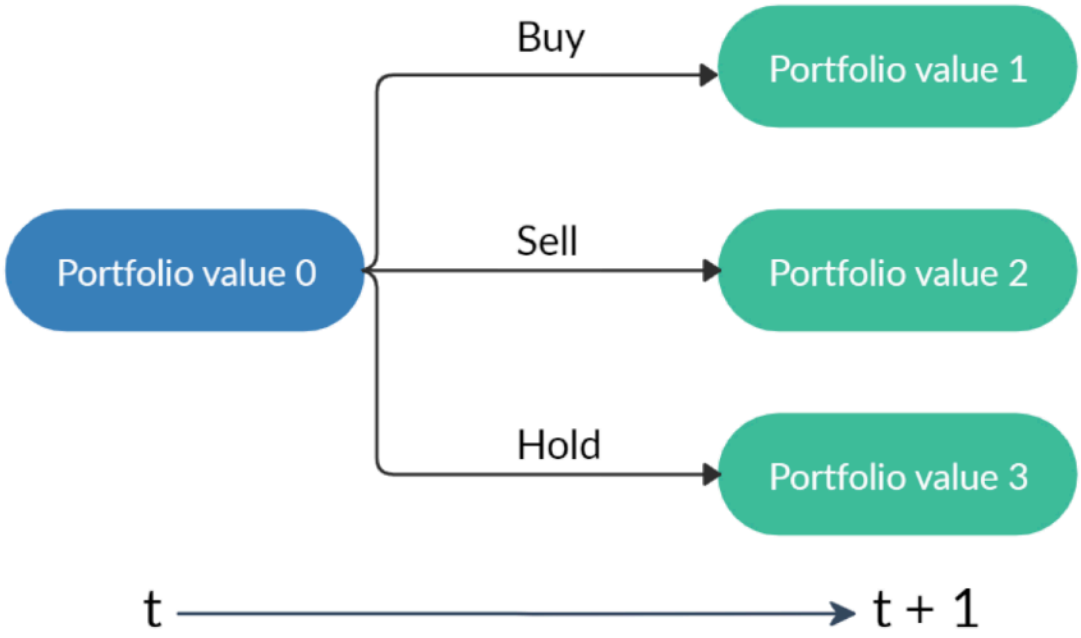
3. Reward $r(s, a, s')$ which is the reward of taking an action a to move from a state s to a new state s'.

4. Policy $\pi(s)$ which denotes the probability distribution of the set of actions a at a state s.

5. Q-function $Q_\pi(s, a)$ which is the reward of taking an action a under a policy π at a state s.

Action scheme

The mentioned actions are Buy, Hold, Sell over D stocks in a portfolio where d is some stock in $d = 1,..,D$. At each time step t, an action is taking to transform the portfolio value from t to $t + 1$



The actions are described as follows:

1. Buying $k[d]$ shares results in $h_{t+1}[d] = h_t[d] + k[d]$.
2. Selling $k[d]$ shares results in $h_{t+1}[d] = h_t[d] - k[d]$.
3. Holding results in $h_{t+1}[d] = h_t[d]$, nothing changes.

There are some constraints that are applied for the agent to suit the stock market nature.

We assume that the market will not be affected by the agent so that we can apply the trading for the closing price.

Also the agent is forced to trade at the objective of a non-negative balance. Meaning that b must remain greater than or equal to zero, $b \geq 0$. So given actions B Buy, H Hold, S Sell, the total actions must not result in a balance that is less than zero so the rule is

$$b_{t+1} = b_t + (P_t^S)^T K_t^S - (P_t^B)^T K_t^B \geq 0$$

We also assume that there are transaction costs C which is the percentages that are deduced for each trade for plenty of reasons such as broker fees, exchange fees, etc. The cost is calculated as follows

$$c_t = p^T k_t \times \ C$$

The last constraint is the Risk-aversion which is avoiding sudden changes due to market crash or any event that greatly changes the prices relative to the overall change of a certain stock. It is accounted for by using a threshold for the turbulence discussed above in the features. $turbulence = (y_t - \mu)\Sigma^-1(y_t - \mu)'$ where y_t is the stock return, μ is the average historical return, and Σ is the covariance of historical return. If the turbulence is greater than a threshold, we stop buying and order the agent to sell all shares until the turbulence is back to normal under the threshold.

Reward Scheme

We define the reward scheme as maximizing the portfolio balance when moving from a state s_t to a new state s_{t+1} when an action a is performed. So the change of portfolio should be maximized with the trading strategy given the set of three actions B Buy, H Hold, and S Sell.

