Reinforcement Learning for Financial Trading: Algorithms, Evaluations and Platforms

Shuo Sun

College of Computing and Data Science

A thesis submitted to the Nanyang Technological University in partial fulfillment of the requirements for the degree of Doctor of Philosophy

2024

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

05/07/2024

Date

Shuo Sun

Shuo Sun

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

05/07/2024

Date

Prof. Bo An

Authorship Attribution Statement

This thesis contains material from five papers published in the following peer-reviewed conferences and journals in which I am the first author.

Chapter 2 and 3 are published as **Shuo Sun**, Rundong Wang, Bo An. Reinforcement learning for quantitative trading. ACM Transactions on Intelligent Systems and Technology, Vol.14, No.3, pp.1-29, 2023.

The contributions of the co-authors are as follows:

- Prof. Bo An provided the initial direction of the survey.
- I conducted literature review and wrote the manuscripts.
- Prof. Bo An and Rundong Wang provided insightful comments and carefully revised the manuscripts.

Chapter 4 is published as **Shuo Sun**, Wanqi Xue, Rundong Wang, Xu He, Junlei Zhu, Jian Li and Bo An. DeepScalper: A risk-aware reinforcement learning framework to capture fleeting intraday trading opportunities. Proceedings of the 31st ACM International Conference on Information & Knowledge Management (**CIKM'22**), pp. 1858-1867, 2022.

- Prof. Bo An proposed the problem.
- I proposed the algorithms, conducted experiments and wrote the manuscripts.
- Prof. Bo An and Prof. Jian Li provided critical comments to the algorithm design and manuscripts.
- The manuscripts were revised by Wanqi Xue, Rundong Wang, Xu He, Junlei Zhu, Prof. Jian Li and Prof. Bo An.

Chapter 5 is published as **Shuo Sun**, Xinrun Wang, Wanqi Xue, Xiaoxuan Lou and Bo An. Mastering stock markets with efficient mixture of diversified trading experts. Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD'23**), pp. 2109-2119, 2023.

- I proposed the idea and implemented the algorithms.
- I conducted the experiments and wrote the manuscripts.
- Xinrun Wang, Wanqi Xue, Xiaoxuan Lou and Prof. Bo An provided insightful comments and carefully revised the manuscripts.

Chapter 6 and 7 are published as **Shuo Sun**, Molei Qin, Xinrun Wang and Bo An. PRUDEX-Compass: Towards systematic evaluation of reinforcement learning in financial markets. Transactions on Machine Learning Research, 2023.

- Xinrun Wang and I proposed the problem.
- I proposed the benchmark and wrote the manuscripts.
- Molei Qin and I conducted experiments and prepared visualization figures.
- Xinrun Wang and Prof. Bo An provided insightful comments and carefully revised the manuscripts.

Chapter 8 is published as **Shuo Sun**, Molei Qin, Wentao Zhang, Haochong Xia, Chuqiao Zong, Jie Ying, Yonggang Xie, Lingxuan Zhao, Xinrun Wang and Bo An. TradeMaster: A holistic quantitative trading platform empowered by reinforcement learning. Proceedings of 37th Conference on Neural Information Processing Systems (**NeurIPS'23**) Datasets and Benchmarks Track, pp.59047-59061, 2023.

- Prof. Bo An proposed the problem.
- I designed the architecture of the software platform and implemented many major components.
- I conducted part of experiments and wrote the manuscripts.
- Molei Qin and Wentao Zhang implemented some components of the platform and helped on experiments.
- Haochong Xia, Chuqiao Zong, Jie Ying, Yonggang Xie, Lingxuan Zhao contributed to the Github repository, software document and official website.
- Xinrun Wang and Prof. Bo An provided insightful comments and carefully revised the manuscripts.

05/07/2024

Date

Shuo Sun

Acknowledgements

As my PhD journey comes to a close, I would like to express my most sincere gratitude to the people who have taught, helped and inspired me. The past three years of study has been unforgettable time of growth, exploration and harvest for me from both personal and professional perspectives. It is a great honor to work with exceptional people, who always inspire me to be curious, resilient and humble.

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Bo An. Thank you so much for offering the excellent opportunity to purse my PhD under your supervision. Your patience and insightful advice have been invaluable in shaping my academic growth throughout the journey. Discussions with you have consistently deepen my understanding of research including emphasizing the importance of solid motivation, fundamental contributions and real-world impact. Your tremendous comments and assistance has significantly improved my writing and presentation skills. As my research role model, I have learned a profound sense of responsibility, professionalism, and much more from you. I will be forever grateful to your supervision and I hope the connections fostered within the AMI group will continue to grow stronger across time and endure indefinitely.

I would like to say thank you for my Thesis Advisory Committee (TAC) members: Prof. Gao Cong and Prof. Zinovi Rabinovich. The meetings and discussions with you are quite inspiring and help me build a solid foundation of my research.

Throughout the three-year PhD journey, I have had the privilege of collaborating with many outstanding researchers including Prof. Xinrun Wang, Dr. Wanqi Xue, Dr. Xu He, Dr. Rundong Wang, Dr. Zhuoyi Lin, Mr. Molei Qin, Mr. Wentao Zhang, Mr. Haochong Xia, Mr. Chuqiao Zong, Mr. Xiaoxuan Lou and Prof. Jian Li. Thank you so much for the guidance, suggestions and collaborations during the projects. I am very proud to have the opportunity working with you.

I would like to express my gratitude to all members in the AMI group who accompanied my wonderful life in the last three years. They are Xinrun Wang, Xu He, Lei

Feng, Wei Qiu, Hongxin Wei, Yanchen Deng, Rundong Wang, Jakub Cerny, Shuxin Li, Shufeng Kong, Renchunzi Xie, Wanqi Xue, Pengdeng Li, Yewen Li, Chaojie Wang, Chuqiao Zong, Xinyu Cai, Haochong Xia, Molei Qin, Wentao Zhang, Pengjie Gu, Zhenghai Xue, Longtao Zheng, Yuzhou Cao, Sheng Zang, Qi Wei and Weihao Tan. The time I spent with you is a cherished memory for me. I will never forget the time of working together for deadlines and rebuttals, discussing on research topics and enjoying group dinners.

Lastly but most importantly, I would like to thank my parents and precious friends Han Song and Wanling Pan for your unwavering supports throughout the journey. I am profoundly grateful to my girlfriend Muqi Yu, whose steadfast support, immeasurable love, and remarkable patience have been a source of strength and inspiration to me. They hold a significance beyond words in this PhD journey. Needless to say, this thesis would not have been possible without your encouragement. This thesis is dedicated to all of you.

Abstract

The financial markets, which involve more than $90 trillion market capitals, attract the attention of innumerable investors around the world. Due to the complexity and low signal-to-noise ratio of financial data, it is a challenging task to make profitable investment decisions. Recently, reinforcement learning (RL) has emerged as a promising direction to train agents for financial decision making. However, most existing techniques focuses on small-scale tasks and deliver a vulnerable solution without systematic evaluation. There is still a long way to go for deploying RL algorithms into real-world finance applications.

This doctoral thesis is devoted to push the real-world deployment of RL in financial decision making (FinRL) from the perspectives of algorithms, evaluation benchmarks and software infrastructures, by leveraging the power of modern RL, ensemble learning and system design techniques. In particular, we tackle the following fundamental research problems and propose a solution for all of them.

First, we study the problem of designing efficient FinRL algorithms under the competitive intraday trading scenarios to actively trade within the same day for profit arbitrage. We provide a formal definition and propose a risk-aware deep RL algorithm. A dueling Q-network with action branching is applied to deal with the large action space for RL optimization. Three key components, namely hindsight bonus, market embedding encoder-decoder and risk-aware auxiliary task, are introduced to further improve the performance. Extensive experiments on real-world financial datasets demonstrate the superior performance of the proposed algorithm comparing to state-of-the-art baselines.

We then investigate the problem of adopting advanced ensemble learning and mixture-of-experts (MoE) techniques for robust decision making in finance. The initial idea is inspired by the real-world bottom-up trading strategy design workflow in the finance industry, where hierarchical decision-making is involved to combine the intelligence of junior traders and senior portfolio managers. We first propose a novel

three-stage MoE framework for stock prediction. In stage one, an efficient ensemble learning method is designed to train multiple trading experts with tiny overhead of costs. In stage two, we build a pool of trading experts utilizing hyperparameter level and initialization level diversity of neural networks for post hoc ensemble construction. In stage three, we design three different mechanisms, namely as-needed router, with-replacement selection and integrated expert soup, to dynamically pick experts from the expert pool. Later on, we extend the idea into the FinRL setting by proposing a simple yet efficient RL algorithm. Risk-aware Bellman backup is introduced to reweight Q-values by adopting estimated uncertainty. To encourage strategy diversity, we leverage the idea of bootstrap with random initialization. Extensive experiments show that the two algorithms that leveraging the power of ensemble learning significantly outperforms other baselines with higher profit, lower risk and better diversity.

Third, we focus on the systematic evaluation of FinRL algorithms. The motivation is that the evaluation of existing FinRL algorithms mostly focuses on profit-related measures and ignores many critical axes. We introduce a systematic evaluation benchmark with six axes, i.e., profitability, risk-control, universality, diversity, reliability, and explainability, and 17 measures from the literature of multiple disciplines (e.g., finance, artificial intelligence, statistics and engineering). We collect the implementation of evaluation metrics and visualization tools into an open-source library and further demonstrate its usage on four datasets with eight FinRL algorithms.

Finally, we study the problem of FinRL software infrastructures. To address the engineering challenges and facilitate the development of new FinRL algorithms, we build a holistic open-source trading platform. The platform is composed of six key modules, which cover the whole pipeline including data preprocessing, market simulators implementation, algorithms design, evaluation toolkits and user interfaces. The platform offers the software infrastructures and relevant materials (e.g., documentation, tutorials) on RL based financial trading and has made great impact in the community with over 1.2k stars on Github.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

With a global capital of over 90 trillion dollars, quantitative trading has been a popular research topic in both academia and financial industry for decades. Recently, reinforcement learning (RL) has become an appealing direction for financial decision making owing to its outstanding ability on complex sequential decision making. However, the potential of RL is not fully released due to the noisy nature of financial market and unstable training procedures of RL. Significant efforts are still required for the wide real-world deployment of RL in finance.

This thesis systematically examines the challenges of deploying RL into real-world financial markets with a focus on the three key perspectives: i) algorithms; ii) evaluation benchmarks and iii) software infrastructures. To tackle these challenges, we propose novel RL algorithms for various trading settings, design systematic evaluation benchmarks and develop holistic software platforms in this thesis.

Our first emphasis is to design profitable FinRL algorithms for intraday trading, where traders actively long/short one selected financial asset within the same day to maximize profit. We provide a formal definition of intraday trading and propose a risk-aware deep RL algorithm called DeepScalper including four key components: i) a dueling Q-network with action branching to deal with the large action space for efficient RL optimization; ii) a novel reward function with a hindsight bonus to encourage RL agents making trading decisions with a long-term horizon; iii) an

encoder-decoder architecture to learn multi-modality temporal market embedding, which incorporates both macro-level and micro-level market information; iv) a risk-aware auxiliary task to maintain a striking balance between maximizing profit and minimizing risk. Extensive experiments on long-term real-world financial data demonstrate that DeepScalper significantly outperforms many state-of-the-art baselines for intraday investment decision making.

We then investigate the problem of adopting advanced ensemble learning and mixture-of-experts (MoE) techniques to pursue more accurate and robust decision making in finance. The initial motivation is inspired by the bottom-up hierarchical trading strategy design workflow in finance, where traders analyze the market from different perspectives and the senior portfolio manager make the final decision based on these suggestions. Two algorithms are introduced for stock prediction (supervised learning) and financial decision making (FinRL) settings, respectively. We propose AlphaMix, a novel three-stage MoE framework for stock prediction. Specifically, we introduce an efficient ensemble learning method, whose computational and memory costs are significantly lower than traditional ensemble methods, to train multiple groups of trading experts with personalised market understanding and trading styles in stage one. In stage two, we collect diversified investment suggestions through building a pool of trading experts utilizing hyperparameter level and initialization level diversity of neural networks for post hoc ensemble construction. In stage three, we design three different mechanisms, namely as-needed router, with-replacement selection and integrated expert soup, to dynamically pick experts from the expert pool, which takes the responsibility of a portfolio manager. Through extensive experiments on US and Chinese stock markets, we demonstrate that AlphaMix achieves the highest profit in terms of 7 popular financial criteria. Furthermore, we propose AlphaMix+, an universal deep RL framework with diversified risk-aware mixture-of-experts. Risk-aware Bellman backup is introduced to reweight Q-values by adopting estimated uncertainty. We further leverage the idea of bootstrap with random initialization to encourage strategy diversity between different trading agents. Extensive experiments show that AlphaMix+ outperforms other baselines on four financial markets.

Third, we pay attention to the evaluation of RL algorithms in finance. The motivation here is that the evaluation of most existing FinRL algorithms only focuses on profit-related measures and many critical axes that industry practitioners

care about are ignored. We introduce PRUDEX-Compass, which has 6 axes, i.e., profitability, risk-control, universality, diversity, reliability, and explainability, with a total of 17 measures for a systematic evaluation. To demonstrate the usage of the benchmark, we evaluate 8 FinRL methods in 4 real-world financial datasets. The source code of evaluation metrics and visualization toolkits is public available.

Fourth, we focus on the software infrastructures for FinRL research to pursue fast implementation and industry deployment. Orchestrating a FinRL project lifecycle poses challenges in engineering (i.e., hard to build), benchmarking (i.e., hard to compare) and usability (i.e., hard to optimize, maintain and use). We build TradeMaster, a holistic open-source platform, that serves as a i) software toolkit, ii) empirical benchmark, and iii) user interface. It covers the whole pipeline of FinRL including data preprocessing, market simulators, algorithms and evaluation toolkits. Our ultimate goal is to provide infrastructures for transparent and reproducible FinRL research and facilitate their real-world deployment with industry impact.

Reinforcement learning for financial decision making presents significant challenges, especially in today's fast-paced and ever-changing financial markets. In this thesis, we conduct comprehensive exploration on the challenges of employing FinRL algorithms to push the real-world deployment from different perspectives. The approaches and software from this thesis are beneficial to both finance and RL community. We hope that this thesis can not only shed light on future research to prevent untrustworthy results from stagnating FinRL into successful industry deployment but also provide a new challenging algorithm evaluation scenario for the RL community.

1.2 Motivation

RL has achieved significant success in various quantitative trading tasks. Specifically, FDDR [2] and iRDPG [3] are designed to learn financial trading signals and mimic behaviors of professional traders for algorithmic trading, respectively. For portfolio management, deep RL methods are proposed to account for the impact of market risk [4] and the commission fee [5]. A PPO-based framework [6] is proposed for order execution and a policy distillation mechanism is added to bridge the gap between imperfect market states and optimal execution actions [7]. For market making,

deep RL methods are introduced from both game-theoretic [8] and adversarial learning [9] perspectives as an adaptation of traditional mathematical models. However, although numerous work has been focused on FinRL, their performance, robustness and generalization ability are still far from satisfactory for real-world deployment. We point out major challenges that motivated this thesis as follows.

For algorithms, popular FinRL methods [4, 5, 7, 10] mostly focus on day-level trading scenarios, which has gap to the high-frequency decision making (e.g., minute-level) user cases for human traders. One direction is to explore more realistic trading setups (e.g., intraday trading) that aligned with real markets. On the other hand, one major limitation of existing works is that investment decisions are made based on one individual neural network with high uncertainty, which is inconsistent with the workflow in real-world trading firms. To achieve more accurate and robust performance, it worth exploring the potential of ensemble learning [11] and mixture-of-experts (MoE) to collaborate into the design of FinRL algorithms.

For evaluation, existing FinRL methods [5, 7, 10, 12] only focuses on profit-related measures, which ignores several critical axes such as risk-control and reliability. In addition to profitability, financial practitioners care about many other aspects of FinRL methods, i.e., how much risk I need to take for per unit of profit; how FinRL algorithms behave when the market status changes. In preliminary experiments, we find many examples that indicate the weakness of existing profit-seeking FinRL algorithms. For instance, IMIT [13] may lead to catastrophic capital loss when black swan events happen and SAC [14] shows poor risk-control ability, which is not an ideal option for conservative traders. In practice, with the existence of low signal-to-noise ratio and distribution shift in financial markets [15], FinRL methods with only great profitability performance on backtesting are very likely to overfit on historical data and become risky to be deployed in real-world scenarios [16]. As Robert Pardo (CEO of a hedge fund) said, he will never trade with a method that does not prove itself through a systematic evaluation [17]. Therefore, a benchmark for the systematic evaluation of FinRL methods is urgently needed.

For software infrastructures, financial practitioners are never keen to share source code of their algorithms and platforms [18] that achieve good performance in real-world settings due to the existence of alpha decay phenomenon [19]. However, building FinRL algorithms encompasses engineering challenges due to the highly composite nature of this domain, which entails design choices and interactions

between components that collect financial data, conduct feature engineering, build market environments, make investment decisions, evaluate model behaviors and provides user interfaces. Significant upfront efforts are required to address the engineering issue. For an immature domain such as FinRL, it is common to spend over 90% of work on implementation details and less than 10% of work on core technical questions [20]. As a FinRL researcher, it is seldom the case that enough resources are available to build a high quality code base of the complete FinRL workflow. There has been many attempts on financial trading platforms. Qlib [18] proposes an AI-oriented quantitative investment platform with a focus on prediction-based supervised learning settings. FinRL [21] makes the first attempt on developing an RL-based quantitative trading platform. FinRL-Meta [22] offers a series of data-driven RL environments for various trading scenarios. However, existing software platforms either not focuses on RL or only cover limited algorithms and part of the pipeline. As a result, a holistic platform that encapsulates all major components of FinRL with comprehensive support of mainstream settings and state-of-the-art algorithms is highly desirable for both academic researchers and industry practitioners.

1.3 Main Contributions

In this thesis, we make multiple significant contributions on RL for financial decision making. We conduct research from the perspective of algorithm design, systematic evaluation and software infrastructure. The major contributions are summarized as follows.

First, we propose a novel risk-aware RL framework namely DeepScalper for intraday trading. We apply the dueling Q-network with action branching to effectively optimize intraday trading agents with high-dimensional fine-grained action space. A reward function with hindsight bonus is designed to encourage a long-time horizon for RL agents. We propose a multi-modality encoder-decoder architecture to learn temporal intraday market embedding. Furthermore, we design a risk-aware auxiliary task to keep balance between profit and risk. Through empirical experiments, we show that DeepScalper significantly outperforms many state-of-the-art baseline methods and demonstrate its practical applicability to intraday trading with a series of exploratory and ablative studies.

Second, we design the first mixture-of-experts framework for quantitative investment called AlphaMix. We first efficiently optimize multiple independent groups of trading experts, then we build a pool of diversified models for post hoc ensemble construction, and finally three different mechanisms are introduced to further improve performance by dynamically picking these experts. Through experiments on real-world data of two influential stock markets spanning over five years, we show that AlphaMix significantly outperforms 11 state-of-the-art baseline methods in terms of seven financial criteria. Later on, we extend the idea into RL setting and introduce AlphaMix+. To make decisions under estimated uncertainty, risk-aware Bellman backup is introduced to reweight Q-values. To encourage strategy diversity, the idea of bootstrap with random initialization is applied. Extensive experiments show that AlphaMix+ significantly outperforms other baselines with higher profit, lower risk and better diversity.

Third, we introduce the first systematic evaluation benchmark for RL in finance. The benchmark includes six axes to evaluate the algorithm from the perspectives of profitability, risk-control, universality, diversity, reliability, and explainability. Dozens of measures and visualization toolkits are developed from the literature of multiple disciplines (e.g., finance, artificial intelligence, statistics and engineering). All source code is public available to inspire future research in this field. Furthermore, we demonstrate the usage of the metrics and toolkits by evaluating 8 FinRL algorithms on 4 datasets.

Finally, we develop a holistic RL based trading platform called TradeMaster. It offers open-source implementations of the whole workflow including 13 real-world financial datasets, high-fidelity RL environments for 6 mainstream quantitative trading tasks, 15 popular RL in finance algorithms and dozens of tools for systematic evaluation and visualization. Software documentation, tutorials and official website are also included to make the platform more accessible. This modular and composable structure enables fast development and decrease the cost of collaboration and code-sharing. TradeMaster serve as software toolkits, empirical benchmark and user interfaces with great impact for the community.

1.4 Outline of the Thesis

The rest part of the thesis is organized as follows. In Chapter 2, we introduce preliminaries of quantitative finance and RL. Chapter 3 conducts literature review on works relevant to this thesis. Chapter 4 focuses on designing FinRL algorithms for intraday trading. Chapter 5 and Chapter 6 improve financial decision making with advanced ensemble learning and MoE techniques. Chapter 7 focuses on the systematic evaluation of FinRL algorithms. Chapter 8 focuses on building software infrastructures on RL based trading applications. Chapter 9 summarise this thesis and point out potential future directions.

Chapter 2

Preliminaries

2.1 Quantitative Trading Background

2.1.1 Overview

For many countries, the financial industry serves as a paramount pillar and spawns the birth of many financial centres (e.g., New York, London, Singapore and Shanghai). Trading exchanges, where trading activities of trillions of dollars volume are executed, play a key role to offer liquidity for the financial markets. Many famous trading exchanges are formed such as NYSE and Nasdaq for stocks, CME for derivatives and Coinbase for cryptocurrencies. Participants in the financial market can be generally categorized as financial intermediaries (e.g., banks and brokers), issuers (e.g., companies and governments), institutional investors (e.g., investment managers and hedge funds) and individual investors. With the development of electronic trading platforms, quantitative trading, which refers to the type of investment strategy that relies on mathematical models to automatically make profitable investment decisions [23], has been demonstrated quite profitable by many leading trading companies (e.g., Jane Street, Two Sigma, Citadel, D.E. Shaw). It is becoming a dominating trading style in the global financial markets that accounts for over 70% and 40% overall trading volume in developed market (e.g., US and Europe) and emerging market (e.g., China and India) respectively at the year of 2020.

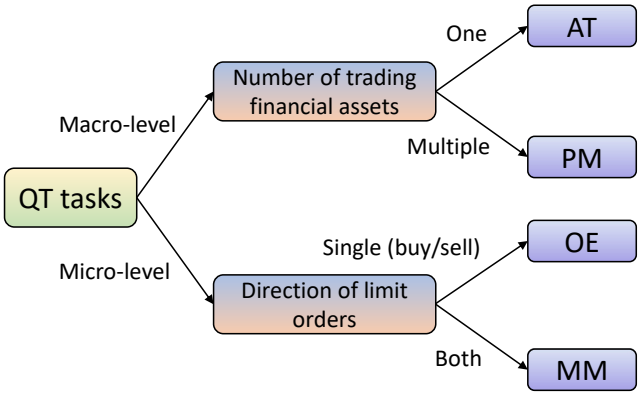


FIGURE 2.1: Tree relationships of different QT Tasks.

As shown in Figure 2.1, we categorize various quantitative trading tasks as four mainstream tasks: algorithmic trading (AT), portfolio management (PM), order execution (OE) and market making (MM). We introduce them one by one in the following part of this subsection.

2.1.2 Algorithmic Trading

Algorithmic trading refers to the process that traders consistently buy and sell one given financial asset to maximize profit. It has been widely applied in trading stocks, commodity futures, foreign exchanges and cryptocurrencies. In algorithmic trading, time is splitted into discrete time steps. At the beginning of a trading period, traders are allocated some cash and set net value as 1. Then, at each time step t, traders have the options to buy, hold or sell some amount of shares to change positions. Net value and position is used to represent traders' status at each time step. The objective of algorithmic trading is to maximize the final net value at the end of the trading period.

Based on trading styles, algorithmic trading is generally divided into five categories: position trading, swing trading, intraday trading, scalp trading and high-frequency trading. Specifically, position trading involves holding the financial asset for a long period of time, which is not sensitive with short-term market fluctuations and only focuses on the overarching market trend. Swing trading is a medium-term style that holds financial assets for several days or weeks. The goal of swing trading is to spot a trend and then capitalise on dips and peaks that provide entry points. Intraday trading tries to capture the fleeting intraday pattern in the financial market and

all positions will be closed at the end of the day to avoid overnight risk. Scalping trading aims at discovering micro-level trading opportunities and makes profit by holding financial assets for only a few minutes. High-frequency trading is a type of trading style characterized by high speeds, high turnover rates, and high order-to-trade ratio. A summary of different trading styles is illustrated in Table 2.1.

Trading Style	**Time Frame**	**Holding Period**
Position trading	Long term	Months to years
Swing trading	Medium term	Days to weeks
Intraday trading	Short term	Within a trading day
Scalping trading	Short term	Seconds to minutes
High-frequency trading	Extreme short term	Milliseconds to seconds

TABLE 2.1: A Summary of Algorithmic Trading Styles

Traditional algorithmic trading methods discover trading signals based on technical indicators or mathematical models. Buy & Hold (BAH) strategy, which invests all capital at the beginning and holds until the end of the trading period, is proposed to reflect the average market condition. Momentum strategies, which assumes the trend of financial assets in the past has the tendency to continue in the future, are another well-known algorithmic trading strategies. Buying-winner-selling-loser [24], time series momentum [25] and cross sectional momentum [26] are three classic momentum strategies. In contrast, mean reversion strategies such as Bollinger bands [27] assume the price of financial assets will finally revert to the long-term mean. Although traditional methods may capture the underlying patterns of the financial market, these simple rule-based methods exhibit limited generalization ability among different market conditions.

2.1.3 Portfolio Management

Portfolio management (PM) is a fundamental trading task, where investors hold a number of financial assets and reallocate them periodically to maximize long-term profit. In the literature, it is also called portfolio optimization, portfolio selection and portfolio allocation. In the real market, portfolio managers work closely with traders, where portfolio managers assign a percentage weighting to every stock in the portfolio periodically and traders focus on finishing portfolio reallocation at

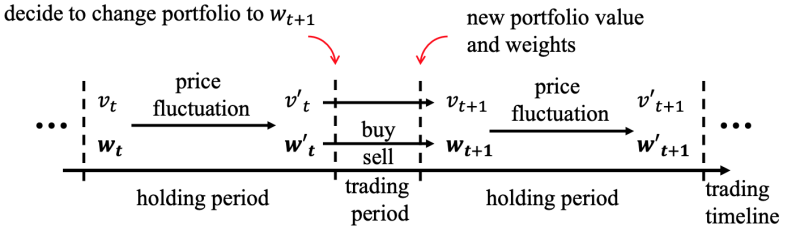


FIGURE 2.2: Portfolio management process

the favorable price to minimize the trading cost. For PM, time is splitted into two types of periods: holding period and trading period as shown in Figure 2.2. At the beginning of a holding period, the agent holds a portfolio $\mathbf{w_t}$ consists of pre-selected financial assets with a corresponding portfolio value v_t. With the fluctuation of the market, the assets' prices would change during the holding period. At the end of the holding period, the agent will get a new portfolio value v'_t and decide a new portfolio weight $\mathbf{w_{t+1}}$ of the next holding period. During the trading period, the agent buys or sells some shares of assets to achieve the new portfolio weights. The lengths of the holding period and trading period are based on specific settings and can change over time. In some previous works, the trading period is set to 0, which means the change of portfolio weight is achieved immediately for convenience. The objective is to maximize the final portfolio value given a long-term time horizon.

PM has been a fundamental problem for both finance and machine learning community for decades. Existing approaches can be grouped into four major categories, which are benchmarks such as constant rebalanced portfolio and uniform constant rebalanced portfolio [28], follow-the-winner approaches such as exponential gradient [29] and winner [30], follow-the-loser approaches such as robust mean reversion [31], passive aggressive online learning [32] and anti-correlation [33], pattern-matching-based approaches such as correlation-driven nonparametric learning [34] and B^K [35], and meta learning algorithms such as online Newton step. More details are available in this survey paper [36].

2.1.4 Order Execution

While adjusting new portfolio, investors need to buy/sell some amount of shares by executing an order of liquidation. Essentially, the objectives of order execution are two-fold: it does not only require to fulfill the whole order but also targets a more economical execution with maximizing profit (or minimizing cost). As

mentioned in [37], the major challenge of order execution lies in the trade-off between avoiding harmful market impact caused by large transactions in a short period and restraining price risk, which means missing good trading windows due to slow execution. Traditional OE solutions are usually designed based on some stringent assumptions of the market and then derive some model-based methods with stochastic control theory.

For instance, time weighted average price (TWAP) evenly splits the whole order and execute at each time step with the assumption that the market price follows the Brownian motion [38]. The Almgren-Chriss model [39] incorporates temporary and permanent price impact functions also with the Brownian motion assumption. Volume weighted average price (VWAP) distributes orders in proportion to the (empirically estimated) market transaction volume. The goal of VWAP is to track the market average execution price [40]. However, traditional solutions are not effective in the real market because of the inconsistency between the assumptions and reality.

Formally, OE is to trade fixed amount of shares within a predetermined time horizon (e.g., one hour or one day). At each time step t, traders can propose to trade a quantity of $q_t \geq 0$ shares at current market price p_t, The matching system will then return the execution results at time $t+1$. Taking the sell side as an example, assuming a total of Q shares required to be executed during the whole time horizon, the OE task can be formulated as:

$$\arg \max_{q_1, q_2, \ldots, q_T} \sum_{t=1}^{T} (q_t \cdot p_t), \quad \text{s.t.} \sum_{t=1}^{T} q_t = Q$$

OE not only completes the liquidation requirement but also the maximize/minimize average execution price for the sell/buy side execution respectively.

2.1.5 Market Making

Market makers are traders who continually quote prices at which they are willing to trade on both buy and sell side for one financial asset. They provide liquidity and make profit from the tiny price spread between buy and sell orders. The main challenge for market making is non-zero inventory. When we submit a limit order on both sides, there is no guarantee that all the orders can be successfully executed.

It is risky when non-zero inventory accumulates to a high level because this means market maker will have to close the inventory by current market price, which could lead to a significant loss. In practice, some market makers keep their inventory at low-level to avoid market exposure and only make profits by repeatedly making their quoted spread. On the other hand, some more advanced market makers may choose to hold a non-zero inventory to capture the market trend, while exploiting the quoted spread simultaneously. Traditional finance methods consider market making as a stochastic optimal control problem [41]. Agent-based method [42] and RL [43] have also been applied to market making.

2.1.6 Terminologies

Definition 2.1. (OHLCV) OHLCV is a type of bar chart directly obtained from the financial market. OHLCV vector at time t is denoted as $\mathbf{x}_t = (p_t^o, p_t^h, p_t^l, p_t^c, v_t)$, where p_t^o is the open price, p_t^h is the high price, p_t^l is the low price, p_t^c is the close price and v_t is the volume.

Definition 2.2. (Technical Indicator) A technical indicator indicates a feature calculated by a formulaic combination of the original OHLCV to uncover the underlying pattern of the financial market. We denote the technical indicator vector at time t: $\mathbf{y}_t = \bigcup_k y_t^k$, where $y_t^k = f_k(\mathbf{x}_{t-h}, ..., \mathbf{x}_t, \theta^k)$, θ^k is the parameter of technical indicator k.

Definition 2.3. (Limit Order) A limit order is an order placed to long/short a certain number of shares at a specific price. It is defined as a tuple $l = (p_{target}, \pm q_{target})$, where p_{target} represents the submitted target price, q_{target} represents the submitted target quantity, and $\pm$ represents the trading direction (long/short).

Definition 2.4. (Limit Order Book) A limit order book (LOB) contains public available aggregate information of limit orders by all market participants. It is widely used as market microstructure [44] in finance to represent the relative strength between buy and sell side. We denote an m level LOB at time t as $\mathbf{b}_t = (p_t^{b_1}, p_t^{a_1}, q_t^{b_1}, q_t^{a_1}, ..., p_t^{b_m}, p_t^{a_m}, q_t^{b_m}, q_t^{a_m})$, where $p_t^{b_i}$ is the level i bid price, $p_t^{a_i}$ is the level i ask price, $q_t^{b_i}$ and $q_t^{a_i}$ are the corresponding quantities.

Definition 2.5. (Matching System) The matching system is an electronic system that matches the buy and sell orders for the financial market. It is usually used to execute orders for different traders in the exchange.

Definition 2.6. (Position) Position pos_t at time t is the amount and direction of a financial asset owned by traders. It represents a long (short) position when pos_t is positive (negative).

Definition 2.7. (Net Value) Net value is the normalised sum of cash and value of financial assets held by a trader. The net value at time t is denoted as $n_t = (c_t + p_t^c \times |pos_t|)/c_1$, where c_t is the cash at time t and c_1 is the initial amount of cash.

Definition 2.8. (Portfolio) Portfolio is the proportion of capitals allocated to each asset that can be represented as a vector:

$$\mathbf{w_t} = [w_t^0, w_t^1, ..., w_t^M] \in R^{M+1} \quad and \quad \sum_{i=0}^{M} w_t^i = 1 \tag{2.1}$$

where $M+1$ is the number of portfolio's constituents, including cash and M financial assets. w_t^i represents the ratio of the total portfolio value invested at time t on asset i and w_t^0 represents cash.

Definition 2.9. (Asset Price) Asset price refers to the vector of close price for each financial asset defined as $\mathbf{p_t} = [p_t^0, p_t^1, ..., p_t^M]$, where p_t^i is the close price of asset i at time t. Note that the price of cash p_t^0 is a constant.

Definition 2.10. (Portfolio Value) Portfolio value v_{t+1} at time $t+1$ is defined based on the asset price change and portfolio weight as:

$$v_{t+1} = v_t \sum_{i=0}^{M} \frac{w_t^i \ p_{t+1}^i}{p_t^i} \tag{2.2}$$

2.1.7 Technical Indicators

Technical indicators play a key role on the performance of machine learning methods. We generate 11 widely used temporal features as shown in Table 2.2 to describe the stock markets following [45, 46]. z_{open}, z_{high} and z_{low} represent the relative values of the open, high, low prices compared with the close price at current time step, respectively. z_{close} and z_{adj_close} represent the relative values of the closing and adjusted closing prices compared with time step $t-1$. z_{dk} represents a long-term moving average of the adjusted close prices during the last k time steps compared to the current close price. To keep a fair and consistent comparison, these features are used for empirical experiments of following chapters in the thesis.

Features	**Calculation Formula**
$z_{open}, z_{high}, z_{low}$ z_{close}, z_{adj_close}	e.g., $z_{open} = open_t/close_t - 1$ e.g., $z_{close} = close_t/close_{t-1} - 1$
$z_{d\_5}, z_{d\_10}, z_{d\_15}$ $z_{d\_20}, z_{d\_25}, z_{d\_30}$	e.g., $z_{d\_5} = \frac{\sum_{i=0}^{4} adj\_close_{t-i}/5}{adj\_close_t} - 1$

TABLE 2.2: Formulas of features to describe the stock markets.

2.1.8 Evaluation Metrics

We introduce all numerical evaluation metrics including profit metrics, risk metrics, risk-adjusted return metrics and diversity metrics.

- **Total Return (TR)** is the overall return rate of the whole trading period. It is defined as $TR = \frac{n_t - n_1}{n_1}$, where n_t is the final net value and n_1 is the initial net value.
- **Volatility (VOL)** is the variance of the daily return defined as $\sigma[\mathbf{ret}]$ to measure the risk level of trading strategies, where $\mathbf{ret} = (ret_1, ret_2, ..., ret_t)$ is a vector of daily return and $\sigma[.]$ is the variance.
- **Downside Deviation (DD)** is the variance of negative return of the whole trading period.
- **Maximum Drawdown (MDD)** measures the largest loss from any peak to show the worst case.
- **Sharpe Ratio (SR)** considers the amount of extra return that a trader receives per unit of increase in risk. It is defined as: $SR = E[\mathbf{ret}]/\sigma[\mathbf{ret}]$, where $E[.]$ is the expected value.
- **Calmar Ratio (CR)** is defined as $CR = \frac{E[\mathbf{ret}]}{MDD}$, which is the expected return divided by the maximum drawdown.
- **Sortino Ratio (SoR)** applies downside deviation as the risk measure. It is defined as: $SoR = \frac{E[\mathbf{ret}]}{DD}$.
- **Entropy (ENT)** [47] is applied to measure the amount of information given by observing the financial market. In a portfolio, it is defined as $H(\omega) = -exp\left(\sum_{i=1}^{n} \omega_i \, log \, \omega_i\right)$, where $\omega = (\omega_1, \omega_2, ..., \omega_n)$ is a portfolio among N financial assets. This measure reach a minimum value 1 if a portfolio is fully concentrated

in one single asset and a maximum equal to N that representation the equally weighted portfolio.

- **Effect Number of Bet (ENT)** As entropy ignore the correlation between different financial assets, effective number of bets (ENB) is proposed to remove the correlation while calculating entropy. We first define diversification distribution as $p_i(\omega) = \omega_{F_i}^2 \lambda_i^2 / \sum_{i=1}^n \omega_{F_i}^2 \lambda_i^2$. We formulate the covariance matrix of N assets Σ as $E'\Sigma E = \lambda$ where λ is a diagonal matrix,λ_i is the element of the diagonal matrix and $\omega_f = E^{-1}\omega$ where ω is the our original portfolios' weights. The effective number of bets is defined as: $ENB(\omega) = -exp\left(\sum_{i=1}^n p_i(\omega) \log p_i(\omega)\right)$, where $p_i(\omega)$ is the i^{th} diversification distribution for the portfolio weights ω.

2.2 Reinforcement Learning Basics

RL is a popular subfield of machine learning that studies complex decision making problems. Sutton and Barto [48] distinguish RL problems by three key characteristics: i) the problem is closed-loop. ii) the agent figures out what to do through trial-and-error. iii) actions have an impact on both short term and long term results. The decision maker is called agent and the environment is everything else except the agent. At each time step, the agent obtains some observations of the environment, which is called state. Later on, the agent takes an action based on the current state. The environment will then return a reward and a new state to the agent. Formally, an RL problem is typically formulated as a Markov decision process (MDP) in the form of a tuple $\mathcal{M} = (S, A, R, P, \gamma)$, where S is a set of states $s \in S$, A is a set of actions $a \in A$, R is the reward function, P is the transition probability, and γ is the discount factor. The goal of an RL agent is to find a policy $\pi(a \mid s)$ that takes action $a \in A$ in state $s \in S$ in order to maximize the expected discounted cumulative reward:

$$\max \ \mathbb{E}[R(\tau)], \ \textit{where } R(\tau) = \sum_{t=0}^{\tau} \gamma^t r(a_t, s_t) \textit{ and } 0 \leq \gamma \leq 1$$

Sutton and Barto [48] summarise the main components of RL as: i) policy, which refers to the probability of taking action a when the agent is in state s. From policy perspective, RL algorithms are categorized into on-policy and off-policy

methods. The goal of on-policy RL methods is to evaluate or improve the policy, which they are now using to make decisions. As for off-policy RL methods, they aim at improving or evaluating the policy that is different from the one used to generate data. ii) reward: after taking selected actions, the environment sends back a numerical signal reward to inform the agent how good or bad are the actions selected. iii) value function, which means the expected return if the agent starts in that state s or state-action pair (s, a), and then acts according to a particular policy π consistently. Value function tells how good or bad the agent's current position is in the long run. iv) model, which is an inference about the behaviour of the environment in different states.

