FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance

Xiao-Yang Liu[1*], **Hongyang Yang**[2,3*], **Qian Chen**[4,2], **Runjia Zhang**[3],
Liuqing Yang[3], **Bowen Xiao**[5], **Christina Dan Wang**[6]
[1]Electrical Engineering, [2]Department of Statistics, [3]Computer Science, Columbia University,
[3]AI4Finance LLC., USA, [4]Ion Media Networks, USA,
[5]Department of Computing, Imperial College, [6]New York University (Shanghai)
Emails: {XL2427, HY2500, QC2231, LY2335}@columbia.edu,
info@ai4finance.net, bowen.xiao20@imperial.ac.uk, christina.wang@nyu.edu

Abstract

As deep reinforcement learning (DRL) has been recognized as an effective approach in quantitative finance, getting hands-on experiences is attractive to beginners. However, to train a practical DRL trading agent that *decides where to trade, at what price, and what quantity* involves error-prone and arduous development and debugging. In this paper, we introduce a DRL library *FinRL* that facilitates beginners to expose themselves to quantitative finance and to develop their own stock trading strategies. Along with easily-reproducible tutorials, FinRL library allows users to streamline their own developments and to compare with existing schemes easily. Within FinRL, virtual environments are configured with stock market datasets, trading agents are trained with neural networks, and extensive backtesting is analyzed via trading performance. Moreover, it incorporates important trading constraints such as transaction cost, market liquidity and the investor's degree of risk-aversion. FinRL is featured with *completeness, hands-on tutorial and reproducibility* that favors beginners: (i) at multiple levels of time granularity, FinRL simulates trading environments across various stock markets, including NASDAQ-100, DJIA, S&P 500, HSI, SSE 50, and CSI 300; (ii) organized in a layered architecture with modular structure, FinRL provides fine-tuned state-of-the-art DRL algorithms (DQN, DDPG, PPO, SAC, A2C, TD3, etc.), commonly-used reward functions and standard evaluation baselines to alleviate the debugging workloads and promote the reproducibility, and (iii) being highly extendable, FinRL reserves a complete set of user-import interfaces. Furthermore, we incorporated three application demonstrations, namely single stock trading, multiple stock trading, and portfolio allocation. The FinRL library will be available on Github at link https://github.com/AI4Finance-LLC/FinRL-Library.

1 Introduction

Deep reinforcement learning (DRL), which balances exploration (of uncharted territory) and exploitation (of current knowledge), has been recognized as an advantageous approach for automated stock trading. DRL framework is powerful in solving dynamic decision making problems by learning through interaction with an unknown environment, and thus providing two major advantages - portfolio scalability and market model independence [5]. In quantitative finance, stock trading is essentially making dynamic decisions, namely *to decide where to trade, at what price, and what*

*Equal contribution.

Deep Reinforcement Learning Workshop, 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

quantity, over a highly stochastic and complex stock market. As a result, DRL provides useful toolkits for stock trading [21, 44, 48, 45, 10, 8, 26]. Taking many complex financial factors into account, DRL trading agents build a multi-factor model and provide algorithmic trading strategies, which are difficult for human traders [3, 47, 24, 22].

Preceding DRL, conventional reinforcement learning (RL) [43] has been applied to complex financial problems [31], including option pricing, portfolio optimization and risk management. Moody and Saffell [36] utilized policy search and direct RL for stock trading. Deng *et al.* [12] showed that applying deep neural networks profits more. There are industry practitioners who have explored trading strategies fueled by DRL, since deep neural networks are significantly powerful at approximating the expected return at a state with a certain action. With the development of more robust models and strategies, general machine learning approaches and DRL methods in specific are becoming more reliable. For example, DRL has been implemented on sentimental analysis on portfolio allocation [27, 22] and liquidation strategy analysis [2], showing the potential of DRL on various financial tasks.