We can write the reward function as $r(s_t, a_t, s_{t+1}) = r_H - r_S + r_B - c_t$ where r_H is the reward of a Holding action, r_S is the reward of a Selling action, r_B is the reward of a Buying action, c_t is the transaction cost moving from time t to time $t + 1$.

We define the rewards for each action as follows

$$r_H = (p_{t+1}^H - p_t^H)^T h_t^H$$

$$r_S = (p_{t+1}^S - p_t^S)^T h_t^S$$

$$r_B = (p_{t+1}^B - p_t^B)^T h_t^B$$

A2C MODEL

The A2C Model (Advantage actor critic) is an algorithm designed to improve the policy-value calculations by using an advantage function. The calculations of the policy and the value functions are handled through deep neural networks as they are perfect universal estimators or approximators to generalize those functions. The idea of the A2C model is to incorporate those two approximators together through another neural network for the advantage function in order to reduce the high variance in the policy function, thus, better results. Incorporating the policy with the value function shows the model how to enhance the action and it won't only judge by the value of the action itself.

The A2C model have two agents that work separately to update the gradients throughout the same environment with different training samples. After the each iteration, the two models hand the average gradients to the advantage network and this network will update the actor and critic altogether.

The objective function of the A2C model is as follows

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

Where $\pi_\theta(a_t \mid s_t)$ is the policy network and the $A(s_t \mid a_t)$ is the advantage network and it is composed of two parts $A(s_t \mid a_t) = Q(s_t \mid a_t) - V(s_t)$

PROJECT EXECUTION

After Applying Mark Minervini stock selection method, we selected 52 stocks out of stocks found in the S & P 500 index. This method provides us with a variety of stocks with good investment opportunities, and also different scenarios for the model, to learn to generalize. Because of the lack of availability of open-source data, we make sure that all selected stocks have enough data equal to the period of training chosen. So for example, We choose to download historical data of the last 10 years of our stocks. Since, not all of our stocks have enough historical data, we ‘ve done an additional filtration process which reduces the number of stocks to 45. The historical data contains basic day data for stocks like open price, close price, high of the day, low of the day and finally total volume traded during the day. We then add our chosen technical indicators [MACD, RSI, DX, DMA, TRIX, SMA] and turbulence index to our data. And we have added a covariance matrix that captures relations with each individual stock, that gives a mathematical modeling of how stocks relate to each other. Now after our data and features are ready to train, we have another design consideration to get the best results. I will discuss now how did we choose our parameters in detail

1- Dataset size: it seems that as increasing our dataset size increases state-space dramatically, also it gives the model more data to generalize its

learning on. Experimenting different data sizes, we have chosen to have historical data of the past 10 years. As we increase data size from 4 years to 10 years, the model gives better results (peak was at 7 years), but the number of stocks decreases because of a lack of data problem. Trying data of more than 10 years, decreases our stock list dramatically, as most of the chosen stocks don't have data to the date described. So 7 years was just suitable to work on.

2- Hmax: the hmax parameters determine the maximum number of shares to buy-sell or hold in each individual stock. We tried different values starting from 100 shares and then increased that number to see the effect on our results. We have seen that increasing hmax does have better results on performance overall , and cumulative return becomes better. But it does decrease sharpe ratio, as trading more stocks relative to an unchanged capital, do associate with more risk taken. Hmax with a value of 1000 shares was just suitable, and balanced between sharpe ratio values and returns.

3- number of steps of the model: choosing n-steps to be 1 which mean 1 day relative to our daily data we use, doesn’t capture a lot of changes since not all stocks have big changes in only 1 day timeframe. Also, choosing it to be more bigger like 7 or 10 days, doesn’t capture great opportunities of price swapping. So, it was suitable to choose it in medium range value like 3 or 5 days, and we have stuck with 5 days.

We have trained our model with 4 different actor-critic Reinforcement models , Table 1 shows the comparison between important resulting metrics of the different models. All models achieved more than 100% in annual return, and cumulative return > 130% during the Trading period of about 22 months. Trading period happened to include the last Coronavirus pandemic which affected many financial markets globally. Despite that, models dealt very well with the bearish market with max drawdown of about -0.13% , thanks to the turbulence index incorporated in our loss function. The sharpe ratios of all models are above 3 which are

considered fantastic sharpe ratios. Also, Annual volatility didn’t exceed 25% which is considered a good score that indicates that our strategy didn’t change our portfolio dramatically.

