DeepScalper: A Risk-Aware Reinforcement Learning Framework to Capture Fleeting Intraday Trading Opportunities

Shuo Sun
Nanyang Technological University

Wanqi Xue
Nanyang Technological University

Rundong Wang*
Nanyang Technological University

Xu He
Huawei Noah Ark Lab

Junlei Zhu
Webank

Jian Li
Tsinghua University

Bo An
Nanyang Technological University

ABSTRACT

Reinforcement learning (RL) techniques have shown great success in many challenging quantitative trading tasks, such as portfolio management and algorithmic trading. Especially, intraday trading is one of the most profitable and risky tasks because of the intraday behaviors of the financial market that reflect billions of rapidly fluctuating capitals. However, a vast majority of existing RL methods focus on the relatively low frequency trading scenarios (e.g., day-level) and fail to capture the fleeting intraday investment opportunities due to two major challenges: 1) how to effectively train profitable RL agents for intraday investment decision-making, which involves high-dimensional fine-grained action space; 2) how to learn meaningful multi-modality market representation to understand the intraday behaviors of the financial market at tick-level.

Motivated by the efficient workflow of professional human intraday traders, we propose DeepScalper, a deep reinforcement learning framework for intraday trading to tackle the above challenges. Specifically, DeepScalper includes four components: 1) a dueling Q-network with action branching to deal with the large action space of intraday trading for efficient RL optimization; 2) a novel reward function with a hindsight bonus to encourage RL agents making trading decisions with a long-term horizon of the entire trading day; 3) an encoder-decoder architecture to learn multi-modality temporal market embedding, which incorporates both macro-level and micro-level market information; 4) a risk-aware auxiliary task to maintain a striking balance between maximizing profit and minimizing risk. Through extensive experiments on real-world market data spanning over three years on six financial futures (2 stock index and 4 treasury bond), we demonstrate that DeepScalper significantly outperforms many state-of-the-art baselines in terms of four financial criteria. Furthermore, we conduct a series of exploratory and ablative studies to analyze the contributions of each component in DeepScalper.

KEYWORDS

Quantitative investment; reinforcement learning

1 INTRODUCTION

The financial market, an ecosystem that involves over 90 trillion[1] market capitals globally in 2020, attracts the attention of hundreds of millions of investors to pursue desirable financial assets and achieve investment goals. Recent years have witnessed significant development of quantitative trading [1], due to its instant and accurate order execution, and capability of analyzing and processing large amount of temporal financial data. Especially, intraday trading[2], where traders actively long/short 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).

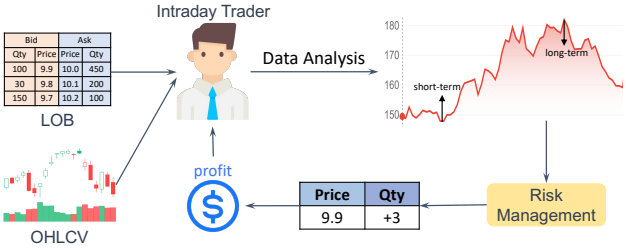


Figure 1: Workflow of professional human intraday trader

Traditional intraday trading strategies use finance knowledge to discover trading opportunities with heuristic rules. For instance, momentum [24] 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 [4] focusing on the investment opportunities at the turning points. However, rule-based traditional methods exhibit poor generalization ability and only perform well in certain market conditions [7]. Another paradigm is to trade based on financial prediction. Many advanced machine learning models such as GCN [13], Transformer [8] and LGBM [19] have been introduced for predicting future prices [9]. Many other data sources such as economic news [15], frequency of price [43], social media [38] and investment behaviors [5] have been added as additional information to further improve prediction performance. However, the high volatility and noisy nature of the financial market make it extremely difficult to accurately predict future prices [10]. Furthermore, there is a noticeable gap between prediction signals and profitable trading actions [13]. Thus, the overall performance of prediction-based methods is not satisfying as well.