Plenty of algorithms have been proposed to address RL problems. Tabular methods and approximation methods are two mainstream directions. For tabular algorithms, a table is used to represent the value function for every action and state pair. The exact optimal policy can be found through checking the table. Due to the curse of dimensionality, tabular methods only work well when the action and state space is small. Dynamic programming (DP), Monto Carlo (MC) and temporal difference (TD) are a few widely studied tabular methods. Under perfect model of environment assumption, DP uses a value function to search for good policies. Policy iteration and value iteration are two classic DP algorithms. MC methods try to learn good policies through sample sequences of states, actions, and reward from the environment. For MC methods, the assumption of perfect environment understanding is not required. TD methods are a combination of DP and MC methods. While they do not need a model from the environment, they can bootstrap, which is the ability to update estimates based on other estimates. From this family, Q-learning [49] and SARSA [50] are popular algorithms, which belong to off-policy and on-policy methods respectively.

On the other hand, approximation methods try to find a great approximate function with limited computation. Learning to generalize from previous experiences (already seen states) to unseen states is a reasonable direction. Policy gradient methods are popular approximate solutions. REINFORCE [51] and actor-critic [52] are two important examples. With the popularity of deep learning, RL researchers use neural networks as function approximator. DRL is the combination of deep learning and RL, which lead to great success in many domains [53, 54]. Popular DRL algorithms for FinRL community include deep Q-network (DQN) [53], deterministic policy

gradient (DPG) [55], deep deterministic policy gradient (DDPG) [56], proximal policy optimization (PPO) [57]. Recurrent reinforcement learning (RRL) is another widely used RL approach for FinRL. Recurrent means the previous output is fed into the model as part of the input here. RRL achieves more stable performance when exposed to noisy data such as financial data.

Chapter 3

Literature Review

In this chapter, we conduct a comprehensive literature review on relevant domains of this thesis. We focus on novel algorithms, systematic evaluation and software platforms for financial decision making.

3.1 RL for Quantitative Trading

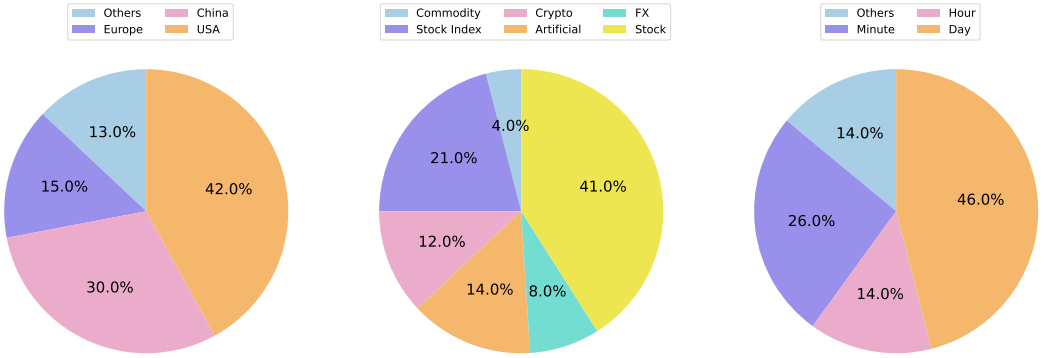
In this subsection, we introduce notable RL methods for quantitative trading. We go through existing works across four mainstream FinRL tasks with a summary table at the end of each subsection.

Category	RL algorithm	Publication
Value-Based	Q-learning	[43, 58–68]
	SARSA	[43, 69–71]
	DQN	[5, 12, 72–78]
Policy-Based	Recurrent RL	[2, 13, 79–85]
	REINFORCE	[5]
	PG	[4, 75, 86–88]
	TRPO	[89, 90]
Actor-Critic	DPG	[3, 91–93]
	PPO	[7, 74, 76, 88, 94, 95]
	DDPG	[88, 95–97]
	SAC	[76, 98]
	A2C	[75, 95]
Others	Model-based RL	[99, 100]
	Multi-Agent RL	[59, 101, 102]

TABLE 3.1: Publications based on different reinforcement learning algorithms

3.1.1 Categories of FinRL methods

In order to provide a bird-eye's view, existing works are classified from different perspectives. Table 3.1 summarizes existing works from the RL algorithm perspective. Q-learning and recurrent RL are the most popular RL algorithms for FinRL. Recent trend indicates deep RL methods such as DQN, DDPG and PPO outperform traditional RL methods. In addition, we use three pie charts to provide taxonomies of existing works based on financial markets, financial assets and data frequencies. The percentage numbers shown in the pie charts are calculated by dividing the number of papers belonging to each type with the total number of papers. We classify existing works based on financial markets (Figure 3.1a). The US market is the most studied market in the literature. Chinese market is getting popular in recent years. The study of European market is mainly in the early era. We classify existing works based on financial assets (Figure 3.1b). Stock data is used for more than 40% of publications. Stock index is the second popular additional option. There are also some works focusing on cryptocurrency in recent years. We classify existing works based on data frequencies (Figure 3.1c). Around half of papers use day-level data since it is more accessible. For order execution and market making, fine-grained data (e.g., second-level and minute-level) are widely used to simulate the micro-level market dynamics.



(A) Financial markets (B) Financial assets (C) Data frequencies

FIGURE 3.1: Categorization of existing works [1]

3.1.2 RL in Algorithmic Trading

Algorithmic trading refers to trade one particular financial asset with signals generated automatically by computer programs. It has been widely applied in

trading all kinds of financial assets. We present a review of most notable RL-based algorithmic trading papers dating back to 1990s.

Policy-based methods. To tackle the limitations of supervised learning methods, Moody and Wu [79] made the first attempt to apply RL in algorithmic trading. An agent is trained with recurrent RL (RRL) to optimize the overall profit directly with a novel evaluation metrics called differential Sharpe ratio. Empirical study on artificial price data shows that it outperforms previous prediction-based methods. With the same algorithm, further experiments are conducted using monthly S&P stock index data [80] and US Dollar/British Pound exchange data [81]. As an extension of RRL [79], Dempster and Leemans [82] proposed an adaptive RRL framework with three layers: i) Layer one adds 14 technical indicators as extra market information; ii) Layer two evaluates trading actions from layer one with consideration of risk factors; iii) The goal of layer three is to search for optimal values of hyperparameters in layer two. With the three-layer architecture, it outperforms baselines on Eur/ US Dollar exchange data. Vittori et al. [89] proposed a risk-averse algorithm called Trust Region Volatility Optimization (TRVO) for option hedging. TRVO trains a sheaf of agents characterized by different risk aversion methods and is able to span an efficient frontier on the volatility-p&l space. Simulation results demonstrate that TRVO outperforms the classic Black & Scholes delta hedge algorithm [103].

With the development of deep learning, several deep RL methods are proposed for algorithmic trading. FDDR [2] enhanced the classic RRL method [80] with deep neural networks. An RNN layer is used to learn meaningful recurrent representations of the market. In addition, a fuzzy extension is proposed to further reduce the uncertainty. FDDR achieves great performance on both stock index and commodity futures. To balance between profit and risk, a multi-objective RL method with LSTM layers [83] is proposed. Through optimizing profit and Sharpe Ratio simultaneously, the agent achieves better performance on 3 Chinese stock index futures.

Value-based methods. QSR [58] used Q-learning to optimize absolute profit and relative risk-adjusted profit respectively. A combination of two networks is employed to improve performance on US Dollar/German Deutschmark exchange data. Lee and Jangmin [59] proposed a multi-agent Q-learning framework for stock trading. Four cooperative agents are designed to generate trading signals and order

prices for both buy and sell side. Through sharing training episodes and learned policies with each other, this method achieves better performance in terms of both profit and risk management on the Korea stock market compared to baselines. In [60], the authors firstly designed some local traders based on dynamic programming and heuristic rules. Later on, they applied Q-learning to learn a meta policy of these local traders on Korea stock markets. de Oliveira et al. [69] implemented a SARSA-based RL method and tested it on 10 stocks in the Brazil market.

DQN is used to enhance trading systems by considering trading frequencies, market confusion and transfer learning [72]. One heuristic function is applied to add a filter as the agent's certainty on market condition. Moreover, the authors train the agent on selected component stocks and apply the pre-train weights as the starting point for different stock indexes. Experiments on 4 different stock indexes demonstrate the effectiveness of the proposed framework. DeepScalper [12] applied branch dueling Q-network [104] for intraday trading with an encoder-decoder architecture to learn market embedding. A novel hindsight bonus is added in the reward function to encourage long-term horizon. The authors also design an auxiliary task by predicting future volatility. DeepScalper significantly outperforms many baselines on Chinese treasury bond and stock index markets. Riva et al. [68] proposed a value-based RL algorithm to train agents for FX trading via fitted-Q iteration. The importance of tuning control frequency is studied in order to obtain effective trading policies. EarnHFT [78] introduced a three-stage hierarchical RL framework for high frequency Crypto trading. It trained low-level agents for different market conditions (e.g., bull/bear) and high-level agents to dynamically pick agents from the pool. The algorithm outperform state-of-art methods with over 30% improvement on 4 Cyrpto pairs.

Other methods. iRDPG [3] is an adaptive DPG-based framework. The authors formulate algorithmic trading as a Partially Observable Markov Decision Process (POMDP). GRU layers are integrated into iRDPG to learn recurrent market embedding. In addition, the authors apply behavior cloning with expert trading actions to guide iRDPG and achieve great performance on two Chinese stock index futures. There are also some works focusing on empirical study to evaluating the performance of different RL algorithms on new datasets. Zhang et al. [75] evaluated DQN, PG and A2C on the 50 most liquid futures contracts. Yuan et al. [76] tested

PPO, DQN and SAC on three selected stocks. DQN achieves the best overall performance among different financial assets.

Summary. Although existing works demonstrate the potential of RL for quantitative trading, there is seemingly no consensus on a general ranking of different RL algorithms (notably, we acknowledge that *no free lunch theorem* exists). The summary of algorithmic trading publications is in Table 3.2. In addition, most existing RL-based works only focus on general AT, which tries to make profit through trading one asset. In finance, extensive trading strategies have been designed based on trading frequency (e.g., high-frequency trading) and asset types (e.g., stock and cryptocurrency).

Paper	RL method	Asset type	Market	Data frequency
[79]	RRL	Artificial	-	-
[80]	RRL	Stock Index	USA	1 Month
[58]	Q-learning	FX	-	1 Day
[81]	RRL	FX	-	30 Min
[59]	Multi-agent RL	Stock Index	Korea	-
[60]	Q-learning	Stock Index	Korea	-
[82]	RRL	FX	-	1 Min
[61]	Q-learning	Artificial Stock	- Italy	- 1 Day
[2]	RRL	Stock Index Commodity	China	1 Min
[83]	RRL	Stock Index	China	1 Min
[72]	DQN	Stock Index	USA, Hong Kong Europe, Korea	1 Day
[69]	SARSA	Stock	Brazil	15 Min
[3]	DDPG	Stock Index	China	1 Min
[89]	TRPO	Artificial	-	-
[75]	DQN, PG, A2C	Stock Index Commodity	-	-
[76]	PPO, DQN, SAC	Stock	China	1 Day
[12]	DQN	Stock Index Treasury Bond	China	1 Min
[68]	Q-learning	FX	-	1 Min
[78]	DDQN	Cryptocurrency	-	1 Sec

TABLE 3.2: Summary of RL for algorithmic trading

3.1.3 RL in Portfolio Management

Portfolio management, which studies the art of balancing between a collection of different financial assets, has become a popular topic for RL researchers. We survey on existing works on RL for portfolio management.

Policy-based methods. Since a portfolio is essentially a weight distribution among different financial assets, policy-based methods are the most widely applied RL methods for PM. Almahdi and Yang [105] proposed an RRL-based algorithm for portfolio management with maximum drawdown as the optimization object to measure downside risk. Furthermore, an adaptive version is designed with a transaction cost and market condition stop-loss retraining mechanism. Investor-Imitator [13] formalized the trading knowledge by imitating the behavior of an investor with a set of logic descriptors. To further instantiate specific logic descriptors, the authors introduced a rank-invest model that can keep the diversity of different logic descriptors through optimizing a variety of evaluation metrics with RRL. Experiments on the Chinese stock market show that Investor-imitator successfully extracts interpretable knowledge of portfolio management that can help human traders better understand the financial market. Alphastock [106] is another policy-based RL method for portfolio management with a cross-asset attention network to describe the interrelationships among stocks. Policy gradient is used to optimize the Sharpe Ratio. Experiments on both US and Chinese stock market show that Alphastock achieves robust performance over different market states. EI^3 [84] is another RRL-based method, which tries to build profitable cryptocurrency portfolios by extracting multi-scale patterns in the financial market. The authors design a multi-scale temporal feature aggregation convolution framework with two CNN branches to extract short-term and mid-term market embedding and a max pooling branch to extract the highest price information. To bridge the gap between the traditional Markowitz portfolio and RL-based methods, Benhamou et al. [86] applied PG with a delayed reward function and showed better performance than the classic Markowitz efficient frontier.

Zhang et al. [87] proposed a cost-sensitive PM framework based on direct policy gradient. To learn more robust market representation, a novel two-stream portfolio policy network is designed to extract both price series pattern and the relationship between different financial assets with a cost-sensitive reward function to take the trading cost constraint into consideration with theoretically near-optimal guarantee. Xu et al. [85] proposed a novel relation-aware transformer under the classic RRL paradigm to capture both sequential patterns and the inner corrections between financial assets. Specifically, it follows an encoder-decoder structure, where the encoder is for sequential feature extraction and the decoder is for decision making. Bisi et al. [90] derived a PG theorem with a novel objective function, which exploited

the mean-volatility relationship. The new objective could be used in actor-only algorithms such as TRPO with monotonic improvement guarantees. Wang et al. [4] proposed DeepTrader, a PG-based DRL method, to tackle the risk-return balancing problem in PM. The model simultaneously uses negative maximum drawdown and price rising rate as reward functions to balance between profit and risk. The authors propose an asset scoring unit with graph convolution layer to capture temporal and spatial interrelations among stocks. Moreover, a market scoring unit is designed to evaluate the market condition.

Actor-critic methods. Jiang et al. [91] proposed a DPG-based RL framework consists of 3 novel components: i) the ensemble of identical independent evaluators topology; ii) a portfolio vector memory; iii) an online stochastic batch learning scheme. To demonstrate the effectiveness of proposed components, extensive experiments using different neural network architectures are conducted on cryptocurrency data. Later on, extra experiments are conducted in an extended version [92]. To model the data heterogeneity and environment uncertainty in PM, Ye et al. [93] proposed a state-augmented RL (SARL) framework based on DPG. SARL learns the price movement prediction with financial news as additional states. Another popular actor-critic RL method for portfolio management is DDPG. Xiong et al. [96] constructed a highly profitable portfolio with DDPG on the Chinese stock market. PROFIT [97] is another DDPG-based approach that makes time-aware decisions on PM with text data. The authors make use of a custom policy network that hierarchically and attentively learns time-aware representations of news and tweets for PM, which is generalizable among various actor-critic RL methods. PROFIT shows promising performance on both China and US stock markets. Murray et al. [107] proposed a novel actor-critic algorithm for solving general risk-reverse stochastic control problems and use it to learn hedging strategies for portfolio management across multiple risk aversion levels simultaneously. EarnMore [98] is proposed to handle customize stock pool scenarios with masked stock representations. The authors introduced a mechanism to mask out the representation of the stocks outside the target pool and learn meaningful stock representations through a self-supervised masking and reconstruction process. Moreover, a re-weighting mechanism is designed to make the portfolio concentrate on favorable stocks and neglect the stocks outside the target pool.

Other methods. Neuneier [62] made an attempt to formalize portfolio management as an MDP and trained an RL agent with Q-learning. Experiments on German stock market demonstrate its superior performance over heuristic benchmark policy. Later on, a shared value-function for different assets and model-free policy-iteration are applied to further improve the performance of Q-learning in [63]. Yu et al. [99] proposed the first model-based RL framework for portfolio management, which supports both off-policy and on-policy settings. The authors designed an infused prediction module to predict future price, a data augmentation module with recurrent adversarial networks to mitigate the data deficiency issue, and a behavior cloning module to reduce the portfolio volatility. MetaTrader [77] introduced a two-stage RL-based approach, which learns to integrate diverse trading policies to adapt to various market conditions. MAPS [101] proposed a cooperative multi-agent RL system in which each agent is an independent investor creating its own portfolio. The authors design a novel loss function to guide each agent to act as diversely as possible while maximizing its long-term profit. MAPS outperforms most of baselines with 12 years of US stock market data. MSPM [102] is a multi-agent RL framework with a modularized and scalable architecture for PM. MSPM consists of the evolving agent module to learn market embedding with heterogeneous input and the strategic agent module to produce profitable portfolios.

Summary. Policy-based methods are the most widely-used RL methods for PM. There are also many successful examples based on actor-critic algorithms. The summary of publications is in Table 3.3. We point out two limitations of existing methods: i) most of them ignore the interrelationship between different financial assets, which is valuable for human portfolio managers. ii) existing works construct portfolios from a relative small pool of stocks (e.g., 20 in total). However, the real market contains thousands of stocks and common RL methods are vulnerable when the action space is very large [108].

3.1.4 RL in Order Execution

Different from algorithmic trading and portfolio management, order execution (OE) is a micro-level QT task, which tries to trade a fixed amount of shares in a given time horizon and minimize the execution cost. In the real financial market, OE is

Reference	RL method	Asset type	Market	Data frequency
[62]	Q-learning	Stock Index	Germany	1 Day
[63]	Q-learning	Stock Index	Germany	1 Day
[91]	DPG	Cryptocurrency	-	30 Min
[105]	RRL	Stock Index	-	-
[92]	DPG	Cryptocurrency	-	30 Min
[88]	DDPG, PPO, PG	Stock	China	1 Day
[13]	RRL	Stock	China	1 Day
[96]	DDPG	Stock	USA	1 Day
[106]	Technical Indicator	Stock	China, USA	-
[99]	Model-based RL	Stock	USA	1 Hour
[84]	RRL	Cryptocurrency	-	-
[86]	PG	-	-	1 Day
[87]	PG	Cryptocurrency	-	-
[95]	PPO, A2C, DDPG	Stock	USA	1 Day
[101]	Multi-agent RL	Stock	USA	1 Day
[97]	DDPG	Stock	USA, China	1 Min
[93]	DPG	Stock Cryptocurrency	USA -	1 Day 30 Min
[85]	RRL	Stock Cryptocurrency	USA -	30 Min
[5]	REINFORCE, DQN	Stock	USA, China	1 Day
[90]	TRPO	Artificial Stock Index	- USA	- 1 Day
[4]	PG	Stock	USA, China Hong Kong	-
[102]	Multi-agent RL	Stock	USA	1 Day
[77]	DQN	Stock	USA, China Hong Kong	-
[107]	Actor-Critic	Artificial	-	
[98]	SAC	Stock	USA	1 Day

TABLE 3.3: Summary of RL for portfolio management

extremely important for institutional traders whose trading volume is large enough to have an obvious impact on the market price.

Nevmyvaka et al. [64] proposed the first RL-based method for large-scale order execution. The authors used Q-learning to train the agent with real-world LOB data. With carefully designed state, action and reward function, the Q-learning framework can significantly outperform traditional baselines. Hendricks and Wilcox [65] implemented another Q-learning based RL method on South Africa market by extending the popular Almgren-Chriss model with linear price impact. Ning et al. [73] proposed an RL framework using double DQN and evaluated its performance on 9 different US stocks. Dabérius et al. [74] implemented DDQN, PPO and compared their performance with TWAP.

PPO is another widely used RL method for OE. Lin and Beling [94] proposed an

Reference	RL method	Asset Type	Market	Data frequency
[64]	Q-learning	Stock	USA	Millisecond
[65]	Q-learning	Stock	South Africa	5 Min
[73]	DQN	Stock	USA	1 Second
[74]	DDQN, PPO	Artificial	-	-
[94]	PPO	Stock	USA	Millisecond
[7]	PPO	Stock	China	1 Min

TABLE 3.4: Summary of RL for order execution

end-to-end PPO-based framework. MLP and LSTM are tested as time dependencies accounting network. The authors design a sparse reward function instead of previous implementation shortfall (IS) or a shaped reward function, which leads to state-of-the-art performance on 14 stocks in the US market. Fang et al. [7] proposed another PPO-based framework to bridge the gap between the noisy yet imperfect market states and the optimal action sequences for OE. The framework leverages a policy distillation method with an entropy regularization term in the loss function to guide the student agent toward learning similar policy by an oracle teacher with perfect information of the financial market. Moreover, the authors design a normalized reward function to encourage universal learning among different stocks. Extensive experiments on Chinese stock market demonstrate that the proposed method significantly outperforms various baselines with reasonable trading actions.

We present a summary of existing RL-based order execution publications in Table 3.4. Although there are a few successful examples using either Q-learning or PPO on order execution, existing works share a few limitations. First, most of algorithms are only tested on stock data. Their performance on different financial assets (e.g., futures and cryptocurrency) is still unclear. Second, the execution time window (e.g., 1 day) is too long, which makes the task easier. In practice, professional traders usually finish the execution process in much shorter time window (e.g., 10 minute). Third, existing works will fail when the trading volume is huge, because all of them assume there is no obvious market impact, which is impossible for large volume settings. In the real-world, the requirement of institutional investors is to execute large amount of shares in a relatively short time window. There is still a long way to go for researchers to tackle these limitations.

3.1.5 RL in Market Making

Market making refers to trading activities that buy and sell one given asset simultaneously at desired price. The goal of a market maker is to provide liquidity to the market and market profit through the tiny price spread of buy/sell orders. We discuss existing RL-based methods for market making.

Chan and Shelton [70] made the first attempt to apply RL for market making without any assumption of the market. Simulation showed that the RL method converged on optimal strategies successfully on a few controlled environments. Spooner et al. [43] focused on designing and analyzing temporal-difference (TD) RL methods for market making. The authors firstly build a realistic, data-driven simulator with millisecond LOB data for market making. With an asymmetrically dampened reward function and a linear combination of tile coding as state, both Q-learning and SARSA outperform previous baselines. Lim and Gorse [66] proposed a Q-learning based algorithm with a novel usage of CARA utility as the terminal reward for market making. Guéant and Manziuk [100] proposed a model-based actor-critic RL algorithm, which focuses on market making optimization for multiple corporate bonds. Zhong et al. [67] proposed a model-free and off-policy Q-learning algorithm to develop trading strategy implemented with a simple lookup table. The method achieves great performance on event-by-event LOB data confirmed by a professional trading firm. For training robust market making agents, Spooner and Savani [71] introduced a game-theoretic adaptation of the traditional mathematical market making model. The authors thoroughly investigate the impact in 3 environmental settings with adversarial RL. Zhao and Linetsky [109] proposed a high-frequency feature called book exhaustion rate, which can serve as a direct measurement of the adverse selection risk from an equilibrium point of view. The authors train a market making agent via RL using three years of LOB data on Chicago Mercantile Exchange S&P 500 and achieve stable performance.

Even though market making is a fundamental task in quantitative trading, research on RL-based market making is still at the early stage. Existing works simply apply different RL methods on their own data. The summary of order execution publications is in Table 3.5. To fully realize the potential of RL for market making, one major obstacle is the lack of high-fidelity micro-level market simulator. At present, there is still no reasonable way to simulate the ubiquitous market impact.

This unignorable gap between simulation and real market limits the usage of RL in market making.

Reference	RL method	Asset Type	Market	Data frequency
[70]	SARSA	Artificial	-	-
[43]	Q-learning, SARSA	-	-	Millisecond
[66]	Q-learning	Artificial	-	-
[100]	Model-based RL	Bond	Europe	-
[67]	Q-learning	-	-	Event
[71]	SARSA	Artificial	-	-
[109]	-	Stock & Treasury	US	5 Min

TABLE 3.5: Summary of RL for market making

3.2 Deep Learning for Financial Prediction

Extensive deep learning methods have been proposed for stock prediction, which can be generally categorised into: i) recurrent neural networks (RNN), ii) non-recurrent neural networks (NRNN), iii) models with alternative data sources, iv) enhancing traditional finance methods.

RNN models. RNN-based models are popular for stock prediction since they are specifically designed to capture temporal patterns in sequential data. Nelson et al. [110] showed that GRU outperforms many baseline methods. Wang et al. [111] combined attention mechanism with LSTM to model correlated time steps. To improve the robustness of LSTM, Feng et al. [45] applied adversarial training techniques for stock prediction. Zhang et al. [112] proposed a State Frequency Memory (SFM) recurrent network with Discrete Fourier Transform (DFT) to discover multi-frequency patterns in stock markets. Huynh et al. [113] proposed a framework for efficient integration of multi-order dynamics and internal dynamics of stock markets, which includes temporal generative filters on LSTM to learn individual patterns per stock and hypergraph attentions to capture the non-pairwise correlations.

NRNN models. There are also many DL approaches that make stock prediction without using RNN-based models. Liu et al. [114] introduced a multi-scale two-way neural network (NN) to predict the stock trend. Ding et al. [115] proposed a hierarchical Gaussian transformer-based model for stock movement prediction.

Another direction of work employs graph-based DL models to model pair-wise relations between stocks. For instance, Feng et al. [116] enhanced graph convolutional networks (GCNs) with temporal convolutions for mining inter-stock relations. Sawhney et al. [117] focused on stock industry data and links between company CEOs. MASTER [118] is proposed as market-guided stock transformer, which leverages the momentary and cross-time stockc correlation for feature selection. Sawhney et al. [119] proposed a spatiotemperal convolution network architectures for stock movement prediction.

Models with alternative data. To further show the potential of deep learning methods, the usage of additional data sources is another direction for trading signal discovery. Xu and Cohen [120] proposed a variational autoencoder architecture to extract latent information from tweets. Chen et al. [121] enhanced trading strategy design with the investment behaviors of professional fund managers. Economics news [122] are used to enhance the prediction accuracy. Earning calls information [123] improved the performance of volatility prediction. Recently, large language models (e.g., GPT4) become one new source of information, where FinAgent [124] are proposed to improve financial prediction.

Enhancing traditional finance methods. Another research direction is to enhance traditional rule-based methods with ML techniques. Lim et al. [125] enhanced time-series momentum with deep learning. Takeuchi and Lee [126] applied NN to enhance cross section momentum. Chauhan et al. [127] took account uncertainty and look-ahead based on factor models. Alphastock [106] proposed a deep reinforcement attention network to improve the classic buying-winners-and-selling-losers strategy [24]. Gu et al. [128] explore ML techniques' ability on asset pricing. In [129], an autoencoder architecture was proposed for asset pricing. Compared to pure ML methods, these methods keep the original financial insight and have better explainability.

3.3 Decision Making with Multiple Experts

3.3.1 Ensemble Learning

Ensemble methods, which combine the output of different models to improve the overall generalization performance, have been studied for decades [130–136]. DeepEns [137] showed its potential for uncertainty estimation. Although ensemble methods achieve stellar performance when each member makes independent errors [138], the significant increase of computation and memory cost becomes a major bottleneck for the wide deployment of ensemble methods. In recent years, the major direction in ensemble research is how to reduce the cost of training, testing and memory. Snapshot ensemble [139], in which a single model is trained by cyclic learning rates, is proposed to encourage visiting multiple local minima within one run. Wen et al. [133] proposed an efficient mini-batch friendly ensemble method called BatchEns with an ensemble weight generation mechanism using Hadamard product. Recent works highlight the benefit of ensemble models with hyperparameter diversity [134] and architecture diversity [135], respectively. Instead of combining the outputs of neural networks, model soup [140] directly average the weights of neural networks and achieves great performance in pretrain settings with no extra inference cost.

Although there is a lot of progress in ensemble learning, most existing ensemble methods for stock markets [141] are only straightforward applications of traditional methods such as bagging and stacking with linear increase of both computation and memory cost. For instance, Xiang and Fu [142] combined different neural networks as sub-models with bagging for stock market prediction. There is still a lack of advanced efficient ensemble learning methods for capturing the fleeting trading opportunities in financial markets. AlphaMix is designed by customizing advanced ensemble techniques [133, 134, 140, 143, 144] into this filed to fill the gap.

3.3.2 Mixture of Experts

Ever since its introduction more than two decades ago [145], the mixture-of-experts (MoE) approach has been the key component for many works. Eigen et al. [146] extended MoE into deep learning by stacking two layers of mixture of experts with

a trainable weighted gating network. Shazeer et al. [147] presented a sparsely-gated MoE layer to enable training extremely large number of experts. Ma et al. [148] proposed a multi-gate MoE framework to model task relationships in multi-task learning. Xu et al. [149] proposed an adaptive Master-Slave regularized model for unexpected revenue prediction enhanced with alternative data. MoE models have achieved stellar performance in many challenging fields such as computer vision [143, 150, 151] and natural language processing [147, 152].

However, even though MoE methods have been successful in many fields, there is a lack of MoE frameworks for quantitative investment in the stochastic financial market. AlphaMix is the first MoE framework to fill this gap with three stages to mimic the efficient hierarchical bottom-up trading strategy design workflow in real-world trading firms.

3.4 Systematic Evaluation of RL Methods

Reinforcement learning has attracted increasing attention from both academia and financial industries [1] due to its stellar performance on solving complex sequential problems such as Go [153], StarCraft-II [54], nuclear fusion [154] and matrix multiplication [155]. However, the performance of RL is unstable and require systematic evaluation before real-world deployment. There are many existing works focusing on rigorous evaluation in RL. Henderson et al. [156] highlights various reproducibility issues in RL including hyperparameters, network architectures, reward scale and random seeds. Colas et al. [157] studies the minimum number of random seeds required to report results with statistical significance. Agarwal et al. [158] recommend for reporting interval estimates of aggregate performance and propose performance profiles for reliable RL evaluation with a handful of runs. Interquartile mean scores is proposed as one robust and efficient aggregate metrics to achieve small uncertainty in RL experimental results. Chan et al. [159] propose metrics to measure the reliability of RL algorithms in terms of their stability during training and their variability and risk in returns across multiple episodes. Jordan et al. [160] propose a game-theoretic evaluation procedure for complete algorithms that do not require any hyperparameter tuning and recommend evaluating between 1000 to 10000 runs per task to detect statistically significant results. A metric called SharpeRatio@k is designed to measure the risk-return

tradeoff of policy portfolios formed by off-policy evaluation estimators under varying online evaluation budgets [161].

3.5 Financial Trading Software and Platforms

There has been many attempts on financial trading platforms. In the early stages, many open-source libraries are proposed for traditional finance approaches [162], event-driven backtesting [163] and portfolio risk analysis [164]. With the advent of AI, recent financial trading platforms mostly focus on data-driven machine learning techniques. OLPS [165] presents a toolbox for online portfolio selection that implements a collection of state-of-the-art online strategies powered by machine learning. The toolbox is implemented in Matlab. Qlib [18] proposes an AI-oriented quantitative investment platform with a focus on prediction-based supervised learning settings. The library contains the whole pipeline of quantitative investment covering a wide range of deep learning models and financial tasks. FinRL [21] makes the first attempt on developing an RL-based quantitative trading platform. It supports five trading applications with implementation of many classic RL algorithms. FinRL-Meta [22] offers a series of data-driven RL environments for various trading scenarios. The focus of FinRL-Meta is to provide realistic simulators as testbed of RL algorithms for trading.

However, due to the existence of alpha decay phenomenon [19], financial practitioners are never keen to share source code of their algorithms and platforms [18] that achieve good performance in real-world settings. There is still a long way to go for open-source trading platforms.

Chapter 4

Reinforcement Learning for Intraday Trading

4.1 Introduction

In the last decade, there has been significant development of quantitative trading [166], due to its instant and accurate order execution, and capability of analyzing and processing large amount of temporal financial data. Especially, intraday trading, where traders actively long/short one pre-selected financial assets (at least a few times per day) to seize intraday trading opportunities, becomes one of the most profitable and risky quantitative trading tasks with the growth of discount brokerages (lower commission fee). In this chapter[1], we study the problem of FinRL algorithms for intraday trading, which is a specific algorithmic trading task with intraday trading decisions.

Traditional intraday trading strategies adopt finance insights to discover trading opportunities with heuristic rules. For instance, momentum [25] trading is designed based on the assumption that the trend of financial assets in the past has the tendency to continue in the future. Mean reversion [27] strategies focusing on the investment opportunities at the turning points. However, rule-based traditional methods exhibit poor generalization ability and only perform well in some certain market conditions [2]. Another popular direction is to trade based on financial

[1]This chapter has been published in [12]

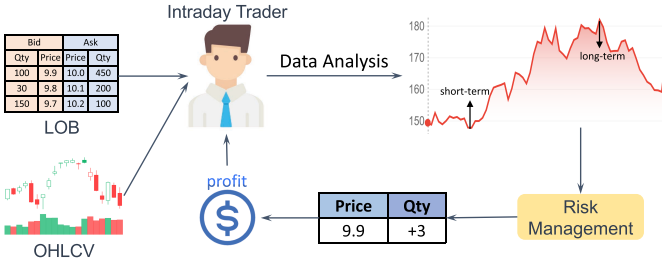


FIGURE 4.1: Workflow of professional human intraday trader

prediction. Many advanced machine learning methods such as GCN [116], Transformer [115] and LGBM [167] have been introduced for predicting future prices [168]. External data sources such as economic news [122], frequency of price [112], social media [120] and investment behaviors [121] are incorporated as extra information to further enhance prediction performance. Nevertheless, the high volatility and noisy nature of the financial market make it extremely difficult to accurately predict future prices [169]. There is a noticeable gap between prediction signals and profitable trading actions [116]. Thus, the overall performance of prediction-based methods is not satisfying as well.

Similar to other application scenarios of RL, quantitative trading also interacts with the environment (financial market) and maximizes the accumulative profit. Recently, FinRL has been considered as an attractive approach to quantitative trading as it allows training agents to directly output profitable trading actions with better generalization ability across various market conditions [3]. Although there have been many successful RL-based quantitative trading methods [5, 7, 93], a vast majority of existing methods focus on the relatively low-frequency trading scenarios (e.g., day-level) and fail to capture the fleeting intraday investment opportunities. To design profitable RL methods for intraday trading, there are two major challenges. First, different from the low-frequency scenarios, intraday trading involves a much larger high-dimensional fine-grained action space to represent the price and quantity of orders for more accurate control of the financial market. Second, we need to learn meaningful multi-modality intraday market representation, which takes macro-level market information, micro-level market information and risk into consideration.

We show the workflow of a professional human intraday trader in Figure 4.1. The trader first collects both micro-level and macro-level market information to analyze the market status. Then, he predicts the short-term and long-term price trend based on the market status. Later on, he conducts risk management and makes

final trading decisions (when and how much to long/short at what price). Among many successful trading firms, this workflow plays a key role for designing robust and profitable intraday trading strategies. Inspired by this, we propose DeepScalper, a novel RL framework for intraday trading to tackle the above challenges by mimicking the information collection, short-term and long-term market analysis and risk management procedures of human intraday traders. Our main contributions are three-fold:

- We apply the dueling Q-network with action branching to effectively optimize intraday trading agents with high-dimensional fine-grained action space. A novel reward function with hindsight bonus is designed to encourage RL agents making trading decisions with a long-term horizon of the entire trading day.
- We propose a multi-modality encoder-decoder architecture to learn meaningful temporal intraday market embedding, which incorporates both micro-level and macro-level market information. Furthermore, we design a risk-aware auxiliary task to keep balance between profit and risk.
- Through extensive experiments on real-world market data spanning over three years on six financial futures, we show that DeepScalper significantly outperforms many state-of-the-art baseline methods in terms of four financial criteria and demonstrate DeepScalper's practical applicability to intraday trading with a series of exploratory and ablative studies.

4.2 Problem Formulation

In this section, we first introduce the objective of intraday trading. Next, we provide a Markov Decision Process (MDP) formulation of intraday trading.

4.2.1 Intraday Trading

Intraday trading is a fundamental quantitative trading task, where traders actively long/short one pre-selected financial asset within the same trading day to maximize future profit. In real-world intraday trading, traders are allocated some cash into the account at the beginning of each trading day. During the trading time, traders

analyze the market and make their trading decisions. Then, they submit their orders (desired price and quantity) to the matching system. The matching system will execute orders at best available price (possibly at multiple price when market liquidation is not enough for large orders) and then return execution results to traders. At the end of the trading day, traders close all remaining positions at market price to avoid overnight risk and hold 100% cash again. The objective of intraday trading is to maximize accumulative profit for a period of multiple continuous trading days (e.g., half a year). Comparing to conventional low-frequency trading scenarios, intraday trading is more challenging since intraday traders need to capture the tiny price fluctuation with much less responsive time (e.g., 1 min). In addition, intraday trading involves a large fine-grained trading action space that represents a limit order to pursue more accurate control of the market.

4.2.2 MDP Formulation

We formulate intraday trading as a MDP, which is constructed by a 5-tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$. Specifically, $\mathcal{S}$ is a finite set of states. $\mathcal{A}$ is a finite set of actions. $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ is a state transaction function, which consists of a set of conditional transition probabilities between states. $R : \mathcal{S} \times \mathcal{A} \longrightarrow \mathcal{R}$ is the reward function, where $\mathcal{R}$ is a continuous set of possible rewards and R indicates the immediate reward of taking an action in a state. $\gamma \in [0, 1)$ is the discount factor. A (stationary) policy $\pi : \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$ assigns each state $s \in \mathcal{S}$ a distribution over actions, where $a \in \mathcal{A}$ has probability $\pi(a|s)$. In intraday trading, $\mathcal{S}, \mathcal{A}, R$ are set as follows.