Model	A2C	PPO	TD3	SAC
Annual return	1.058821	1.074517	1.014310	1.057710
Cumulative returns	1.382808	1.404666	1.321004	1.381261
Annual Volatility	0.245292	0.241528	0.234066	0.251107
Sharpe ratio	3.070505	3.146100	3.112963	3.003008
Stability	0.977314	0.978894	0.980532	0.973708
Max Drawdown	-0.136333	-0.130428	-0.128370	-0.144274

We will then backtest our A2C Model against the S & P 500 index, since we already chose our stocks among stocks found in this index.

Figure 1 shows A2C cumulative returns in green, with respect to S&p 500 cumulative returns in the same period. the model already over-performed

the index during all period achieving 140% in cumulative returns with respect to only 40% of cumulative return of the index.

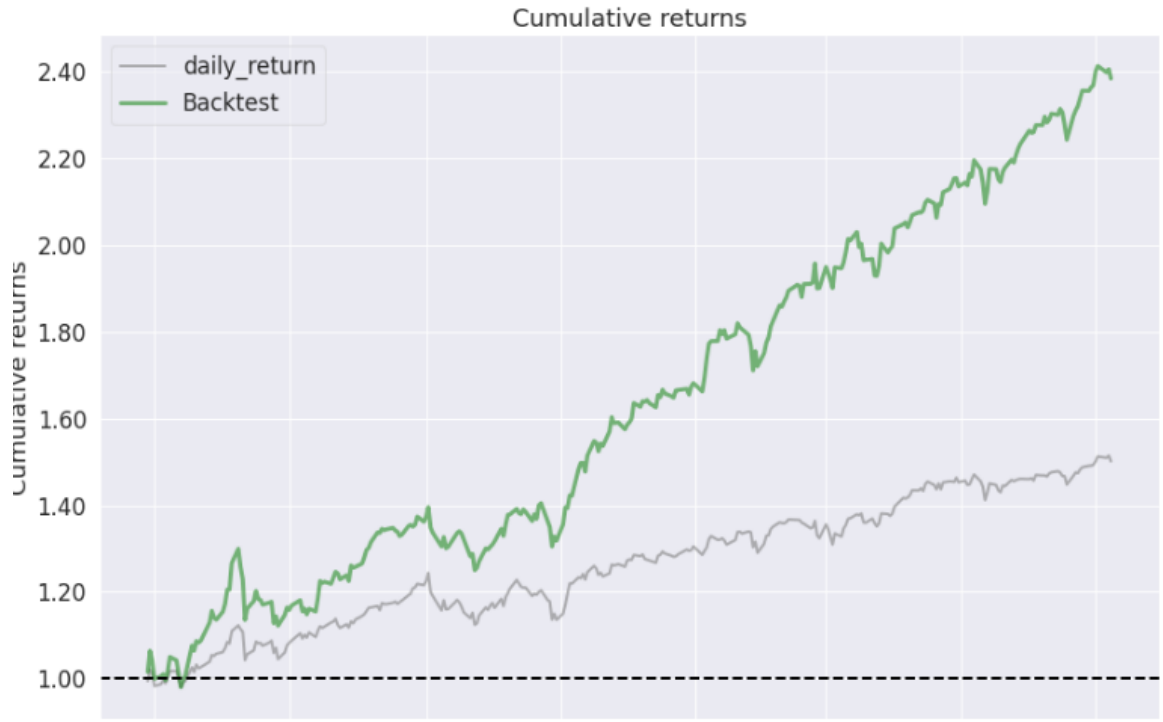


Figure 2 shows rolling volatility in 6 months period of our model compared to benchmark volatility. It shows that our model over-performed benchmark volatility.