*Corresponding author.
[1]https://data.worldbank.org/indicator/CM.MKT.LCAP.CD/
[2]https://www.investopedia.com/articles/trading/05/011705.asp

Similar to other application scenarios of reinforcement learning (RL), quantitative trading also interacts with the environment (financial market) and maximizes the accumulative profit. Recently, RL 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 [21]. Although there have been many successful RL-based quantitative trading methods [11, 35, 39], 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, micro-level market and risk into consideration.

Considering the workflow of a professional human intraday trader (Figure 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 it, 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 an 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.

2 RELATED WORK

2.1 Traditional Finance Methods

Technical analysis [25], which believes that past price and volume data have the ability to reflect future market conditions [10], is the foundation of traditional finance methods. Millions of technical indicators are designed to generate trading signals [18]. Momentum [14] and mean reversion [28] are two well-known types of traditional finance methods based on technical analysis. Momentum trading assumes the trend of financial assets in the past has the tendency to continue in the future. Time Series Momentum [24] and Cross Sectional Momentum [16] are two classic momentum strategies. In contrast, mean reversion strategies such as Bollinger bands [4] assume that the price of financial assets will finally revert to the long-term mean.

However, traditional finance methods are not perceptive enough to capture fleeting intraday patterns and only perform well in certain market conditions [7]. In recent years, many advanced machine learning methods have significantly outperformed traditional finance methods.

2.2 Prediction-Based Methods

As for prediction-based methods, they first formulate quantitative trading as a supervised learning task to predict the future return (regression) or price movement (classification). Later on, trading decisions are generated by the prediction results with a heuristic strategy generator (e.g., top-k in [40]). Specifically, Wang et al. [34] combine attention mechanism with LSTM to model correlated time steps. To improve the robustness of LSTM, Feng et al. [12] apply adversarial training techniques for stock prediction. Zhang et al. [43] propose a novel State Frequency Memory (SFM) recurrent network with Discrete Fourier Transform (DFT) to discover multi-frequency patterns in stock markets. Liu et al. [20] introduce a multi-scale two-way neural network to predict the stock trend. Sun et al. [32] propose an ensemble learning framework to train mixture of trading experts.

However, the high volatility and noisy nature of the financial market make it extremely difficult to accurately predict future prices [10]. Furthermore, there is a noticeable gap between prediction signals and profitable trading actions [13]. Thus, the overall performance of prediction-based methods is not satisfying as well.

2.3 Reinforcement Learning Methods

Recent years have witnessed the successful marriage of reinforcement learning and quantitative trading as RL allows training agents to directly output profitable trading actions with better generalization ability across various market conditions [31]. Neuneier [26] make the first attempt to learn trading strategies using Q-learning. Moody and Saffell [23] propose a policy-based method, namely recurrent reinforcement learning (RRL), for quantitative trading. However, traditional RL approaches have difficulties in selecting market features and learning good policy in large scale scenarios. To tackle these issues, many deep RL approaches have been proposed to learn market embedding through high dimensional data. Jiang et al. [17] use DDPG to dynamically optimize cryptocurrency portfolios. Deng et al. [7] apply fuzzy learning and deep learning to improve financial signal representation. Yu et al. [41] propose a model-based RL framework for daily frequency portfolio trading. Liu et al. [21] propose an adaptive DDPG-based framework with imitation learning. Ye et al. [39] proposed a State-Augmented RL (SARL) framework based on DPG with financial news as additional states.

Although there are many efforts on utilizing RL for quantitative trading, a vast majority of existing RL methods focus on the relatively low-frequency scenarios (e.g., day-level) and fail to capture the fleeting intraday investment opportunities. We propose DeepScalper to fill this gap by mimicking the workflow of human intraday traders.

3 PROBLEM FORMULATION

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

3.1 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. Below are some necessary definitions for understanding the problem:

Definition 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. (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 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 4. (Limit Order Book) As shown in Figure 2, a limit order book (LOB) contains public available aggregate information of limit orders by all market participants. It is widely used as market microstructure [22] 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 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 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 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.

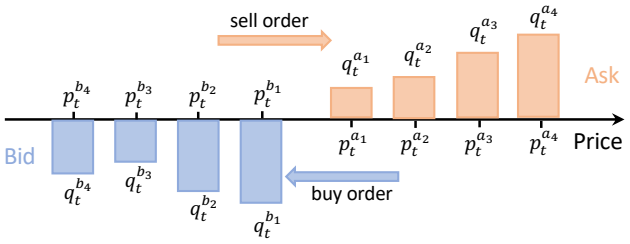


Figure 2: A snapshot of 4-level limit order book (LOB)

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.