State. Due to the particularity of the financial market, the state $s_t \in \mathcal{S}$ at time t can be divided into three parts: macro-level market state $s_t^a \in \mathcal{S}^a$, micro-level market state $s_t^i \in \mathcal{S}^i$ and trader's private state set $s_t^p \in \mathcal{S}^p$. Specifically, we use a vector y_t of 11 technical indicators described in chapter 2 and the original OHLCV vector $\mathbf{x_t}$ as macro-level state following [46], the historical LOB sequence $(\mathbf{b_{t-h}}, ..., \mathbf{b_t})$ with length $h + 1$ as micro-level state and trader's private state $\mathbf{z_t} = (pos_t, c_t, t_t)$, where pos_t, c_t and t_t are the current position, cash and remaining time. The entire set of states can be denoted as $\mathcal{S} = (\mathcal{S}^a, \mathcal{S}^i, \mathcal{S}^p)$. Compared to previous formulations, we introduce the LOB and trader's private state as additional information to effectively capture intraday trading opportunities.

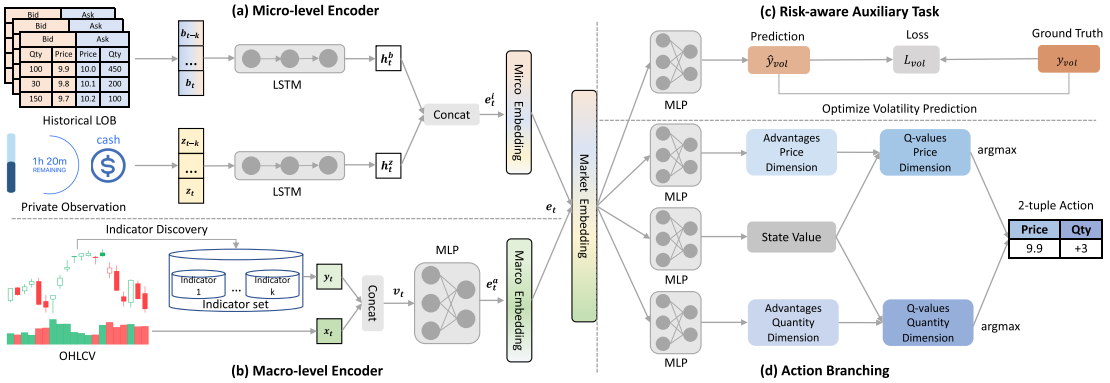


FIGURE 4.2: An overview of the proposed DeepScalper framework. We show four individual building blocks: (a) micro-level encoder, (b) macro-level encoder, (c) risk-aware auxiliary task, and (d) RL optimization with action branching.

Action. Previous works [2, 3] lie in low-frequency trading scenarios, which generally stipulate that the agent trades a fixed quantity at market price and applies a coarse action space with three options (long, hold, and short). However, when focusing on relatively high-frequency trading scenarios (e.g., intraday trading), tiny price fluctuation (e.g., 1 cent) is of vital importance to final profit that makes the market price execution and fixed quantity assumptions unacceptable. In practice, traders have the freedom to decide both the target price and the quantity by submitting limit orders. We use a more realistic two-dimensional fine-grained action space for intraday trading, which represents a limit order as a tuple $(p_{target}, \pm q_{target})$. p_{target} is the target price, q_{target} is the target quantity and $\pm$ is the trading direction (long/short). It is also worth pointing out that when the quantity is zero, it indicates that we skip the current time step with no order placement.

Reward. We define the reward function as the change of account pnl, which shows the value fluctuation (profit & loss) of the account:

$$r_t = \underbrace{(p^c_{t+1} - p^c_t) \times pos_t}_{instant\ \ profit} - \underbrace{\delta \times p^c_t \times \left|pos_t - pos_{t-1}\right|}_{transaction\ \ fee}$$

where p^c_t is the close price at time t, δ is the transaction fee rate and pos_t is the position at time t.

4.3 Risk-Aware RL Framework

In this section, we introduce DeepScalper framework as shown in Figure 4.2 for intraday trading. We describe the four components of DeepScalper: i) RL optimization with action branching; ii) reward function with a hindsight bonus; iii) intraday market embedding; iv) risk-aware auxiliary task orderly.

4.3.1 RL Optimization with Action Branching

Different from day-level scenarios, intraday trading tries to seize the fleeting tiny price fluctuation with much less responsive time. To provide more accurate trading decisions, intraday trading involves a much larger two-dimensional (price and quantity) fine-grained action space. However, learning from scratch for tasks with large action spaces remains a critical challenge for RL algorithms [170, 171]. For intraday trading, while human traders can usually detect the subset of feasible trading actions in a given market condition, RL agents may attempt inferior actions, thus wasting computation time and leading to capital loss.

As possible intraday trading actions can be naturally divided into two components (e.g., desired price and quantity), we propose to adopt the branching dueling Q-network (BDQ) [172] for decision-making. Particularly, as shown in Figure 4.2 (d), BDQ distributes the representation of the state-dependent action advantages in both the price and quantity branches. Later, it simultaneously adds a single additional branch to estimate the state-value function. Finally, the advantages and the state value are combined via an aggregating layer to output the Q-values for each action dimension. During the inference period, these Q-values are then queried with argmax to generate a joint action tuple to determine the final trading actions.

Formally, intraday trading is formulated as a sequential decision making problem with two action dimensions of $|p| = n_p$ discrete relative price levels and $|q| = n_q$ discrete quantity proportions. The individual branch's Q-value Q_d at state $s \in S$ and the action $a_d \in \mathcal{A}_d$ are expressed in terms of the common state value $V(s)$ and the corresponding (state-dependent) action advantage [104] $Adv_d(s, a_d)$ for $d \in \{p, q\}$:

$$Q_d(s, a_d) = V(s) + (Adv_d(s, a_d) - \frac{1}{n} \sum_{a_d^{'} \in A_d} Adv_d(s, a_d^{'}))$$

We train our Q-value function approximator as Q-network with parameter θ_q based on the one-step temporal-difference learning with target y_d in a recursive fashion:

$$y_d = r + \gamma \max_{a_d' \in A_d} Q_d^-(s', a_d', \theta_q)), d \in \{p, q\}$$

where Q_d^- denoting the branch d of the target network Q^-, r is the reward function result and γ is the discount factor.

Finally, we calculate the following loss function:

$$L_q(\theta_q) = E_{(s,a,r,s')\sim D}[\frac{1}{N} \sum_{d \in \{p,q\}} (y_d - Q_d(s, a_d, \theta_q))^2]$$

where D denotes a prioritized experience replay buffer. a denotes the joint-action tuple (p, q). By differentiating the Q-network loss function with respect to θ_q, we get the following gradient:

$$\begin{aligned} \bigtriangledown_{\theta_q} L_q(\theta_q) = E_{(s,a,r,s')\sim D}[(r + \gamma \max_{a_d' \in A_d} Q_d(s^{'}, a_d^{'}, \theta_q) \\ -Q_d(s, a_d, \theta_q)) \bigtriangledown_{\theta_q} Q_d(s, a_d, \theta_q)] \end{aligned}$$

In practice, we optimize the loss function by stochastic gradient descent, rather than computing the full expectations in the above gradient, to maintain computational efficiency.

4.3.2 Reward Function with Hindsight Bonus

One major issue for training directly with the profit & loss reward is that RL agents tend to pay too much attention to the short-term price fluctuation [4]. Although the agent performs well in capturing local trading opportunities, ignoring the overall long-term price trend could lead to significant loss. Here, we design a novel reward function with a hindsight bonus to tackle this issue. To demonstrate the motivation

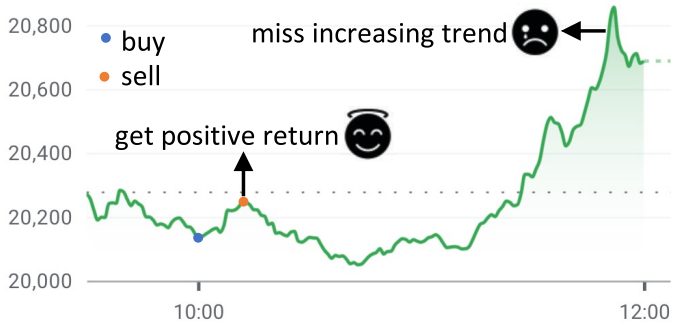


FIGURE 4.3: Illustration of the motivation of the hindsight bonus

of the hindsight bonus, considering a pair of buy/sell actions in Figure 4.3, the trader feels happy at the point of selling the stock, since the price of the stock increases. However, this sell decision is actually a bad decision in the long run. The trader feels disappointed before 12:00 since he/she misses the main increasing wave due to the short horizon. It is more reasonable for RL agents to evaluate one trading action from both short-term and long-term perspectives. Inspired by this, we add a hindsight bonus, which is the expected profit for holding the assets for a longer period of h with a weight term w, into the reward function to add a long-term horizon while training intraday RL agents:

$$r_t^{hind} = r_t + \underbrace{w \times (p_{t+h}^c - p_t^c) \times pos_t}_{hindsight\ \ bonus}$$

where p_t^c is the close price at time t, w is the weight of the hindsight bonus, h is the horizon of the hindsight bonus and pos_t is the position at time t.

Noticeably, we only use the reward function with a hindsight bonus for training to better understand the market. During the test period, we continue to use the original reward r_t to calculate the profits. Furthermore, the hindsight reward function ignores details of price fluctuation between $t+2$ to $t+h-1$ and focuses on the trend of this period, which is computational efficient and shows robust performance in practice.

4.3.3 Intraday Market Embedding

To learn a meaningful multi-modality intraday market embedding, we propose an encoder-decoder architecture to represent the market from the micro-level and macro-level, respectively.

For micro-level encoder, we choose LOB data and trader's private state to learn the micro-level market embedding. LOB is widely used to analyze the relative strength of the buy and sell side based on micro-level trading behaviors, and private state of traders is considered insightful to capture micro-level trading opportunities [64]. At time t, we have a sequence of historical LOB embeddings $(\mathbf{b_{t-k}}, ..., \mathbf{b_t})$ and trader's private state embedding $(\mathbf{z_{t-k}}, ..., \mathbf{z_t})$, where $k+1$ is the sequence length. As shown in Figure 4.2 (a), we feed them into two different LSTM layers and concatenate the last hidden states $\mathbf{h_t^b}$ and $\mathbf{h_t^z}$ of the two LSTM layers as the micro-level embedding $\mathbf{e_t^i}$ at time t.

For macro-level encoder, we pick raw OHLCV data and technical indicators to learn the macro-level embedding. The intuition here is that OHLCV reflects the original market status, and technical indicators offer additional information. At time t, we firstly concatenate OHLCV vector $\mathbf{x_t}$ and technical indicator vector $\mathbf{y_t}$ as input $\mathbf{v_t}$. As shown in Figure 4.2(b), the concatenated embedding is then fed into a multilayer perceptron (MLP). The MLP output is applied as the macro-level embedding $\mathbf{e_t^a}$ at time t. Finally, we concatenate micro-level embedding and macro-level embedding together as the market embedding $\mathbf{e_t}$. Our market embedding is better than that of previous work, since it incorporates the micro-level market information.

4.3.4 Risk-Aware Auxiliary Task

As risk management is of vital importance for intraday trading, we propose a risk-aware auxiliary task by predicting volatility to take into account market risk as shown in Figure 4.2(c). Volatility is widely used as a coherent measure of risk that describing the statistical dispersion of returns in finance [173]. We analyze the reasons why volatility prediction is an effective auxiliary task to improve the trading policy learning as follows.

First, it is consistent with the general trading goal, which is to maximize long-term profit under certain risk tolerance. Second, future volatility is easier to predict

compared to future price. For instance, considering the day that the president election result will be announced, nobody can know the result in advance, which will lead the stock market to increase or decrease. However, everyone knows that there would be a huge fluctuation in the stock market, which increases future volatility. Third, predicting future price and volatility are two closely related tasks. Learning value function approximation and volatility prediction simultaneously can help the agent learn a more robust market embedding. The definition of volatility is the variance of return sequence $y_{vol} = \sigma(\mathbf{r})$, where $\mathbf{r}$ is the vector of return at each time step. Volatility prediction is a regression task with market embedding $\mathbf{e_t}$ as input and y_{vol} as target. We feed the market embedding into a single layer MLP with parameters θ_v. The output $\hat{y_{vol}}$ is the predicted volatility. We train the network by minimizing the mean squared error.

$$\hat{y_{vol}} = MLP(e_t, \theta_v) \qquad L_{vol}(\theta_v) = (y_{vol} - \hat{y_{vol}})^2$$

The overall loss function is defined as:

$$L = L_q + \eta * L_{vol}$$

where L_q is the Q value loss and η is the relative importance of the auxiliary task.

4.4 Experiment Setup

4.4.1 Datasets and Features

To conduct a comprehensive evaluation of DeepScalper, we evaluate it on *six* financial assets from *two* real-world datasets (stock index and treasury bond) spanning over *three* years in the Chinese market collected from Wind[2]. We summarize the statistics of the two datasets in Table 8.1 and further elaborate on them as follows:

Stock index is a dataset containing the minute-level OHLCV and 5-level LOB data of two representative stock index futures (IC and IF) in the Chinese market. IC is a stock index future calculated based on 500 small and medium market capitalization stocks. IF is another stock index future that focuses on the top 300

[2]https://www.wind.com.cn/

large capitalization stocks. For each stock index future, we split the dataset with May-Dec, 2019 for training and Jan-April, 2020 for testing.

Treasury bond is a dataset containing the minute-level OHLCV and 5-level LOB data of four treasury bond futures (T01, T02, TF01, TF02). These treasury bond futures are mainstream treasury bond futures with the highest liquidity in the Chinese market. For each treasury bond, we use 2017/11/29 - 2020/4/29 for training and 2020/04/30 - 2020/07/17 for testing.

To describe macro-level financial markets, we generate 11 temporal features from the original OHLCV as shown in chapter 2.1.7 following [46]. For micro-level markets, we extract a 20-dimensional feature vector from the 5-level LOB where each level contains bid, ask price and bid, ask quantity following [5]. For evaluation metrics, we follow that in chapter 2.1.8 to the profitability, risk-adjusted return and risk.

Dataset	Freq	Number	Days	From	To
Stock index	1min	2	251	19/05/01	20/04/30
Treasury bond	1min	4	662	17/11/29	20/07/17

TABLE 4.1: Dataset statistics detailing data frequency, number of financial assets, trading days and chronological period

4.4.2 Baseline

We compare DeepScalper with nine baseline methods consisting of three traditional finance methods, three prediction-based methods, and three reinforcement learning methods.

- **Buy & Hold (BAH)**, which is usually used to reflect the market average, indicates the trading strategy that buys the pre-selected financial assets with full position at the beginning and holds until the end of the trading period.
- **Mean Reversion (MV)** [174] is a traditional finance method designed under the assumption that the price of financial assets will eventually revert to the long-term mean. In practice, it shows stellar performance under volatile market conditions.
- **Time Series Momentum (TSM)** [25] is an influential momentum-based method, which long (short) financial assets with increasing (decreasing) trend

in the past. This is in line with the principle that the stronger is always stronger in the financial market.

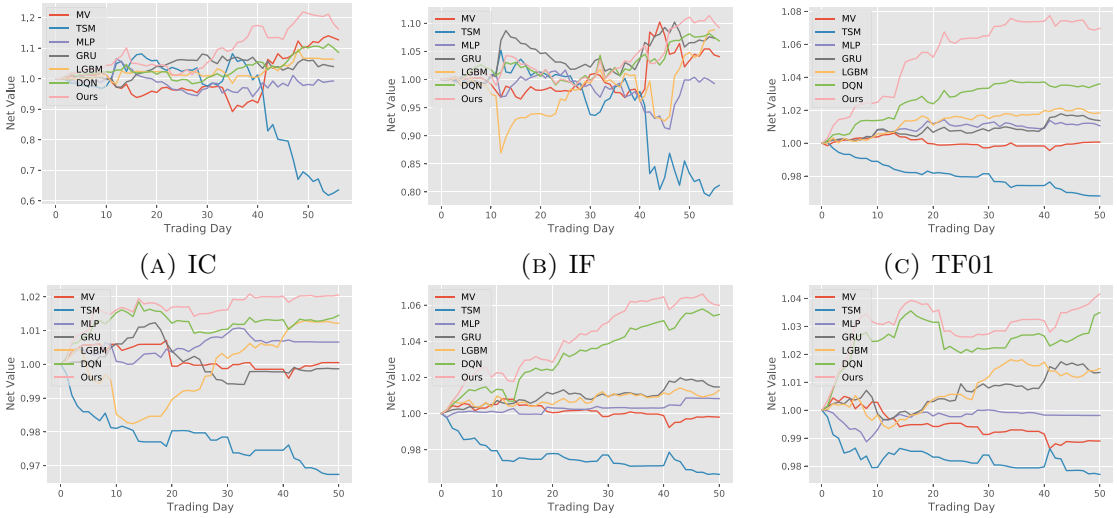
- **MLP** [175] use the classic multi-layer perceptron for future return prediction. We apply a three-layer MLP with hidden size 128.
- **GRU** [176] use a newer generation of recurrent networks with gated recurrent units for future return prediction. We apply a two-layer GRU with hidden size 64.
- **LGBM** [167] is an efficient implementation of the gradient boosting decision tree with gradient-based one-side sampling and exclusive feature bundling.
- **DQN** [75] applies the deep Q-network with a novel state representation and reward function for quantitative trading, which shows stellar performance in more than 50 financial assets.
- **DS-NH** is a variant of DeepScalper (DS), which removes the hindsight bonus from the reward function.
- **DS-NA** is a variant of DeepScalper (DS), which removes the risk-aware auxiliary task.

4.4.3 Preprocessing and Setup

For macro-level features, we directly calculate the 11 technical indicators following the formulas in Table 2.2. For micro-level features, we divide the order price and quantity of each level by the first-level price and quantity, respectively, for normalization. For missing values, we fill the empty price with the previous one and empty quantity as zero to maintain the consistency of time series data. To make the evaluation more realistic, we further consider many practical real-world constraints. The transaction fee rate δ is set as 2.3×10^{-5} and 3×10^{-6} for stock index futures and treasury bond futures respectively, which is consistent with the real-world scenario. Since leverage such as margin loans is widely used for intraday trading, we apply a fixed five-times leverage to amplify profit and volatility. Time is discretized into 1 min interval and we assume that the agent can only long/short a financial future at the end of each minute. The account of RL agents is initialized

with enough cash to buy 50 shares of the asset at the beginning. The maximum holding position is 50.

We perform all experiments on a Tesla V100 GPU. Grid search is applied to find the optimal hyperprarameters. We explore the look-ahead horizon h in [30, 60, 90, 120, 150, 180], importance of hindsight bonus w in $[1e^{-3}, 5e^{-3}, 1e^{-2}, 5e^{-2}, 1e^{-1}]$ and importance of auxiliary task η in [0.5, 1.0]. As for neural network architectures, we search the hidden units of MLP layers and GRU layer in [32, 64, 128] with ReLU as the activation function. we use Adam as the optimizer with learning rate $\alpha \in (1e^{-5}, 1e^{-3})$ and train DeepScalper for 5 epochs in all financial assets. Following the iterative training scheme in [64], we augment traders' private state repeatedly during the training to improve data efficiency. We run experiments with 5 different random seeds and report the average performance. It takes 1.5 and 3.5 hours to train and test Deepscalper on each financial asset in the stock index and treasury bond datasets, respectively. As for other baselines, we use the default settings in their public implementations [18, 22].



(A) IC (B) IF (C) TF01

(D) TF02 (E) T01 (F) T02

FIGURE 4.4: Trading day vs. net value curve of different baselines and DeepScalper on stock index (IC and IF) and treasury bond (TF01, TF02, T01, T02) datasets. DeepScalper achieves the highest profit in all six financial assets.

Stock Index					
Type	Models	TR(%)↑	SR↑	CR↑	SoR↑
FIN	BAH	5.65	0.15	0.02	0.27
	MV	8.39	1.18	0.21	2.22
	TSM	-27.62	-2.83	-0.21	-3.08
PRE	MLP	-0.73 ± 7.11	-0.14 ± 1.02	-0.01 ± 0.58	-0.24 ± 1.77
	GRU	5.66 ± 4.98	1.25 ± 0.66	0.24 ± 0.18	2.40 ± 0.82
	LGBM	7.62 ± 1.14	1.26 ± 0.22	0.28 ± 0.05	1.59 ± 0.22
RL	DQN	7.74 ± 3.52	1.25 ± 0.62	0.28 ± 0.17	1.79 ± 0.91
	DS-NH	8.17 ± 5.07	0.98 ± 0.77	0.17 ± 0.17	1.37 ± 0.88
	DS-NA	9.74 ± 5.12	1.32 ± 0.76	0.26 ± 0.21	2.19 ± 1.11
	DS	12.74♣ ± 4.65	1.76♣ ± 0.61	0.34♣ ± 0.16	2.58♣ ± 0.72
% Improvement		30.80↑	33.33↑	21.42↑	7.50↑
Treasury Bond					
Type	Models	TR(%)↑	SR↑	CR↑	SoR↑
FIN	BAH	-14.26	-3.42	-0.25	-4.40
	MV	-0.29	-0.59	-0.04	-0.62
	TSM	-3.02	-5.35	-0.31	-6.20
PRE	MLP	0.59 ± 1.11	1.42 ± 1.30	0.42 ± 0.51	2.33 ± 1.42
	GRU	1.02 ± 2.10	1.90 ± 1.72	0.55 ± 0.57	3.69 ± 2.09
	LGBM	1.45 ± 0.17	2.43 ± 0.43	0.58 ± 0.09	3.68 ± 0.52
RL	DQN	3.51 ± 1.05	4.01 ± 1.27	1.15 ± 0.39	5.66 ± 1.33
	DS-NH	3.38 ± 1.28	4.42 ± 1.21	1.39 ± 0.45	6.85 ± 1.19
	DS-NA	4.17 ± 1.44	4.27 ± 0.99	1.38 ± 0.43	7.59 ± 1.49
	DS	4.79♣ ± 0.99	4.75♣ ± 1.25	1.82♣ ± 0.41	7.4 ± 1.22
% Improvement		14.87↑	7.47↑	30.94↑	2.57↓

TABLE 4.2: Profitability comparison (mean and standard deviation of 5 individual runs) with 9 baselines including traditional finance (FIN), prediction based (PRE) and reinforcement learning (RL) methods. All three FIN models are *deterministic* methods without the performance standard deviation. Purple and pink show best & second best results. ♣ indicates improvement over SOTA baseline is statistically significant ($p < 0.01$) under Wilcoxon's signed rank test.

4.5 Results and Analysis

4.5.1 Profitability Comparison with Baselines

We compare DeepScalper with 9 state-of-the-art baselines in terms of four financial metrics in Table 5.1. We observe that DeepScalper consistently generates significantly ($p < 0.01$) higher performance than all baselines on 7 out of 8 metrics across the two datasets. In the stock index dataset, DeepScalper performs best on all four metrics. Specifically, it outperforms the second best by 30.80%, 33.33%, 21.42% and 7.50% in terms of TR, SR, CR and SoR, respectively. As for the treasury bond dataset, DeepScalper outperforms the second best by 14.87%, 7.47%, 30.94% in terms of TR, SR and CR. For SoR, DS-NA performs slightly better than DS (2%

without statistical significance). One possible reason is that volatility prediction auxiliary task is not directly relevant to control downside return variance.

Furthermore, we show the trading day vs. net value trading days of the test period for each financial future from the two datasets in Figure 4.4. We intentionally exclude BAH, DS-NH and DS-NA to make the figure easy to follow. For traditional methods, we find that MV achieves decent performance for most financial futures. In comparison, TSM's performance is much worse. One possible reason for TSM's failure is that there is no evident momentum effect within the market for intraday trading. For deep learning models, the overall performance of GRU is better than that of MLP due to its ability to learn the temporal dependency of indicators. As for LGBM, it achieves slightly better performance than deep learning models. The average performance of RL methods is the best.

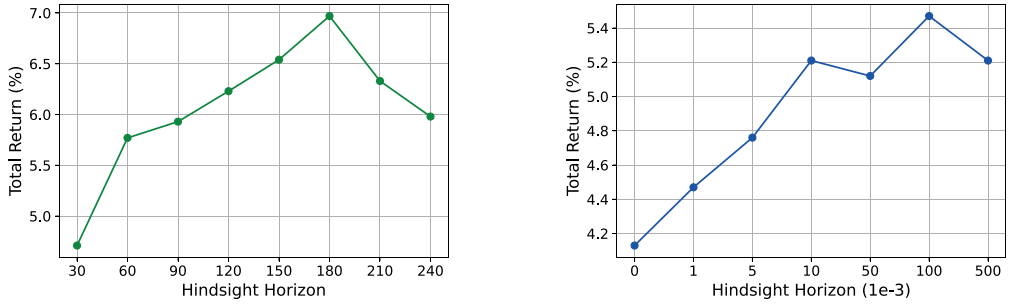
Macro	Micro	Hindsight	Volatility	TR(%)↑	SR↑
√				3.45	4.42
	√			3.47	4.43
√	√			3.62 (+0.15)	4.81 (+0.38)
√	√		√	4.05 (+0.58)	5.03 (+0.60)
√	√	√		5.36 (+1.89)	5.72 (+1.29)
√	√	√	√	**6.97** (+3.50)	**6.10** (+1.67)

TABLE 4.3: Ablation studies over different DeepScalper components. √ indicates adding the component to DeepScalper.

4.5.2 Model Component Ablation Study

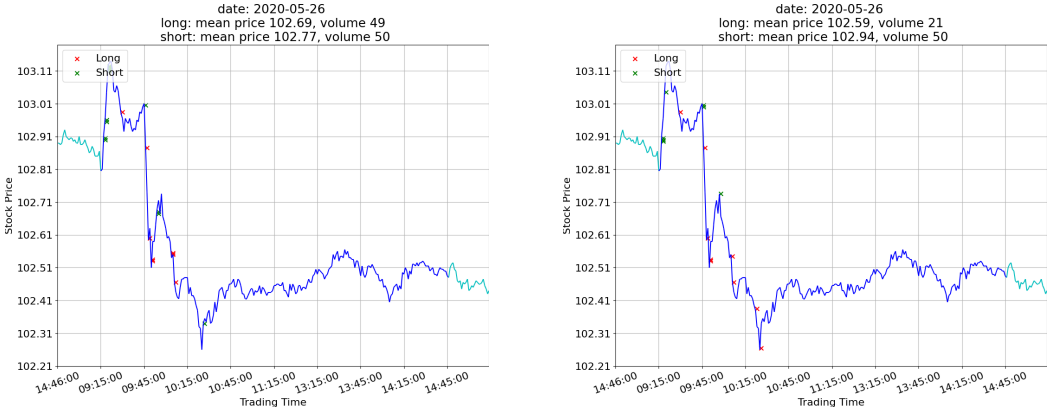
We conduct comprehensive ablation studies on DeepScalper's investment profitability benefits from each of its components in Table 4.3. First, we observe that the encoder-decoder architecture can learn better multi-modality market embedding than agents trained with other macro-level or micro-level market information, which leads to 0.15% and 0.38 improvement of TR and SR, respectively. Next, we find that adding the volatility prediction auxiliary task into DeepScalper can further improve performance, indicating that taking risk into consideration can lead to robust market understanding. In addition, we observe that the hindsight bonus can significantly improve DeepScalper's ability for the evaluation of trading decisions and further enhance profitability. Finally, we add all these components into DeepScalper and achieve the best performance in terms of TR and SR. Comprehensive ablation studies demonstrate: i) each individual component in DeepScalper is effective; ii)

these components are largely orthogonal and can be fruitfully integrated to further improve performance.



(A) TF02 (B) T02

FIGURE 4.5: Hyperparameter sensitivity of hindsight bonus: (a) effect of importance (b) effect of horizon



(A) May 26 (DS-NH) (B) May 26 (DS)

FIGURE 4.6: Trading behavior comparison of DS-NH and DS to show the effectiveness of the hindsight bonus

4.5.3 Effectiveness of Hindsight Bonus

We analyze the effectiveness of the hindsight bonus from two perspectives. First, we explore the impact of the hindsight bonus horizon and weight. As shown in Figure 4.5a, with the increase of w, the agent tends to trade with a long-term horizon and achieves a higher profit. DeepScalper with $w = 0.1$ achieves the highest profit. Figure 4.5b shows the impact of hindsight horizon h on DeepScalper's performance. We observe that DeepScalper's total return gradually increases by moving h from 30 to 180 and decreases when $h > 180$.

Moreover, we compare the detailed trading behaviors of agents trained with and without hindsight bonus on a trading day with the decreasing trend in Figure 4.6. The general good intraday trading strategy for that day is to short at the start of the day and long at the end of the day. We find that the agent trained without the hindsight bonus (Figure 4.6a) performs well in capturing local trading opportunities and overlooks the long-term trend of the entire trading day. In comparison, an agent trained with the hindsight bonus (Figure 4.6b) trades a large volume of short actions at the beginning of the trading day, indicating that it is aware of the decreasing trend in advance. This kind of trading action is smart, since it captures the big price gap of the overall trend and somehow ignores the local gain or loss.

4.5.4 Effectiveness of Risk-aware Auxiliary Task

Since the financial market is noisy and the RL training process is unstable, the performance variance among different random seeds is a major concern of RL-based trading algorithms. Intuitively, taking market risk into account can help the RL agent behave more stable with lower performance variance. We run experiments 5 times with different random seeds and report the relative variance relationship between RL agents trained with/without the risk-aware auxiliary task in Figure 4.7. We find that RL agents trained with the risk-aware auxiliary task achieve a lower TR variance in all six financial assets and a lower SR variance in 67% of financial assets. Furthermore, we test the impact of auxiliary task importance η on DeepScalper's performance. Naturally, the volatility value scale is smaller than return, which makes $\eta = 1$ a decent option to start. In practice, we test $\eta \in [0, 0.5, 1]$ and find the improvement of the auxiliary task is robust to different importance weights as shown in Figure 4.8.

4.5.5 Generalization Ability

We further test the generalization ability of our framework among different financial futures (TF02 and T02). In Figure 4.9, it is clear that the price trend of TF02 and T02 is similar. We assume similar price curves shares and similar trading patterns. Then, we train DeepScalper using the TF02 training set and test it on the test set of both TF02 and T02. We compare the performance of MV, GRU, LGBM, and

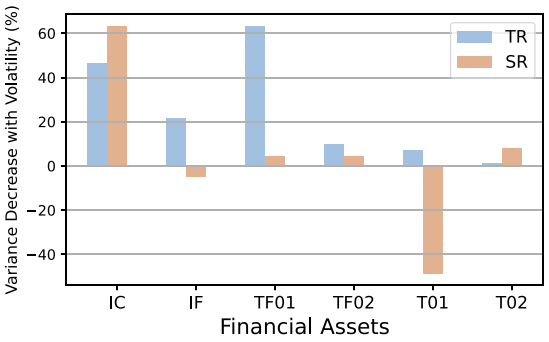


FIGURE 4.7: Effect of the auxiliary task on performance variance (>0 means RL agents trained with the risk-aware auxiliary task get lower standard deviation)

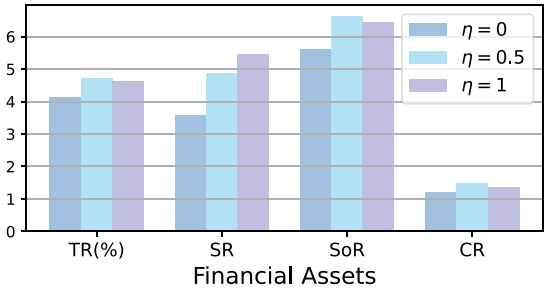


FIGURE 4.8: Sensitivity analysis on relative importance η in terms of TR, SR, SoR and CR

DeepScalper in Figure 4.10. The red and blue lines represent the performance on TF02 and T02, respectively. We observe that the performance of MV, GRU, and LGBM among these two assets is quite different, demonstrating that they have poor generalization ability on our task. One possible reason is that the trading signal of MV, GRU and LGBM involves heuristic rules or threshold. There will be some potential trading opportunities that are close to meet the trading rules or threshold, but MV, GRU, and LGBM will lose the opportunities. At the same time, our DeepScalper achieves robust performance on both TF02 and T02 as shown in Figure 4.10 (d), although it has never seen T02 data before. All these experiments demonstrate that DeepScalper can learn a robust representation of the market and achieve good generalization ability.

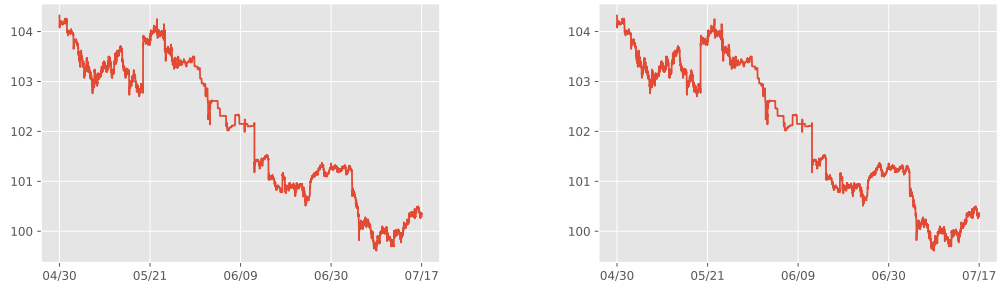
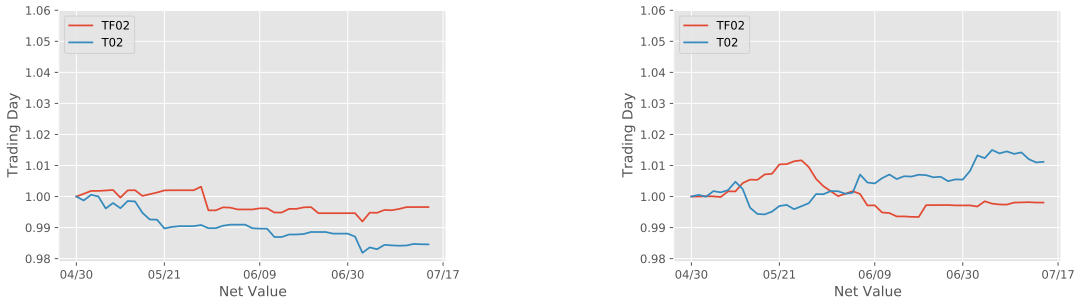
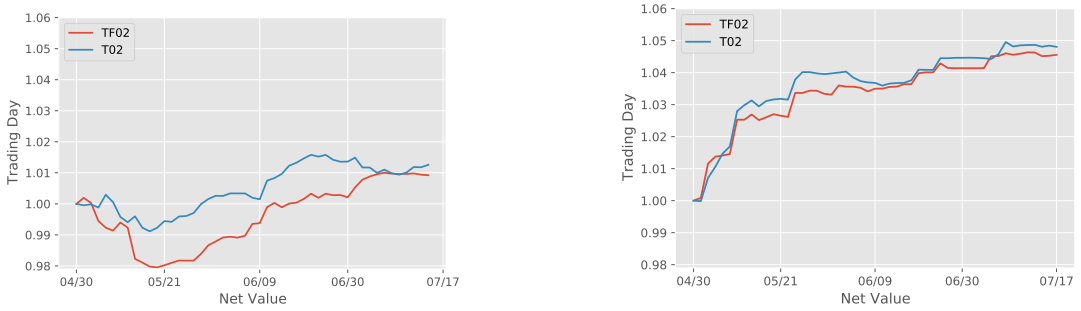


FIGURE 4.9: Price curve of TF02 and T02



(A) MV

(B) GRU



(C) LGBM

(D) DeepScalper

FIGURE 4.10: Net value curve of MV, GRU, LGBM and DeepScalper on TF02 and T02 to show the generalization ability.

4.6 Chapter Summary

In this chapter, we focus on intraday trading and propose DeepScalper to mimic the workflow of professional intraday traders. First, we apply the dueling Q-network with action branching to efficiently train intraday RL agents. Then, we design a novel reward function with a hindsight bonus to encourage a long-term horizon to capture the overall price trend. In addition, we design an encoder-decoder architecture to learn robust market embedding by incorporating both micro-level and macro-level market information. Finally, we propose volatility prediction as an auxiliary task to help agents be aware of market risk while maximizing profit.

Extensive experiments on two stock index futures and four treasury bond futures demonstrate that DeepScalper significantly outperforms many advanced methods.

Chapter 5

Mixture of Experts for Stock Prediction

5.1 Introduction

The stock market is one of the most popular channels for investors to pursue desirable financial assets and achieve investment goals. As the famous efficient market hypothesis [169] claims, forecasting stock prices accurately and making highly profitable investment decisions are extremely challenging tasks due to the noisy nature of the financial markets. In the last decade, deep learning (DL) methods have become an appealing direction for stock prediction [120] and quantitative investment [177] owing to its ability to learn insightful market representations in an end-to-end manner. Most existing DL methods adopt various advanced network architectures (e.g., LSTM [110], Attention [178] and Transformer [115]) to extract meaningful features from heterogeneous financial data such as fundamental factors [127], economics news [122], social media [120] and investment behaviors [121].

To apply deep learning models for quantitative investment, the vast majority of existing methods [45, 46, 179, 180] first adopt stock prediction as a supervised learning task. Observable market information (e.g., historical price) is used as the feature vector and the future price movement direction [46] or return rate [180] is applied as the target for classification or regression, respectively. One neural network (NN) predictor is optimized to capture the underlying pattern of financial markets by minimizing the corresponding loss function. Later on, the trained

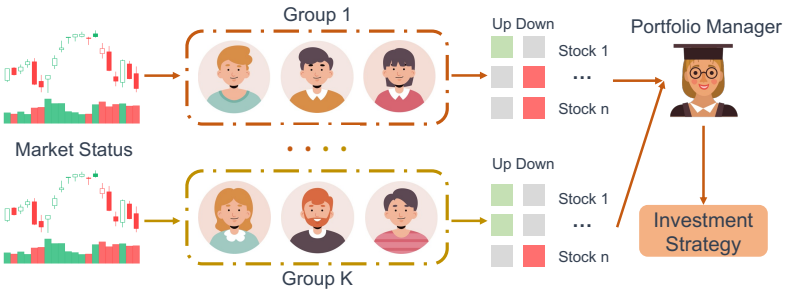


FIGURE 5.1: Overview of the bottom-up hierarchical trading strategy design workflow in real-world trading companies.

predictor is applied on new test sets for prediction. Finally, investment strategies are constructed by picking stocks with the top predicted price rising probability [46] or highest predicted return rate [180]. However, the performance of previous methods is unstable and sensitive to many factors such as random seeds [181]. The potential reason of this phenomenon is that existing methods make investment decisions based on only one individual predictor, whose predictions are usually uncertain [182] and fail to seize fleeting trading opportunities in stock markets.