However, to implement a DRL or RL driven trading strategy is nowhere near as easy. The development and debugging processes are arduous and error-prone. Training environments, managing intermediate trading states, organizing training-related data and standardizing outputs for evaluation metrics - these steps are standard in implementation yet time-consuming especially for beginners. Therefore, we come up with a beginner-friendly library with fine-tuned standard DRL algorithms. It has been developed under three primary principles:

- **Completeness**. Our library shall cover components of the DRL framework completely, which is a fundamental requirement;
- **Hands-on tutorials**. We aim for a library that is friendly to beginners. Tutorials with detailed walk-through will help users to explore the functionalities of our library;
- **Reproducibility**. Our library shall guarantee reproducibility to ensure the transparency and also provide users with confidence in what they have done.

In this paper, we present a three-layered *FinRL* library that streamlines the development stock trading strategies. FinRL provides common building blocks that allow strategy builders to configure stock market datasets as virtual environments, to train deep neural networks as trading agents, to analyze trading performance via extensive backtesting, and to incorporate important market frictions. On the lowest level is environment, which simulates the financial market environment using actual historical data from six major indices with various environment attributes such as closing price, shares, trading volume, technical indicators etc. In the middle is the agent layer that provides fine-tuned standard DRL algorithms (DQN [29][34], DDPG [29], Adaptive DDPG [27], Multi-Agent DDPG [30], PPO [40], SAC [18], A2C [33] and TD3 [11], etc.), commonly used reward functions and standard evaluation baselines to alleviate the debugging workloads and promote the reproducibility. The agent interacts with the environment through properly defined reward functions on the state space and action space. The top layer includes applications in automated stock trading, where we demonstrate three use cases, namely single stock trading, multiple stock trading and portfolio allocation.

The contributions of this paper are summarized as follows:

- FinRL is an open source library specifically designed and implemented for quantitative finance. Trading environments incorporating market frictions are used and provided.
- Trading tasks accompanied by hands-on tutorials with built-in DRL agents are available in a beginner-friendly and reproducible fashion using Jupyter notebook. Customization of trading time steps is feasible.
- FinRL has good scalability, with a broad range of fine-tuned state-of-the-art DRL algorithms. Adjusting the implementations to the rapid changing stock market is well supported.
- Typical use cases are selected and used to establish a benchmark for the quantitative finance community. Standard backtesting and evaluation metrics are also provided for easy and effective performance evaluation.

The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 presents FinRL Library. Section 4 provides evaluation support for analyzing stock trading performance. We conclude our work in Section 5.

2 Related Works

We review related works on relevant open source libraries and existing applications of DRL in finance.

2.1 State-of-the-Art Algorithms

Recent works can be categorized into three approaches: value based algorithm, policy based algorithm, and actor-critic based algorithm. FinRL has consolidated and elaborated upon those algorithms to build financial DRL models. There are a number of machine learning libraries that share similar features as our FinRL library.

- **OpenAI Gym** [4] is a popular open source library that provides a standardized set of task environments. OpenAI Baselines [13] implements high quality deep reinforcement learning algorithms using gym environments. Stable Baselines [19] is a fork of OpenAI Baselines with code cleanup and user-friendly examples.
- **Google Dopamine** [7] is a research framework for fast prototyping of reinforcement learning algorithms. It features plugability and reusability.
- **RLlib** [28] provides high scalability with reinforcement learning algorithms. It has modular framework and is very well maintained.
- **Horizon** [17] is a DL-focused framework dominated by PyTorch, whose main use case is to train RL models in the batch setting.

2.2 DRL in Finance

Recent works show that DRL has many applications in quantitative finance [14].

Stock trading is usually considered as one of the most challenging applications due to its noisy and volatile features. Many researchers have explored various approaches using DRL [38, 37, 10, 9, 48, 16]. Volatility scaling can be incorporated with DRL to trade futures contracts [48]. By adding a market volatility term to reward functions, we can scale up the trade shares with low volatility, and vice versa. News headline sentiments and knowledge graphs can also be combined with the time series stock data to train an optimal policy using DRL [37]. High frequency trading with DRL is also a hot topic [16]. Deep Hedging [5, 6] represents hedging strategies with neural networks learned by modern DRL policy search. This application has shown two key advantages of the DRL approach in quantitative finance, which are scalability and model independent. It uses DRL to manage the risk of liquid derivatives, which indicates further extension of our library into other asset classes and topics.