3.2 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, $\mathcal{O}, \mathcal{A}, R$ 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 $\mathbf{y_t}$ of 11 technical indicators and the original OHLCV vector $\mathbf{x_t}$ as macro-level state following [40], 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.

Action. Previous works [7, 21] lie in low-frequency trading scenarios, which generally stipulate that the agent trades a fixed

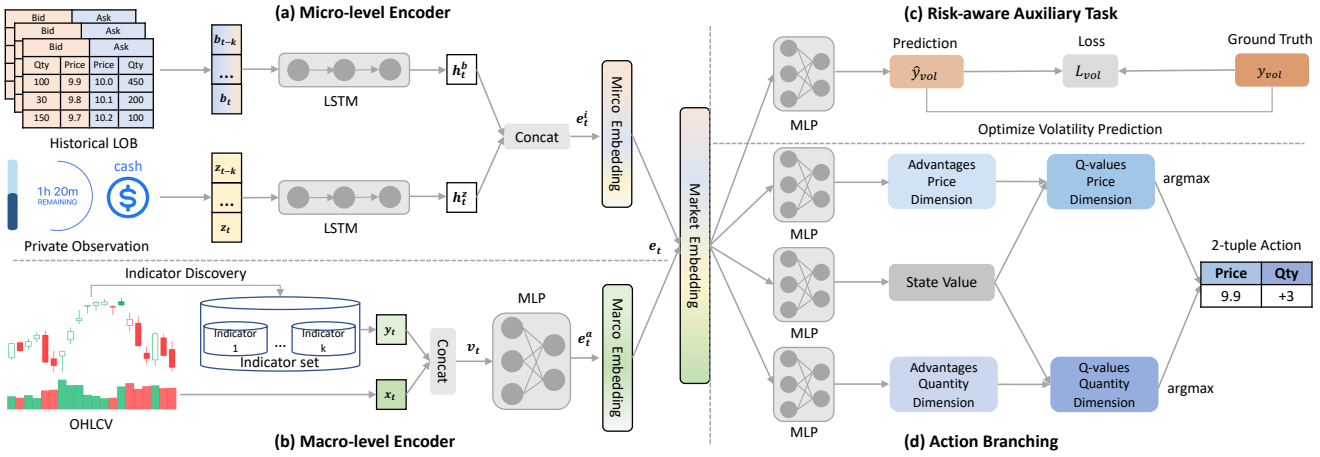


Figure 3: An overview of the proposed DeepScalper framework. We show four individual building blocks: (a) micro-level encoder, (b) macro-level encoder, (c) risk-aware auxiliary task, and (d) RL optimization with action branching.

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 the real-world financial market, 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 P&L, which shows the value fluctuation (profit & loss) of the account:

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

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 DEEPSCALPER

In this section, we introduce DeepScalper (an overview in Figure 3) for intraday trading. We describe the four components of DeepScalper: 1) RL optimization with action branching; 2) reward function with a hindsight bonus; 3) intraday market embedding; 4) risk-aware auxiliary task orderly.

4.1 RL Optimization with Action Branching

Comparing to conventional low-frequency trading 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 [3, 42]. 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) [33] for decision-making. Particularly, as shown in Figure 3(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 [37] $Adv_d(s, a_d)$ 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:

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

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.

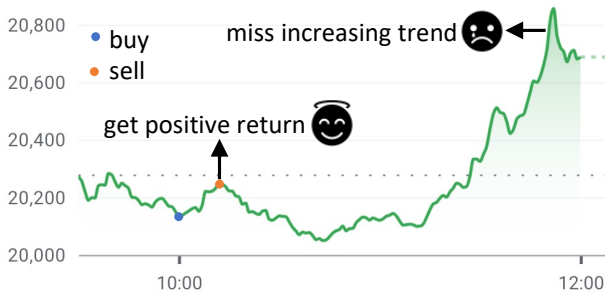


Figure 4: Illustration of the motivation of the hindsight bonus

4.2 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 [36]. 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 of the hindsight bonus, considering a pair of buy/sell actions in Figure 4, 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 somehow 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 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 [27]. 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 3(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 3(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.