In Figure 5.1, we plot an overview of trading strategy design workflow in real-world trading firms. There are multiple groups of trading experts with diversified background knowledge (e.g., finance and computer science). Within one group, they collaboratively work together to analyze the market from the same perspective. At the same time, groups work independently with different trading styles and risk tolerance to avoid mutual impact and correlated decisions. Then, a senior portfolio manager summarizes their results and makes the final investment decisions. This bottom-up hierarchical workflow plays a key role in designing robust and profitable trading strategies [183]. In this chapter[1], we propose AlphaMix, a novel three-stage mixture-of-experts (MoE) framework for quantitative investment. In stage one, an efficient ensemble learning method, whose computational and memory costs are significantly lower than traditional ensemble methods, is proposed to train multiple trading experts with personalised market understanding and trading styles. In stage two, we build a pool of diversified trading experts by utilizing both hyperparameter level and initialization level diversity of neural networks for post hoc ensemble construction. In stage three, we design three different mechanisms, namely with-replacement selection, integrated expert soup and as-needed router, to dynamically pick experts from the expert pool to take the responsibility of a portfolio manager. As illustrated in Figure 5.2, AlphaMix achieves the best performance with much

[1]This chapter has been published in [184].

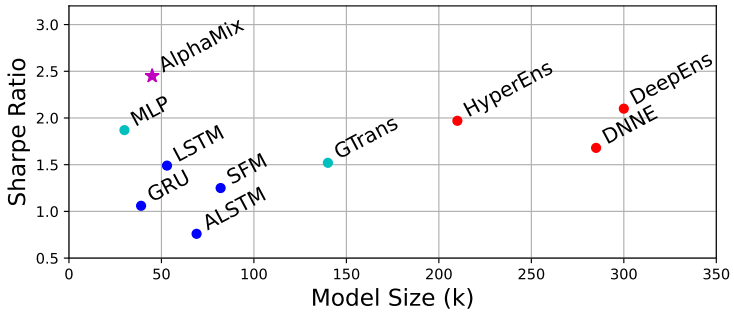


FIGURE 5.2: Model size vs. Sharpe ratio dot plot on US stock market. Ensemble methods (red) generally outperform other RNN (dark blue) and Non-RNN (light blue) methods. Our AlphaMix (purple) performs the best (17% improvement) with tiny computation overhead than the base model (MLP).

smaller model size comparing to three prevalent ensemble methods [134, 137, 185]. The main contributions of this work are two-fold:

- To the best of our knowledge, AlphaMix is the first mixture-of-experts framework for quantitative investment. In AlphaMix, we first efficiently optimize multiple independent groups of trading experts, then we build a pool of diversified models for post hoc ensemble construction, and finally three different mechanisms are introduced to further improve performance by dynamically picking these experts.
- Through experiments on real-world data of two influential stock markets spanning over 5 years, we show that AlphaMix significantly outperforms 11 state-of-the-art baseline methods in terms of 7 financial criteria and demonstrate AlphaMix's practical applicability to quantitative investment with a series of exploratory and ablative studies.

5.2 Generating Diversified Experts

In this section, we first provide a formulation of quantitative investment. We then introduce an efficient ensemble approach to train multiple groups of trading experts with tiny computation and memory overhead. Finally, we conduct model augmentation to build a pool of diversified trading experts, which utilizes two sources of diversity: varied hyperparameters and random initialization.

5.2.1 Problem Formulation

We formulate quantitative investment as a supervised learning task by generating trading strategies based on stock movement predictions following [46]. Consider a stock pool of size N, where for each stock i on trading day t, there is a corresponding closing price p_t^i and a feature vector x_t^i.

Formalizing stock movement prediction. We define stock movement prediction with label $y_{[t,t+\tau]}^i$ over a period of τ days as a classification task following [46, 120]. Here, we define the label using the closing price of a given stock p_t^i, that can either rise or fall on day $t+\tau$ compared to day t as:

$$y_{[t,t+\tau]}^i = \begin{cases} 1, & p_{t+\tau}^i > p_t^i \\ 0, & p_{t+\tau}^i \leq p_t^i \end{cases} \tag{5.1}$$

Unless otherwise stated, we use x and y to denote the sequential feature vectors $(x_{t-k}^i, ..., x_t^i)$ and next day stock movement $y_{[t,t+1]}^i$ of stock i on time t in the following of this work for simplicity.

Trading strategy generation. To evaluate the performance of quantitative investment methods, we apply a widely used simple yet robust trading strategy called daily top k buy & hold as in [46, 180]. Specifically, we rebalance our portfolio by evenly allocating money into the top k stocks with the highest probability to increase from the stock pool at the end of each trading day.

5.2.2 An Efficient Ensemble Approach

To fully utilize the potential of ensemble learning methods in financial markets, there are two major challenges: i) how to train diversified ensemble members by taking personal market status understanding and trading styles into consideration; ii) how to address the significant time and memory cost of traditional ensemble methods for real-world deployment. To address these two challenges, we introduce an efficient ensemble approach to train a group of trading experts simultaneously with limited cost overhead leveraging the idea of [133]. As shown in Figure 5.3, we first allocate each ensemble member two trainable vectors representing personal

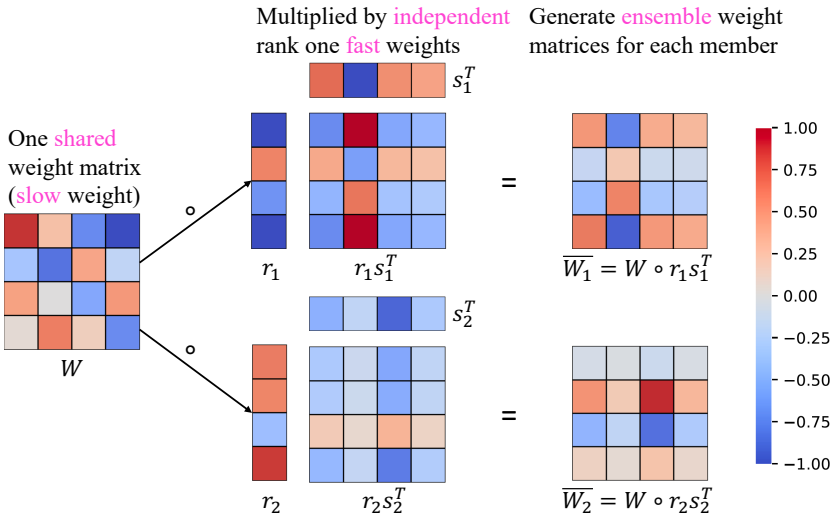


FIGURE 5.3: Illustration of the efficient ensemble approach on how to generate the weights for two ensemble members.

market understanding and trading style. Then, we generate an independent rank-one matrix of each ensemble member by multiplying the two personalised vectors. Finally, the ensemble weights are generated by calculating the Hadamard product of the shared weight matrix and the independent rank-one matrix.

Formally, let W be the weights in a neural network layer. Denote the input dimension as m and the output dimension as n, i.e., $W \in \mathbb{R}^{m \times n}$. M trading experts are trained, where each expert has weight matrix $\overline{W_i}$. Specifically, each expert i owns a tuple of trainable vectors r_i and s_i with dimension m and n, which represents the expert's personal market understanding and trading style, respectively. Our algorithm generates a group of ensemble weights $\overline{W_i}$ as follows:

$$\overline{W_i} = W \circ F_i \ , \quad where \ F_i = r_i s_i^T \tag{5.2}$$

For each training sample in the mini-batch, it receives an ensemble weights $\overline{W_i}$ by computing Hadamard product of W (shared slow weights) and a rank-one matrix F_i (personal fast weights). The subscript i represent expert i. The insight here is that one can modulate intermediate features of neural networks to achieve promising performance [133, 186] instead of modulating the whole weight matrices.

Vectorization for Parallelization. To further elaborate the efficiency of this approach, we show that the above ensemble weight generation mechanism can be parallelizable within one device based on [133]. Specifically, one computes a forward

pass with respect to multiple ensemble members in parallel by manipulating the matrix computations for a mini-batch [187]. We denote a as the activations of the incoming neurons in a neural network layer. The next layer's activations c are calculated as:

$$\begin{aligned} c_n &= \phi\ (\overline{W}_i^T a_n) = \phi\ ((W \circ r_i s_i^T)^T a_n) \\ &= \phi\ ((W^T(a_n \circ r_i)) \circ s_i) \end{aligned} \tag{5.3}$$

where ϕ represents the activation function and the subscript n denotes the index in the mini-batch. The output is next layer's activations from the i^{th} ensemble member. In order to vectorize these computations, we define matrices R and S whose rows consist of the vectors r_i and s_i for all data samples in the mini-batch. The vectorized form of the above equation is as follows:

$$C = \phi(((A \circ R)W) \circ S) \tag{5.4}$$

where A and C are the mini-batch input and output of this layer. By applying equation 5.4, the next layer's activations for each ensemble member can be computed in a mini-batch friendly way. As a results, this ensemble method allows us to take full advantage of parallel accelerators for efficiently training multiple trading experts simultaneously. In practice, we can divide the input mini-batch in M sub-batches and each sub-batch receives ensemble weight $\overline{W}_i$ to match the input and the ensemble weight,.

Individual Ensemble Loss. For the computation of ensemble loss, we feed the feature vector x into the neural network and get predicted stock movement $\hat{y}_i$ of expert i. To combine the loss of multiple experts, one way is to use the aggregated classification logits. We define the collaborated aggregated loss as:

$$L^{classify}_{collaborative}(x, y) = L_{classify}\left(\frac{1}{n}\sum\nolimits_{i=1}^{n} \hat{y}_i, y\right) \tag{5.5}$$

where $L_{classify}$ denotes any classification loss (e.g., cross-entropy). However, collaborative loss leads to correlated decisions [179] instead of complementary experts in our preliminary experiments. To discourage correlations, we require each expert to do the job well by itself. Therefore, we apply an effective individual loss as:

$$L^{classify}_{individual}(x, y) = \frac{1}{n}\sum\nolimits_{i=1}^{n} L_{classify}(\hat{y}_i, y)) \tag{5.6}$$

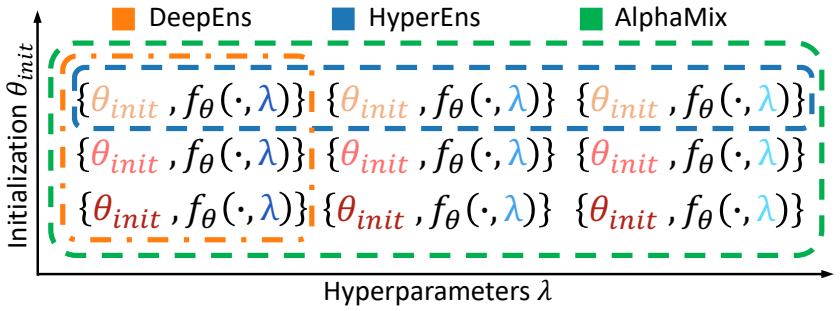


FIGURE 5.4: Pictorial view of the expert pool. Our AlphaMix can search in the whole "block" (green), while Deepens and Hyperens are built on one column (orange) or row (blue).

Ensembling During Testing. For inference, we first acquire the predictions of each ensemble member and then average them as the final prediction. For the scenarios of batch size B and M ensemble members, we repeat the data of the input mini-batch M times to get an effective batch size $B * M$, which enables all ensemble members to compute the output of the same B input data points in a single forward pass. As a result, it reduces the computation cost by eliminating the need to calculate the output of each ensemble member sequentially [133].

5.2.3 Hyper-Level Expert Augmentation

In professional trading firms, there is usually an administrative staff who collects and organizes the investment suggestions from different groups of trading experts as supportive information for portfolio managers' final decisions. Motivated by this, we introduce a way to construct an expert pool exploiting two sources of diversity: varied hyperparameters [134] and random initialization [188], which fully utilizes the investment decisions diversity of different groups of trading experts. We proceed in three main steps as summarized in Algorithm 1 following [134]. Specifically, random_search is a function that train models[2], i.e., $f_\theta(\cdot,\lambda)$, with random selected hyperparameters λ (e.g., learning rate, hidden size). topk_ens is a function that pick models with top K performance. In lines 1-2, we run random search of hyperparameters, which leads to a set of N models (i.e., $\mathcal{M}_0$). Then, we generate one "row" of K models using topk_ens to pick K models with better performance. We then tile and stratify that "row" by training the models for K different random initialization, thus $O(K^2)$ models to train in total (lines 4-7). A

[2]each $f_\theta(\cdot,\lambda)$ is a model trained with the efficient ensemble approach in Section 5.2.2 that represents the investment decision of one group of experts.

pictorial view of the expert pool is illustrated in Figure 5.4. We show that the expert pool of two strong ensemble methods, i.e., DeepEns [137] and HyperEns [134], are subsets of that for AlphaMix, which indicates that AlphaMix has the potential to achieve better performance by utilizing both sources of diversity.

Algorithm 1: Construct A Diversified Expert Pool

$\mathcal{M}_0 = \{f_{\theta_j}(\cdot, \lambda_j)\}_{j=1}^{N} \leftarrow$ random_search(N);
$\varepsilon_0 \leftarrow$ topk_ens($\mathcal{M}_\prime, K$) and $\varepsilon_{strat} = \{\}$;
foreach $f_\theta(\cdot, \lambda) \in \varepsilon_0$.unique() **do**
  foreach $k \in \{1, ..., K\}$ **do**
    $\theta' \longleftarrow$ random initialization;
    $f_{\theta_k}(\cdot, \lambda) \longleftarrow$ train $f_{\theta'}(\cdot, \lambda)$;
    $\varepsilon_{strat} = \varepsilon_{strat} \cup \{f_{\theta_k}(\cdot, \lambda)\}$
return ε_{strat}

5.3 Routing Experts Dynamically

The role of a senior portfolio manager is to make the final investment decisions based on the suggestions from different groups of junior trading experts. We try three different mechanisms to act as a portfolio manager by dynamically picking experts from the pool. First, we propose a novel routing mechanism to route the experts on an as-needed basis [143]. Second, we provide one simple yet robust ensemble aggregation option named with-replacement selection [144] that builds ensembles where the contribution of each ensemble member is weighted. Further, we customize the idea of model soup [140] from pre-train settings, which directly mixes the weights of neural networks, to incorporate different trading experts' knowledge without any extra training or inference cost.

5.3.1 Training As-Needed Routers

For a portfolio manager, it is a smart way to listen to suggestions from junior traders sequentially and make final investment decisions once they believe necessarily enough suggestions are already considered. This helps them avoid hesitation due to too many suggestions and make timely profitable decisions. Motivated by this, we train $n-1$ routers with parameters $h_{\theta_1}, ..., h_{\theta_{n-1}}$ to dynamically deploy these (arbitrarily

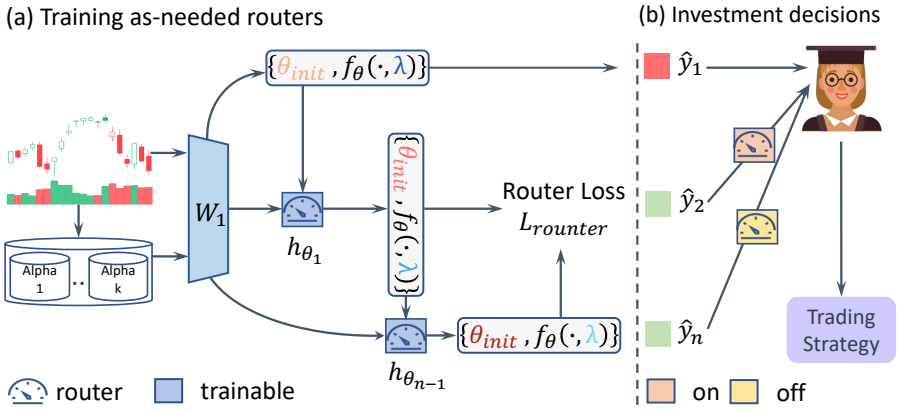


FIGURE 5.5: Illustration of AlphaMix with as-needed routers.

ordered) experts sequentially on an as-needed basis as shown in Figure 5.5 inspired by [143]. When the mean logits of the first k-th expert provide the correct prediction ($\hat{y} = y$), which indicates the first k experts are enough, the router should ideally switch off the $k+1$ expert with $y_{on} = 0$. Otherwise, the router should turn on the $k+1$ expert to explore more possibility with $y_{on} = 1$. In practice, we build a binary classifier with two fully connected layers to learn each router. Each of the $n-1$ routers for n experts has a shared component to reduce feature dimensions and an individual component to make decisions. Specifically, we normalize the market feature x and reduce the embedding dimension (8 in our experiments) by a fully connected layer W_1, which is shared with all routers, followed by LeakyReLU, and then concatenate it with the mean logits $\hat{y}^k_{mean} = \frac{1}{k}\sum_{i=1}^{k} \hat{y}_i$ of the first k experts. Furthermore, we project it to a scalar by $W_2^{(k)}$ which is independent between routers, and finally apply Sigmoid function $S(x) = \frac{1}{1+e^{-x}}$ to get a continuous activation value $v(x) \in [0, 1]$:

$$v(x) = S\left(W_2^{(k)} \begin{bmatrix} LeakyRelu(W_1 \frac{x}{\|x\|}) \\ \hat{y}^k_{mean} \end{bmatrix}\right) \tag{5.7}$$

The routers are optimized with a weighted variant of binary cross-entropy loss:

$$L_{router} = -\omega y_{on} log(v(x)) - (1 - y_{on}) log(1 - v(x)) \tag{5.8}$$

where ω controls the easiness to switch on the routers. We find $\omega = 1.7$ to be a good trade-off between performance and computational cost for both datasets. At

the test time, we simply threshold the activation with 0.5 to decide the on/off of the current router.

5.3.2 With-Replacement Experts Selection

In real-world trading, a portfolio manager may choose to give more weights on suggestions of some traders based on his/her preferences. Correspondingly, we adopt a simple yet efficient mechanism called with-replacement model selection [144] as shown in Algorithm 2 to enable weighted contribution of each member. Given a set of models (a pool of trading experts), we build an ensemble greedily until the target ensemble size is met by selecting the model leading to the optimal improvement of profits with replacement. This selection strategy enables to construct ensembles where the contribution of each member is weighted. To properly deal with the situation that there might be multiple times the same model is selected, we use .unique() here to address this issue.

Algorithm 2: With-Replacement Selection

Input: ensemble $\varepsilon = \{\}$, ensemble size K, validation profit $\mathsf{Pro}(\cdot)$, $\mathsf{Pro}_{\mathsf{best}} = +\infty$, model set $\mathcal{M} = \{f_{\theta_1}, ..., f_{\theta_n}\}$

while $\varepsilon.\mathsf{unique}() \leq \mathsf{K}$ **do**
- $f_{\theta^*} = \mathsf{argmax}_{f_{\theta_i} \in \mathcal{M}} \mathsf{Pro}(\varepsilon \bigcup \{f_{\theta_i}\})$;
- **if** $\mathsf{Pro}(\varepsilon \cup \{f_{\theta^*}\}) < \mathsf{Pro}_{\mathsf{best}}$ **then**
  - $\varepsilon = \varepsilon \cup \{f_{\theta^*}\}, \mathsf{Pro}_{\mathsf{best}} = \mathsf{Pro}(\varepsilon)$;
- **else**
  - **return** ε;

return ε

5.3.3 Integrated Model Soup

Instead of making investment decision directly based on suggestions of other trading experts, another working paradigm for portfolio managers is to integrate the market understanding and analysis of others and derive investment decisions based on their own experience. Inspired by it, we customize the idea of model soup [140] into financial markets, which directly mixs the network weights of different experts, i.e., their knowledge and market understanding. Unlike the above two mechanisms that mix the output of different experts, it is perhaps surprising that averaging

the weights of independently trained models achieves high performance since the loss landscape of neural network training is nonconvex with many solutions in different loss basins [140]. The insight here is that fine-tuned models optimized independently with the same initialization or hyperparameters may lie in the same basin of the error landscape [189]. As shown in Algorithm 3, the greedy soup sequentially adds each model as a potential ingredient in the soup, and only pick the model in the soup if performance on the validation set improves. We sort the models in decreasing order of validation set performance to ensure that the greedy soup can be no worse than the best individual model. In the end, we average the weights of all ingredient networks in the soup with function $\mathsf{average}()$ and construct one network without any extra inference cost.

Algorithm 3: Greedy Expert Soup

Input: Model set $\mathcal{M} = \{f_{\theta_1}, ..., f_{\theta_n}\}$ (sorted in decreasing order of validation profit $\mathsf{Pro}(f_{\theta_i})$,
soup ingredients $\varphi = \{\}$

for $i \leftarrow 1$ **to** k **do**
  if $\mathsf{Pro}(\mathsf{average}(\varphi \bigcup \{f_{\theta_i}\}) \geq \mathsf{Pro}(\mathsf{average}(\varphi))$ **then**
    $\varphi = \varphi \bigcup \{f_{\theta_i}\}$

return $\mathsf{average}(\varphi)$

5.4 Experiment Setup

5.4.1 Datasets

To conduct a comprehensive evaluation of AlphaMix, we evaluate it on two real-world datasets from US and Chinese stock markets spanning over five years as representatives of developed and developing financial markets, respectively. We summarize statistics of the two datasets in Table 8.1 and further elaborate on them as follows:

ACL18 [120] is a widely used public dataset collected using Yahoo Finance[3], which contains historical stock prices of 88 US stocks with top capital size from 9 different

[3]Yahoo Finance: https://github.com/yahoo-finance

industry categories: Basic Materials, Consumer Goods, Healthcare, Services, Utilities, Conglomerates, Financial, Industrial Goods and Technology, in the Nasdaq exchange.

SZ50 is a dataset with 4-year (2016 - 2020) historical prices of 47 Chinese stocks collected from Yahoo Finance. The stock pool contains component stocks of SSE50 Index, which represents top 50 companies by float-adjusted capitalization in Shanghai exchange.

For dataset split, we use data of the last year for test, penultimate year for validation and the remaining for training on both datasets. It is worth mentioning that the test period of ACL18 (16/09/01 to 17/09/01) is a stable period of economic growth while the test period of SZ50 (19/09/01 to 20/09/01) goes through an unstable period due to the panic decline during the COVID-19 pandemic. Experimental results in Table 5.1 show the great performance of AlphaMix under different market status.

5.4.2 Training Setup

To describe the financial markets, we generate 11 temporal features from the original OHLCV as shown in Chapter 2.1.7 following [46]. For evaluation metrics, we follow that in Chapter 2.1.8 to the profitability, risk-adjusted return and risk. We conduct all experiments on a Tesla V100 GPU. To build the expert pool of AlphaMix, we apply grid search to train experts with batch size in list $[32, 64, 128, 256]$, hidden size in range $[32, 64, 128]$, and learning rate $\alpha \in [5e^{-5}, 1e^{-4}, 5e^{-4}, 1e^{-3}]$. In addition, we explore the number of experts in range 2 to 32. Adam is used as the optimizer and we train AlphaMix for 10 epochs on all datasets. It takes about 1.5 and 1.1 hours to train and test on ACL18 and SZ50 datasets, respectively. As for other baselines, there are two conditions: i) there are authors' official or open-source library [18] implementations, we apply the same hyperparameters for a fair comparison[4]. ii) if there are no publicly available implementations, we reimplement the algorithms and try our best to maintain consistency based on the original papers.

[4]This condition fits for most baselines except GTrans and DNNE.

5.4.3 Baselines

To provide a comprehensive comparison of AlphaMix, we select 11 state-of-the-art and representative stock prediction methods of 4 different types consisting of recurrent neural network (RNN), non-recurrent neural network (NRNN), boosting decision tree (BDT) and ensemble learning (ENS) methods. Descriptions of baselines are as follows:

Recurrent Neural Network (RNN)

- **ALSTM** [178] adds an external attention layer into LSTM to attentively aggregate information from all hidden states.
- **LSTM** [110] uses a two-layer vanilla LSTM network with hidden size 32 to predict stock movement.
- **SFM** [112] is a state frequency memory recurrent network, which decomposes prices into signals of different frequencies via Discrete Fourier Transform.
- **GRU** [190] is a new version of recurrent networks with gated recurrent units. We apply a two-layer GRU with hidden size 64 for stock movement.

Non-Recurrent Neural Network (NRNN)

- **GTrans** [115] is a novel hierarchical Gaussian transformer-based model for stock prediction.
- **MLP** [191] applies multi-layer perceptron for stock movement prediction with hidden size 128.

Boosting Decision Tree (BDT)

- **LightGBM (LGB)** [167] is an efficient implementation of gradient boosting decision tree with gradient-based one-side sampling and exclusive feature bundling.
- **CatBoost (CatB)** [192] is a gradient boosting toolkit with ordered boosting and categorical feature processing.

Ensemble Learning Methods (ENS)

- **DNNE** [185] is a deep neural network ensemble method for stock market prediction based on bagging.
- **DeepEns** [137] is a simple yet efficient ensemble method with remarkable accuracy and robust uncertainty estimates.
- **HyperEns** [134] builds ensembles that involve a random search over different hyperparameters.

5.5 Results and Analysis

5.5.1 Comparison with Baselines

We compare AlphaMix[5] with 11 state-of-the-art baselines in terms of 7 financial criteria in Table 5.1. We observe that AlphaMix consistently generates significantly higher performance than other baselines on one profit and three risk-adjusted profit metrics across both US and China markets. In the US market, AlphaMix outperforms the second best by 12%, 17%, 58%, 5% in terms of TR, SR, CR, and SoR. As for the China market, AlphaMix outperforms the second best by 30%, 27%, 116%, 17% in terms of TR, SR, CR, and SoR. For risk metrics, AlphaMix performs the best in MDD, which shows its resistance to drawdown. For VOL and DD, AlphaMix achieves 1st and 2nd places on the US market that demonstrates its stellar performance on risk control. As for the China market, AlphaMix comes up with higher VOL and DD due to the trade-off between fluctuation and return in developing stock markets. In general, ensemble learning methods perform better than single deep learning methods (RNN and NRNN), as they combine multiple submodels for more robust and profitable investment decisions. We postulate that both RNN and NRNN methods have the potential to capture the internal patterns of financial data since both RNN and NRNN methods achieve decent performance on the two datasets. Boosting decision tree models tend to overfit to the noise in training data and generalize poorly in test data with the lowest profitability. In

[5]We report the experiment results of AlphaMix with as-needed routers in this Section if not particularly pointed out.

addition, we observe that DeepEns is the most powerful baseline with at least the second best performance on 7 out of 14 metrics. MLP achieves robust performance on both markets and CatBoost performs particularly well for risk control in the China market.

			Profit	Risk-Adjusted Profit			Risk Metrics		
Market	Type	Model	TR(%)↑	SR↑	CR↑	SoR↑	VOL(%)↓	DD(%)↓	MDD(%)↓
ACL18	RNN	ALSTM	9.9±8.1	0.76±0.61	1.76±1.91	1.14±0.99	0.82±0.03	0.56±0.04	6.88±1.43
		LSTM	19.5±4.3	1.49±0.34	3.31±1.15	2.30±0.59	0.82±0.02	0.54±0.04	6.21±1.26
		SFM	15.1±3.0	1.25±0.23	1.86± 0.48	1.94± 0.32	0.76±0.02	0.49±0.04	8.40± 1.58
		GRU	14.1±7.9	1.06±0.61	2.27±1.92	1.52±0.96	0.84±0.02	0.61±0.05	7.96± 2.59
	NRNN	GTrans	19.8±9.1	1.52 ±0.67	3.45 ±2.37	2.42±1.16	0.81±0.06	0.52±0.06	7.08 ±2.57
		MLP	22.1±6.3	1.87±0.57	3.69±1.77	2.52±0.88	0.75±0.03	0.57±0.07	6.82±2.15
	BDT	LGB	4.8	0.4	0.70	0.53	0.81	0.57	6.76
		CatB	9.3±3.3	0.70±0.25	1.31±0.55	1.00±0.37	0.83±0.03	0.59±0.03	7.33±0.91
	ENS	DNNE	23.3±9.4	1.68±0.72	3.76±2.52	2.73±1.24	0.90±0.14	0.56±0.09	7.99±3.37
		DeepEns	22.4	2.10	4.79	3.42	0.69	0.42	5.53
		HyperEns	23.3±3.2	1.96±0.28	3.39±0.78	2.67±0.42	0.74±0.02	0.54±0.02	6.88±0.97
	Ours	AlphaMix	26.1	2.45	7.59	3.59	0.67	0.45	3.43
% Improvement over SOTA			12%	17%	58%	5%	3%	-	61%
SZ50	RNN	ALSTM	23.2±7.0	1.07±0.26	1.59±0.48	1.34±0.33	1.38±0.09	1.10±0.08	14.74±1.89
		LSTM	20.7±7.8	1.01±0.33	1.57±0.52	1.28±0.42	1.30±0.07	1.03±0.05	13.02±0.85
		SFM	17.3±2.3	0.79±0.08	1.16±0.15	1.04±0.11	1.40±0.17	1.07±0.15	14.95±2.17
		GRU	36.5±27.9	1.26±0.76	2.00±1.35	1.74±1.14	1.66±0.39	1.22±0.22	16.81±2.96
	NRNN	GTrans	16.8±22.2	0.59±0.77	0.85±1.19	0.80±1.08	1.90±0.28	1.44±0.21	25.41±6.97
		MLP	43.5±7.3	1.82±0.15	2.87±0.30	2.46±0.24	1.53±0.15	1.13±0.09	15.36±3.23
	BDT	LGB	15.3	0.67	0.85	0.85	1.48	1.16	17.91
		CatB	21.0±1.8	1.06±0.08	1.72±0.23	1.43±0.12	1.27±0.11	0.95±0.10	12.28±0.72
	ENS	DNNE	40.8±8.1	1.66±0.35	3.25±0.90	2.27±0.56	1.58±0.05	1.17±0.08	12.82±1.43
		DeepEns	48.2	1.79	3.13	2.39	1.73	1.29	15.57
		HyperEns	44.3±7.9	1.83±0.31	2.92±0.56	2.51±0.46	1.51±0.06	1.10±0.08	15.32±1.68
	Ours	AlphaMix	62.6	2.34	7.04	2.95	1.72	1.37	8.89
% Improvement over SOTA			30%	27%	116%	17%	-	-	38%

TABLE 5.1: Performance comparison (mean and standard deviation of 10 individual runs) on ACL18 (US) and SZ50 (China) stock markets with 11 baselines including recurrent network (RNN), non-recurrent network (NRNN), boosting decision tree (BDT) and ensemble (ENS) learning methods. LightGBM (LGB), DeepEns and AlphaMix are three *deterministic* methods without the performance standard deviation. Results in pink, green and blue show best, second best and third best results on each dataset.

	ACL18 (US)			SZ50 (China)		
Models	TR(%)↑	SR↑	MDD↓	TR(%)↑	SR↑	MDD↓
DNNE	23.3	1.68	7.99	40.8	1.66	12.82
DeepEns	22.4	2.10	5.53	48.2	1.79	15.57
HyperEns	23.3	1.97	6.88	44.3	1.83	15.32
AlphaMix-V	24.5	2.39	3.95	53.7	2.08	11.19
AlphaMix-R	26.1	2.45	**3.43**	**62.6**	**2.34**	**8.89**
AlphaMix-W	**26.7**	**2.54**	5.83	61.0	2.24	12.10
AlphaMix-S	26.0	2.31	6.64	52.7	2.04	13.80

TABLE 5.2: Ablation studies over the effectiveness of different routing mechanisms of AlphaMix on US and China markets. Red indicates worse than either DNNE, DeepEns or HyperEns. Bold shows the best results.

5.5.2 Ablation

We conduct ablation studies on AlphaMix's performance benefits from different ensemble construction mechanisms. There are four variants of AlphaMix: i) AlphaMix-V is a vanilla version that simply averages the prediction of each group of expert; ii) AlphaMix-R trains extra as-needed routers; iii) AlphaMix-W uses the heuristic with-replacement selection and iv) AlphaMix-S applies model soup (Sec 5.3.3). We note that AlphaMix-V and AlphaMix-S can be considered as an enhanced version of [133, 140] and outperform the original version in our preliminary experiments. In this subsection, we compare them with the three ensemble baselines in Table 6.4. First, we observe that AlphaMix outperforms the three ensemble baselines in most cases (red indicates worse than baselines). The AlphaMix-V is one simple yet efficient choice on both markets. With extra training of as-needed routers, AlphaMix-R performs the best. For AlphaMix-W, it outperforms AlphaMix-V in terms of profit, but the MDD is higher. For AlphaMix-S with model soup, it does not perform as well as it is in computer vision [140] due to the low signal-to-noise ratio of stock markets. In summary, we recommend AlphaMix-V or AlphaMix-R for users, who prefer simple yet robust methods or best performance, respectively.

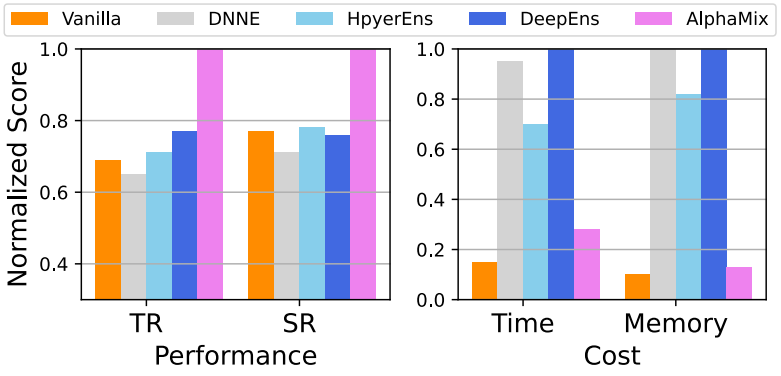


FIGURE 5.6: AlphaMix and several other methods on China market. AlphaMix achieves the best trade-off among performance (↑) and time & memory (↓) costs.

5.5.3 Computational Cost

We show AlphaMix is an efficient ensemble method that addresses the extensive computation cost overhead of traditional ensemble methods. Figure 5.6 displays results on China market in term of performance (TR & SR) and cost (time & memory). Although DeepEns and HyperEns slightly outperform the vanilla single

model, their computation cost is much higher. For AlphaMix, it significantly outperforms all three other methods (over 20% improvement) with much less computation overhead. The poor performance of DNNE further shows that straightforward applications of traditional ensemble methods are not enough in financial markets.

5.5.4 Diversity Analysis

To evaluate diversity, we use the predictive disagreement metric from [188]. This metric is based on the average of the pairwise comparisons of the predictions across the ensemble members. For a given pair of members, it is zero when they are making identical predictions, and one when all their predictions differ. We report the diversity of AlphaMix-V and two baselines in Table 5.3. The diversity of AlphaMix is significantly better than other two baselines as it utilizes two sources of diversity: hyperparameter (H) level and initialization (I) level.

	Diversity	Ens Size	ACL18 (US)	SZ50 (China)
DeepEns	I	4	0.4194	0.2832
HyperEns	H	4	0.3454	0.3992
AlphaMix	I & H	4	**0.5947**	**0.6124**

TABLE 5.3: Diversity comparison of the predictive disagreement. AlphaMix performs the best by utilizing two sources of diversity: hyperparameter (H) level and initialization (I) level.

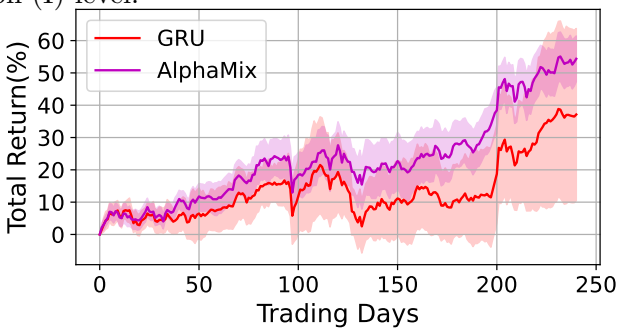


FIGURE 5.7: Trading days vs. total return curve. AlphaMix achieves higher return with much lower standard deviation.

5.5.5 Reduce Uncertainty with AlphaMix

We conduct experiments to show a single neural network is sensitive to random seeds with high uncertainty and AlphaMix can tackle this issue with mixture-of-experts. We train a popular two-layer GRU [190] model and our AlphaMix for stock

movement prediction and generate investment decisions with the classic top-4 buy & hold trading strategy with 10 independent runs. The trading days vs. total return curve (mean and standard deviation) of the two models is reported in Figure 5.7. AlphaMix achieves higher return with much lower standard deviation, which makes investment decisions from AlphaMix more reliable. To analysis the potential reason of this phenomenon, we plot the confidence score (largest logits after softmax) for stock movement prediction (binary classification) in Figure 5.8. The confidence score of GRU is quite low (between 0.5 and 0.6 for most samples) that indicates one single predictor is not enough. We observe that AlphaMix achieves much higher confidence score with mixture-of-experts. It is reasonable that higher confidence score leads to lower performance variance.

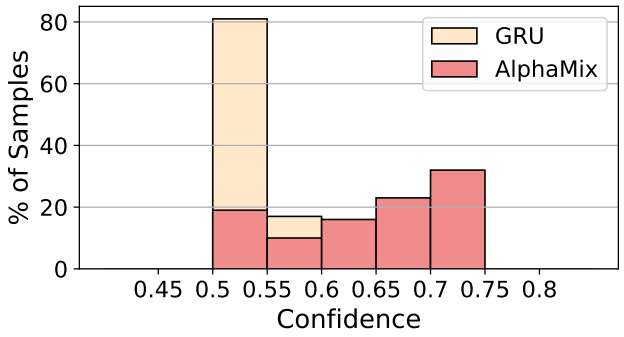


FIGURE 5.8: Confidence histogram to show AlphaMix has higher decision certainty comparing to GRU.

5.5.6 Parameter Analysis: Probing Sensitivity

Number of selected top k stocks. We analyze AlphaMix's performance (TR & SR) variation with the number of selected top stocks on ACL18 and SZ50 in Figure 5.9. We find that AlphaMix performs consistently well in terms of TR and SR on both datasets, which shows AlphaMix's robustness and suitability to strategies with different risk taking appetites. We pick $k = 4$ in practice.