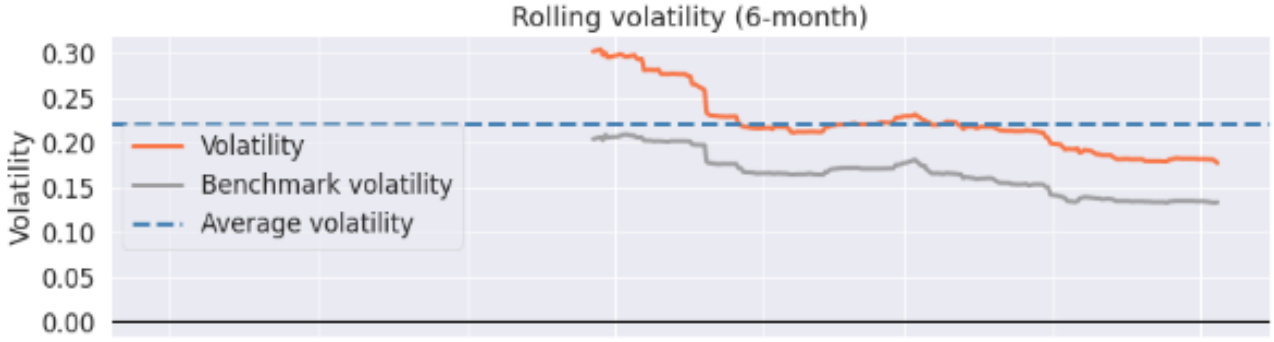


Figure 3 shows how this high volatility score helped the ease monetization of assets in portfolio, and over-performed benchmark s&p 500. Ease of monetization helps the model to exploit more trading opportunities and ensure flexibility to different market scenarios.

Figure 4 shows rolling sharpe ratio in 6 months period. It shows that our model already has a minimum sharpe ratio of 2 and maximum of 5 with mean above 3. It showed that in even worst scenarios, the model already did achieve a good sharpe ratio of 2. While in the best case scenarios, the model achieved a very high sharpe ratio of 5.

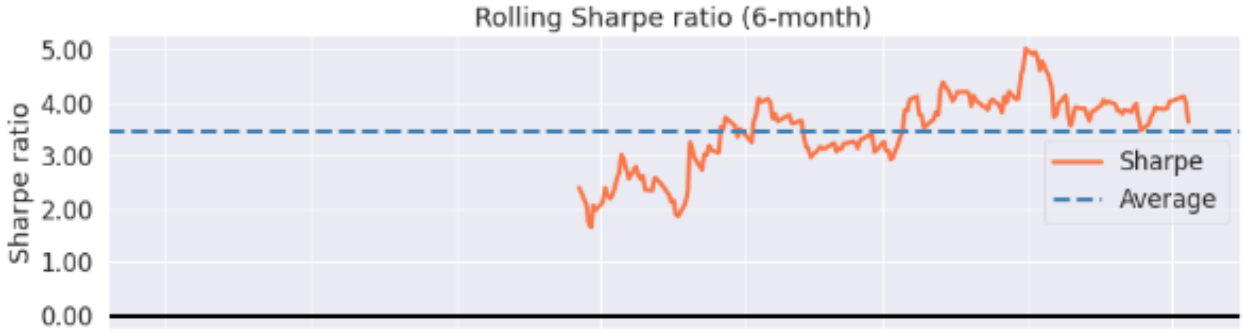


Figure 5 shows top 5 drawdown periods along with their drawdown scores, which means periods when the model didn't perfor, well compared to its overall performance. Worst drawdown was about -14% and least significant one was near 1%.