3 The Proposed FinRL Library

FinRL library consists of three layers: environments, agents and applications. We first describe the overall architecture, and then present each layer.

3.1 Architecture of the FinRL Library

The architecture of the FinRL library is shown in Fig. 1, and its features are summarized as follows:

- **Three-layer architecture**: The three layers of FinRL library are stock market environment, DRL trading agent, and stock trading applications. The agent layer interacts with the environment layer in an exploration-exploitation manner, whether to repeat prior working-well decisions or to make new actions hoping to get greater rewards. The lower layer provides APIs for the upper layer, making the lower layer transparent to the upper layer.

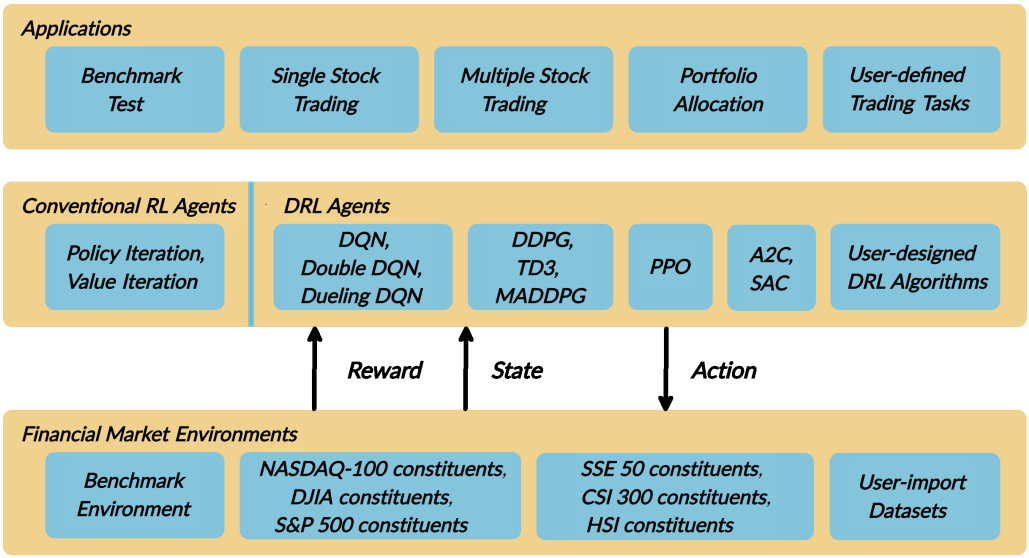

Figure 1: An overview of our FinRL library. It consists of three layers: application layer, DRL agent layer, and the finance market environment layer.

- **Modularity**: Each layer includes several modules and each module defines a separate function. One can select certain modules from any layer to implement his/her stock trading task. Furthermore, updating existing modules is possible.
- **Simplicity, Applicability and Extendibility**: Specifically designed for automated stock trading, FinRL presents DRL algorithms as modules. In this way, FinRL is made accessible yet not demanding. FinRL provides three trading tasks as use cases that can be easily reproduced. Each layer includes reserved interfaces that allow users to develop new modules.
- **Better Market Environment Modeling**: We build a trading simulator that replicates live stock market and provides backtesting support that incorporates important market frictions such as transaction cost, market liquidity and the investor's degree of risk-aversion. All of those are crucial among key determinants of net returns.

3.2 Environment: Time-driven Trading Simulator

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's calculation to have the agent adjusting its strategy accordingly. By interacting with the environment, the trading agent will derive a trading strategy with the maximized rewards as time proceeds.