4.4 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 3(c). Volatility is widely used as a coherent measure of risk that describing the statistical dispersion of returns in finance [2]. 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

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

Features	Calculation Formula
$z_{open}, z_{high}, z_{low}$	e.g., $z_{open} = open_t/close_t - 1$
z_{close}, z_{adj_close}	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: Features to describe macro-level financial markets

$\mathbf{r}$ 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.

$$\hat{y_{vol}} = MLP(e_t, \theta_v)$$

$$L_{vol}(\theta_v) = (y_{vol} - \hat{y_{vol}})^2$$

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.

5 EXPERIMENT SETUP

5.1 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[3]. We summarize the statistics of the two datasets in Table 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 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 Table 2 following [40]. z_{open}, z_{high} and z_{low} represent the relative values of the open, high, and low prices compared to the close price at the current time step, respectively. z_{close} and z_{adj_close} represent the relative values of the closing and adjusted closing prices compared to the 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. 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 [35].

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

5.2 Evaluation Metrics

We evaluate DeepScalper on four different financial metrics, including one profit criterion and three risk-adjusted profit criteria:

- **Total Return (TR)** is the overall return rate for the entire 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.
- **Sharpe Ratio (SR)** [30] considers the amount of extra return that a trader receives per unit of increased risk. It is defined as: $SR = \mathbb{E}[\mathbf{r}]/\sigma[\mathbf{r}]$, where $\mathbf{r}$ denotes the historical sequence of the return rate.
- **Calmar Ratio (CR)** is defined as $CR = \frac{\mathbb{E}[\mathbf{r}]}{MDD}$. It is calculated as the expected return divided by the maximum drawdown (MDD) of the entire trading period, where MDD measures the largest loss from any peak to show the worst case.
- **Sortino Ratio (SoR)** applies the downside deviation (DD) as the risk measure. It is defined as: $SoR = \frac{\mathbb{E}[\mathbf{r}]}{DD}$, where DD is the variance of the negative return.

5.3 Baseline

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

Traditional Finance 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)** [28] 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)** [24] 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.

Prediction Based Methods

- **MLP** [29] use the classic multi-layer perceptron for future return prediction. We apply a three-layer MLP with hidden size 128.
- **GRU** [6] 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** [19] is an efficient implementation of the gradient boosting decision tree with gradient-based one-side sampling and exclusive feature bundling.

Reinforcement Learning Methods

- **DQN** [44] 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.

5.4 Preprocessing and Experiment Setup

For macro-level features, we directly calculate the 11 technical indicators following the formulas in Table 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[4]. 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 [27], 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[5] [6].

6 RESULTS AND ANALYSIS

6.1 Profitability Comparison with Baselines

We compare DeepScalper with 9 state-of-the-art baselines in terms of four financial metrics in Table 3. 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 5. 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.