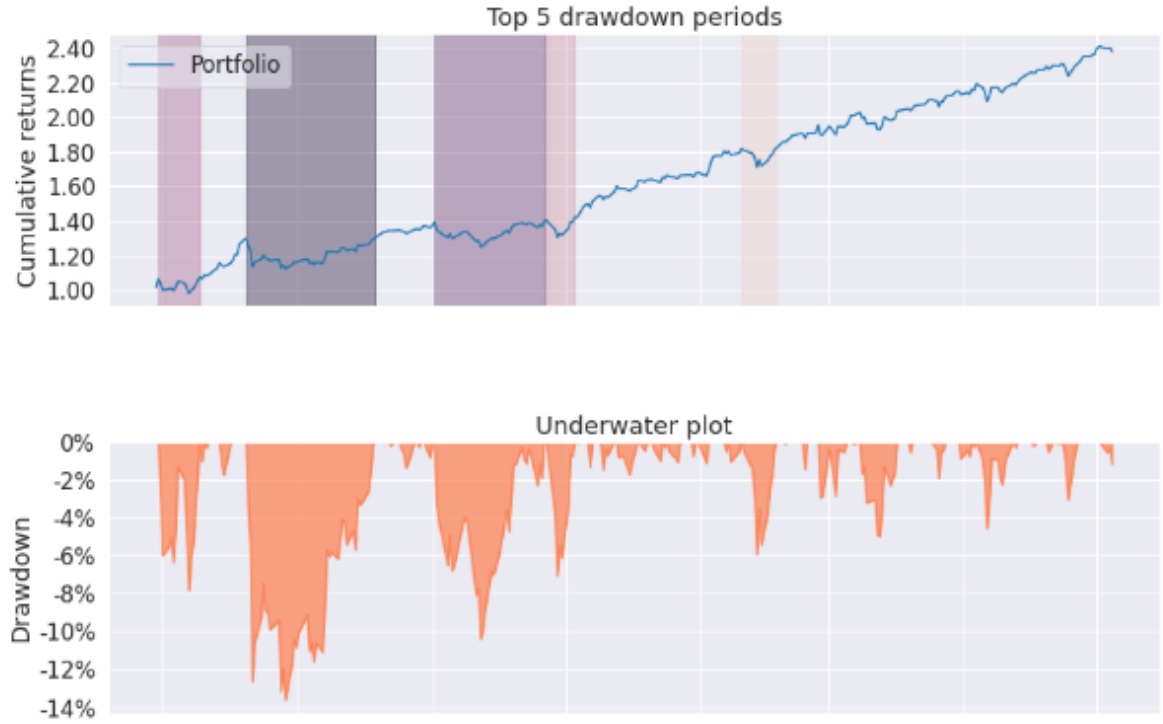
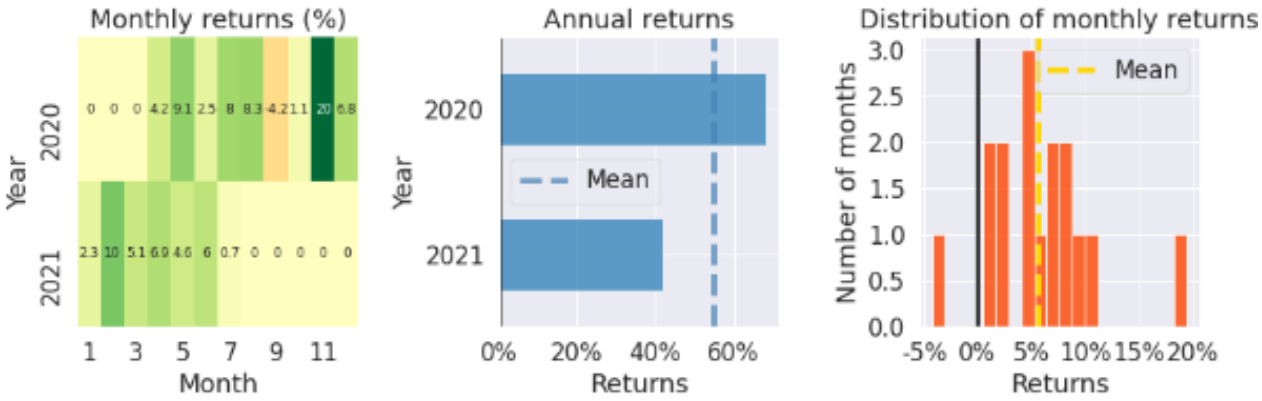


Figure 6 shows the distribution of monthly and annual returns of our model, as it shows that 2020 year contributed to more than 60% of our return. Also, shows that in the same year, the best monthly return was in December, as it contributed to 20% of our total return. The mean monthly return was about 5% , which is a good investment return in only 30 days.



COST ANALYSIS

To be able to Trade in many stocks as in our case, we need an adequate capital. We used 1 Million dollars as an initial amount of our portfolio, and it can be scaled to a much higher number in which it returns to a bigger cumulative returns of our portfolio. The strategy can be scaled to include intraday trading instead of using daily data such as in our case. Another important point to be discussed is that stocks chosen must have good liquidity to ensure orders execution. For the product to be scalable many requirements must be met achieving that purpose. Firstly, a good low fault-tolerance infrastructure to retrieve prices on a low-latency basis, also to send orders to different markets. Secondly, the proposed models must be backtesting first on simulated environments before putting it into production and executing real-life trades. The development of HFT systems and any other type of Automated Algorithms, has led the algorithms to learn spoofing. Spoofing in Financial markets is to send false orders to the market without the intention of actually executing those

orders, in order to manipulate prices moving direction for the spoofer's interest. Sending such big orders to the market, specially in low-volatility stocks can lead to huge price movements. This has a very bad economic impact and is considered a crime in the financial market, so we can ensure that our models will not reach this state through putting constraints on the orders they send, to ensure that all orders must be executed.