Our trading environments, based on OpenAI Gym framework, simulate live stock markets with real market data according to the principle of time-driven simulation [4]. FinRL library strives to provide trading environments constructed by six datasets across five stock exchanges.

3.2.1 State Space, Action Space, and Reward Function

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

State space $\mathcal{S}$. 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 $b_t \in \mathbb{R}_+$: the amount of money left in the account at the current time step t.
- Shares own $\boldsymbol{h}_t \in \mathbb{Z}_+^n$: current shares for each stock, n represents the number of stocks.
- Closing price $\boldsymbol{p}_t \in \mathbb{R}_+^n$: one of the most commonly used feature.
- Opening/high/low prices $\boldsymbol{o}_t, \boldsymbol{h}_t, \boldsymbol{l}_t \in \mathbb{R}_+^n$: used to track stock price changes.
- Trading volume $\boldsymbol{v}_t \in \mathbb{R}_+^n$: total quantity of shares traded during a trading slot.
- Technical indicators: Moving Average Convergence Divergence (MACD) $\boldsymbol{M}_t \in \mathbb{R}^n$ and Relative Strength Index (RSI) $\boldsymbol{R}_t \in \mathbb{R}_+^n$, etc.
- Multiple-level of granularity: we allow data frequency of the above features to be daily, hourly or on a minute basis.

Action space $\mathcal{A}$. The action space describes the allowed actions that the agent interacts with the environment. Normally, $a \in \mathcal{A}$ includes three actions: $a \in \{-1, 0, 1\}$, where $-1, 0, 1$ represent selling, holding, and buying one stock. Also, an action can be carried upon multiple shares. We use an action space $\{-k, ..., -1, 0, 1, ..., k\}$, where k denotes the number of shares. For example, "Buy 10 shares of AAPL" or "Sell 10 shares of AAPL" are 10 or -10, respectively.

Reward function $r(s, a, s')$ is the incentive mechanism for an agent to learn a better action. There are many forms of reward functions. We provide commonly used ones [14] as follows:

- The change of the portfolio value when action a is taken at state s and arriving at new state s' [12, 44, 10, 37, 45], i.e., $r(s, a, s') = v' - v$, where v' and v represent the portfolio values at state s' and s, respectively.
- The portfolio log return when action a is taken at state s and arriving at new state s' [20], i.e., $r(s, a, s') = \log(\frac{v'}{v})$.
- The Sharpe ratio for periods $t = \{1, ..., T\}$ [23, 35], i.e., $S_T = \frac{\text{mean}(R_t)}{\text{std}(R_t)}$, where $R_t = v_t - v_{t-1}$.
- FinRL also supports user defined reward functions to include risk factor or transaction cost term such as in [12, 48, 5]

3.2.2 Standard and User Import Datasets

The application of DRL in finance is different from that in other fields, such as playing chess and card games [42, 46]; the latter inherently have clearly defined rules for environments. Various finance markets require different DRL algorithms to get the most appropriate automated trading agent. Realizing that setting up training environment is a time-consuming and laborious work, FinRL provides six environments based on representative listings, including NASDAQ-100, DJIA, S&P 500, SSE 50, CSI 300, and HSI, plus one user-defined environment. With those efforts, this library frees users from tedious and time-consuming data pre-processing workload.

We are well aware that users may want to train trading agents on their own data sets. FinRL library provides convenient support to user imported data to adjust the granularity of time steps. We specify the format of the data for each of the use cases. Users only need to pre-process their data sets according to our data format instructions.

3.3 Deep Reinforcement Learning Agents

FinRL library includes fine-tuned standard DRL algorithms, namely, DQN [29][34], DDPG [29], Multi-Agent DDPG [30], PPO [40], SAC [18], A2C [33] and TD3 [11]. We also allow users to design their own DRL algorithms by adapting these DRL algorithms, e.g., Adaptive DDPG [27], or employing ensemble methods [45]. The comparison of DRL algorithms is shown in Fig. 2.