Number of experts e. We analyze AlphaMix's performance (TR & SR) variation with the number of experts e on ACL18 and SZ50 in Figure 5.10. We observe that AlphaMix performs the best when e is relatively small. Then, the performance drops a lot with the increase of expert number. In addition, when the number of experts goes to very large (e.g., 32). The performance increases again. This indicates that it is better to make investment decision based on the discussion of a

few elite or a large population of experts. Decisions form medium size experts may lead to worse decisions.

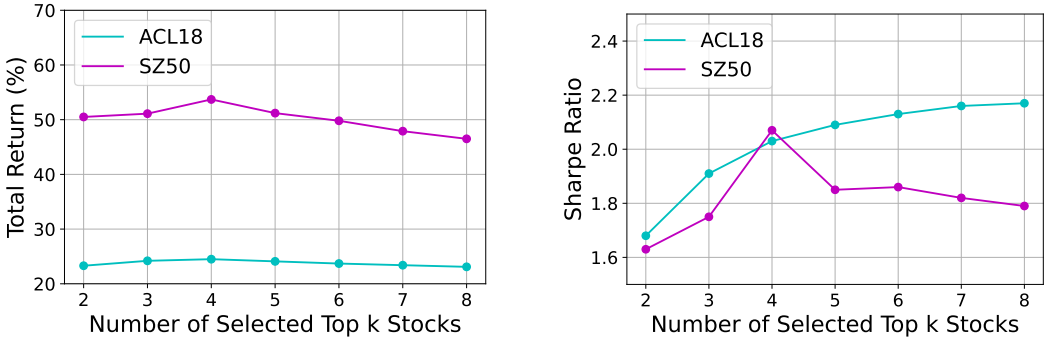


FIGURE 5.9: The impact of the number of top k stocks on AlphaMix in terms of TR and SR on ACL18 and SZ50.

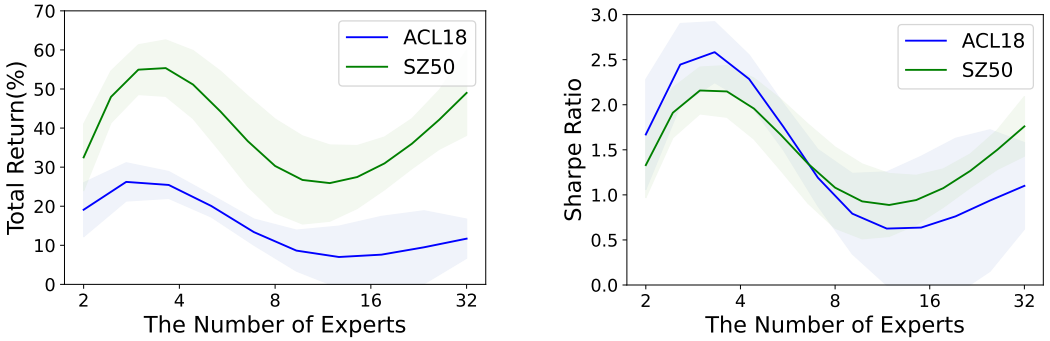


FIGURE 5.10: The impact of the number of experts on AlphaMix in terms of TR and SR on ACL18 and SZ50.

5.6 Chapter Summary

In this chapter, we propose AlphaMix, a novel three-stage mixture-of-experts framework, to mimic the efficient bottom-up workflow in real-world trading firms for quantitative investment. Specifically, we introduce an efficient ensemble learning method to train multiple trading experts with personalised market understanding and trading styles. Later on, we conduct model augmentation to construct a pool of diversified trading experts. Finally, we leverage three different mechanisms, namely as-needed router, integrated soup and with-replacement selection, to further improve performance by dynamically routing experts from the expert pool. Extensive experiments on both China and US stock markets demonstrate that AlphaMix significantly outperforms many strong baselines. Ablation studies show the effectiveness of the proposed components.

Chapter 6

Financial Decision Making with Ensemble Learning

6.1 Introduction

As stated in the last chapter, ensemble learning and mixture-of-experts methods achieve great performance on financial data under the supervised prediction setting. On the other hand, reinforcement learning has also be benefited with ensemble learning [11, 193] that takes uncertainty into consideration for robust decision making. It is natural to explore algorithms with ensemble learning based RL algorithms for financial decision making such as portfolio management. In this chapter[1], AlphaMix+ is proposed for financial decision making as an universal simple yet efficient algorithm. Risk-aware Bellman backup is introduced to reweight Q-values by adopting estimated uncertainty. To encourage strategy diversity between different trading agents, we leverage the idea of bootstrap with random initialization. Extensive experiments demonstrate the superior performance of the algorithm.

6.2 Soft Actor-Critic (SAC)

We introduce SAC [14], which is the base model of many popular FinRL methods [76]. SAC is a popular off-policy actor-critic RL method based on the maximum

[1]This chapter has been published in [194].

entropy RL framework [195], which maximizes a weight objective of the reward and the policy entropy, to encourage robustness to noise and exploration. For parameter updating, SAC alternates between a soft policy evaluation and a soft policy improvement. At the soft policy evaluation step, a soft Q-function $Q_\theta(s_t, a_t)$, which is modeled as a neural network with parameters θ, is updated by minimizing the following soft Bellman residual:

$$\mathcal{L}^{\text{SAC}}_{critic}(\phi) = \mathbb{E}_{\tau_t \sim \mathcal{D}}[L_Q(\tau_t, \theta)], \tag{6.1}$$

$$L_Q(\tau_t, \theta) = (Q_\theta(s_t, a_t) - r_t - \gamma \bar{V}(s_{t+1}))^2, \tag{6.2}$$

$$where \quad \bar{V}(s_t) = \mathbb{E}_{a_t \sim \pi_\phi}[Q_{\bar{\theta}}(s_t, a_t) - \alpha \log \pi_\phi(a_t \mid s_t)]. \tag{6.3}$$

where $\tau_t = (s_t, a_t, r_t, s_{t+1})$ represents a transition, $\mathcal{D}$ indicates a replay buffer, $\bar{\theta}$ are the delayed parameters, and α is used as a temperature parameter. To conduct steps of soft policy improvement, the policy π with its parameter θ is updated through minimizing the following objective:

$$\mathcal{L}^{\text{SAC}}_{actor}(\theta) = \mathbb{E}_{s_t \sim \mathcal{D}}[L_\pi(s_t, \phi)], \tag{6.4}$$

$$L_\pi(s_t, \phi) = \mathbb{E}_{a_t \sim \pi_\phi}[\alpha \log \pi_\phi(a_t \mid s_t) - Q_\theta(s_t, a_t)]. \tag{6.5}$$

To handle continuous action spaces, the policy is modeled as a Gaussian with mean and covariance given by neural networks. In addition to SAC, A2C [196], a popular actor-critic RL method, shows stellar performance in algorithmic trading [75]. The simple and efficient policy gradient method PPO [57] performs well in capturing trading opportunities for order execution [6]. EIIE [91] and Investor-Imitator (IMIT) [13] are two pioneering works that apply deep RL for quantitative trading. Furthermore, SARL [93] and Deeptrader [4] are proposed with augmented market embedding to take market risk into account for portfolio management.

6.3 AlphaMix+: A Strong Baseline

AlphaMix+, a FinRL algorithm based on ensemble learning, is proposed to fill the gap due to the unstable performance of existing FinRL methods under uncertainty.

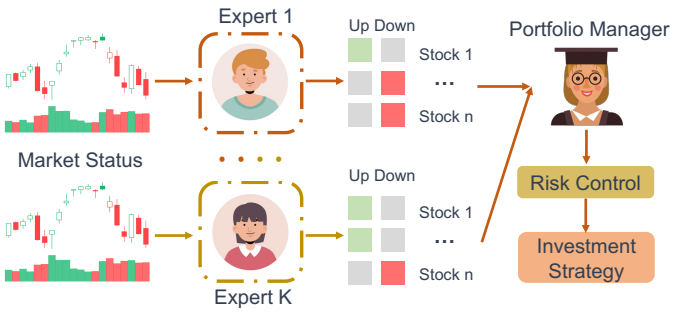


FIGURE 6.1: Workflow of real-world trading firms.

Considering the limitation of existing FinRL methods, the major one is that investment decisions are made by a single agent with high potential risk. The success of real-world trading firms relies on an efficient bottom-up hierarchical workflow with risk management as illustrated in Figure 6.1. First, multiple experts conduct data analysis and build models independently based on personal trading style and risk tolerance. Later on, a senior portfolio manager summarizes their results, manage risk and makes final investment decisions. Inspired by it, we propose AlphaMix+, a universal deep RL framework with diversified risk-aware mixture-of-experts following the idea of [11], to mimic this efficient workflow. In principle, AlphaMix+ can be combined with most modern off-policy RL algorithms for any quantitative trading task. As SAC [14] is a sample-efficiency algorithm in quantitative trading [76], we pick it as the base model of AlphaMix+ here for exposition. An overview of AlphaMix+ is shown in Figure 6.2.

Risk-aware Bellman backup. We consider a trading firm with N trading experts, i.e., $\{Q_{\theta_i}, \pi_{\phi_i}\}_{i=1}^{N}$, where θ_i and ϕ_i denote the parameters of the i-th expert's soft Q-function and policy. To apply classic Q-learning based on the Bellman backup in Eq. 6.1 in FinRL, one major issue is the severe negative impact of error propagation that induces the market noise to the learning trading signals (true Q-value) of the current Q-function [197] , which can cause unstable convergence. In order to overcome this issue, for trading expert i, we apply a risk-aware Bellman backup [11] as follows:

$$\mathcal{L}_{WQ}(\tau_t, \theta_i) = w(s_{t+1}, a_{t+1})(Q_{\theta_i}(s_t, a_t) - r_t - \gamma \bar{V}(s_{t+1}))^2 \tag{6.6}$$

where $\tau_t = (s_t, a_t, r_t, s_{t+1})$ is a transition, $a_{t+1} \sim \pi_{\phi_i}(a \mid s_t)$, and $w(s, a)$ is a confidence weight based on the ensemble of target Q-functions:

$$w(s, a) = \sigma(-\bar{Q}_{std}(s, a) * T) + k \tag{6.7}$$

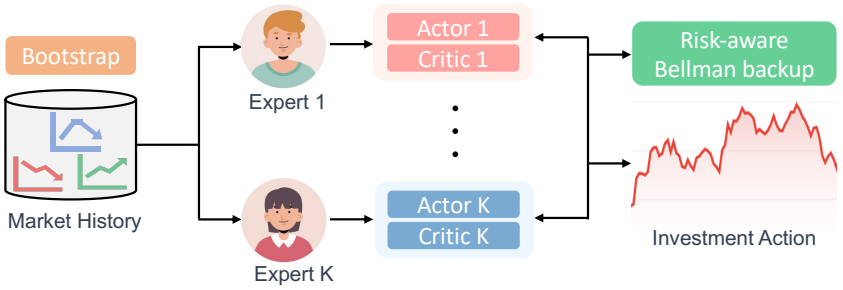


FIGURE 6.2: An illustration of AlphaMix+, a universal deep RL framework with diversified risk-aware mixture-of-experts.

where $\bar{Q}_{std}(s,a)$ indicates the empirical standard deviation of target Q-functions $\{Q_{\bar{\theta}_i}\}_{i=1}^{N}$ of all trading experts and $T > 0$ is a temperature parameter [198] to adapt the scale of $\bar{Q}_{std}(s,a)$. σ is the sigmoid function and $k > 0$ is used to control the value range of confidence weight. The confidence weight is bounded in $[k, k+0.5]$ as $\bar{Q}_{std}(s,a)$ is always a positive number. Intuitively, the objective $\mathcal{L}_{WQ}$ down weights the sample transitions with inconsistent trading suggestions from different experts (high variance across target Q-functions), resulting in a loss function for the Q-updates with better risk management.

Diversified Experts. We encourage the diversity between trading experts with two simple yet efficient tricks based on [11]. First, we initialize the model parameters of all trading experts with random parameter values for inducing an initial diversity in the models following [134, 137]. Second, we apply different samples to train each agent based on similar idea in BatchEnsemble [133]. For each trading expert i at each time step t, we construct the binary masks $m_{t,i}$ by sampling from the Bernoulli distribution with parameter $\beta \in (0, 1]$. Later on, we multiply the bootstrap mask to each objective function (e.g., $m_{t,i}\mathcal{L}_{\pi}(s_t, \phi_i)$ and $m_{t,i}\mathcal{L}_{WQ}(\tau_t, \theta_i)$ in Eq.6.5 and Eq.6.6 while updating parameters of trading experts. This encourages each expert to think individually with diversified strategies. We find it sufficient for AlphaMix+ to have desired diversity with the two simple tricks. Other tricks such as a KL divergence [199] loss term is not further incorporated to keep simplicity. We remark that although similar diversity encouragement techniques have been shown effective in classic RL tasks [11, 200], this work is the first to explore their potential in financial markets. We conduct ablation studies on the effectiveness of each component in AlphaMix+ and parameter analysis to probe sensitivity.

6.4 Problem Formulation

6.4.1 Portfolio Management

Portfolio management is a fundamental quantitative trading task [1], where investors hold a pool of different financial assets, i.e., stocks, bonds, as well as cash, and reallocate the proportion of capitals invested in each asset periodically to maximize future profit. The objective of portfolio management is to maximize the final portfolio value given a long-term time horizon by dynamically tuning the portfolio weight at each time step.

6.4.2 MDP Formulation

We consider a standard RL scenario in which an agent (investor) interacts with an environment (the financial market) in discrete time. Formally, we introduce MDP, which is defined by the tuple: MDP $= (\mathcal{S}, \mathcal{A}, P, R, \gamma, H)$. Specifically, $\mathcal{S}$ is a finite set of states. $\mathcal{A}$ is a finite set of actions. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ is a state transaction function, which consists of a set of conditional transition probabilities between states. $R : \mathcal{S} \times \mathcal{A} \longrightarrow \mathcal{R}$ is the reward function, where $\mathcal{R}$ is a continuous set of possible rewards and R indicates the immediate reward of taking an action in a state. $\gamma \in [0, 1)$ is the discount factor and H is a time horizon indicating the length of the trading period. A (stationary) policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$, parameterized by θ, assigns each state $s \in \mathcal{S}$ a distribution over actions, where $a \in \mathcal{A}$ has probability $\pi(a|s)$. A Q-value function gives expected accumulated reward when executing action a_t in state s_t and following policy π in the future, which is $Q(s_t, a_t) = \mathbb{E}_{(s_{t+1},\ldots,\pi)}\left[\sum_{i=t}^{T} \gamma^i r(s_i, a_i)\right]$. During training, one episode corresponds to adjusting the portfolio at each time step through the whole trading periods, i.e., time scope of training set, with time horizon H. The objective of the agent is to learn an optimal policy: $\pi_{\theta^*} = argmax_{\pi_\theta} \mathbb{E}_{\pi_\theta}\left[\sum_{i=t}^{T} \gamma^i r_{t+i} \mid s_t = s\right]$.

State $s_t \in \mathcal{S}$ at time t involves the concatenation of technical indicator vectors of M financial assets $(\mathbf{y_t^1}, \mathbf{y_t^2}, ..., \mathbf{y_t^i})$ as the external markets state, where $\mathbf{y_t^i}$ represent the technical indicator vector of asset i at time t. In addition, a 2-dimension tuple of investors' position and remaining cash is added as internal state. **Action space** at time t is an $M+1$ dimension vector $[w_t^0, w_t^1, ..., w_t^M]$ as a portfolio $\mathbf{w}_t$ to represent

the proportion of capitals invested at each asset. **Reward** r_t at time t is the change of portfolio value: $r_t = v_{t+1} - v_t$, where positive/negative values indicate earning/losing money, respectively.

6.5 Experimental Setup

6.5.1 Datasets

We collect real-world financial datasets spanning over 15 years of US stock, China stock, Cryptocurrency (Crypto) and Foreign Exchange (FX) from Yahoo Finance and Kaggle. All raw data and related processing scripts are publicly available. We summarize statistics of the 4 datasets with further elaboration in Table 8.1. US Stock dataset contains 10-year historical prices of 29 influential stocks with top unit price as a strong assessment of the market's overall health and tendencies. China Stock dataset contains 4-year historical prices of 47 influential stocks with top capitalization from the Shanghai exchange. Both US and China stock data is collected from Yahoo-Finance[2]. Crypto[3] dataset contains 6-year historical prices of 9 influential virtual currency with top unit price and trading volume collected. FX[4] dataset contains 20-year historical prices of 22 most popular currency with top foreign exchange reserves for US dollars. For each dataset, we filter out financial assets with missing values. For data split, we apply the similar split procedure in [119] with rolling window for all four datasets. As shown in Figure 6.3, phase 3 uses the last year for test, penultimate year for validation and the remaining of the dataset for training. For phase one and two, their validation/test sets roll back one and two years, respectively. For features, we follow that in Chapter 2.1.7.

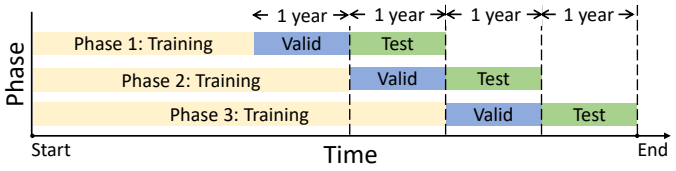


FIGURE 6.3: Train/valid/test split procedure with rolling windows.

[2] https://github.com/yahoo-finance
[3] https://www.kaggle.com/datasets/sudalairajkumar/cryptocurrencypricehistory
[4] https://www.kaggle.com/datasets/brunotly/foreign-exchange-rates-per-dollar-20002019

TABLE 6.1: Dataset statistics

Dataset	Market	Freq	Number	Days	From	To	Source
US Stock	US	1d	29	2517	12/01/03	21/12/31	Yahoo
China Stock	China	1h	47	1036	16/06/01	20/09/01	Yahoo
Crypto	-	1d	9	2014	16/01/01	21/07/06	Kaggle
FX	-	1d	22	5015	00/01/03	19/12/31	Kaggle

6.5.2 Baselines

We conduct experiments with 7 representative FinRL methods including 3 classic RL methods: A2C [196], PPO [57], SAC [14], 4 RL-based trading methods: EIIE [91], Investor-Imitator (IMIT) [13], SARL [93] and DeepTrader (DT) [4] and our AlphaMix+. Descriptions of baselines are as follows:

- A2C [196] is a classic actor-critic RL algorithms that introduce an advantage function to enhance policy gradient update by reducing variance.
- PPO [57] is a proximal policy optimization that constrain the difference between current policy and updated policy with simplified clipping term in the objective function.
- SAC [14] is a widely-used off-policy actor-critic method based on the maximum entropy RL framework to encourage algorithm robustness.
- SARL [93] proposes a state-augmented RL framework, which leverages the price movement prediction as additional states, based on deterministic policy gradient [55] methods.
- DeepTrader (DT) [4] is a policy gradient based method. To tackle the risk-return balancing issue, it simultaneously uses negative maximum drawdown and price rising rate as reward functions to balance between profit and risk with an asset scoring unit.
- EIIE [91] is a deterministic policy gradient based RL framework, which contains: 1) an ensemble of identical independent evaluators topology; 2) a portfolio vector memory; 3) an online stochastic batch learning scheme.
- Investor-Imitator (IMIT) [13] imitates behaviors of different investors (e.g., oracle/collaborate/public investor) using investor-specific reward functions with a set of logic descriptors.

6.5.3 Training Setup

We perform all experiments on an RTX 3090 GPU with 5 fixed random seeds. We apply grid search for AlphaMix+ on Crypto and FX datasets and apply the same hyperparameters on China and US stock datasets. We try scale parameter k in list $[0.3, 0.5, 0.7, 0.9]$, binomial sample parameter β in list $[0.3, 0.4, 0.5, 0.6, 0.7]$ and temperature T in list $[18, 19, 20, 21, 22]$.We explore batch size in list $[256, 512, 1024]$ and hidden size in range $[64, 128]$, We apply learning rate $7e^{-4}$ for both actor and critic. Adam is used as the optimizer. One full list of hyperparameters is available in Table 6.2. We train AlphaMix+ for 10 epochs on all datasets. It takes about 60 minutes to train and test on China stock, US stock, FX and Crypto datasets, respectively.

For other FinRL methods, there are two conditions: i) if there are authors' official or open-source FinRL library [10] implementations, we apply the same hyperparameters[5] for a fair comparison. This condition applies for A2C, PPO, SAC, SARL and DeepTrader. ii) if there are no publicly available implementations, we reimplement the algorithms and try our best to maintain consistency based on the original papers. This applies for EIIE and IMIT.

6.5.4 RL Environment Implementation

In this work, we apply the popular portfolio management environment [10] implemented based on OpenAI Gym [201], which simulates live financial markets with realistic historical market data according to the principle of time-driven simulations. During training, we feed observations of technical indicators as input of RL agents. RL agents generate a portfolio (action) and the environment returns the net value change at each time step as reward. By interacting with the environment, the trading agents will try to derive a trading strategy with high profits. In addition, the environment assumes the trading volume of agents is not very large and has little impact on the market. Then, it is reasonable to use the price fluctuation of offline historical financial data to build a model for reward calculation during online simulation. We provide a concrete example here to further clarify how FinRL environment is built. Considering a simple trading scenario with only one stock, we

[5]Both authors' official or open-source FinRL library [10] implementations are tuned in FinRL domains.

obtain p_t^c and p_{t+1}^c, which denote the close price of the stock at time t and $t+1$, from historical data. The action at time t is to buy k shares of the stock. Then, the reward r_t at time t is the account profit defined as $k * (p_{t+1}^c - p_t^c)$. For state, we use historical market data to calculate technical indicators in Chapter 2 as external state and investors' private information such as remaining cash and current position is applied as internal state. Similar procedures to build RL environment with historical market data have been applied in many FinRL work [4, 10, 93]. Readers may check our open source code for more details of the RL environment.

TABLE 6.2: Hyperparameters of AlphaMix+

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Replay buffer size	10000	Initial step	10000
Layer(MLP)	(128,128)	Stacked frame	3
Evaluation episodes	10	Optimizer	Adam
Temperature	20	Uncertainty	0.5
Actor learning rate	0.0007	Critic learning rate	0.0007
Batch size	256	Action numbers	29(US) 47(China) 22(FX) 9(Crypto)
Discount γ	0.99	Ber_mean	0.5
Non-linear	Sigmoid	Observation	Number of assets $\times$ 11

6.6 Empirical Results

6.6.1 Comparison with Baseline

We compare AlphaMix+ with 7 modern RL algorithms in terms of total return (TR), Sharpe ratio (SR) and entropy (ENT). As shown in Table 6.3, we observe that the MoE framework of AlphaMix+ significantly improve the diversity of strategy and AlphaMix+ generally perform the best across different datasets and time period. IMIT performs the best on some dataset but the performance variance across different year is quite large. In addition, AlphaMix+ mostly outperforms SAC, which is the base model to show the effectiveness of MoE here. For Crypto market, AlphaMix+ does not achieve top performance. One possible reason is that the Crypto market is quite volatile and unstable, it is hard to achieve consensus opinions from different experts. The performance of other 7 FinRL methods are quite unstable and highly relies on time and markets. The performance volatility of Crypto market is the highest and FX market is the lowest.

	US Stock 2019			US Stock 2020			US Stock 2021		
Models	TR(%)	SR	ENT	TR(%)	SR	ENT	TR(%)	SR	ENT
A2C	20.5	1.47	1.83	7.46	0.36	1.96	12.4	1.03	2.26
PPO	21.5	1.53	1.82	18.3	0.62	1.82	13.1	0.94	1.82
SAC	27.3	1.72	1.63	7.77	0.37	1.61	20.0	1.30	1.67
SARL	27.7	2.00	2.08	9.23	0.41	2.07	17.3	1.37	2.79
DeepTrader	22.9	1.60	1.81	10.2	0.42	1.82	17.2	1.22	1.82
EIIE	30.2	2.12	2.32	15.8	0.55	2.41	16.9	1.39	2.90
IMIT	20.6	1.28	1.89	-20.6	-0.28	1.85	4.0	0.35	1.89
AlphaMix+	25.1	1.98	3.27	12.7	0.52	3.22	20.7	1.64	3.25
	China Stock 2018			China Stock 2019			China Stock 2020		
Models	TR(%)	SR	ENT	TR(%)	SR	ENT	TR(%)	SR	ENT
A2C	-6.86	-0.28	1.54	31.9	1.59	1.54	5.52	0.35	1.85
PPO	-6.56	-0.31	2.85	29.7	1.65	2.85	6.20	0.40	2.82
SAC	-4.05	-0.12	1.01	25.4	1.13	1.02	13.4	0.59	1.10
SARL	-5.77	-0.25	2.41	32.6	1.80	2.47	11.4	0.63	2.60
DeepTrader	5.67	0.37	1.53	22.1	1.19	1.53	12.3	0.62	1.53
EIIE	-2.27	-0.05	2.55	36.2	1.77	1.97	13.0	0.73	2.39
IMIT	-7.14	-0.34	1.30	23.2	1.21	1.66	52.6	2.68	1.30
AlphaMix+	-0.70	0.03	3.12	32.2	1.79	3.15	14.8	0.79	3.12
	Crypto 2019			Crypto 2020			Crypto 2021		
Models	TR(%)	SR	ENT	TR(%)	SR	ENT	TR(%)	SR	ENT
A2C	82.8	1.37	0.60	61.4	1.12	0.59	223	1.38	1.02
PPO	85.8	1.39	0.62	59.1	1.12	0.59	146	1.22	0.72
SAC	73.9	1.27	0.43	16.0	0.51	0.45	377	1.13	0.47
SARL	61.0	1.26	1.21	31.4	0.81	1.32	199	1.41	1.42
DeepTrader	39.1	0.83	0.59	37.9	0.85	0.60	398	1.71	0.56
EIIE	50.8	1.15	1.24	30.7	0.79	1.24	146	1.46	1.53
IMIT	110.6	2.06	0.58	55.3	1.03	0.58	140	1.24	0.58
AlphaMix+	68.3	1.43	2.15	33.8	0.86	2.17	291	1.87	2.09
	FX 2017			FX 2018			FX 2019		
Models	TR(%)	SR	ENT	TR(%)	SR	ENT	TR(%)	SR	ENT
A2C	6.93	1.34	1.34	-5.43	-1.01	1.38	-0.94	-0.22	1.44
PPO	6.85	1.32	1.35	-5.25	-0.97	1.34	-1.49	-0.34	1.37
SAC	8.25	1.41	0.90	-5.61	-0.91	0.94	-1.02	-0.21	0.93
SARL	5.81	1.34	2.17	-4.52	-0.93	2.23	0.96	0.29	2.55
DeepTrader	5.94	1.13	1.37	-4.99	-0.91	1.37	-0.19	-0.03	1.35
EIIE	6.71	1.26	1.41	-6.15	-1.15	1.55	1.42	0.39	2.23
IMIT	6.42	0.81	3.08	-5.53	-0.97	3.08	-5.53	-0.97	3.08
AlphaMix+	7.16	1.72	2.98	-4.92	-1.15	2.98	1.22	0.41	2.97

TABLE 6.3: Performance comparison of FinRL algorithms across different time period and financial markets

6.6.2 Ablation Studies

We further conduct ablation studies on AlphaMix+'s performance to show the effectiveness of proposed components. There are 4 variants of AlphaMix+ where different components are added or removed indicated by check marks. We observe that the overall performance with all components is the best in terms of TR and SR for all datasets. By introducing ensemble learning, the performance of all AlphaMix+ is much better than the base model SAC.

US Stock											
Models	Ensemble	Weight BB	Diveristy	TR(%)↑	SR↑	CR↑	SoR↑	MDD(%)↓	VOL(%)↓	ENT↑	ENB↑
SAC				7.77	0.37	0.30	0.50	42.8	2.45	1.61	1.10
AlphaMix+	√			10.2	0.46	0.50	0.55	31.6	2.15	3.25	1.01
	√	√		10.1	0.45	0.48	0.54	32.3	2.15	3.28	1.01
	√		√	10.9	0.47	0.52	0.57	32.0	2.19	3.26	1.02
	√	√	√	12.7	0.52	0.47	0.62	38.2	2.16	3.22	1.00
China Stock											
Models	Ensemble	Weight BB	Diveristy	TR(%)↑	SR↑	CR↑	SoR↑	MDD(%)↓	VOL(%)↓	ENT↑	ENB↑
SAC				13.4	0.59	0.41	0.71	29.9	0.81	1.10	1.27
AlphaMix+	√			13.6	0.75	0.77	0.83	19.0	0.69	3.15	1.01
	√	√		11.6	0.66	0.68	0.73	18.7	0.68	3.09	1.02
	√		√	12.0	0.68	0.70	0.76	18.7	0.68	3.20	1.02
	√	√	√	14.8	0.79	0.53	0.89	29.1	0.69	3.12	1.02
Models	Ensemble	Weight BB	Diveristy	TR(%)↑	SR↑	CR↑	SoR↑	MDD(%)↓	VOL(%)↓	ENT↑	ENB↑
Crypto											
SAC				16.0	0.51	0.59	0.59	55.7	4.46	0.45	1.07
AlphaMix+	√			24.4	0.73	0.72	0.76	44.8	3.26	2.08	1.00
	√	√		27.7	0.77	0.78	0.82	46.5	3.42	2.14	1.00
	√		√	32.2	0.83	0.86	0.89	48.0	3.62	2.19	1.00
	√	√	√	33.8	0.86	0.87	0.94	46.5	3.46	2.17	1.00
Foreign Exchange											
Models	Ensemble	Weight BB	Diveristy	TR(%)↑	SR↑	CR↑	SoR↑	MDD(%)↓	VOL(%)↓	ENT↑	ENB↑
SAC				-1.02	-0.21	-0.02	-0.26	6.94	0.31	0.93	2.02
AlphaMix+	√			1.06	0.33	0.26	0.58	4.24	0.21	2.98	1.22
	√	√		1.11	0.34	0.27	0.60	4.47	0.22	2.91	1.19
	√		√	1.04	0.33	0.25	0.57	4.34	0.21	3.01	1.19
	√	√	√	1.22	0.41	0.36	0.74	3.77	0.19	2.97	1.18

TABLE 6.4: Ablation study on four datasets of year 2020 to show the effectiveness of proposed components

6.6.3 Parameter Analysis: Probing Sensitivity

To avoid parameter sensitivity, we study the performance of AlphaMix+ with different hyperparameters. Specifically, we show the total return and sharpe ratio with different values of binomial sample β, temperature T and scale factor k. As shown in Figure 6.4, we find that AlphaMix performs consistently well with different choose of paprameters that proofs its robustness. The best performance is gotten when $\beta = 0.7$, $T = 19$ and $k = 0.3$.

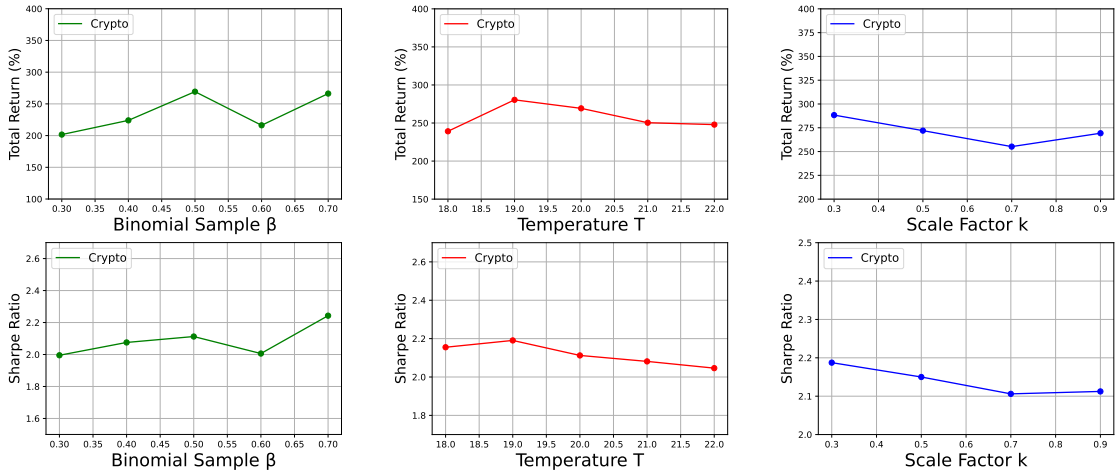


FIGURE 6.4: Hyperparameter sensitivity experimental results on Crypto

Chapter 7

Systematic Evaluation of RL Algorithms for Quantitative Trading

7.1 Introduction

In the previous chapters, we introduced novel FinRL algorithms with great performance for different trading scenarios. However, the evaluation of existing FinRL methods [5, 7, 10, 12] only focuses on profit-related measures, which ignores several critical axes, such as risk-control and reliability. In addition to profitability, there are many other aspects of FinRL algorithms that financial practitioners care about, i.e., how much risk I need to take for per unit of profit; how FinRL algorithms behave when the market status suddenly changes. In preliminary experiments, we find many examples that indicate the weakness of existing profit-seeking FinRL algorithms. For instance, IMIT [13] may lead to catastrophic capital loss when black swan events happen and SAC [14] shows poor risk-control ability, which is not an ideal option for conservative traders. In practice, with the existence of low signal-to-noise ratio and distribution shift in financial markets [15], FinRL methods with only great profitability performance on backtesting are very likely to overfit on historical data and become risky to be deployed in real-world scenarios [16]. As Robert Pardo (CEO of a hedge fund) said, he will never trade with a method that

does not prove itself through a systematic evaluation [17]. Therefore, a benchmark for the systematic evaluation of FinRL methods setting is urgently needed.

In this chapter[1], we first introduce a benchmark called PRUDEX-Compass, which has 6 axes with a total of 17 measures for systematic evaluation of FinRL methods. In addition, we evaluate 8 widely used FinRL methods on 4 long-term real-world datasets on popular trading tasks to demonstrate the usage of PRUDEX-Compass. Accompanied with an open-source library[2] of datasets, baseline implementation, RL environment and evaluation toolkits, we call for a change in how we evaluate FinRL methods to facilitate the industry deployment of FinRL methods. Moreover, PRUDEX-Compass also provides the RL community new algorithm evaluation scenarios to test the effectiveness of novel RL algorithms in the ever-changing financial markets.

7.2 PRUDEX-Compass: Systematic Evaluation of FinRL

To provide a clear exposition of FinRL evaluation, we introduce PRUDEX-Compass to provide an intuitive visual means to give financial practitioners a sense of comparability and positioning of FinRL methods. PRUDEX-Compass is composed of two central elements: i) the axis-level (inner), which specifies the different axes considered for FinRL evaluation and ii) measure-level (outer), which specifies the measures used for benchmarking FinRL methods. Figuratively speaking, the axis-level maps out the relative strength of FinRL methods in terms of each axis, whereas the measure-level provides a compact way to visually assess which setup and evaluation measures are practically reported to point out how comprehensive the evaluation are for FinRL algorithms. To provide a practical basis, we have directly filled the exemplary compass visualization in Figure 7.1. The contributions of PURDEX-Compass are three-fold: i) carefully collecting 17 measures from the literature of multiple disciplines (e.g., finance, artificial intelligence, statistics and engineering) and properly categorizing them into 6 axes; ii) proposing customized version of the measures suitable for FinRL together with a set of easy-to-use visualization tools; iii) introducing PRUDEX-Compass, a unified visual interface

[1]This chapter has been published in [194].
[2]https://github.com/TradeMaster-NTU/PRUDEX-Compass

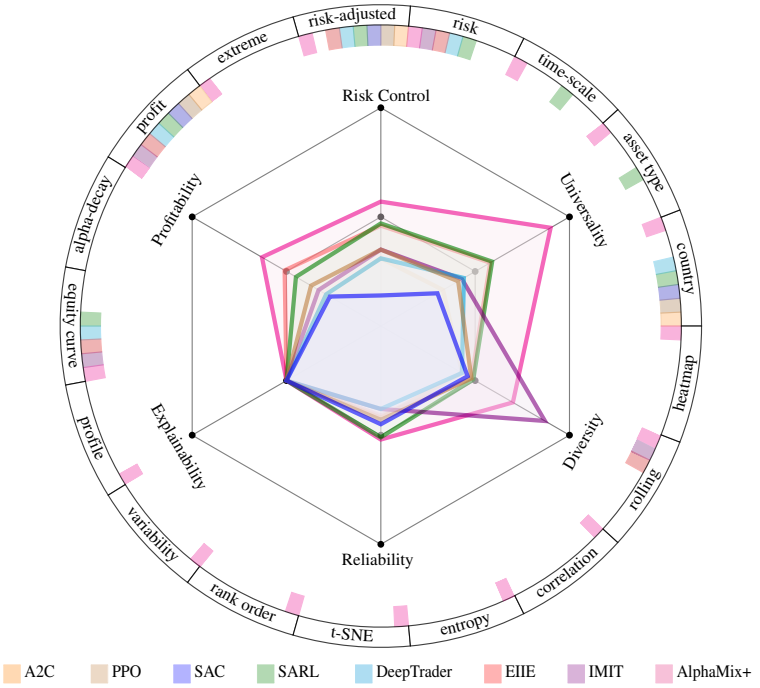


FIGURE 7.1: Illustration of our PRUDEX-Compass. The inner star plot provides a visual indication of the relative strength of different FinRL methods in terms of six axes. A mark on the star plot's inner circle suggests the market average. The outer level of the PRUDEX-Compass specifies which particular measures have been evaluated in practice to show the status of evaluation. We fill the compass with 8 widely-used FinRL methods to provide an intuitive example.

composed of two core elements to indicate the relative performance strength of FinRL methods (inner part) and their evaluation completeness (outer part). We remark that the visualization design of PRUDEX-Compass is inspired and built on top of the template code provided in CLEVA-Compass [202].