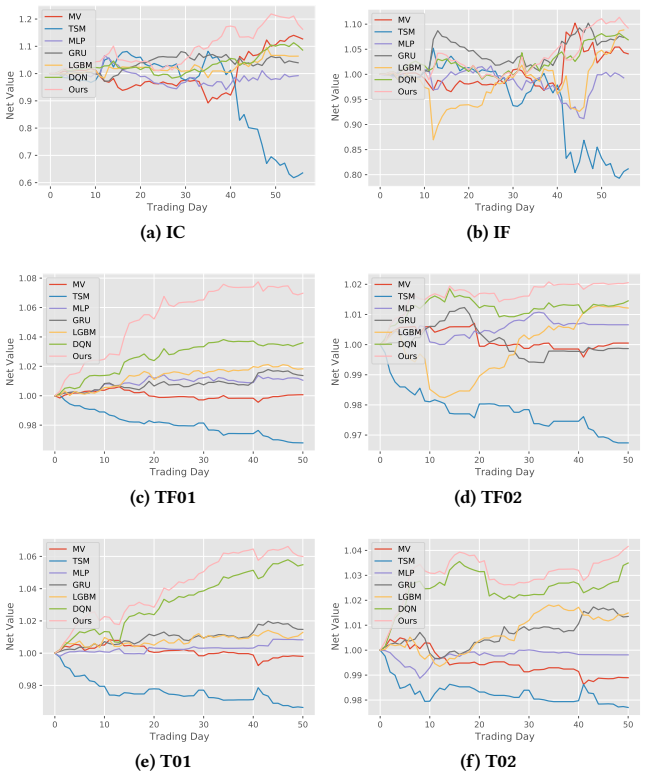


Figure 5: Trading day vs. net value curve of different baselines and DeepScalper on stock index (IC and IF) and treasury bond (TF01, TF02, T01, T02) datasets. DeepScalper achieves the highest profit in all six financial assets.

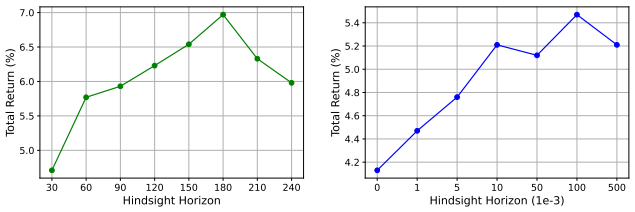
[4]China Financial Futures Exchange: http://www.cffex.com.cn/en_new/index.html
[5]Qlib: https://github.com/microsoft/qlib
[6]FinRL: https://github.com/AI4Finance-Foundation/FinRL

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

Table 3: 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.

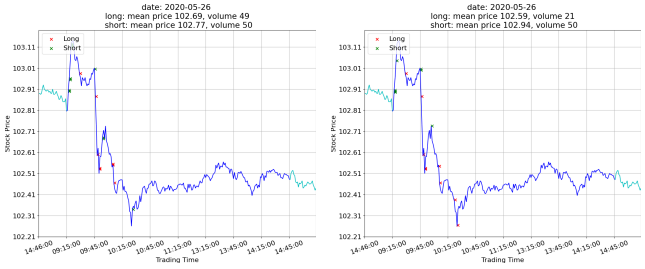
Macro	Micro	Hindsight	Volatility	TR(%)↑	SR↑
√				3.45	4.42
	√			3.47	4.43
√	√			3.62 (+0.15)	4.81 (+0.38)
√	√		√	4.05 (+0.58)	5.03 (+0.60)
√	√	√		5.36 (+1.89)	5.72 (+1.29)
√	√	√	√	**6.97** (+3.50)	**6.10** (+1.67)

Table 4: Ablation studies over different DeepScalper components. √ indicates adding the component to DeepScalper.



(a) TF02 **(b) T02**

Figure 6: Hyperparameter sensitivity of hindsight bonus: (a) effect of importance (b) effect of horizon



(a) May 26 (DS-NH) **(b) May 26 (DS)**

Figure 7: Trading behavior comparison of DS-NH and DS to show the effectiveness of the hindsight bonus

6.2 Model Component Ablation Study

We conduct comprehensive ablation studies on DeepScalper's investment profitability benefits from each of its components in Table 4. 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: 1) each individual component in DeepScalper is effective; 2) these components are largely orthogonal and can be fruitfully integrated to further improve performance.

6.3 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 6a, 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 6b 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 7. 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

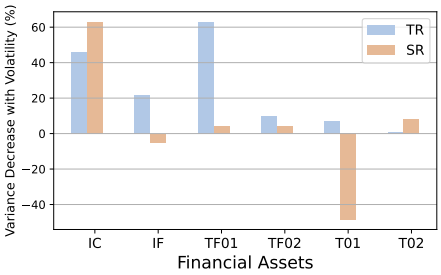


Figure 8: Effect of the auxiliary task on performance variance (>0 means RL agents trained with the risk-aware auxiliary task get lower standard deviation)