The implementation of the DRL algorithms are based on OpenAI Baselines [13] and Stable Baselines [19].

4 Evaluation of Trading Performance

Standard metrics and baseline trading strategies are provided to support trading performance analysis. FinRL library follows a training-validation-testing flow to design a trading strategy.

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

Figure 2: Comparison of DRL algorithms.

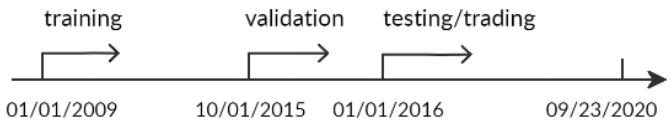

Figure 3: Data splitting.

4.1 Standard Performance Metrics

FinRL provides five evaluation metrics to help users evaluate the stock trading performance directly, which are final portfolio value, annualized return, annualized standard deviation, maximum drawdown ratio, and Sharpe ratio.

4.2 Baseline Trading Strategies

Baseline trading strategies should be well-chosen and follow industrial standards. The strategies will be universal to measure, standard to compare with, and easy to implement. In FinRL library, traditional trading strategies serve as the baseline for comparing with DRL strategies. Investors usually have two objectives for their decisions: the highest possible profits and the lowest possible risks of uncertainty [41]. FinRL uses five conventional strategies, namely passive buy-and-hold trading strategy [32], mean-variance strategy [1], and min-variance strategy [1], momentum trading strategy [15], and equal-weighted strategy to address these two mutually limiting objectives and the industrial standards.

4.3 Training-Validation-Testing Flow

With our use cases as instances, the stock market data are divided into three phases in Fig. 3. Training dataset is the sample of data to fit the DRL model. The model sees and learns from the training dataset. Validation dataset is used for parameter tuning and to avoid overfitting. Testing (trading) dataset is the sample of data to provide an unbiased evaluation of a fine-tuned model. Rolling window is usually associated with the training-validation-testing flow in stock trading because investors and portfolio managers may need to rebalance the portfolio and retrain the model periodically. FinRL provides flexible rolling window selection such as on a daily basis, monthly, quarterly, yearly or by user specified.

Figure 4: Performance of single stock trading using PPO in the FinRL library.

Figure 5: Performance of multiple stock trading and portfolio allocation using the FinRL library.

4.4 Backtesting with Trading Constraints

In order to better simulate practical trading, we incorporate trading constraints, risk-aversion and automated backtesting tools.

Automated Backtesting. Backtesting plays a key role in performance evaluation. Automated backtesting tool is preferable because it reduces the human error. In FinRL library, we use the Quantopian pyfolio package [39] to backtest our trading strategies. This package is easy to use and consists of various individual plots that provide a comprehensive image of the performance of a trading strategy.

Incorporating Trading Constraints. Transaction costs incur when executing a trade. There are many types of transaction costs, such as broker commissions and SEC fee. We allow users to treat transaction costs as a parameter in our environments:

- Flat fee: a fixed dollar amount per trade regardless of how many shares traded.
- Per share percentage: a per share rate for every share traded, for example, 1/1000 or 2/1000 are the most commonly used transaction cost rate for each trade.

Moreover, we need to consider market liquidity for stock trading, such as bid-ask spread. Bid-ask spread is the difference between the prices quoted for an immediate selling action and an immediate buying action for stocks. In our environments, we can add the bid-ask spread as a parameter to the stock closing price to simulate real world trading experience.

Risk-aversion. Risk-aversion reflects whether an investor will choose to preserve the capital. It also influences one's trading strategy when facing different market volatility level.