7.2.1 Axes of PRUDEX-Compass

We choose the design of the axis-level compass as a star diagram following the idea of [202, 203]. Specifically, the axis-level element contains two hexagons, marks on all vertices of the two hexagons[3], and lines connecting the central point and vertices of hexagons. We add marks in the middle of central point and outer circle of the axis-level to indicate the performance of market average for convenient check. To indicate the relative strength of FinRL methods in terms of 6 critical evaluation

[3]The edges and vertices of the inner hexagon become slightly blurry but still distinguishable after filling in the results of 8 FinRL algorithms.

perspectives, we first calculate the normalized score of each FinRL algorithm in terms of each axis by normalizing the numeric values of original experimental results into an integer score $t \in [0, 100]$. Inspired by the widely used performance ratio in Arcade Learning Environment [53, 204], we choose to normalize FinRL methods performance based on their relative performance to market average strategy, which is considered as the golden standard for many financial practitioners. Specifically, we assign score $t = 50$ to market average and introduce a parameter $k\%$, where $k\%$ better or worse relative to the market average is assigned 100 and 0, respectively. We further clip score values, which are larger than 100 to 100 and lower than 0 to 0, to alleviate the impact of extreme values (e.g., extremely high return rate under markets with sudden increase of some Crypto assets). For the value choice of k, we find it is an empirically robust option to set $k = 20$ for our experiments on 8 FinRL methods across 4 financial markets. Moreover, we consult with our industry collaborators and they agree that it is reasonable to consider 20% better than the market average as a very successful trading strategy (e.g., $t = 100$). Users can directly follow our settings and compare the score (larger is better). We then decide the position of vertices based on the score and connect them to provide a visual impression on the performance of each FinRL method. The advantages of this design are three-fold: i) it provides an ideal visual representation when comparing plot elements [205], ii) it allows human perceivers to quickly learn more facts by fast visual search [206], and iii) the geometric region boundaries in the star plot have high priority in perception [207]. We introduce the 6 axis-level elements of PRUDEX-Compass as follows:

Profitability. Aligned with the key objective of financial trading, profitability focuses on the evaluation of FinRL methods' ability to gain market capital. Besides pure return, it also measures how stable [208] and persistent [19] FinRL methods are to achieve high profit.

Risk-Control. Due to the well-known tradeoff between profit and risk in finance [209], financial practitioners take great efforts on the assessment and control of both systematic risk and idiosyncratic risk [210], which is also of vital importance in FinRL evaluation.

Universality. The financial market is a complex ecosystem that involves innumerable assets, countries, time-scale and trading styles. Universality tries to evaluate FinRL's ability to achieve satisfied performance (e.g., better than market average)

in various quantitative trading scenarios. Designing FinRL methods with better universality [7] is in line with popular machine learning topics such as transfer learning [211] and meta learning [212].

Diversity. In finance, diversification refers to the process of allocating capitals in a way that reduces the exposure to any one particular asset or risk. As Markowitz (Nobel Laureate in Economics) said [213], diversity is the only free lunch in investing that plays an indispensable role on enhancing profitability and risk-control. In RL community, diversity is widely used to encourage exploration [214]. This axis of PRUDEX-Compass tends to address the lack of diversity evaluation of FinRL methods.

Reliability. RL methods have the tendency to be highly variable in performance and sensitive to many factors such as random seeds [156] and market stationarity shift across time [215]. This variability issue hinders a reliable method and can be costly or even dangerous for high-stake applications such as quantitative trading. This axis introduces techniques on RL reliability evaluation [158, 159] with a focus on quantitative trading.

Explainability. Psychologically speaking, if the users do not trust a model, they will not use it [216]. Explainability generally refers to any technique that helps users or developers of models understand why models behave the way they do. In FinRL, it can come in the form that tells traders which model is effective under what market conditions or why one trading action is mistaken and how to fix it. Rigorous regulatory requirements in financial markets further enhance its importance for model debugging [217], monitoring [218] and audit [217].

7.2.2 Measures of PRUDEX-Compass

As the inner star plot contextualizes macroscopic axes of FinRL evaluation, the outer measure-level places emphasis on detailed evaluation setup and metrics. In essence, a mark on the measure-level indicates that a method practically reports corresponding measures in its empirical investigation, where more marks indicate a more comprehensive evaluation. We list the 17 measures on the outer level of the PRUDEX-Compass in Table 7.1 with brief descriptions. In addition, we leave measures of FinRL explainability as future work due to the lack of FinRL algorithms

with solid design of explainability. We conduct literature review on RL explainability and point their potential application in FinRL as follows. DSP [219] is proposed to discover symbolic policy with expert knowledge. Differentiable decision trees are incorporated into RL for better explainability [220]. Another line of works tries to discover interpretable features with techniques such as self-supervised learning [221] and adversarial learning [222]. Open-XAI [223] offers a comprehensive open-source framework for evaluating and benchmarking post hoc explanation methods. We plan to incorporate suitable evaluation methods, i.e., LIME [216] and SHAP [224], from Open-XAI into PRUDEX-Compass with customized adoption on decision-based FinRL methods.

7.2.3 Creating a PRUDEX-Compass

To make the PRUDEX-Compass as accessible as possible and disseminate in a convenient way, we provide two options for practical use based on the template provided in CLEVA-Compass [202].

- We provide a **LaTeX template** for PRUDEX-Compass, making use of the TikZ library to draw the compass with. We envision that such a template makes it easy for readers to include a compass into their future research, where they can adapt the naming and values of the entries respectively.
- We further provide a **Python script** to generate the PRUDEX-Compass. In fact, because the use of drawing in LaTeX with TikZ may be unintuitive for some, we have written a Python script that automatically fills above LaTeX template, so that it can later simply be included into a LaTeX document. The Python script takes a path to a JSON file that needs to be filled by the user. There is a default JSON file that is easy to adopt.

7.2.4 Computing Scores for PRUDEX-Compass Axes

We introduce how the value of each axe for each FinRL method is computed in this subsection. The basic principle is to propose a distinguishable and robust way to show the performance difference of FinRL methods across multiple evaluation measures in terms of each axe. Generally speaking, we normalize the performance

TABLE 7.1: Brief summary of evaluation measures in outer level of PRUDEX-Compass: Profitability, Risk-control, Universality, Diversity, rEliability, and eXplainability.

Axes	Measures	Descriptions
P	Profit	A class of metrics to assess FinRL's ability to gain market capital.
	Alpha Decay	Loss in the investment decision making ability of FinRL methods over time due to distribution shift in financial markets [225].
	Equity Curve	A graphical representation of the value changes over time.
R	Risk	A class of metrics to assess the risk level of FinRL methods [226].
	Risk-adjusted Profit	A class of metrics that calculate the normalized profit with regards to different kinds of risks, i.e., volatility and downside risk [227].
	Extreme Market	The relative performance of FinRL methods on extreme market condition during black swan events [228] such as war and covid-19.
U	Country	Financial market across both developed countries (e.g., US and Europe) and developing countries (e.g., China and India).
	Asset Type	Various financial asset types, i.e., stock, future, FX and Crypto
	Time-Scale	Both coarse-grained (e.g., day level) and fine-grained (e.g., second level) financial data to match different trading styles.
D	t-SNE	A statistical visualization tool to map high-dimensional time-series data points into 2-D dimension [229] to assess the data-level diversity.
	Entropy	Entropy-based metrics from information theory [230] to show the diversity of FinRL methods' trading behaviors.
	Correlation	Metrics that account the correlation [231] between financial assets to assess the diversity of FinRL methods.
	Diversity Heatmap	A visualization tool to demonstrate the diversity of investment decisions among different financial assets with heatmap [232]
E	Performance Profile	A visualization of FinRL methods' empirical score distribution [233], which is easy to read with qualitative comparisons.
	Variability	The performance standard deviation across different random seeds and hyperparameters [156].
	Rolling Window	Using rolling time window to retrain or fine-tune FinRL methods and evaluate the performance on multiple test periods [16].
	Rank Comparison	A visualization toolkit to show the rank of FinRL methods across different metrics, which will not be dominated by extreme values [158].
X	-	We discuss current status and highlight promising further directions.

of different measures on the 6 axes to a score from 0 to 100. We mark market average strategy (evenly invest on all assets) as 50. All scores are calculated with the average of 4 financial markets. We define the metrics value for FinRL methods and market average as m_{rl} and m_{ave}, respectively. For the 4 profit-related metrics (TR, SR, CR, SoR), we normalized them into a score S_{pro} with range

$[0, 100]$: $S_{pro} = (m_{rl}/m_{ave} - 0.8) * 250$, where 20% higher profit than market average is scored 100. We clip values lower than 0 and higher than 100 as 0 and 100, respectively. For the 2 risk metrics (Vol, MDD), we normalized them into a score S_{risk} with range $[0, 100]$: $S_{risk} = (1.2 - m_{rl}/m_{ave}) * 250$, where 20% lower risk than market average is scored 100. We clip values lower than 0 and higher than 100 as 0 and 100, respectively.

For universality, we directly use the raw return rate and calcualte the 4 indicators of profitability, we then plot a rank graph and for each measures of profitability, we multiply the probability obtained from the rank matrix and the rankscore(if it's 1st, the score is 100 and if 6th, the score is 0) to get a score for that measure. Then we average the 4 measures together to get the university score for that algorithm. For the 2 diversity metrics (ENT, ENB), we normalized them into a score S_{div}:

$$S_{div} = \begin{cases} m_{rl}/m_{ave} \times 100, & \text{for ENT} \\ m_{rl}/m_{ave} \times 50, & \text{for ENB} \end{cases}$$

where the diversity of uniform policy is scored 100 for ENT and 50 for ENB. For explainability, we set the explainability score 50 for all FinRL methods. For reliability, we use the total return rate score we just normalized using average policy as a indicator and draw a performance profile graph, then we use the area under the curve for each algorithm to calculate the reliability. The constants in the equations (e.g., 20%) are applied to make the plot distinguishable and easy to follow, which does not influence the robustness of axes in PRUDEX-Compass.

7.3 Demonstrative Usages of Evaluation Toolkits

In this section, we conduct experiments on portfolio management with real-world datasets of 4 influential financial markets to demonstrate the usage of PRUDEX-Compass and related evaluation toolkits. In section 7.3.1, we show how different investors can get a general impression on FinRL algorithms' performance with PRUDEX-Compass. Moreover, we provide example usage of other evaluation toolkits with a focus on one particular perspective including: i) t-SNE plot to show data-level diversity, ii) PRIDE-Star to report the performance of 8 point-wise financial metrics for evaluating profitability, risk-control and diversity, iii)

performance profile and rank distribution plot as unbiased and robust measures towards reliable FinRL methods, iv) portfolio diversity heatmap to evaluate the decision-level diversity, v) extreme market scenarios with black swan events to evaluate the risk-control and generalization ability of FinRL algorithms. In particular, investors can either use these evaluation toolkits together with PRUDEX-Compass to pursue a systematic evaluation or as an independent measure with a focus on the perspective they care about.

7.3.1 A General Impression with PRUDEX-Compass

As shown in Figure 7.1, we fill the PRUDEX-Compass based on the experimental results of the 8 FinRL methods. For axis-level, it directly illustrates the relative performance of each method in terms of 6 axes to provide a general impression. We normalized the score into 0 to 100 with 50 as the market average. For explainability, all methods are scored 50 as we leave it as future direction. AlphaMix+ performs best in all 5 remaining axes. Specifically, it outperforms other FinRL methods 53% and 43% in universality and diversity, respectively, which demonstrates the effectiveness of the weighted Bellman backup and diversified bootstrap initialization.

For measure-level, we give a mark if one measure is used in the evaluation of the FinRL methods, the goal here is to show how comprehensive the methods are evaluated. Together with all measures we proposed, AlphaMix+ clearly has a more rigorous evaluation, which makes the results more trustworthy. Arguably, PRUDEX-Compass provides a compact visualization to evaluate FinRL methods that is much better than only looking at a result table of different metrics especially when lots of FinRL methods are involved. In other words, the compass highlights the required subtleties, that may otherwise be challenging to extract from text descriptions, potentially be under-specified and prevent readers to misinterpret results based on result table display. With PRUDEX-Compass, users can flexibly pick suitable methods with regards to their personal interests. Conservative traders may prefer methods with a relative stable profit rate and low risk. Aggressive traders may pay more attention on profitability, as they are willing to take high risk to pursue extremely high profit. For international trading firms, they may have high expectation on universality and diversity.

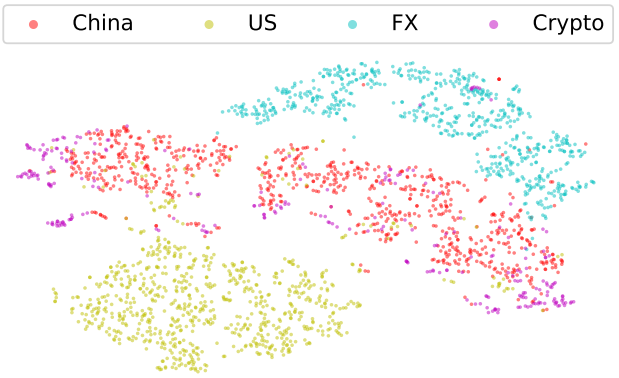


FIGURE 7.2: t-SNE market visualization.

7.3.2 Visualizing Financial Markets with t-SNE

Even though it is a wide consensus that different financial markets share different trading patterns [234], there is a lack of visualization tool to demonstrate how different are these markets. To show data-level diversity of evaluation, we use t-SNE [229] here to map all 4 datasets into a 2-D dimension plot with the 11 features described in Table 2.2 as the input. To avoid overlapping in financial data, we pick a data point every 30 time step for each asset across 4 financial markets. Each data point corresponds to the value vector of 11 features at each time step, where necessary temporal information is maintained. In Figure 7.2, the US stock and FX datasets lie in the lower left and upper right corner, respectively, as a whole cluster, which is consistent with their status as relative mature and stable markets [235]. For China stock and the Crypto, data points are scattered with more outliers that demonstrate their essence as emerging and violate markets [236]. The t-SNE plot is useful to provide an intuitive expression on the data-level diversity of different markets while evaluating FinRL methods.

7.3.3 Evaluating Profitability, Risk-Control and Diversity

As the evaluation measures for Profitability, RIsk and DivErsity (PRIDE) are point-wise metrics with real number values, we use the PRIDE-star, which is a star plot to show the relative strength of 8 metrics including 1 profit metrics: total return (TR), 2 risk metrics: volatility (Vol) [226] and maximum drawdown (MDD) [227], 3 risk-adjusted profit metrics: Sharpe ratio (SR) [237], Calmar ratio (CR) and Sortino ratio (SoR), and 2 diversity metrics: entropy (ENT) [238] and effect number

of bets (ENB) [231]. We report the overall performance across the 4 financial markets of the 6 FinRL methods in Figure 7.3, where the inner circle represents market average. In general, AlphaMix+ performs best in the PRIDE-Star plot. In addition, AlphaMix+ outperforms the second best by 25% in terms of ENT that shows the effectiveness of the boostrap with random initialization component in AlphaMix+.

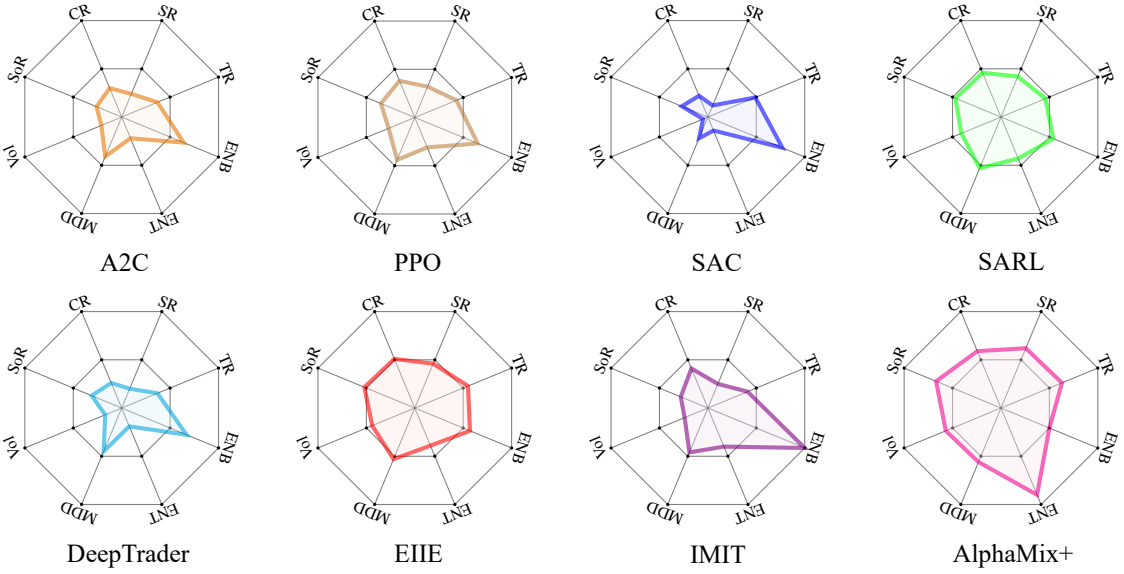
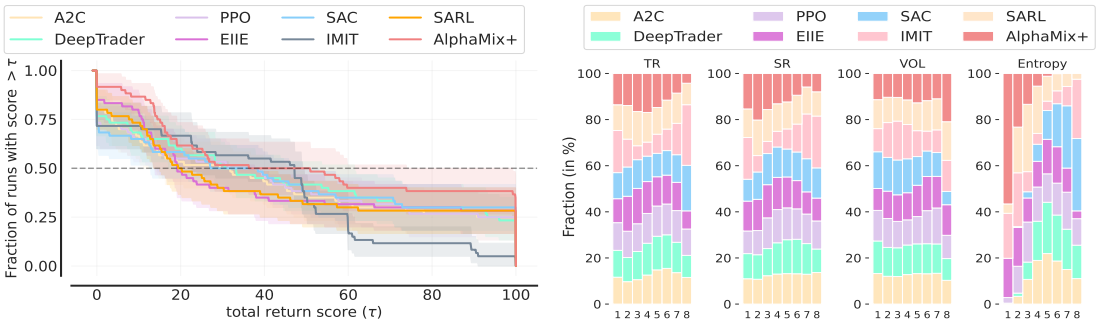


FIGURE 7.3: Overall performance across 4 financial markets on PRIDE-Star to evaluate profitability, risk-control and diversity, where AlphaMix+ achieves the best performance in 7 out of 8 metrics.



(A) (B)

FIGURE 7.4: (a) Performance profile of total return score distributions across 4 financial markets. Shaded regions show pointwise 95% confidence bands based on percentile bootstrap with stratified sampling. (b) Rank distribution in terms of TR, SR, Vol and ENT across 4 financial markets.

7.3.4 An Unbiased Approach to Report Performance

The performance profile [158] reports the score distribution of all runs across the 4 financial markets that are statistically unbiased and more robust to outliers compared to the widely-used mean performance. Performance profiles proposed herein visualize the empirical tail distribution function of a random score (higher curve is better), with point-wise confidence bands based on stratified bootstrap [239]. A score distribution shows the fraction of runs above a certain normalized score that is an unbiased estimator of the underlying performance distribution. As shown in Figure 7.4a, AlphaMix+ is generally a robust but conservative FinRL methods that shows the least bad runs, which makes it an attractive option for conservative investors that care more about risk. However, radical investors may pick SAC as it has the largest probability of achieving score 100, which indicates a return rate higher than twice the market average.

7.3.5 Rank Distribution to Demonstrate the Rank of FinRL methods

In Figure 8.3, we plot the rank distribution [158] of 8 FinRL methods in terms of TR, SR, VOL and Entropy across 4 financial markets with results of 5 random seeds in each market. The i-th column in the rank distribution plot shows the probability that a given method is assigned rank i in the corresponding metrics. For x-axis, rank 1 and 8 indicate the best and worst performance.[4] For y-axis, the bar length of a given method on a given metric with rank i corresponds to the % fraction of rank i it achieves across the 4 financial markets, 3 test periods and 5 random seeds ($4 \times 3 \times 5 = 60$ in total), i.e., if the rank 1 column of TR is purely red, it indicates AlphaMix+ achieves the highest TR in all random seeds across 4 financial markets.

For TR and SR, AlphaMix+ slightly outperforms other methods with 27% and 35% probability to achieve top 2 performance. For Vol, SAC gets the overall best performance while AlphaMix+ goes through higher volatility. For ENT, AlphaMix+ significantly outperforms other FinRL methods with over 56% probability for rank 1, which demonstrates its ability to train mixture of diversified trading experts.

[4]For TR, SR and entropy, higher values indicate better performance. For VOL, lower values indicate better performance

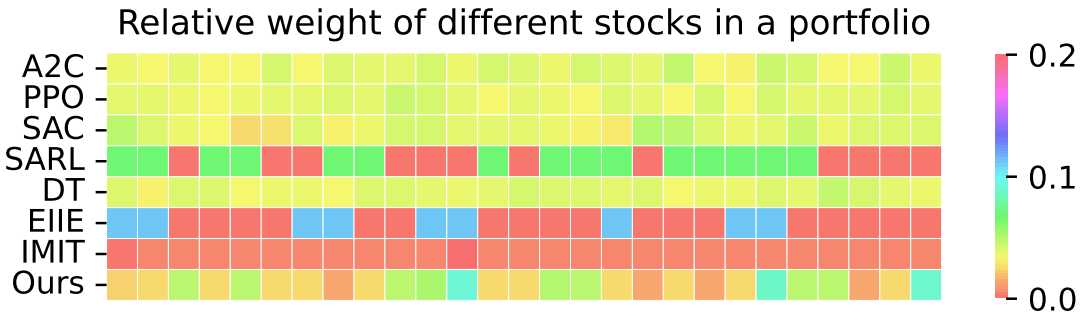


FIGURE 7.5: Heatmap of average portfolio on China stock market.

7.3.6 Visualizing Strategy Diversity with Heatmap

To demonstrate the overall investment diversity of FinRL methods, we show the average portfolio across the test period as a heatmap in Figure 7.5. Formally, we define the average portfolio as $\bar{w}$:

$$\bar{\mathbf{w}} = [\frac{\sum_{j=t}^{t+h} w_i^0}{h+1}, \frac{\sum_{j=t}^{t+h} w_i^1}{h+1}, ..., \frac{\sum_{j=t}^{t+h} w_i^M}{h+1}] \tag{7.1}$$

where $M+1$ is the number of portfolio's constituents, including cash and M financial assets. w_t^i represents the ratio of the total portfolio value invested at time t on asset i, h represents the length of the evaluation period and w_t^0 represents cash.

For IMIT, it puts all capital in one or two assets. The portfolio of SARL and EIIE is not that diversified with near 0 weight on many assets more (red). For A2C, DT, PPO and SAC, the portfolio is closed to uniform, which is not desirable due to poor profitability. Our AlphaMix+ achieves an ideal investment portfolio, which is generally diversified and allocate more weights on a few bullish stocks.

7.3.7 The Impact of Extreme Market Conditions

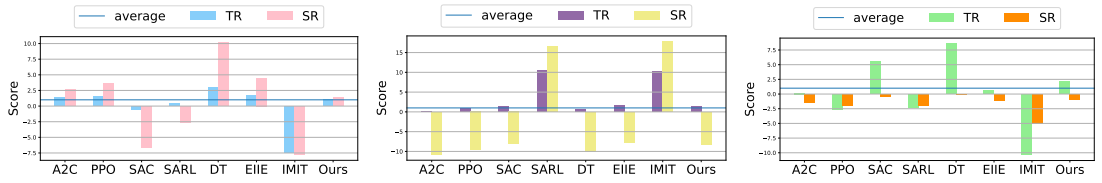
To further evaluate the risk-control and reliability, we pick three extreme market periods with black swan events. For China stock market, the period is from February 1st to March 31st 2021 when strict segregation policy is implemented in China to fight against the COVID-19 pandemic [240]. For US stock market, the period is from March 1st to April 30th 2020, when the global financial markets are violate due the global pandemic of COVID-19 [241]. For Crypto, the period is from April 1st to May 31st 2021 when many countries posed regulation against Crypto oligarch.

To report the results of different metrics with different numerical value scale, we normalize them into a score m_{score} as follows:

$$m_{score} = (m_{ave}/|m_{ave}|)(m_{rl}/m_{ave} - 1) * k + 1 \tag{7.2}$$

We define the metrics value for FinRL methods and market average as m_{rl} and m_{ave}, respectively. k is a scale parameter.

In Figure 7.6, we plot the bar chart of TR and SR during the period of extreme market conditions. As a conservative method, AlphaMix+'s performance is unsatisfactory in extreme market conditions, which proves the general consensus that radical methods such as DeepTrader (DT) and SARL are more suitable for extreme markets [242]. Analyzing the performance on extreme market conditions can shed light on the design of FinRL methods, which is in line with economists' efforts on understanding the financial markets. For instance, incorporating volatility-aware auxiliary task [12] and multi-objective RL [243] in AlphaMix+ may further make it be aware of extreme market conditions in advance and behave as a profit-seeking agents to achieve better performance during extreme market conditions.



(A) US (B) China (C) Crypto

FIGURE 7.6: Performance of FinRL methods during extreme market conditions.

7.4 Discussion

Complementary Related Efforts. Apart from the above aspects described in PRUDEX-Compass, there also exist several orthogonal perspectives for FinRL evaluation, which encompass a check-list [244] for quantitative experiments, the construction of elaborate dataset sheets [245], and the creation of model cards [246]. We stress that these perspectives remain indispensable, as novel datasets and their variants are regularly suggested in FinRL and the PRUDEX-Compass does not disclose intended use with respect to human-centered application domains, ethical

considerations, or their caveats. We believe that it is best to report both the prior works and PRUDEX-Compass together to further improve the evaluation quality.

Potential Impact. For FinRL community, PRUDEX-Compass can serve as i) a systematic evaluation toolkit; ii) a benchmark for comprehensive comparison of FinRL methods and iii) a standard code base for future design and development of novel FinRL algorithms. We hope that PRUDEX-Compass could encourage both researchers and financial practitioners to avoid fooling themselves by evaluating FinRL methods in a systematic way and facilitate the design of stronger FinRL methods. While accounting for all elements in PRUDEX-Compass is not a panacea, we believe it is a good start with more trustworthy results for the community and further increase the confidence for real-world industry deployment. For RL community, PRUDEX-Compass introduces a new challenging scenario with different evaluation axes for the test of novel RL algorithms in financial market. For FinTech community, this work enables industry practitioners with limited RL background to easily explore the potential of RL in different FinTech scenarios. In addition, the usage of PRUDEX-Compass is not limited in RL settings, most elements of PRUDEX-Compass can be easily generalized to supervised learning settings with broader impact.

Future Plans. We plan to improve PRUDEX-Compass from the following perspectives: i) For axis-level, we plan to explore the evaluation of FinRL explainability with measures and plots; ii) For measure-level, we plan to include metrics to evaluate alpha decay and more metrics, e.g., optimality gaps, for profits and risks; iii) To further improve the accuracy and reliability of PRUDEX-Compass, we plan to customize existing scientific and non-exploitative RL evaluation procedures [160] into FinRL domains for the normalization and summarization of experimental results. iv) For a comprehensive evaluation under different market stationarity assumptions, we plan to add one toolkit to automatically categorize market into different styles (e.g., bull/bear) and evaluate the performance of FinRL methods under different styles. Furthermore, there can be a data-driven simulator to generate unseen stylized data for further evaluation; v) For visualization, we plan to further develop a GUI software version accompanied with a website to further lower the barrier for dissemination and use; vi) As most axes and measures in PRUDEX-Compass are also key points for non-RL trading scenarios, we plan to bring PRUDEX-Compass

into more general machine learning settings with implementations and results of non-RL methods for broader impact.

Hosting, Maintenance, Licensing. The PRUDEX-Compass datasets are hosted on Google Drive. The authors will provide important bug fixes to the community as commits to the GitHub repository. There will be summary of changes to the code and the datasets in the README web page of the GitHub repository. In the unlikely case that the Google Drive link stops operating, we will migrate the dataset to another hosting and announce the new links in the GitHub repository. The provided source code and dataset are copyrighted by us and under the MIT license. Users have the permission to reuse the codes for any purpose.

7.5 Chapter Summary

In this chapter, we design a systematic evaluation benchmark on RL for fianncial trading called PRUDEX-Compass. It includes 6 axes with a total of 17 measures for systematic evaluation of RL methods. In addition, we evaluate 8 widely used FinRL methods on 4 long-term real-world datasets on popular trading tasks to demonstrate the usage of PRUDEX-Compass and the superiority of AlphaMix+. Accompanied with an open-source library[5] of datasets, baseline implementation, RL environment and evaluation toolkits, we call for a change in how we evaluate FinRL methods to facilitate the industry deployment of FinRL methods. Moreover, PRUDEX-Compass also provides the RL community new algorithm evaluation scenarios to test the effectiveness of novel RL algorithms in the ever-changing financial markets.

[5]https://github.com/TradeMaster-NTU/PRUDEX-Compass

Chapter 8

A Holistic Trading Platform Empowered by Reinforcement Learning

8.1 Introduction

With the increasing accessibility of financial data [16] and the development of RL techniques [54, 153–155], FinRL has made great strides in offering profitable financial trading models [1] in recent years. From an algorithmic perspective, many FinRL algorithms are proposed for core quantitative trading tasks such as portfolio management [4, 5, 92, 93], algorithmic trading [2, 3, 12], and order execution [6, 7, 247], respectively. In addition, a plethora of research has tried to address other key problems alongside the FinRL pipeline, including preprocessing financial data [21, 248], alpha discovery [249, 250], market simulation [22, 251, 252], systematic evaluation [194] and online deployment [16]. However, the vast majority of existing works are formulated, addressed and implemented separately, resulting in a stylized range of methods with limited acknowledgement on the complexities and interdependencies in the FinRL pipeline (as a composite). This leads to an often punishing translational barrier between state-of-the-art FinRL research and investors' usage of RL techniques to earn real financial benefits [1, 16, 253].

Three Challenges. To facilitate FinRL research and their industry deployments, we call for a holistic platform for the design, development and evaluation of FinRL

applications. Specifically, due to the ever-changing nature of financial markets [254] and high performance variability of RL [156], managing real-world FinRL workflows is non-trivial with the following three challenges:

- First and foremost, the engineering problem is that building sophisticated FinRL pipelines requires significant upfront efforts. For an immature domain (e.g., FinRL), it is common to spend over 90% of work on implementation details and less than 10% of work on core technical questions [20]. As an FinRL researcher, it is seldom the case that enough resources are available to build a high quality code base of the complete FinRL workflow. As a result, a holistic platform that encapsulates all major components of FinRL with comprehensive support of mainstream settings and state-of-the-art algorithms is highly desirable for both academic researchers and industry practitioners.

- Second, the benchmarking problem is that the performance of any component depends on its context. For instance, the profitability of an FinRL algorithm is intimately tied to the features used to train RL agents [249] and the market status during evaluation [17]. However, existing works typically examine the merits of each component individually and arbitrarily configure surrounding settings for only ensuring "all else equal" conditions, which lead to inconsistent reporting of revenues with no general consensus on the relative strength of popular FinRL methods [1]. A reproducible empirical benchmark that honestly reflects interactions of FinRL components is urgently needed.

- Third, the usability problem is incurred by the complexities of FinRL, which requires extensive interdisciplinary knowledge to be properly optimized and maintained in practice. Specifically, many knobs (e.g., hyperparameters) need to be tuned for FinRL methods [255] and regular maintenance (e.g., model retraining with up-to-date market data) is unavoidable due to the significant temporal distribution shift of financial markets [256]. These difficulties are usually insurmountable obstacles for non-expert users (e.g., individual investors) and hinder the wide utilization of FinRL methods. A user-friendly platform where the whole process is highly automated with excellent user interfaces is indispensable for this domain.

Contributions. In this chapter[1], TradeMaster is proposed to tackle the above three challenges in one unified FinRL platform with the following contributions: i) As a software toolkit, it offers open-source implementations of the whole FinRL workflow including 13 real-world financial datasets, high-fidelity RL environments for 6 mainstream quantitative trading tasks, 15 popular FinRL algorithms and dozens of tools for systematic evaluation and visualization. This modular and composable structure enables fast development and decrease the cost of collaboration and code-sharing. ii) As an empirical benchmark, it provides rigorous comparison of state-of-the-art FinRL algorithms to determine their effectiveness across different tasks, financial markets and evaluation metrics. The standardized pipelines of TradeMaster ensure that the comparison is transparent, fair and reproducible. iii) As a user interface, TradeMaster provides multiple options for practical use: an open-source Github repository, a Python package, an online web service with graphical user interface (GUI), multiple Jupyter Notebook tutorials and a cloud version using Colab. In addition, we customize advanced AutoML techniques into TradeMaster to finish many key steps of FinRL (e.g., feature engineering and hyperparameter tuning) in an automatic way. These efforts significantly improve the usability of TradeMaster and make it accessible for users with various background and investment goals.

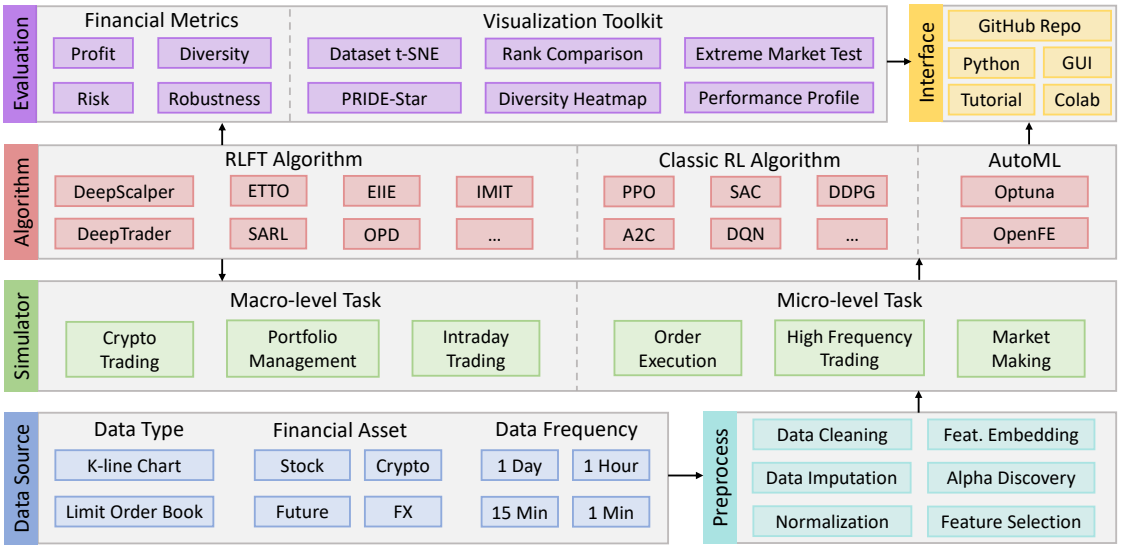


FIGURE 8.1: Overview of the TradeMaster platform, which includes six key components for FinRL.

[1]This chapter has been published in [257].

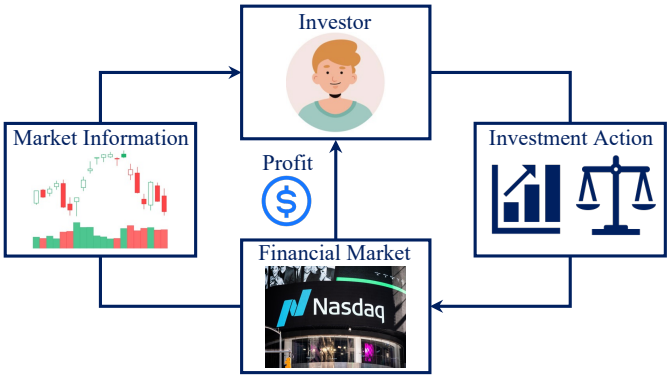


FIGURE 8.2: MDP formulation in FinRL

8.2 Quantitative Trading as a Markov Decision Process

As shown in Figure 8.2, we formulate quantitative trading tasks as a Markov Decision Process (MDP) following a standard RL scenario, where an agent (investor) interacts with an environment (the financial markets) in discrete time to make actions (investment decision) and get reward (profits). Formally, we define the MDP as a 6-tuple: $(\mathcal{S}, \mathcal{A}, P, R, \gamma, H)$. Specifically, $\mathcal{S}$ is a finite set of states. $\mathcal{A}$ is a finite set of actions. $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \longrightarrow [0, 1]$ is s state transition function, which consists of a set of conditional transition probabilities between states. $R : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}$ is the reward function. $\gamma \in [0, 1)$ is the discount factor and H is a time horizon indicating the length of the trading period. A (stationary) policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$, parameterized by θ, assigns each state $s \in \mathcal{S}$ a distribution over actions, where $a \in \mathcal{A}$ has probability $\pi(a|s)$. During training, one episode corresponds to making investment decisions at each time step through one whole trading period. The objective of the agent is to learn an optimal policy (investment strategy) that maximizes the expected sum of discounted reward (overall profit): $\pi_{\theta^*} = argmax_{\pi_\theta} \mathbb{E}_{\pi_\theta} \left[\sum_{i=0}^{T} \gamma^i r_{t+i} \mid s_t = s \right]$. We further describe details on the definition of $\mathcal{S}$, $\mathcal{A}$, R and P in existing works as follows:

State Space ($\mathcal{S}$): A state represents RL agents' perception on market status. Specifically, price information (e.g., open, high, low and close price), trading volume, technical indicators and snapshot of limit order book are used in existing works for different trading tasks [5, 7, 12, 95]. In addition, some works feed alternative data, i.e., financial news [97], prediction of future trend [93], experts' investment

behaviors [13] and overall market status [4], into states to further improve the performance of RL agents. Furthermore, trader's personal information (e.g., holding position and remaining cash) is also applied as private state in [12, 64, 247] to help investment decision making.

Action Space ($\mathcal{A}$): At each time step, the action space represents all possible actions of RL agents' investment decisions that vary under different trading tasks [1]. For single asset trading, the actions are the number of shares to buy/sell or long/short the asset as used in [2, 3]. For portfolio management, the actions represent the proportions of capitals allocated into each financial asset as in [4, 5]. For order execution and market making, the actions correspond to a limit order that shows traders' desired price and quantity as in [6, 7].

Reward Function (R): Previous works [3, 5, 91] commonly apply the change of capitals (how much money earned/losed) as the reward function, which is natural and easy to implement. Many practical constraints such as commission fee and slippage are further included to be more realistic. In addition, there are also several alternative options designed for specific settings including Sharpe ratio in [4], the hindsight reward function in [12] and the binary sparse reward function in [6].

Transition Function (P) is defined by the financial market progression following the investment actions. The progression is defined by building realistic data-driven simulated environment based on real-world financial data following the paradigm in [22, 258].

Examples of the MDP in FinRL. Considering a simple trading scenario with only one stock, we obtain p_t^c and p_{t+1}^c, which denote the close price of the stock at time t and $t+1$, from historical data. The action at time t is to buy k shares of the stock. Then, the reward r_t at time t is the account profit defined as $k * (p_{t+1}^c - p_t^c)$. For state, we use historical market data to calculate technical indicators as external state and investors' private information such as remaining cash and current position is applied as internal state. Similar procedures to build RL environments with historical market data have been applied in many FinRL work [4, 21, 93].