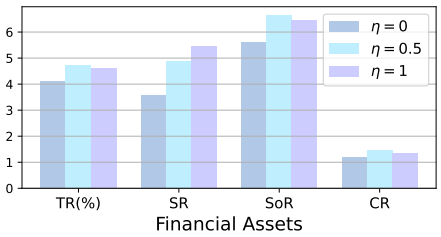


Figure 9: Sensitivity to relative importance η in terms of TR, SR, SoR and CR

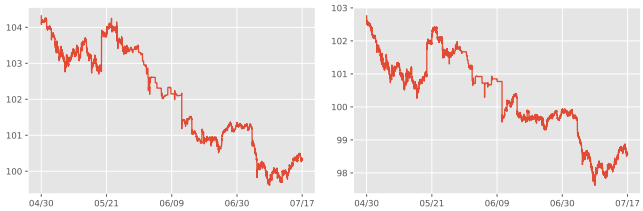
bonus (Figure 7a) performs well in capturing local trading opportunities and overlooks the long-term trend of the entire trading day. In comparison, an agent trained with the hindsight bonus (Figure 7b) trades a large volume of short actions at the beginning of the trading day, indicating that it is 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.

6.4 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 8. 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 9.

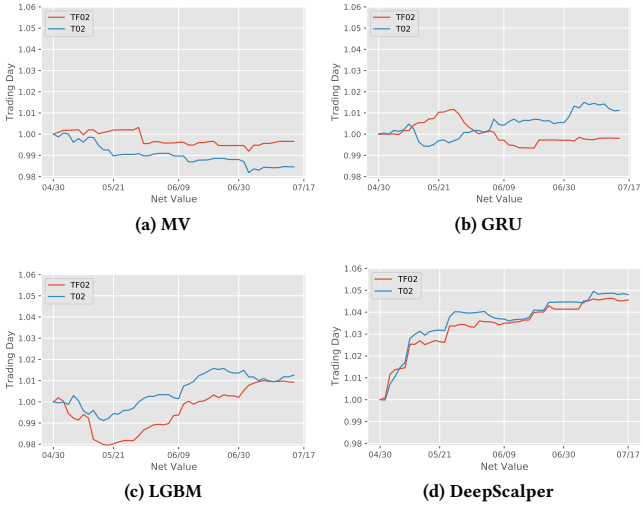
6.5 Generalization Ability

We further test the generalization ability of our framework among different financial futures (TF02 and T02). In Figure 10, 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 DeepScalper in Figure 11. 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 11 (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.



(a) TF02 (b) T02

Figure 10: Price curve of TF02 and T02



(a) MV (b) GRU

(c) LGBM (d) DeepScalper

Figure 11: Net value curve of MV, GRU, LGBM and DeepScalper on TF02 and T02 to show the generalization ability.

7 CONCLUSION

In this article, 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.

REFERENCES

[1] Bo An, Shuo Sun, and Rundong Wang. 2022. Deep Reinforcement Learning for Quantitative Trading: Challenges and Opportunities. *IEEE Intelligent Systems* 37, 2 (2022), 23–26.

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

[3] Marc G Bellemare, Will Dabney, and Rémi Munos. 2017. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*. 449–458.

[4] John Bollinger. 2002. *Bollinger on Bollinger Bands*. McGraw-Hill New York.

[5] Chi Chen, Li Zhao, Jiang Bian, Chunxiao Xing, and Tie-Yan Liu. 2019. Investment behaviors can tell what inside: Exploring stock intrinsic properties for stock trend prediction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2376–2384.

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

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

[8] Qianggang Ding, Sifan Wu, Hao Sun, Jiadong Guo, and Jian Guo. 2020. Hierarchical multi-scale Gaussian transformer for stock movement prediction.. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*. 4640–4646.

[9] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence*. 2327–2333.

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

[11] Yuchen Fang, Kan Ren, Weiqing Liu, Dong Zhou, Weinan Zhang, Jiang Bian, Yong Yu, and Tie-Yan Liu. 2021. Universal trading for order execution with oracle policy distillation. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*. 107–115.

[12] Fuli Feng, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, and Tat-Seng Chua. 2018. Enhancing stock movement prediction with adversarial training. *arXiv preprint arXiv:1810.09936* (2018).

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

[14] Harrison Hong and Jeremy C Stein. 1999. A unified theory of underreaction, momentum trading, and overreaction in asset markets. *The Journal of Finance* 54, 6 (1999), 2143–2184.