To control the risk in a worst-case scenario, such as financial crisis of 2007–2008, FinRL employs the financial turbulence index $turbulence_t$ that measures extreme asset price fluctuation [25]:

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

2019/01/01-2020/09/23	SPY	QQQ	GOOGL	AMZN	AAPL	MSFT	S&P 500
Initial value	100,000	100,000	100,000	100,000	100,000	100,000	100,000
Final value	127,044	163,647	174,825	192,031	173,063	172,797	133,402
Annualized return	14.89%	32.33%	37.40%	44.94%	36.88%	36.49%	17.81%
Annualized Std	9.63%	27.51%	33.41%	29.62%	25.84%	33.41%	27.00%
Sharpe ratio	1.49	1.16	1.12	1.40	1.35	1.10	0.74
Max drawdown	20.93%	28.26%	27.76%	21.13%	22.47%	28.11%	33.92%

Table 1: Performance of single stock trading using PPO in the FinRL library. The Sharpe ratio of all the ETFs and stocks outperform the market, namely the S&P 500 index.

2019/01/01-2020/09/23	TD3	DDPG	Min-Var.	DJIA
Initial value	1,000,000	1,000,000	1,000,000	1,000,000
Final value	1,403,337; 1,381,120	1,396,607; 1,281,120	1,171,120	1,185,260
Annualized return	21.40%; 17.61%	20.34%; 15.81%	8.38%	10.61%
Annualized Std	14.60%; 17.01%	15.89%; 16.60%	26.21%	28.63%
Sharpe ratio	1.38; 1.03	1.28; 0.98	0.44	0.48
Max drawdown	11.52% 12.78%	13.72%; 13.68%	34.34%	37.01%

Table 2: Performance of multiple stock trading and portfolio allocation over the DJIA constituents stocks using the FinRL library. The Sharpe ratios of TD3 and DDPG excceed the DJIA index, and the traditional min-variance portfolio allocation strategy.

where $\boldsymbol{y_t} \in \mathbb{R}^n$ denotes the stock returns for current period t, $\boldsymbol{\mu} \in \mathbb{R}^n$ denotes the average of historical returns, and $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$ denotes the covariance of historical returns. It is used as a parameter that controls buying or selling action, for example if the turbulence index reaches a pre-defined threshold, the agent will halt buying action and starts selling the holding shares gradually.

4.5 Demonstration of Three Use Cases

We demonstrate with three use cases: single stock trading [10, 8, 26, 48], multiple stock trading [44, 45], and portfolio allocation [22, 27]. FinRL library provides practical and reproducible solutions for each use case, with online walk-through tutorial using Jupyter notebook (e.g., the configurations of the running environment and commands). We select three use cases and reproduce the results using FinRL to establish a benchmark for the quantitative finance community.

Fig. 4 and Table 1 demonstrate the performance evaluation of single stock trading. We pick large-cap ETFs such as SPDR S&P 500 ETF Trust (SPY) and Invesco QQQ Trust Series 1 (QQQ), and stocks such as Google (GOOGL), Amazon (AMZN), Apple (AAPL), and Microsoft (MSFT). We use PPO algorithm in FinRL and train a trading agent. The maximum drawdown in Table 1 is large due to Covid-19 market crash.

Fig. 5 and Table 2 show the performance and multiple stock trading and portfolio allocation over the Dow Jones 30 constitutes. We use DDPG and TD3 to trade multiple stocks, and allocate portfolio.

5 Conclusions

In this paper, we have presented FinRL library that is a DRL library designed specifically for automated stock trading with an effort for educational and demonstrative purpose. FinRL is characterized by its extendability, more-than-basic market environment and extensive performance evaluation tools also for quantitative investors and strategy builders. Customization is easily accessible on all layers, from market simulator, trading agents' learning algorithms up towards profitable strategies. In a trading strategy design, FinRL follows a training-validation-testing flow and provides automated backtesting as well as benchmark tests. As a walk-through tutorial in Jupyter notebook format, we demonstrate easily reproducible profitable strategies under different scenarios using FinRL: (i) single stock trading; (ii) multiple stock trading; (iii) incorporating the mechanism of stock information penetration. With FinRL Library, implementation of powerful DRL driven trading strategies is made an accessible, efficient and delightful experience.