Further Clarification. We would like to highlight that the above descriptions on the MDP formulation is a general version. As a holistic platform, TradeMaster covers a wide range of trading scenarios, which is impossible to offer detailed MDP formulation for each scenario here due to page limit. For readers unfamiliar with

FinRL, this survey [12] offers a detailed introduction on related terminologies and detailed MDP formulation of each FinRL task.

8.3 TradeMaster as a Software Toolkit

To provide a high-quality implementation of the complete FinRL workflow, we first introduce our 3 design principles [253, 259] based on our experience in prototyping and developing FinRL projects. Then, we discuss details on the six component modules of TradeMaster as shown in Figure 8.1.

TradeMaster Design Principles

⋆ **Pipeline First, Algorithms Second:** Due to the strict "separation of concerns" produced by the high-level API functions of each component, the goal of TradeMaster is to first enable the proper pipeline functionality, while the intricacies and configurations of individual algorithm choices on the back burner. ⋆ **Be Minimal and Unintrusive:** While workflow development needs to be unified and systematic, we believe learning to use the platform needs to be easy and intuitive. Concretely, this manifests in the clear code structure and user-friendly interfaces of TradeMaster that enables easy usage and adoption for different users. ⋆ **Encourage Extension:** Since FinRL is an emerging research direction with many novel methods proposed rapidly, the platform components should be extensible for the incorporation of new methods. Concretely, TradeMaster encapsulates the design of functions within each component module with support of easy extension.

Data Sources. We include 13 long-term real-world financial datasets that are highly diverse in terms of financial assets (e.g., stock, FX, and Crypto), financial markets (e.g., US and China), data granularity (e.g., day-level and minute-level) and dataset sizes (e.g., small, medium and large-scale). We summarize statistics of these datasets in Table 8.1. We host all raw data and corresponding datasheets [245] accessible for users to download on both Google Drive and Baidu Cloud. We further offer scripts for calling API data providers, i.e., Yahoo Finance and AKShare, for users who desire to build personalized datasets. We also plan to further expand our collection of datasets in future updates.

Preprocessing. We follow the DataOps paradigm [22, 260, 261] in this component to implement an efficient pipeline for financial data engineering. During data

TABLE 8.1: Dataset statistics of market, frequency, asset amounts, size, chronological period and source

Dataset	Market	Freq	# Assets	Size	From	To	Source
DJ30	US Stock	1 day	29	72994	12/01/03	21/12/31	Yahoo
SP500	US Stock	1 day	363	2009568	00/01/01	22/01/01	Yahoo
Russell	US Stock	1 day	691	2391650	07/09/26	22/06/29	Yahoo
KDD17	US Stock	1 day	41	149407	07/09/26	22/06/29	Yahoo
ACL18	US Stock	1 day	74	264954	07/09/26	22/06/29	Yahoo
SSE50	China Stock	1 hour	26	134680	16/06/01	20/08/30	Yahoo
HSTech	HK Stock	1day	30	60120	88/12/30	23/03/27	AKShare
HSI	HK Stock	1 day	72	206866	07/09/27	22/06/29	AKShare
Future	Future	5 min	20	20370	23/03/07	23/03/28	AKShare
FX	FX	1 day	22	110330	00/01/01	19/12/31	Kaggle
USDCNY	FX	1 day	1	5014	00/01/01	19/12/31	Kaggle
Crypto	Crypto	1 day	1	2991	13/04/29	21/07/06	Kaggle
BTC	Crypto	1 min	1	17113	21/04/07	21/04/19	Binance

cleaning, we offer scripts to remove duplicated data and merge data from different sources into one unified format. In addition, we apply a conditional score-based diffusion model [248] to conduct data imputation for missing values. For technical indicators, TradeMaster supports both a version of 11 features [46, 184] and Alpha 158 [18], which are widely used for research papers [45, 194] and industry scenarios, respectively. Multiple data normalization methods are supported, where z-score serves as the default choice. In addition, users can choose to pick a threshold that filters out valuable features with sufficiently large information coefficient on the return for feature selection. After preprocessing, the raw data is converted into high quality ready-to-use data that can be directly used to build realistic data-driven RL environments.

Environments. We build data-driven market environments with real-world financial data following the de facto standard of OpenAI gym [22, 201] and the paradigm in [22, 258]. Each environment has three functions: `reset()` function that resets the environment back to the initial state s_0; `step()` function that takes an action a_t of the agent and updates state from s_t to s_{t+1}; `reward()` function computes the reward value generated by action a_t. To make the environments more realistic, we incorporate many practical constraints (e.g.,, transaction costs, slippage and non-negative balance) for different scenarios. There are also options for leveraging, short and loss stopping.

Specifically, the current version supports 6 mainstream quantitative trading tasks

[1]: i) For Crypto trading, investors continuously buy/sell cryptocurrencies (e.g., Bitcoin) to make profits. ii) For portfolio management, investors hold a pool of different financial assets and reallocate the proportion of capitals invested in each asset periodically to pursue stable long-term profits. iii) For intraday trading, investors finish the buy/sell or long/short actions within the same trading day to capture the fleeting intraday opportunities and all positions are sold out on market price at the end of the day to avoid overnight risk. iv) Order execution is a micro-level trading task that finish the execution goal within a fixed time horizon (e.g., selling 10 thousand shares of Apple stock in 20 minutes) and try to minimize the costs at the same time. v) Market making focuses on providing liquidity of the market by trading on both buy/sell sides at the same time. vi) High frequency trading utilizes micro-level information and trades in a high-frequency to capture the tiny local price fluctuation.

Algorithms. Although there has been extensive works on RL algorithms for different trading scenarios [4, 7, 12], only a tiny fraction of authors choose to make their code publicly available due to the intense competitions and zero-sum game essence in financial markets [262]. As far as we know, TradeMaster is the first open-source platform that offers standardized implementations of a wide range of state-of-the-art FinRL algorithms. For algorithms with official source code, we convert the code into TradeMaster's code structures following the same networks, hyperparameters and other details. This condition fits for SARL [93], DeepTrader [4] and OPD [7]. For other algorithms without publicly available implementations, we reimplement them and try our best to maintain consistency based on the original papers. Brief descriptions of FinRL algorithms implemented in TradeMaster are as follows:

- SARL [93] proposes a state-augmented RL framework, which leverages the price movement prediction as additional states, based on deterministic policy gradient [55] methods.
- DeepTrader (DT) [4] is a policy gradient based methods, which leverages both maximum drawdown and return rate as reward functions to balance between risk and profit.

- EIIE [91] is considered as the first deep FinRL method with an ensemble of identical independent evaluators topology, a portfolio vector memory, and an online stochastic learning scheme.
- Investor-Imitator (IMIT) [13] is an RL approach that imitates behaviors of different types of investors using investor-specific reward functions with a set of logic descriptions.
- DeepScalper (DS) [12] is a risk-aware RL framework for intraday trading, which contains both micro-level and macro-level market embedding, a hindsight reward function to capture long-term trend, and a volatility prediction auxiliary task to model risk.
- ETEO [6] is designed based on the classic proximal policy optimization (PPO) algorithm for order execution with a sparse reward function and a market encoder using LSTM.
- OPD [7] is developed based on ETEO [6] and further adds a policy distillation mechanism to learn from teacher trained based on perfect market information.

Besides advanced FinRL algorithms, TradeMaster also includes 9 efficient customized implementations of classic RL algorithms based on the widely used RLib library [263] including PPO [57], A2C [196], Rainbow [264], SAC [14], DDPG [56], DQN [265], DDQN [266], PG [267], TD3 [268] for various quantitative trading tasks.

Evaluations. To provide a systematic evaluation, TradeMaster offers a plethora of evaluation measures and visualization tools based on PRUDEX-Compass [194] to assess FinRL algorithms from 5 axes: profitability, risk-control, universality, diversity and reliability. Specifically, we implement a range of financial metrics including profit metrics (e.g., return rate), risk-adjusted profit metrics (e.g., Sharpe ratio [237]) and risk metrics (e.g., volatility and maximum drawdown). Besides evaluating on regular point-wise financial metrics, we further provide many robust measures and evaluation tools to pursue a comprehensive evaluation. For instance, the performance profile [233] and rank distribution [158] plots are included as two unbiased and robust measures towards reliable FinRL methods. We apply t-SNE [229] and heatmap to show data-level and decision-level diversity in different financial markets. We also provide tools to evaluate FinRL methods under extreme market conditions with black swan events.

Interfaces. TradeMaster provides multiple options for practical use. Specifically, we host an open-source Github repository, build a Python package, serve an online web service with graphical user interface (GUI)[2], offer dozens of hands-on Jupyter Notebook tutorials for quick start and develope a cloud version using Colab to improve accessibility. More details on the usage of different user interfaces are available in Section 8.5.

8.4 TradeMaster as an Empirical Benchmark

Evaluating any algorithm depends on its context. For instance, how well an FinRL algorithm ultimately performs depends on the choices of many factors such as datasets, preprocessing and evaluation measures. While current FinRL works typically seek to isolate individual gains through all-else-equal configurations in experiments, there is often limited overlap in pipeline configurations across different studies. To promote transparency and comparability, TradeMaster aims to serve as a structured evaluation framework by providing a comprehensive FinRL empirical standard. Next, we describe how we benchmark FinRL algorithms using TradeMaster with an demonstrative example of portfolio management task on US stock markets. We first introduce experimental setup and analyze the performance of FinRL algorithms. To offer a more comprehensive analysis, we further report the results of an unbiased and robust measure called rank distribution plots and performance on extreme market conditions with black swan events.

8.4.1 Experimental Setup

We conduct reproducible experiments of the classic portfolio management task on a stock pool of Dow Jones 30 index using the datasets in Table 8.1 to benchmark results of state-of-the-art FinRL algorithms. Due to limited space, we briefly introduce the experimental setup in the following table with red outlines.

[2]http://trademaster.ai

TABLE 8.2: Performance comparison (mean of 5 individual runs) on the US stock market of 8 FinRL algorithms in terms of 8 financial metrics. Pink and green indicate best and second best results.

	TR(%)↑	SR↑	CR↑	SoR↑	MDD(%)↓	VOL(%)↓	ENT↑	ENB↑
A2C	51.92	0.750	1.510	1.468	32.44	1.373	2.161	1.373
DDPG	56.95	0.800	1.614	1.562	32.37	1.372	2.667	1.265
EIIE	49.66	0.756	1.526	1.468	30.86	1.358	3.368	1.110
IMIT	-4.95	0.102	0.240	0.234	50.52	2.078	1.460	2.149
SARL	52.13	0.752	1.544	1.490	32.23	1.420	2.758	1.192
TD3	57.15	0.804	1.616	1.564	32.36	1.373	2.608	1.214
PG	51.17	0.742	1.492	1.452	32.49	1.373	2.146	1.385
PPO	50.99	0.742	1.490	1.450	32.49	1.372	2.136	1.393

8.4.2 Performance Comparison

We report the overall performance of 8 methods in Table 8.2. TD3 achieves the best profitability with highest values in TR, SR, CR and SoR, while DDPG is also a good option with slightly lower performance. For RL algorithms designed for trading (e.g., EIIE, IMIT and SARL), they fail to outperform properly tuned classic RL algorithms such as TD3 in terms of profitability. For risk-control, EIIE shows best performance with the lowest MDD and VOL. For diversity measures, EIIE and IMIT perform the best for ENT and ENB, respectively. The overall performance of IMIT is much worse comparing to other methods. One possible reason is the lack of high-quality data of professional investmentors' behaviors [13]. Generally speaking, no existing algorithm achieves dominating results on all financial metrics and there are still vast opportunities to improve in this filed. In addition, we observe that classic RL algorithms (e.g., DDPG and TD3) still outperform FinRL trading algorithms with proper hyperparameter tuning in terms of profit-related metrics (e.g., TR, SR, CR and SoR). At the same time, FinRL algorithms (e.g., EIIE, IMIT and SARL) achieve portfolios with better risk-control and diversity, which demonstrate the effectiveness of the risk-related components. We hope the reproducible benchmark results from TradeMaster can help researchers easily notice the relative strength of existing algorithms and inspire the design of new algorithms.

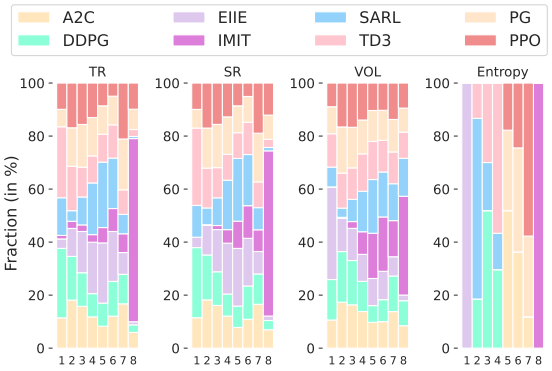


FIGURE 8.3: Rank distribution in terms of 4 financial metrics on the US stock market.

8.4.3 Rank Distribution to Demonstrate the Rank of FinRL methods

In Figure 8.3, we plot the rank distribution plot [158] of 8 FinRL methods in terms of TR, SR, VOL and Entropy across 3 test periods with results of 5 random seeds in each period. It is more robust to outliers comparing to the widely-used mean performance, where the i-th column in the rank distribution plot shows the probability that a given method is assigned rank i in the corresponding metrics. For x-axis, rank 1 and 8 indicate the best and worst performance. For y-axis, the bar length of a given method on a given metric with rank i corresponds to the % fraction of rank i it achieves across the 3 test periods and 5 random seeds ($3 \times 5 = 15$ in total). For TR and SR, TD3 slightly outperforms DDPG with 24% and 27% probability to achieve the best performance. For Vol, EIIE gets the overall best performance while TD3 goes through higher volatility. For ENT, EIIE significantly outperforms other methods with 100% probability for rank 1, which demonstrates its ability to construct portfolios with enough diversity.

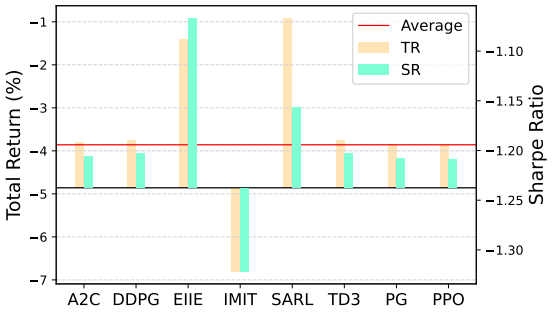


FIGURE 8.4: Performance of FinRL methods during extreme market conditions.

8.4.4 The Impact of Extreme Market Conditions

To further provide an evaluation of risk-control and reliability, we pick one extreme market period with black swan events during testing. In this case, the period is September 1-30 in 2021, when the US stock market is violate and goes through the largest decrease after COVID-19 due to the concern on interest rate increase and the congressional shutdown. In Figure 8.4, we plot the bar chart of TR and SR during the period of extreme market conditions. The red line indicates the market average. Classic RL methods (e.g., A2C, DDPG, TD3, PG, PPO) achieves similar performance comparing to the market average as they have the tendency to keep conservative during extreme market conditions. In contrast, radical methods such as EIIE and SARL are more suitable options with better performance [242]. This analysis on extreme market conditions can shed light on the design of FinRL methods, which is in line with economists' efforts on understanding the financial markets. For instance, researchers may incorporate volatility-aware auxiliary task [12] into the design of FinRL algorithms to make them be aware of extreme market conditions in advance and behave as a profit-seeking agents during extreme market conditions.

8.4.5 Demonstrative Usage of the AutoML Component

To compare the results with or without automatic feature generation, we run PPO & SARL algorithms on DJ30 dataset for 5 epochs. We utilize the OpenFE library [269] to expand the candidate features using the original technical indicators of the dataset, and select the top-ranked 10 features as generated features to be appended to the dataset for the model training and testing. The quantitative results in Table 8.3 mirror the relative improvement of the total return using automatic feature generation compared to directly training by the base dataset.

TABLE 8.3: Comparison of results with/without feature generation & selection

Algorithm	Feature Generation	TR(%)	SR	VOL(%)	MDD (%)
PPO	×	16.42±0.38	1.38±0.03	0.73±0.01	6.73±0.06
	✓	**16.60±0.27**	**1.40±0.03**	**0.73±0.01**	**6.72±0.10**
SARL	×	17.05±2.07	1.46±0.22	0.73±0.03	6.55±0.44
	✓	**17.49±2.95**	**1.50±0.17**	**0.72±0.03**	**6.51±1.01**

8.4.6 Convergence Curve of Different Algorithms

In figure 8.5, we show the convergence curve of all 8 algorithms after training for 10 epochs. It shows that 10 epochs, which corresponds to over 20 years of market data, is enough for all algorithms to converge. Unlike most algorithms that train RL agents to make investment decisions directly, IMIT focuses on imitating the trading behaviors of different experts with two stages. In stage one, IMIT trains multiple supervised learning trading experts through optimizing various financial metrics (e.g., return rate and Sharpe ratio). In stage two, IMIT trains RL agents to construct profitable portfolios by dynamically picking pre-trained experts. The goal of IMIT's RL training is to dynamically pick different pre-trained experts under suitable market status. This is a hard task because the advantage difference among various experts can be not obvious and usually require lots of efforts to identify. For instance, one expert with the best return rate can also achieve good Sharpe ratio at the same time because these two metrics are correlated. As a result, it takes IMIT more epochs to converge due to the difficulty of the task. The start point (performance on epoch 0) of IMIT is higher than other algorithms because all pre-trained supervised learning experts already have reasonably good performance comparing to other algorithms that start from random portfolios.

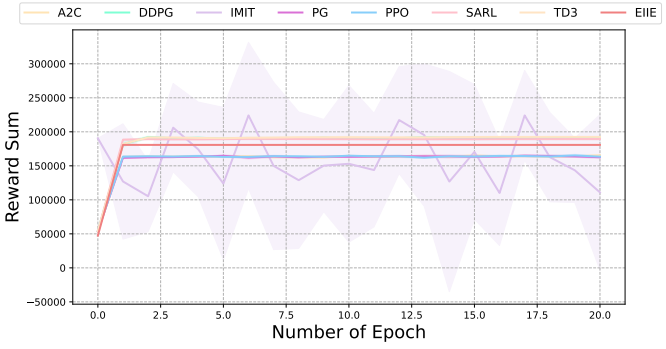


FIGURE 8.5: Convergence curve of different algorithms for training. All 8 algorithms converge after training epochs, where each epoch goes through 10 years of stock market history.

Demonstrative Code of TradeMaster Platform

```
# Load Configuration File
args = parse_args()
cfg = Config.fromfile(args.config)

# Build Environments for Train/Valid/Test
dataset = build_dataset(cfg)
train_env = build_env(cfg, dataset,
   default_args=dict(…, task="train"))
valid_env = build_env(cfg, dataset,
   default_args=dict(…, task="valid"))
test_env = build_env(cfg, dataset,
   default_args=dict(…, task="test"))

# Setup Network and Optimizer
net = build_net(cfg.act)
optimizer = build_optimizer(cfg,
   default_args=dict(…))

# Setup Loss and Transition Function
criterion = build_loss(cfg)
transition = build_transition(cfg)

# Build Reinforcement Learning Agent
agent = build_agent(cfg,default_args=dict(…))

# Build Trainer Based on Environments
trainer = build_trainer(cfg,default_args=dict(
   train_env=train_env, valid_env=valid_env,
   test_env=test_env, agent=agent))

# The Procedure of Training and Validation
trainer.train_and_valid()

# The Procedure of Testing
trainer.test()
```

8.5 TradeMaster as a User Interface

In this section, we introduce our efforts to make TradeMaster a great user interface. First, we show the clean and extendable code structure of TradeMaster with a demonstrative example. Second, we describe the multiple practical usage options for users from different background in TradeMaster. Third, we discuss on the AutoML component to further improve usability.

Demonstrative Code. The code snippet above shows the code of the whole pipeline in TradeMaster including 8 main steps (green code comments), which is simple, clean and easy to extend. We first load the configuration file and setup the environment, network, optimizer, loss and transition function based on it. Then, we build RL agents and trainer, respectively. Later on, we call `train_and_valid()` function for the training and validation of RL agents. Finally, we call `test()` function to test the performance of RL agents on various financial metrics.

Multiple Options for Practical Use. We provide different ways to use TradeMaster for users with different background and investment goals. First, we host a Github repository for TradeMaster including source code, update log and supplementary materials. Second, we provide a Colab version of TradeMaster for users who prefer using cloud computation resources that can run directly. Third, we offer an online web GUI interface. Users can simply pick the settings they are interested in. By clicking on the button, they can train and evaluate FinRL algorithms in a few minutes. This serve as an accessible interface for users with no need on any

expert knowledge. Fourth, we provide dozens of Jupyter Notebook tutorials for educational purposes covering key features of TradeMaster.

Improving Usability with AutoML. To achieve decent performance, many knobs (e.g., features and hyperparameters) need to be properly tuned for FinRL algorithms [1]. However, this requires in-depth knowledge of RL and coding, which is an insurmountable obstacle for many users without AI background. To make TradeMaster more accessible to these users, we implement an AutoML component [270] to finish the FinRL pipeline in a more automatic way. We aim to provide an interface that enables existing AutoML implementations to be conveniently plugged in. Specifically, we support automatic feature generation and feature selection by customizing the OpenFE [269] library, which firstly generates a set of new features by combining many useful operator (e.g., +, -, min, max) of raw data and then efficiently filters out top-ranked features following a synergistic two-stage evaluation module. In addition, we offer an automatic hyperparameter tuning interface based on Optuna library [271] that enables automatic search for optimal values of learning rate, exploration rate and hidden node amounts. We note that the current version based on [269, 271] shows the effectiveness of AutoML for the FinRL pipeline and our unified interface design enables easy extension with other advanced AutoML techniques.

Website GUI Interface. We show an example of using the TradeMaster GUI online service for training and testing RL agents. As shown in Figure 8.6, users can pick the task, financial market and RL algorithm by clicking the corresponding buttons. Later on , we click on the "train" button to start the session for training. The backend server will automatically call the functions to train and test the RL algorithms. The final results are shown in Figure 8.7. We believe the TradeMaster GUI online service is accessible to all types of users and can further benefit the wide distribution of TradeMaster.

8.6 Uniqueness of TradeMaster

As shown in Table 8.4, we compare TradeMaster with existing trading platforms [18, 21] to show its uniqueness as follows: i) while existing platforms claims support of various markets, they only offers links to many data providers that users have

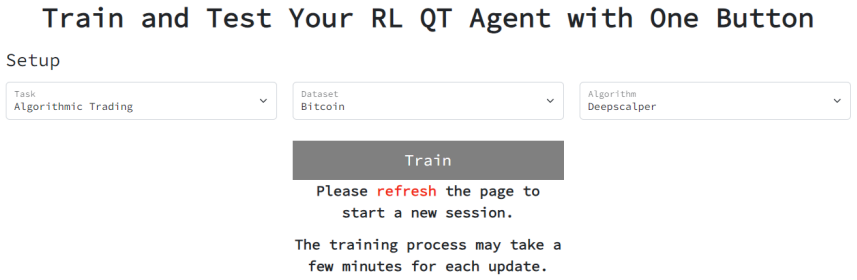


FIGURE 8.6: Experiment configuration page of TradeMaster GUI online service

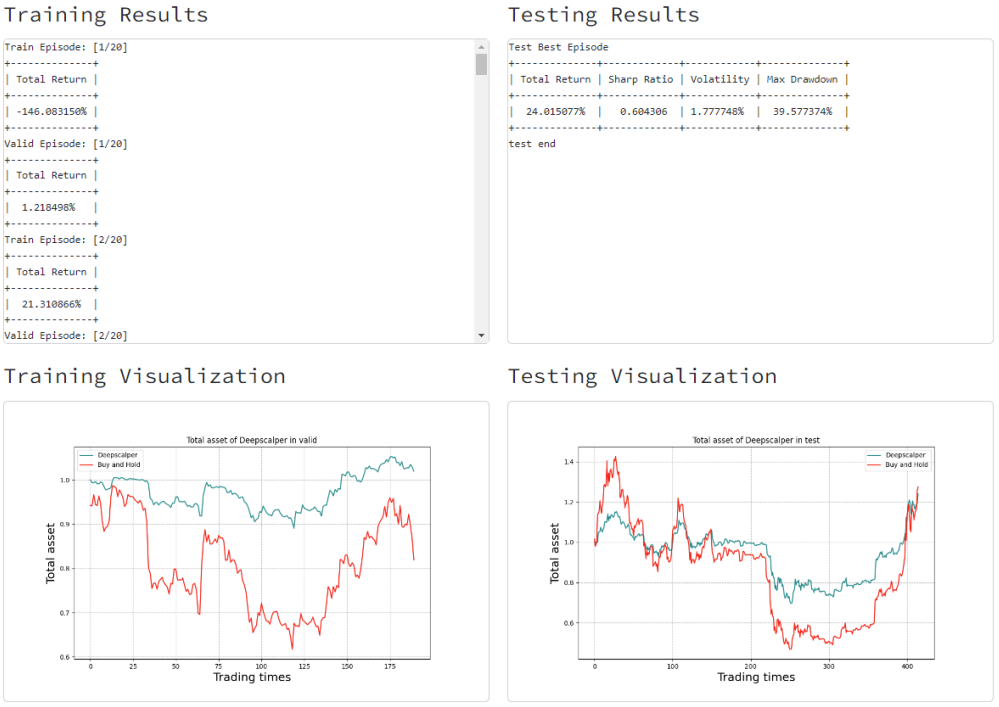


FIGURE 8.7: Demonstrative results of training and testing

to write scripts to acquire data flexibly. TradeMaster prepares 13 well-processed datasets and serve them on Google Drive to be downloaded directly. ii) TradeMaster covers a wide range of trading scenarios with realistic simulation under practical constraints. As far as we know, many scenarios (e.g., intraday trading and market making) are not covered by any existing platforms. iii) TradeMaster provides high-quality implementations of 9 classic RL algorithms and 7 RL for trading algorithms, where about half of them is not covered by existing platforms. iv) TradeMaster include a series of evaluation measures and visualization toolkits to provide a systematic evaluation instead of pure financial metrics in existing platforms. v) We build a GUI interface on the official TradeMaster website for users

without in-depth coding skills to explore the potential of FinRL. vi) We include an AutoML module for automatic feature selection and hyperparameter tuning to further improve usability.

TABLE 8.4: Comparison of TradeMaster and existing trading platforms. # indicates "the number of".

Platform	**# financial markets**	**# algo**	**# trading scenarios**	**# eval metrics**	**# plot toolkits**	**GUI**	**AutoML**
Qlib [18]	2	1	1	4	-	×	×
FinRL [21]	2	8	3	5	-	×	×
TradeMaster	13	16	6	10	6	√	√

8.7 Discussion

Potential Impacts. We hope that TradeMaster can facilitate the development of the FinRL domain with a broad impact on both academic researchers and financial practitioners. For FinRL researchers, they can rapidly implement and compare their own methods with a focus on real scientific problems instead of engineering details. For RL researchers, TradeMaster introduces many challenging trading scenarios for the test of novel RL algorithms in financial markets. For finance researchers and individual investors, we can have a taste on RL-based trading methods without in-depth knowledge of AI and coding. For professional trading firms, TradeMaster can serve as the code base to enhance their internal trading systems with advanced RL techniques.

Desiderata. Machines will never fully replace investors' understanding of the complex economic world, nor researchers' effort on designing novel RL algorithms [253]. TradeMaster enables rapid prototyping, benchmarking and deployment of FinRL procedures, so that investors concentrate on discovering the essential laws of financial markets and researchers can spend more time on core technical challenges. To help grease the wheels, we hope the release of TradeMaster can facilitate the FinRL research and serve as a bridge between finance and AI for broader interdisciplinary impact.

Chapter 9

Conclusion and Future Directions

9.1 Conclusion

Reinforcement learning has shown great potential on various financial decision making tasks. However, there are still many challenges for widely usage of RL in the finance sector. This thesis systematically examines the challenges of deploying RL into real-world financial markets with a focus on the three perspectives of algorithms, evaluation, and software infrastructures.

In Chapter 4, we pay attention to intraday trading, which is one competitive yet highly profitable setting. We propose a risk-aware RL framework namely DeepScalper. First, dueling Q-network with action branching is applied to efficiently train RL agents. Second, we add a hindsight bonus into the reward function to encourage a long-term horizon of price trend. Third, we design an encoder-decoder architecture to learn robust market embedding by incorporating both micro-level and macro-level market information. Finally, volatility prediction is used as an auxiliary task to help agents analyzing market risk while maximizing profit. Extensive empirical evaluations on two stock index futures and four treasury bond futures indicates that DeepScalper significantly outperforms many state-of-the-art methods.

In Chapter 5, we develop a novel three-stage mixture-of-experts framework to mimic the efficient bottom-up workflow in real-world trading firms for financial investment. Specifically, we introduce an efficient ensemble learning method to train multiple trading experts simultaneously with tiny overhead of time and memory.

Then, we conduct model augmentation to construct a pool of diversified trading experts. Finally, we leverage three different mechanisms, namely as-needed router, integrated soup and with-replacement selection, to further improve performance by dynamically routing experts from the expert pool. Extensive experiments on both China and US stock markets indicate that AlphaMix significantly outperforms many strong baselines. Further ablation studies show the effectiveness of the proposed components.

In Chapter 6, we extend the idea of mixture-of-experts into the RL setting. AlphaMix+ is proposed as an universal efficient RL framework. Risk-aware Bellman backup is introduced to reweight Q-values by adopting estimated uncertainty for risk-control. The idea of bootstrap with random initialization is applied to further encourage strategy diversity between different trading agents. Extensive experiments show that AlphaMix+ significantly outperforms other baselines with higher profit, lower risk and better diversity.

In Chapter 7, we develop a systematic evaluation benchmark of RL in finance. The major motivation is that current evaluation mostly uses profit-related numerical metrics and ignores many critical axes. We introduce PRUDEX-Compass, which has 6 axes, i.e., profitability, risk-control, universality, diversity, reliability, and explainability, with a total of 17 measures for a systematic evaluation. We demonstrate the usage of the benchmark and related visualization toolkits. We hope this benchmark can not only shed light on both finance and RL communities.

In Chapter 8, we notice that one bottleneck of FinRL is the lack of software infrastructures. We develop a holistic RL-based trading platform that implements the whole workflow including 13 real-world financial datasets, high-fidelity RL environments for 6 mainstream quantitative trading tasks, 15 popular RLFT algorithms and dozens of tools for systematic evaluation and visualization. This modular and composable structure enables fast development and decrease the cost of collaboration and code-sharing. On the other hand, the platform can also be used as an empirical benchmark and user interfaces. The platform benefits multiple communities with over 1.2k stars on Github.

9.2 Future Directions

Even though existing works have demonstrated the success of RL methods on quantitative trading tasks, this section will point out a few prospective future research directions. Several critical open issues and potential solutions are also elaborated.

Advanced RL techniques. The rapid development of RL can benefit financial decision making. We point out a few possible directions here. First, data scarcity is a major challenge on applying RL for trading tasks. Model-based RL can speed up the training process by learning a model of the financial market [99]. The worst-case (e.g., financial crisis) can be used as a regularizer for maximizing the accumulated reward. Second, the key objective of quantitative trading is to balance between maximizing profit and minimizing risk. Multi-objective RL techniques provide a weapon to balance the trade-off between profit and risk. Training diversified trading policies with different risk tolerance is an interesting direction. Third, the severe distribution shift of financial market makes RL-based methods exhibit poor generalization ability in new market condition. Meta RL and transfer learning techniques can improve RL based trading models' generalization performance across different financial assets or market. Finally, for quantitative trading, learning through directly interacting with the real market is risky and impractical. FinRL normally uses historical data to learn a policy, which fits in offline RL settings. Offline RL techniques can help to model the distribution shift and risk of financial market while training RL agents.

Alternative data and new trading scenarios. Intuitively, alternative data can provide extra information to learn better representation of the financial market. Economic news [122], frequency of prices [112], social media [120], financial events [272] and investment behaviors [121] have been applied to improve performance of financial prediction. For RL-based methods, price movement embedding [93] and market condition embedding [4] are incorporated as extra information to improve an RL agent's performance. However, existing works simply concatenate extra features or embedding from multiple data source as market representation. One interesting forward-looking direction is to utilize multi-modality learning techniques to learn more meaningful representations with both original price and alternative data while training RL agents. Besides alternative data, there are still some important financial

trading settings unexplored by RL researchers. High frequency trading and pairs trading are a few examples. High frequency trading aims at capturing the fleeting micro-level trading opportunities; pairs trading focuses on analyzing the relative trend of two highly correlated assets.

Enhance with auto-ML. Due to the noisy nature of financial data and brittleness of RL methods, the success of FinRL methods highly relies on carefully designed RL components (e.g., reward function) and proper-tuned hyperparameters (e.g., network architecture). As a result, it is still difficult for people without in-depth knowledge of RL such as economists and professional traders to design profitable RL-based trading strategies. Auto-ML, which tries to design high-quality ML applications automatically, can enhance the development of FinRL algorithms from three perspectives: i) For feature engineering, auto-ML can automatically construct, select and collect meaningful features. ii) For hyperparameter tuning, auto-ML can automatically search for proper hyperparameters such as update rule, learning rate and reward function. iii) For neural architecture search, auto-ML can automatically search for suitable neural network architectures for training RL agents. With the assistance of auto-ML, FinRL methods can be more usable for people without in depth knowledge of RL. We believe that it is a promising research direction to facilitate the development of FinRL methods with auto-ML techniques.

Realistic simulation. High-fidelity simulation is the key foundation of RL methods' success. Although existing works take many practical constraints such as transaction fee [273], execution cost [5] and slippage [3] into consideration, current simulation is far from realistic. The ubiquitous market impact, which refers to the effect of one trader's actions to other traders, is ignored. For leading trading firms, their trading volume can account for over 10% of the total volume with a significant impact of other traders in the market. As a result, simulation with only historical market data is not enough. There are some research efforts focusing on dealing with the market impact. Spooner et al. [43] tried to take market impact into consideration for market making with event-level data. Byrd et al. [274] proposed Aides, an agent-based financial market simulator to model market impact. Vyetrenko et al. [275] made a survey on current status for market simulation and proposed a series of stylized metrics to test the quality of simulation. It is a very challenging but important research direction to build high-fidelity market simulators.

Solid benchmark and software infrastructures. In Chapter 7 and 8, we highlight the importance of evaluation benchmark and software platform for research on RL-based financial decision making. There is large space to improve and we offer some options here. For evaluation, current algorithms are tested on a fixed period of time, it will be very useful to build a tool for automatic market style categorization. Then, we can see algorithms' performance under different market style. Explainability is another direction to explore for understanding RL's performance. For software platforms, TradeMaster build a pipeline but can be improved. Distribution RL to accelarate. More state-of-the-art algorithms, more datasets and technical indicators.