CONCLUSION AND FUTURE WORK

We have built a successful trading bot that trades in many stocks simultaneously. We built our stock screener to find best opportunities in markets , using the brilliant way of the legend Mark Minervini. We 've used FinRl open-source library to train-evaluate and backtest our trading strategy. We have adequately designed the environment variables to be compatible with our investing purposes. With proper choosing of stocks, and different environment variables and model hyper-parameters, we achieved a legendary sharpe ratio and annual return scores. For Hedge funds and Investment institutions, to achieve 20-30 % is considered a huge profit for them. We already outperformed those scores with an annual return of nearly 100%, which is nearly 4 times bigger than average successful institutions. The results are very impressive and show the massive opportunities in applying reinforcement learning in even more complex Financial problems. The limitation of our model can be scaling both capital to a much bigger numbers, also the size of our orders to suit capital then. We cannot directly send our huge orders into bulks like in our case, as it will drive the market against our interest. A good order placement strategy should be made then to execute trades at a good average price that serve our strategy. Also, with much larger capital, investment should be expanded to include many markets and different assets, the only thing to be considered then is making sure that those assets have adequate liquidity and high volume of shares.

Future work may include subscribing to an intraday data providing, to assess how our model will behave with a smaller time frame. Also assemble many different models, to improve the overall efficiency of trading strategy. We can also put our trading agent into trading in the real-market , with the help of Brokers API's like Amiterade and Interactive brokers. We can then tackle another area in the financial market, which is the limit order book. We can research how reinforcement learning can be used to explore the world of chaotic and stochastic order book data, trying to find best orders to send in order to maximize profit.

REFERENCES

[1] Taghian, M., Asadi, A., & Safabakhsh, R. (2020, October 27). *Learning Financial Asset-Specific Trading Rules via Deep Reinforcement Learning*. arXiv.org. https://arxiv.org/abs/2010.14194._

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

[3] Schnaubelt, Matthias. (2020). Deep reinforcement learning for the optimal placement of cryptocurrency limit orders. 10.13140/RG.2.2.14315.57127.

[4] 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.

[5] Benhamou, Eric and Saltiel, David and Ohana, Jean-Jacques and Atif, Jamal and Laraki, Rida, Deep Reinforcement Learning (DRL) for Portfolio Allocation (2020). ECML PKDD Demo track 2020, Available at SSRN: https://ssrn.com/abstract=3871071 or http://dx.doi.org/10.2139/ssrn.3871071

[6] John Hull et al. Options, futures and other derivatives/John C. Hull. Upper Saddle River, NJ: Prentice Hall, 2009

[7] Hans Buehler, Lukas Gonon, Josef Teichmann, Ben Wood, Baranidharan Mohan, and Jonathan Kochems. Deep hedging: Hedging derivatives under generic market frictions using reinforcement learning. Swiss Finance Institute Research Paper, 2019.

[8] Prakhar Ganesh and Puneet Rakheja. Deep reinforcement learning in high frequency trading. ArXiv, abs/1809.01506, 2018.

[9] Jay Cao, J. Chen, John C. Hull, and Zissis Poulos. Deep hedging of derivatives using reinforcement learning. Risk Management & Analysis in Financial Institutions eJournal, 2019.

[10]David G Luenberger et al. Investment science. OUP Catalogue, 1997.

[11] A. Cartea, S. Jaimungal, and J. Penalva, Algorithmic and high-frequency trading. Cambridge University Press, 2015.

[12] 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.

[13] 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.

[14] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. arXiv preprint arXiv:1606.01540, 2016.

[15] 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.

[16] 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

[17] Pablo Samuel Castro, Subhodeep Moitra, Carles Gelada, Saurabh Kumar, and Marc G. Bellemare. Dopamine: A research framework for deep reinforcement learning. http://arxiv.org/abs/1812.06110, 2018.

[18] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph E. Gonzalez, Michael I. Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In International Conference on Machine Learning (ICML), 2018.

[19] Jason Gauci, Edoardo Conti, Yitao Liang, Kittipat Virochsiri, Zhengxing Chen, Yuchen He, Zachary Kaden, Vivek Narayanan, and Xiaohui Ye. Horizon: Facebook's open source applied reinforcement learning platform. arXiv preprint arXiv:1811.00260, 2018.

[20] 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

[21] Liu, X. Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., & Wang, C. D. (2020). FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance. arXiv preprint arXiv:2011.09607.

[22] Yang, H., Liu, X. Y., Zhong, S., & Walid, A. (2020). Deep reinforcement learning for automated stock trading: An ensemble strategy. Available at SSRN.