References

[1] Andrew Ang. Mean-variance investing. *Columbia Business School Research Paper No. 12/49.*, August 10, 2012.

[2] 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.

[3] Stelios D Bekiros. Fuzzy adaptive decision-making for boundedly rational traders in speculative stock markets. *European Journal of Operational Research*, 202(1):285–293, 2010.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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)*, pages 29–33, 2019.

[9] Marco Corazza and Francesco Bertoluzzo. Q-learning-based financial trading systems with applications. *Econometric Modeling: International Financial Markets - Developed Markets eJournal*, 2014.

[10] Quang-Vinh Dang. Reinforcement learning in stock trading. In *ICCSAMA*, 2019.

[11] Stephen Dankwa and Wenfeng Zheng. Twin-delayed DDPG: A deep reinforcement learning technique to model a continuous movement of an intelligent robot agent. *Proceedings of the 3rd International Conference on Vision, Image and Signal Processing*, 2019.

[12] Yue Deng, F. Bao, Youyong Kong, Zhiquan Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28:653–664, 2017.

[13] 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.

[14] Thomas G. Fischer. Reinforcement learning in financial markets - a survey. Fau discussion papers in economics, Friedrich-Alexander University Erlangen-Nuremberg, Institute for Economics, 2018.

[15] Bryan Foltice and T. Langer. Profitable momentum trading strategies for individual investors. *Financial Markets and Portfolio Management*, 29:85–113, 2015.

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

[17] 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.

[18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning*, 2018.

[19] 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.

[20] Chien Yi Huang. Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint arXiv:1807.02787*, 2018.

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

[22] Zhengyao Jiang, Dixing Xu, and J. Liang. A deep reinforcement learning framework for the financial portfolio management problem. *ArXiv*, abs/1706.10059, 2017.

[23] Olivier Jin and Hamza El-Saawy. Portfolio management using reinforcement learning. *Stanford University*, 2016.

[24] 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*, 55:127–140, 2017.

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

[26] 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)*, 02:66–71, 2018.

[27] Xinyi Li, Yinchuan Li, Yuancheng Zhan, and Xiao-Yang Liu. Optimistic bull or pessimistic bear: Adaptive deep reinforcement learning for stock portfolio allocation. *ICML Workshop on Applications and Infrastructure for Multi-Agent Learning*, 2019.

[28] 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.

[29] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *ICLR*, 2016.

[30] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

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

[32] B. G. Malkiel. Passive investment strategies and efficient markets. *European Financial Management*, 9:1–10, 2003.

[33] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[35] J. Moody, L. Wu, Y. Liao, and M. Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17:441–470, 1998.

[36] John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 12(4):875–889, 2001.

[37] Abhishek Nan, Anandh Perumal, and Osmar R Zaiane. Sentiment and knowledge based algorithmic trading with deep reinforcement learning. *ArXiv*, abs/2001.09403, 2020.

[38] PG Nechchi. Reinforcement learning for automated trading. *Mathematical EngineeringPolitecnico di Milano: Milano, Italy*, 2016.

[39] Quantopian. Pyfolio: A toolkit for portfolio and risk analytics in python. `https://github.com/quantopian/pyfolio`, 2019.

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

[41] William F Sharpe. *Portfolio theory and capital markets*. McGraw-Hill College, 1970.

[42] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[43] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[44] Zhuoran Xiong, Xiao-Yang Liu, Shan Zhong, Hongyang Yang, and Anwar Walid. 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.

[45] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. *ACM International Conference on AI in Finance (ICAIF)*, 2020.

[46] Daochen Zha, Kwei-Herng Lai, Kaixiong Zhou, and X. X. Hu. Experience replay optimization. In *IJCAI*, 2019.

[47] Yong Zhang and Xingyu Yang. Online portfolio selection strategy based on combining experts' advice. *Computational Economics*, 50(1):141–159, 2017.

[48] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep reinforcement learning for trading. *The Journal of Financial Data Science*, 2(2):25–40, 2020.