Bibliography

[1] Shuo Sun, Rundong Wang, and Bo An. Reinforcement learning for quantitative trading. *arXiv preprint arXiv:2109.13851*, 2021. xvii, 20, 33, 78, 101, 102, 105, 108, 116

[2] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, 28(3):653–664, 2016. 3, 19, 21, 23, 35, 39, 101, 105

[3] Yang Liu, Qi Liu, Hongke Zhao, Zhen Pan, and Chuanren Liu. Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *AAAI*, pages 2128–2135, 2020. 3, 19, 22, 23, 36, 39, 101, 105, 122

[4] Zhicheng Wang, Biwei Huang, Shikui Tu, Kun Zhang, and Lei Xu. Deeptrader: A deep reinforcement learning approach to risk-return balanced portfolio management with market conditions embedding. In *AAAI*, pages 643–650, 2021. 3, 4, 19, 25, 27, 41, 75, 80, 82, 101, 105, 108, 121

[5] Rundong Wang, Hongxin Wei, Bo An, Zhouyan Feng, and Jun Yao. Commission fee is not enough: A hierarchical reinforced framework for portfolio management. In *AAAI*, pages 626–633, 2021. 3, 4, 19, 27, 36, 45, 85, 101, 104, 105, 122

[6] Siyu Lin and Peter A Beling. An end-to-end optimal trade execution framework based on proximal policy optimization. In *IJCAI*, pages 4548–4554, 2020. 3, 75, 101, 105, 109

[7] Yuchen Fang, Kan Ren, Weiqing Liu, Dong Zhou, Weinan Zhang, Jiang Bian, Yong Yu, and Tie-Yan Liu. Universal trading for order execution with oracle policy distillation. In *AAAI*, pages 107–115, 2021. 3, 4, 19, 28, 36, 85, 89, 101, 104, 105, 108, 109

[8] Thomas Spooner, John Fearnley, Rahul Savani, and Andreas Koukorinis. Market making via reinforcement learning. In *AAMAS*, page 434–442, 2018. 4

[9] Thomas Spooner and Rahul Savani. Robust market making via adversarial reinforcement learning. In *IJCAI*, pages 4590–4596, 2020. 4

[10] Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, and Christina Dan Wang. Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance. *ICAIF*, 2022. 4, 81, 82, 85

[11] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *ICML*, pages 6131–6141, 2021. 4, 74, 76, 77

[12] Shuo Sun, Wanqi Xue, Rundong Wang, Xu He, Junlei Zhu, Jian Li, and Bo An. Deepscalper: A risk-aware reinforcement learning framework to capture fleeting intraday trading opportunities. In *CIKM*, pages 1858–1867, 2022. 4, 19, 22, 23, 35, 85, 98, 101, 104, 105, 106, 108, 109, 113

[13] Yi Ding, Weiqing Liu, Jiang Bian, Daoqiang Zhang, and Tie-Yan Liu. Investor-imitator: A framework for trading knowledge extraction. In *KDD*, 2018. 4, 19, 24, 27, 75, 80, 85, 105, 109, 111

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, pages 1861–1870, 2018. 4, 74, 76, 80, 85, 109

[15] Burton G Malkiel. The efficient market hypothesis and its critics. *Journal of Economic Perspectives*, 17(1):59–82, 2003. 4, 85

[16] Marcos Lopez De Prado. *Advances in financial machine learning*. John Wiley & Sons, 2018. 4, 85, 91, 101

[17] Robert Pardo. *The evaluation and optimization of trading strategies*, volume 314. John Wiley & Sons, 2011. 4, 86, 102

[18] Xiao Yang, Weiqing Liu, Dong Zhou, Jiang Bian, and Tie-Yan Liu. Qlib: An ai-oriented quantitative investment platform. *arXiv preprint arXiv:2009.11189*, 2020. 4, 5, 34, 47, 66, 107, 116, 118

[19] Nicolae Gârleanu and Lasse Heje Pedersen. Dynamic trading with predictable returns and transaction costs. *The Journal of Finance*, 68(6):2309–2340, 2013. 4, 34, 88

[20] David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. *NeurIPS*, 2015. 5, 102

[21] Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, and Christina Dan Wang. Finrl: Deep reinforcement learning framework to automate trading in quantitative finance. In *ICAIF*, pages 1–9, 2021. 5, 34, 101, 105, 116, 118

[22] Xiao-Yang Liu, Ziyi Xia, Jingyang Rui, Jiechao Gao, Hongyang Yang, Ming Zhu, Christina Dan Wang, Zhaoran Wang, and Jian Guo. FinRL-Meta: Market environments and benchmarks for data-driven financial reinforcement learning. In *NeurIPS*, 2022. 5, 34, 47, 101, 105, 106, 107

[23] Ernest P Chan. *Quantitative trading: how to build your own algorithmic trading business*. John Wiley & Sons, 2021. 8

[24] Narasimhan Jegadeesh and Sheridan Titman. Returns to buying winners and selling losers: Implications for stock market efficiency. *The Journal of Finance*, 48(1):65–91, 1993. 10, 31

[25] Tobias J Moskowitz, Yao Hua Ooi, and Lasse Heje Pedersen. Time series momentum. *Journal of Financial Economics*, 104(2):228–250, 2012. 10, 35, 45

[26] Louis KC Chan, Narasimhan Jegadeesh, and Josef Lakonishok. Momentum strategies. *The Journal of Finance*, 51(5):1681–1713, 1996. 10

[27] John Bollinger. *Bollinger on Bollinger Bands*. 2002. 10, 35

[28] Thomas M. Cover. Universal portfolios. *Mathematical Finance*, 1(1):1–29, 1991. 11

[29] David P Helmbold, Robert E Schapire, Yoram Singer, and Manfred K Warmuth. On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8(4):325–347, 1998. 11

[30] Alexei A Gaivoronski and Fabio Stella. Stochastic nonstationary optimization for finding universal portfolios. *Annals of Operations Research*, 100(1):165–188, 2000. 11

[31] Dingjiang Huang, Junlong Zhou, Bin Li, Steven HOI, and Shuigeng Zhou. Robust median reversion strategy for on-line portfolio selection. In *IJCAI*, page 2006–2012, 2013. 11

[32] Bin Li, Peilin Zhao, Steven CH Hoi, and Vivekanand Gopalkrishnan. Pamr: Passive aggressive mean reversion strategy for portfolio selection. *Machine Learning*, 87(2):221–258, 2012. 11

[33] Allan Borodin, Ran El-Yaniv, and Vincent Gogan. Can we learn to beat the best stock. *Journal of Artificial Intelligence Research*, 21:579–594, 2004. 11

[34] Bin Li, Steven CH Hoi, and Vivekanand Gopalkrishnan. Corn: Correlation-driven nonparametric learning approach for portfolio selection. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–29, 2011. 11

[35] László Györfi, Gábor Lugosi, and Frederic Udina. Nonparametric kernel-based sequential investment strategies. *Mathematical Finance: An International Journal of Mathematics, Statistics and Financial Economics*, 16(2):337–357, 2006. 11

[36] Bin Li and Steven CH Hoi. Online portfolio selection: A survey. *ACM Computing Surveys*, 46(3):1–36, 2014. 11

[37] Álvaro Cartea, Sebastian Jaimungal, and José Penalva. *Algorithmic and High-frequency Trading*. 2015. 12

[38] Dimitris Bertsimas and Andrew W Lo. Optimal control of execution costs. *Journal of Financial Markets*, 1(1):1–50, 1998. 12

[39] Robert Almgren and Neil Chriss. Optimal execution of portfolio transactions. *Journal of Risk*, 3:5–40, 2001. 12

[40] Sham M Kakade, Michael Kearns, Yishay Mansour, and Luis E Ortiz. Competitive algorithms for vwap and limit order trading. In *EC*, pages 189–198, 2004. 12

[41] Álvaro Cartea, Sebastian Jaimungal, and Jason Ricci. Buy low, sell high: A high frequency trading perspective. *SIAM Journal on Financial Mathematics*, 5(1):415–444, 2014. 13

[42] Dhananjay K Gode and Shyam Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of Political Economy*, 101(1):119–137, 1993. 13

[43] Thomas Spooner, John Fearnley, Rahul Savani, and Andreas Koukorinis. Market making via reinforcement learning. In *AAMAS*, pages 434–442, 2018. 13, 19, 29, 30, 122

[44] Ananth Madhavan. Market microstructure: A survey. *Journal of Financial Markets*, 3(3):205–258, 2000. 13

[45] Fuli Feng, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, and Tat-Seng Chua. Enhancing stock movement prediction with adversarial training. In *IJCAI*, pages 5843–5849, 2019. 14, 30, 55, 107

[46] Jaemin Yoo, Yejun Soun, Yong-chan Park, and U Kang. Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts. In *KDD*, pages 2037–2045, 2021. 14, 38, 45, 55, 56, 58, 66, 107

[47] Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948. 15

[48] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 16

[49] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992. 17

[50] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning Using Connectionist Systems*. University of Cambridge, Department of Engineering Cambridge, UK, 1994. 17

[51] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 17

[52] Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. pages 1008–1014, 2000. 17

[53] 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. 17, 88

[54] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. 17, 33, 101

[55] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, pages 387–395, 2014. 18, 80, 108

[56] 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. In *ICLR*, 2016. 18, 109

[57] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 18, 75, 80, 109

[58] Xiu Gao and Laiwan Chan. An algorithm for trading and portfolio management using q-learning and sharpe ratio maximization. In *NeurIPS*, pages 832–837, 2000. 19, 21, 23

[59] Jae Won Lee and O Jangmin. A multi-agent q-learning framework for optimizing stock trading systems. In *DESA*, pages 153–162, 2002. 19, 21, 23

[60] O Jangmin, Jongwoo Lee, Jae Won Lee, and Byoung-Tak Zhang. Adaptive stock trading with dynamic asset allocation using reinforcement learning. *Information Sciences*, 176(15):2121–2147, 2006. 22, 23

[61] Francesco Bertoluzzo and Marco Corazza. Testing different reinforcement learning configurations for financial trading: Introduction and applications. *Procedia Economics and Finance*, 3:68–77, 2012. 23

[62] Ralph Neuneier. Optimal asset allocation using adaptive dynamic programming. In *NeurIPS*, pages 952–958, 1996. 26, 27

[63] Ralph Neuneier. Enhancing q-learning for optimal asset allocation. In *NeurIPS*, pages 936–942, 1998. 26, 27

[64] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *ICML*, pages 673–680, 2006. 27, 28, 43, 47, 105

[65] Dieter Hendricks and Diane Wilcox. A reinforcement learning extension to the almgren-chriss framework for optimal trade execution. In *Proceedings of the IEEE Conference on Computational Intelligence for Financial Engineering & Economics*, pages 457–464, 2014. 27, 28

[66] Ye-Sheen Lim and Denise Gorse. Reinforcement learning for high-frequency market making. In *ESANN*, 2018. 29, 30

[67] Yueyang Zhong, YeeMan Bergstrom, and Amy Ward. Data-driven market-making via model-free learning. In *IJCAI*, page 2327–2333, 2020. 29, 30

[68] Antonio Riva, Lorenzo Bisi, Pierre Liotet, Luca Sabbioni, Edoardo Vittori, Marco Pinciroli, Michele Trapletti, and Marcello Restelli. Learning fx trading strategies with fqi and persistent actions. In *ICAIF*, pages 1–9, 2021. 19, 22, 23

[69] Renato Arantes de Oliveira, Heitor S Ramos, Daniel Hasan Dalip, and Adriano César Machado Pereira. A tabular sarsa-based stock market agent. In *ICAIF*, 2020. 19, 22, 23

[70] Nicholas Tung Chan and Christian Shelton. An electronic market-maker. technical report. 2001. 29, 30

[71] Thomas Spooner and Rahul Savani. Robust market making via adversarial reinforcement learning. In *IJCAI*, pages 4590–4596, 2020. 19, 29, 30

[72] Gyeeun Jeong and Ha Young Kim. Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, 117:125–138, 2019. 19, 22, 23

[73] Brian Ning, Franco Ho Ting Lin, and Sebastian Jaimungal. Double deep q-learning for optimal execution. *arXiv preprint arXiv:1812.06600*, 2018. 27, 28

[74] Kevin Dabérius, Elvin Granat, and Patrik Karlsson. Deep execution-value and policy based reinforcement learning for trading and beating market benchmarks. *Available at SSRN 3374766*, 2019. 19, 27, 28

[75] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep reinforcement learning for trading. *The Journal of Financial Data Science*, 2(2):25–40, 2020. 19, 22, 23, 46, 75

[76] Yuyu Yuan, Wen Wen, and Jincui Yang. Using data augmentation based reinforcement learning for daily stock trading. *Electronics*, 9(9), 2020. 19, 22, 23, 74, 76

[77] Hui Niu, Siyuan Li, and Jian Li. Metatrader: An reinforcement learning approach integrating diverse policies for portfolio optimization. In *CIKM*, pages 1573–1583, 2022. 26, 27

[78] Molei Qin, Shuo Sun, Wentao Zhang, Haochong Xia, Xinrun Wang, and Bo An. Earnhft: Efficient hierarchical reinforcement learning for high frequency trading. In *AAAI*, pages 14669–14676, 2024. 19, 22, 23

[79] John Moody and Lizhong Wu. Optimization of trading systems and portfolios. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering*, pages 300–307, 1997. 19, 21, 23

[80] John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, 17(5-6):441–470, 1998. 21, 23

[81] John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, 12(4):875–889, 2001. 21, 23

[82] Michael AH Dempster and Vasco Leemans. An automated fx trading system using adaptive reinforcement learning. *Expert Systems with Applications*, 30 (3):543–552, 2006. 21, 23

[83] Weiyu Si, Jinke Li, Peng Ding, and Ruonan Rao. A multi-objective deep reinforcement learning approach for stock index future's intraday trading. In *ISCID*, pages 431–436, 2017. 21, 23

[84] Si Shi, Jianjun Li, Guohui Li, and Peng Pan. A multi-scale temporal feature aggregation convolutional neural network for portfolio management. In *CIKM*, pages 1613–1622, 2019. 24, 27

[85] Ke Xu, Yifan Zhang, Deheng Ye, Peilin Zhao, and Mingkui Tan. Relation-aware transformer for portfolio policy learning. In *IJCAI*, pages 4647–4653, 2020. 19, 24, 27

[86] Eric Benhamou, David Saltiel, Sandrine Ungari, and Abhishek Mukhopadhyay. Bridging the gap between markowitz planning and deep reinforcement learning. *arXiv preprint arXiv:2010.09108*, 2020. 19, 24, 27

[87] Yifan Zhang, Peilin Zhao, Bin Li, Qingyao Wu, Junzhou Huang, and Mingkui Tan. Cost-sensitive portfolio selection via deep reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, 2020. 24, 27

[88] Zhipeng Liang, Hao Chen, Junhao Zhu, Kangkang Jiang, and Yanran Li. Adversarial deep reinforcement learning in portfolio management. *arXiv preprint arXiv:1808.09940*, 2018. 19, 27

[89] Edoardo Vittori, Michele Trapletti, and Marcello Restelli. Option hedging with risk averse reinforcement learning. In *ICAIF*, pages 1–8, 2020. 19, 21, 23

[90] Lorenzo Bisi, Luca Sabbioni, Edoardo Vittori, Matteo Papini, and Marcello Restelli. Risk-averse trust region optimization for reward-volatility reduction. In *IJCAI*, pages 4583–4589, 2019. 19, 24, 27

[91] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059*, 2017. 19, 25, 27, 75, 80, 105, 109

[92] Zhengyao Jiang and Jinjun Liang. Cryptocurrency portfolio management with deep reinforcement learning. In *IntelliSys*, pages 905–913, 2017. 25, 27, 101

[93] Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Ju Xiao, and Bo Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *AAAI*, pages 1112–1119, 2020. 19, 25, 27, 36, 75, 80, 82, 101, 104, 105, 108, 121

[94] Siyu Lin and Peter A Beling. An end-to-end optimal trade execution framework based on proximal policy optimization. In *IJCAI*, pages 4548–4554, 2020. 19, 27, 28

[95] Hongyang Yang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. Deep reinforcement learning for automated stock trading: An ensemble strategy. In *ICAIF*, pages 1–8, 2020. 19, 27, 104

[96] Zhuoran Xiong, Xiao-Yang Liu, Shan Zhong, Hongyang Yang, and Anwar Walid. Practical deep reinforcement learning approach for stock trading. *arXiv preprint arXiv:1811.07522*, 2018. 25, 27

[97] Ramit Sawhney, Arnav Wadhwa, Shivam Agarwal, and Rajiv Shah. Quantitative day trading from natural language using reinforcement learning. In *ACL*, pages 4018–4030, 2021. 19, 25, 27, 104

[98] Wentao Zhang, Yilei Zhao, Shuo Sun, Jie Ying, Yonggang Xie, Zitao Song, Xinrun Wang, and Bo An. Reinforcement learning with maskable stock representation for portfolio management in customizable stock pools. In *WWW*, pages 187–198, 2024. 19, 25, 27

[99] Pengqian Yu, Joon Sern Lee, Ilya Kulyatin, Zekun Shi, and Sakyasingha Dasgupta. Model-based deep reinforcement learning for dynamic portfolio optimization. *arXiv preprint arXiv:1901.08740*, 2019. 19, 26, 27, 121

[100] Olivier Guéant and Iuliia Manziuk. Deep reinforcement learning for market making in corporate bonds: Beating the curse of dimensionality. *Applied Mathematical Finance*, 26(5):387–452, 2019. 19, 29, 30

[101] Jinho Lee, Raehyun Kim, Seok-Won Yi, and Jaewoo Kang. Maps: Multi-agent reinforcement learning-based portfolio management system. *IJCAI*, 2020. 19, 26, 27

[102] Zhenhan Huang and Fumihide Tanaka. A modularized and scalable multi-agent reinforcement learning-based system for financial portfolio management. *arXiv preprint arXiv:2102.03502*, 2021. 19, 26, 27

[103] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973. 21

[104] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *ICML*, pages 1995–2003, 2016. 22, 40

[105] Saud Almahdi and Steve Y Yang. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, 87:267–279, 2017. 24, 27

[106] Jingyuan Wang, Yang Zhang, Ke Tang, Junjie Wu, and Zhang Xiong. Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In *KDD*, pages 1900–1908, 2019. 24, 27, 31

[107] Phillip Murray, Ben Wood, Hans Buehler, Magnus Wiese, and Mikko Pakkanen. Deep hedging: Continuous reinforcement learning for hedging of general portfolios across multiple risk aversions. In *ICAIF*, pages 361–368, 2022. 25, 27

[108] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015. 26

[109] Muchen Zhao and Vadim Linetsky. High frequency automated market making algorithms with adverse selection risk control via reinforcement learning. In *ICAIF*, pages 1–9, 2021. 29, 30

[110] David M. Q. Nelson, Adriano C. M. Pereira, and Renato A. de Oliveira. Stock market's price movement prediction with lstm neural networks. In *IJCNN*, pages 1419–1426, 2017. 30, 55, 67

[111] Jia Wang, Tong Sun, Benyuan Liu, Yu Cao, and Hongwei Zhu. Clvsa: A convolutional lstm based variational sequence-to-sequence model with attention for predicting trends of financial markets. In *IJCAI*, page 3705–3711, 2019. 30

[112] Liheng Zhang, Charu Aggarwal, and Guo Jun Qi. Stock price prediction via discovering multi-frequency trading patterns. In *KDD*, pages 2141–2149, 2017. 30, 36, 67, 121

[113] Thanh Trung Huynh, Minh Hieu Nguyen, Thanh Tam Nguyen, Phi Le Nguyen, Matthias Weidlich, Quoc Viet Hung Nguyen, and Karl Aberer. Efficient integration of multi-order dynamics and internal dynamics in stock movement prediction. In *WSDM*, pages 850–858, 2023. 30

[114] Guang Liu, Yuzhao Mao, Qi Sun, Hailong Huang, Weiguo Gao, Xuan Li, Jianping Shen, Ruifan Li, and Xiaojie Wang. Multi-scale two-way deep neural network for stock trend prediction. In *IJCAI*, pages 4555–4561, 2020. 30

[115] Qianggang Ding, Sifan Wu, Hao Sun, Jiadong Guo, and Jian Guo. Hierarchical multi-scale gaussian transformer for stock movement prediction. In *IJCAI*, pages 4640–4646, 2020. 30, 36, 55, 67

[116] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems*, 37(2):1–30, 2019. 31, 36

[117] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, and Rajiv Shah. Deep attentive learning for stock movement prediction from social media text and company correlations. In *EMNLP*, pages 8415–8426, 2020. 31

[118] Tong Li, Zhaoyang Liu, Yanyan Shen, Xue Wang, Haokun Chen, and Sen Huang. Master: Market-guided stock transformer for stock price forecasting. In *AAAI*, pages 162–170, 2024. 31

[119] Ramit Sawhney, Shivam Agarwal, Arnav Wadhwa, and Rajiv Ratn Shah. Spatiotemporal hypergraph convolution network for stock movement forecasting. In *ICDM*, pages 482–491, 2020. 31, 79

[120] Yumo Xu and Shay B Cohen. Stock movement prediction from tweets and historical prices. In *ACL*, pages 1970–1979, 2018. 31, 36, 55, 58, 65, 121

[121] Chi Chen, Li Zhao, Jiang Bian, Chunxiao Xing, and Tie-Yan Liu. Investment behaviors can tell what inside: Exploring stock intrinsic properties for stock trend prediction. In *KDD*, pages 2376–2384, 2019. 31, 36, 55, 121

[122] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *WSDM*, pages 261–269, 2018. 31, 36, 55, 121

[123] Ramit Sawhney, Piyush Khanna, Arshiya Aggarwal, Taru Jain, Puneet Mathur, and Rajiv Shah. Voltage: Volatility forecasting via text-audio fusion with graph convolution networks for earnings calls. In *EMNLP*, pages 8001–8013, 2020. 31

[124] Wentao Zhang, Lingxuan Zhao, Haochong Xia, Shuo Sun, Jiaze Sun, Molei Qin, Xinyi Li, Yuqing Zhao, Yilei Zhao, Xinyu Cai, et al. Finagent: A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist. *KDD*, 2024. 31

[125] Bryan Lim, Stefan Zohren, and Stephen Roberts. Enhancing time-series momentum strategies using deep neural networks. *The Journal of Financial Data Science*, 1(4):19–38, 2019. 31

[126] Lawrence Takeuchi and Yu-Ying Albert Lee. Applying deep learning to enhance momentum trading strategies in stocks. technical report. 2013. 31

[127] Lakshay Chauhan, John Alberg, and Zachary Lipton. Uncertainty-aware lookahead factor models for quantitative investing. In *ICML*, pages 1489–1499, 2020. 31, 55

[128] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Empirical asset pricing via machine learning. *The Review of Financial Studies*, 33(5):2223–2273, 2020. 31

[129] Shihao Gu, Bryan Kelly, and Dacheng Xiu. Autoencoder asset pricing models. *Journal of Econometrics*, 222(1):429–450, 2021. 31

[130] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990. 32

[131] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. *NeurIPS*, 1994.

[132] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[133] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *ICLR*, 2019. 32, 58, 59, 61, 70, 77

[134] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *NeurIPS*, pages 6514–6527, 2020. 32, 57, 61, 62, 68, 77

[135] Raphael Gontijo-Lopes, Yann Dauphin, and Ekin Dogus Cubuk. No one representation to rule them all: Overlapping features of training methods. In *ICLR*, 2022. 32

[136] Tim Whitaker and Darrell Whitley. Prune and tune ensembles: low-cost ensemble learning with sparse independent subnetworks. In *AAAI*, pages 8638–8646, 2022. 32

[137] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *NeurIPS*, 2017. 32, 57, 62, 68, 77

[138] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 32

[139] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *ICLR*, 2017. 32

[140] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *ICML*, 2022. 32, 62, 64, 65, 70

[141] Isaac Kofi Nti, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. A comprehensive evaluation of ensemble learning for stock-market prediction. *Journal of Big Data*, 7(1):1–40, 2020. 32

[142] Cheng Xiang and WM Fu. Predicting the stock market using multiple models. In *ICARCV*, pages 1–6, 2006. 32

[143] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella Yu. Long-tailed recognition by routing diverse distribution-aware experts. In *ICLR*, 2020. 32, 33, 62, 63

[144] Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. Ensemble selection from libraries of models. In *ICML*, 2004. 32, 62, 64

[145] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. 32

[146] David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013. 32

[147] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ICLR*, 2017. 33

[148] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *KDD*, pages 1930–1939, 2018. 33

[149] Jin Xu, Jingbo Zhou, Yongpo Jia, Jian Li, and Xiong Hui. An adaptive master-slave regularized model for unexpected revenue prediction enhanced with alternative data. In *ICDE*, pages 601–612, 2020. 33

[150] Carlos Riquelme Ruiz, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In *NeurIPS*, 2021. 33

[151] Xin Wang, Fisher Yu, Lisa Dunlap, Yi-An Ma, Ruth Wang, Azalia Mirhoseini, Trevor Darrell, and Joseph E Gonzalez. Deep mixture of experts via shallow embedding. In *UAI*, pages 552–562, 2020. 33

[152] Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, et al. Efficient scaling of language models with mixture-of-experts. In *ICML*, 2022. 33

[153] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017. 33, 101

[154] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022. 33

[155] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J R Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610(7930):47–53, 2022. 33, 101

[156] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI*, 2018. 33, 89, 91, 102

[157] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*, 2018. 33

[158] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *NeurIPS*, 2021. 33, 89, 91, 96, 109, 112

[159] Stephanie CY Chan, Samuel Fishman, Anoop Korattikara, John Canny, and Sergio Guadarrama. Measuring the reliability of reinforcement learning algorithms. In *ICLR*, 2019. 33, 89

[160] Scott Jordan, Yash Chandak, Daniel Cohen, Mengxue Zhang, and Philip Thomas. Evaluating the performance of reinforcement learning algorithms. In *ICML*, pages 4962–4973. PMLR, 2020. 33, 99

[161] Haruka Kiyohara, Ren Kishimoto, Kosuke Kawakami, Ken Kobayashi, Kazuhide Nakata, and Yuta Saito. Towards assessing and benchmarking risk-return tradeoff of off-policy evaluation. In *ICLR*, 2024. 34

[162] N Firth. Why use quantlib. *URL http://www. maths. ox. ac. uk/ firth/research/quantlib. pdf*, 2004. 34

[163] Quantopian. https://github.com/quantopian/zipline. *Github*, 2016. 34

[164] Quantopian. https://github.com/quantopian/pyfolio. *Github*, 2015. 34

[165] Bin Li, Doyen Sahoo, and Steven CH Hoi. Olps: a toolbox for on-line portfolio selection. *Journal of Machine Learning Research*, 17(1):1242–1246, 2016. 34

[166] Bo An, Shuo Sun, and Rundong Wang. Deep reinforcement learning for quantitative trading: Challenges and opportunities. *IEEE Intelligent Systems*, 37(2):23–26, 2022. 35

[167] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. pages 3146–3154, 2017. 36, 46, 67

[168] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Deep learning for event-driven stock prediction. In *IJCAI*, page 2327–2333, 2015. 36

[169] Eugene F Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, 1970. 36, 55

[170] Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. Learn what not to learn: Action elimination with deep reinforcement learning. *NeurIPS*, 2018. 40

[171] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *ICML*, pages 449–458, 2017. 40

[172] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. Action branching architectures for deep reinforcement learning. In *AAAI*, pages 4131–4138, 2018. 40

[173] Gurdip Bakshi and Nikunj Kapadia. Delta-hedged gains and the negative market volatility risk premium. *The Review of Financial Studies*, 16(2): 527–566, 2003. 43

[174] James M Poterba and Lawrence H Summers. Mean reversion in stock prices: Evidence and implications. *Journal of Financial Economics*, 22(1):27–59, 1988. 45

[175] Karsten Schierholt and Cihan H Dagli. Stock market prediction using different neural network classification architectures. In *CIFEr*, pages 72–78, 1996. 46

[176] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014. 46

[177] Rodolfo C Cavalcante, Rodrigo C Brasileiro, Victor LF Souza, Jarley P Nobrega, and Adriano LI Oliveira. Computational intelligence and financial markets: A survey and future directions. *Expert Systems with Applications*, 55:194–211, 2016. 55

[178] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison W Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *IJCAI*, pages 2627–2633, 2017. 55, 67

[179] Hengxu Lin, Dong Zhou, Weiqing Liu, and Jiang Bian. Learning multiple stock trading patterns with temporal routing adaptor and optimal transport. In *KDD*, pages 1017–1026, 2021. 55, 60

[180] Wentao Xu, Weiqing Liu, Chang Xu, Jiang Bian, Jian Yin, and Tie-Yan Liu. Rest: Relational event-driven stock trend forecasting. In *WWW*, pages 1–10, 2021. 55, 56, 58

[181] Michel Ballings, Dirk Van den Poel, Nathalie Hespeels, and Ruben Gryp. Evaluating multiple classifiers for stock price direction prediction. *Expert Systems with Applications*, 42(20):7046–7056, 2015. 56

[182] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56:1513–1589, 2023. 56

[183] Bernard Dumas. The theory of the trading firm revisited. *The Journal of Finance*, 33(3):1019–1030, 1978. 56

[184] Shuo Sun, Xinrun Wang, Wanqi Xue, Xiaoxuan Lou, and Bo An. Mastering stock markets with efficient mixture of diversified trading experts. In *KDD*, page 2109–2119, 2023. 56, 107

[185] Bing Yang, Zi-Jia Gong, and Wenqi Yang. Stock market index prediction using deep neural network ensemble. In *CCC*, pages 3882–3887, 2017. 57, 68

[186] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, pages 3942–3951, 2018. 59

[187] Yeming Wen, Paul Vicol, Jimmy Ba, Dustin Tran, and Roger Grosse. Flipout: Efficient pseudo-independent weight perturbations on mini-batches. In *ICLR*, 2018. 60

[188] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019. 61, 71

[189] Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *NeurIPS*, pages 512–523, 2020. 65

[190] Guizhu Shen, Qingping Tan, Haoyu Zhang, Ping Zeng, and Jianjun Xu. Deep learning with gated recurrent unit networks for financial sequence predictions. *Procedia Computer Science*, 131:895–903, 2018. 67, 71

[191] Mahdi Pakdaman Naeini, Hamidreza Taremian, and Homa Baradaran Hashemi. Stock market value prediction using neural networks. In *CISIM*, pages 132–136, 2010. 67

[192] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: Unbiased boosting with categorical features. In *NeurIPS*, 2018. 67

[193] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *NeurIPS*, pages 7436–7447, 2021. 74

[194] Shuo Sun, Molei Qin, Xinrun Wang, and Bo An. PRUDEX-compass: Towards systematic evaluation of reinforcement learning in financial markets. *Transactions on Machine Learning Research*, 2023. 74, 86, 101, 107, 109

[195] Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010. 75

[196] 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 *ICML*, pages 1928–1937, 2016. 75, 80, 109

[197] Aviral Kumar, Abhishek Gupta, and Sergey Levine. Discor: Corrective feedback in reinforcement learning via distribution correction. *NeurIPS*, pages 18560–18572, 2020. 76

[198] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 77

[199] Dong Yu, Kaisheng Yao, Hang Su, Gang Li, and Frank Seide. Kl-divergence regularized deep neural network adaptation for improved large vocabulary speech recognition. In *ICASSP*, pages 7893–7897, 2013. 77

[200] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. *NeurIPS*, 2016. 77

[201] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 81, 107

[202] Martin Mundt, Steven Lang, Quentin Delfosse, and Kristian Kersting. Clevacompass: A continual learning evaluation assessment compass to promote research transparency and comparability. In *ICLR*, 2021. 87, 90

[203] Wenhui Wang, Hangbo Bao, Li Dong, Johan Bjorck, Zhiliang Peng, Qiang Liu, Kriti Aggarwal, Owais Khan Mohammed, Saksham Singhal, Subhojit Som, et al. Image as a foreign language: Beit pretraining for all vision and vision-language tasks. In *CVPR*, pages 19175–19186, 2023. 87

[204] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018. 88

[205] Johannes Fuchs, Petra Isenberg, Anastasia Bezerianos, Fabian Fischer, and Enrico Bertini. The influence of contour on similarity perception of star glyphs. *IEEE Transactions on Visualization and Computer Graphics*, 20(12): 2251–2260, 2014. 88

[206] James Elder and Steven Zucker. The effect of contour closure on the rapid discrimination of two-dimensional shapes. *Vision Research*, 33(7):981–991, 1993. 88

[207] James H Elder and Steven W Zucker. Evidence for boundary-specific grouping. *Vision Research*, 38(1):143–152, 1998. 88

[208] Franco Modigliani and Leah Modigliani. Risk-adjusted performance. *Journal of Portfolio Management*, 23(2):45–54, 1997. 88

[209] Lars Brink and Bruce McCarl. The tradeoff between expected return and risk among cornbelt farmers. *American Journal of Agricultural Economics*, 60(2): 259–263, 1978. 88

[210] Amit Goyal and Pedro Santa-Clara. Idiosyncratic risk matters! *The Journal of Finance*, 58(3):975–1007, 2003. 88

[211] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2009. 89

[212] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, 2021. 89

[213] Jun Tu and Guofu Zhou. Markowitz meets talmud: A combination of sophisticated and naive diversification strategies. *Journal of Financial Economics*, 99(1):204–215, 2011. 89

[214] Jack Parker-Holder, Aldo Pacchiano, Krzysztof M Choromanski, and Stephen J Roberts. Effective diversity in population based reinforcement learning. *NeurIPS*, pages 18050–18062, 2020. 89

[215] Chien-Chiang Lee, Jun-De Lee, and Chi-Chuan Lee. Stock prices and the efficient market hypothesis: Evidence from a panel stationary test with structural breaks. *Japan and the World Economy*, 22(1):49–58, 2010. 89

[216] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *KDD*, pages 1135–1144, 2016. 89, 90

[217] Umang Bhatt, Alice Xiang, Shubham Sharma, Adrian Weller, Ankur Taly, Yunhan Jia, Joydeep Ghosh, Ruchir Puri, José MF Moura, and Peter Eckersley. Explainable machine learning in deployment. In *FAT*, pages 648–657, 2020. 89

[218] Fábio Pinto, Marco OP Sampaio, and Pedro Bizarro. Automatic model monitoring for data streams. *arXiv preprint arXiv:1908.04240*, 2019. 89

[219] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. Discovering symbolic policies with deep reinforcement learning. In *ICML*, pages 5979–5989, 2021. 90

[220] Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *AISTATS*, pages 1855–1865, 2020. 90

[221] Wenjie Shi, Gao Huang, Shiji Song, Zhuoyuan Wang, Tingyu Lin, and Cheng Wu. Self-supervised discovering of interpretable features for reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(5):2712–2724, 2020. 90

[222] Piyush Gupta, Nikaash Puri, Sukriti Verma, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: understanding agent actions using specific and relevant feature attribution. In *ICLR*, 2020. 90

[223] Chirag Agarwal, Eshika Saxena, Satyapriya Krishna, Martin Pawelczyk, Nari Johnson, Isha Puri, Marinka Zitnik, and Himabindu Lakkaraju. Openxai: Towards a transparent evaluation of model explanations. *NeurIPS*, pages 15784–15799, 2022. 90

[224] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *NeurIPS*, 2017. 90

[225] Julien Pénasse. Understanding alpha decay. *Management Science*, 2022. 91

[226] Robert J Shiller. *Market volatility*. MIT press, 1992. 91, 94

[227] Malik Magdon-Ismail and Amir F Atiya. Maximum drawdown. *Risk Magazine*, 17(10):99–102, 2004. 91, 94

[228] Terje Aven. On the meaning of a black swan in a risk context. *Safety Science*, 57:44–51, 2013. 91

[229] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11), 2008. 91, 94, 109

[230] Fazlollah M Reza. *An introduction to information theory*. 1994. 91

[231] Ulrich Kirchner and Caroline Zunckel. Measuring portfolio diversification. *arXiv preprint arXiv:1102.4722*, 2011. 91, 95

[232] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585 (7825), 2020. 91

[233] Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, 2002. 91, 109

[234] John Y Campbell, Andrew W Lo, A Craig MacKinlay, and Robert F Whitelaw. The econometrics of financial markets. *Macroeconomic Dynamics*, 2(4):559–562, 1998. 94

[235] Emmanuel N Emenyonu and Sidney J Gray. International accounting harmonization and the major developed stock market countries: an empirical study. *The International Journal of Accounting*, 31(3):269–279, 1996. 94

[236] Giorgio De Santis et al. Stock returns and volatility in emerging financial markets. *Journal of International Money and Finance*, 16(4):561–579, 1997. 94

[237] William F Sharpe. The sharpe ratio. *Journal of Portfolio Management*, pages 169–185, 1998. 94, 109

[238] Lou Jost. Entropy and diversity. *Oikos*, 113(2):363–375, 2006. 94

[239] B Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979. 96

[240] Chi-Chuan Lee, Chien-Chiang Lee, and Yizhong Wu. The impact of covid-19 pandemic on hospitality stock returns in china. *International Journal of Finance & Economics*, 2021. 97

[241] Mieszko Mazur, Man Dang, and Miguel Vega. Covid-19 and the march 2020 stock market crash. evidence from s&p1500. *Finance Research Letters*, 38, 2021. 97

[242] Velayoudoum Marimoutou, Bechir Raggad, and Abdelwahed Trabelsi. Extreme value theory and value at risk: application to oil market. *Energy Economics*, 31(4):519–530, 2009. 98, 113

[243] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1), 2022. 98

[244] Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research: A report from the neurips 2019 reproducibility program. *Journal of Machine Learning Research*, 22, 2021. 98

[245] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021. 98, 106

[246] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *FAT*, pages 220–229, 2019. 98

[247] Feiyang Pan, Tongzhe Zhang, Ling Luo, Jia He, and Shuoling Liu. Learn continuously, act discretely: Hybrid action-space reinforcement learning for optimal execution. In *IJCAI*, pages 3912–3918, 2022. 101, 105

[248] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. Csdi: Conditional score-based diffusion models for probabilistic time series imputation. *NeurIPS*, pages 24804–24816, 2021. 101, 107

[249] Can Cui, Wei Wang, Meihui Zhang, Gang Chen, Zhaojing Luo, and Beng Chin Ooi. Alphaevolve: A learning framework to discover novel alphas in quantitative investment. In *SIGMOD*, pages 2208–2216, 2021. 101, 102

[250] Tianping Zhang, Yuanqi Li, Yifei Jin, and Jian Li. Autoalpha: an efficient hierarchical evolutionary algorithm for mining alpha factors in quantitative investment. *arXiv preprint arXiv:2002.08245*, 2020. 101

[251] Zijian Shi and John Cartlidge. State dependent parallel neural hawkes process for limit order book event stream prediction and simulation. In *KDD*, pages 1607–1615, 2022. 101

[252] Andrea Coletta, Aymeric Moulin, Svitlana Vyetrenko, and Tucker Balch. Learning to simulate realistic limit order book markets from data as a world agent. In *ICAIF*, pages 428–436, 2022. 101

[253] Daniel Jarrett, Jinsung Yoon, Ioana Bica, Zhaozhi Qian, Ari Ercole, and Mihaela van der Schaar. Clairvoyance: A pipeline toolkit for medical time series. In *ICLR*, 2021. 101, 106, 118

[254] Marco Pagano. Financial markets and growth: an overview. *European Economic Review*, 37(2-3):613–622, 1993. 102

[255] Mariam Kiran and Melis Ozyildirim. Hyperparameter tuning for deep reinforcement learning applications. *arXiv preprint arXiv:2201.11182*, 2022. 102

[256] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *ICML*, pages 5637–5664, 2021. 102

[257] Shuo Sun, Molei Qin, Wentao Zhang, Haochong Xia, Chuqiao Zong, Jie Ying, Yonggang Xie, Lingxuan Zhao, Xinrun Wang, and Bo An. Trademaster: A holistic quantitative trading platform empowered by reinforcement learning. In *NeurIPS*, pages 59047–59061, 2023. 103

[258] Rong-Jun Qin, Xingyuan Zhang, Songyi Gao, Xiong-Hui Chen, Zewen Li, Weinan Zhang, and Yang Yu. Neorl: A near real-world benchmark for offline reinforcement learning. *NeurIPS*, pages 24753–24765, 2022. 105, 107

[259] Mario Hermann, Tobias Pentek, and Boris Otto. Design principles for industrie 4.0 scenarios. In *HICSS*, pages 3928–3937, 2016. 106

[260] Julian Ereth. Dataops-towards a definition. *LWDA*, 2191:104–112, 2018. 106

[261] Harvinder Atwal. *Practical DataOps: Delivering agile data science at scale*. Springer, 2019. 106

[262] Stuart M Turnbull. Swaps: A zero sum game? *Financial Management*, pages 15–21, 1987. 108

[263] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. In *ICML*, pages 3053–3062, 2018. 109

[264] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018. 109

[265] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 109

[266] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 30, 2016. 109

[267] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *NeurIPS*, 1999. 109

[268] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *ICML*, pages 1587–1596, 2018. 109

[269] Tianping Zhang, Zheyu Zhang, Zhiyuan Fan, Haoyan Luo, Fengyuan Liu, Wei Cao, and Jian Li. Openfe: Automated feature generation beyond expert-level performance. *ICML*, 2023. 113, 116

[270] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019. 116

[271] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *KDD*, pages 2623–2631, 2019. 116

[272] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Knowledge-driven event embedding for stock prediction. In *COLING*, pages 2133–2142, 2016. 121

[273] Edoardo Vittori, Martino Bernasconi de Luca, Francesco Trovò, and Marcello Restelli. Dealing with transaction costs in portfolio optimization: Online gradient descent with momentum. In *ICAIF*, pages 1–8, 2020. 122

[274] David Byrd, Maria Hybinette, and Tucker Hybinette Balch. Abides: Towards high-fidelity market simulation for ai research. *arXiv preprint arXiv:1904.12066*, 2019. 122

[275] Svitlana Vyetrenko, David Byrd, Nick Petosa, Mahmoud Mahfouz, Danial Dervovic, Manuela Veloso, and Tucker Hybinette Balch. Get real: Realism metrics for robust limit order book market simulations. *arXiv preprint arXiv:1912.04941*, 2019. 122