[15] Ziniu Hu, Weiqing Liu, Jiang Bian, Xuanzhe Liu, and Tie-Yan Liu. 2018. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 261–269.

[16] Narasimhan Jegadeesh and Sheridan Titman. 2002. Cross-sectional and time-series determinants of momentum returns. *The Review of Financial Studies* 15, 1 (2002), 143–157.

[17] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. 2017. A deep reinforcement learning framework for the financial portfolio management problem. *arXiv preprint arXiv:1706.10059* (2017).

[18] Zura Kakushadze. 2016. 101 formulaic alphas. *Wilmott* 2016, 84 (2016), 72–81.

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

[20] Guang Liu, Yuzhao Mao, Qi Sun, Hailong Huang, Weiguo Gao, Xuan Li, Jianping Shen, Ruifan Li, and Xiaojie Wang. 2020. Multi-scale two-way deep neural network for stock trend prediction.. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI)*. 4555–4561.

[21] Yang Liu, Qi Liu, Hongke Zhao, Zhen Pan, and Chuanren Liu. 2020. Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proceedings of 35th the AAAI Conference on Artificial Intelligence*. 2128–2135.

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

[23] John E Moody and Matthew Saffell. 1999. Reinforcement learning for trading. (1999), 917–923.

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

[25] John J Murphy. 1999. *Technical Analysis of The Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. Penguin.

[26] Ralph Neuneier. 1996. Optimal asset allocation using adaptive dynamic programming. In *Proceedings of the 10th Neural Information Processing Systems*. 952–958.

[27] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. 2006. Reinforcement learning for optimized trade execution. In *Proceedings of the 23rd International Conference on Machine learning*. 673–680.

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

[29] Karsten Schierholt and Cihan H Dagli. 1996. Stock market prediction using different neural network classification architectures. In *IEEE/IAFE 1996 Conference on Computational Intelligence for Financial Engineering*. 72–78.

[30] William F Sharpe. 1994. The sharpe ratio. *Journal of Portfolio Management* 21, 1 (1994), 49–58.

[31] Shuo Sun, Rundong Wang, and Bo An. 2021. Reinforcement learning for quantitative trading. *arXiv preprint arXiv:2109.13851* (2021).

[32] Shuo Sun, Rundong Wang, and Bo An. 2022. Quantitative Stock Investment by Routing Uncertainty-Aware Trading Experts: A Multi-Task Learning Approach. *arXiv preprint arXiv:2207.07578* (2022).

[33] Arash Tavakoli, Fabio Pardo, and Petar Kormushev. 2018. Action branching architectures for deep reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*. 4131–4138.

[34] Jia Wang, Tong Sun, Benyuan Liu, Yu Cao, and Hongwei Zhu. 2019. CLVSA: A convolutional LSTM based variational sequence-to-sequence model with attention for predicting trends of financial markets. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*. 3705–3711.

[35] Rundong Wang, Hongxin Wei, Bo An, Zhouyan Feng, and Jun Yao. 2021. Commission fee is not enough: A hierarchical reinforced framework for portfolio management. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

[36] Zhicheng Wang, Biwei Huang, Shikui Tu, Kun Zhang, and Lei Xu. 2021. DeepTrader: A deep reinforcement learning approach to risk-return balanced portfolio management with market conditions embedding. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*.

[37] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *Proceedings of 35th International Conference on Machine Learning*. 1995–2003.

[38] Yumo Xu and Shay B Cohen. 2018. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 1970–1979.

[39] Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Ju Xiao, and Bo Li. 2020. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence (AAAI)*. 1112–1119.

[40] Jaemin Yoo, Yejun Soun, Yong-chan Park, and U Kang. 2021. Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2037–2045.

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

[42] Tom Zahavy, Matan Haroush, Nadav Merlis, Daniel J Mankowitz, and Shie Mannor. 2018. Learn what not to learn: Action elimination with deep reinforcement learning. *Advances in Neural Information Processing Systems* (2018).

[43] Liheng Zhang, Charu Aggarwal, and Guo-Jun Qi. 2017. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2141–2149.

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