MetaTrader: An Reinforcement Learning Approach Integrating Diverse Policies for Portfolio Optimization

Hui Niu
niuh17@mails.tsinghua.edu.cn
Tsinghua University
Beijing, China

Siyuan Li
lisiyuan199511@gmail.com
Harbin Institute of Technology
Harbin, China

Jian Li
lapordge@gmail.com
Tsinghua University
Beijing, China

ABSTRACT
Portfolio management is a fundamental problem in finance. It involves periodic reallocations of assets to maximize the expected returns within an appropriate level of risk exposure. Deep reinforcement learning (RL) has been considered a promising approach to solving this problem owing to its strong capability in sequential decision making. However, due to the non-stationary nature of financial markets, applying RL techniques to portfolio optimization remains a challenging problem. Extracting trading knowledge from various expert strategies could be helpful for agents to accommodate the changing markets. In this paper, we propose MetaTrader, a novel two-stage RL-based approach for portfolio management, which learns to integrate diverse trading policies to adapt to various market conditions. In the first stage, MetaTrader incorporates an imitation learning objective into the reinforcement learning framework. Through imitating different expert demonstrations, MetaTrader acquires a set of trading policies with great diversity. In the second stage, MetaTrader learns a meta-policy to recognize the market conditions and decide on the most proper learned policy to follow. We evaluate the proposed approach on three real-world index datasets and compare it to state-of-the-art baselines. The empirical results demonstrate that MetaTrader significantly outperforms those baselines in balancing profits and risks. Furthermore, thorough ablation studies validate the effectiveness of the components in the proposed approach.

CCS CONCEPTS
• Computing methodologies → Reinforcement learning; • Applied computing → Economics.

KEYWORDS
Portfolio Management, Deep Reinforcement Learning, Imitation Learning, Meta-Policy Learning

ACM Reference Format:
Hui Niu, Siyuan Li, and Jian Li. 2022. MetaTrader: An Reinforcement Learning Approach Integrating Diverse Policies for Portfolio Optimization. In

1 INTRODUCTION
Portfolio management has long been an important and challenging problem in quantitative trading [10, 22–24], which aims to maximize the expected returns with acceptable risks through reallocation of portfolio weights. Generally, portfolio management is a sequential decision making problem.

The Markowitz model [24], also called the mean-variance model, is the first theoretical work to formally investigate portfolio optimization. This model formulates portfolio management as a single-period optimization problem, and is then generalized to multiperiod optimization in several works [23, 29, 38]. However, the mathematical solutions for those models require explicit models of both temporal dynamics of individual assets and their co-movements, which are difficult to acquire in practice [10]. Some empirical work devotes to solving the portfolio management problem based on typically observed patterns, including momentum investing [18], mean reversion [17], and multi-factor models [7]. Nevertheless, these traditional investment strategies could hardly adapt to the changing markets since they only perform well in certain cases.

More recently, a variety of machine learning methods are developed to address the portfolio management problem. In particular, deep neural networks such as graph models [4, 35, 36] and recurrent networks [14, 43] are widely employed to extract temporal and spatial relationships of asset data, benefitting from their strong abilities in representation learning. Several supervised learning methods are proposed to predict price movements based on various kinds of financial information [13, 15, 33, 49]. Nevertheless, these supervised learning algorithms require data with a set of labels that are associated with certain tasks (e.g. classification and regression). The design of labels for financial data is nontrivial [5], especially in the context of portfolio management. Moreover, as the portfolio weights are generated indirectly, these supervised learning methods suffer from fragile hyper-parameter tuning.

Efforts have also been made to adopt deep reinforcement learning (RL) techniques to address the portfolio management problem [6, 16, 19, 37, 44, 46]. Compared to supervised learning, RL is considered a more flexible framework for portfolio management since it takes advantage of various reward functions to achieve better risk-return balancing. Despite promising results achieved by the previous RL-based methods, applying RL to real-world portfolio management still faces challenges. The non-stationary nature of the financial markets induces a challenging exploration problem for RL algorithms: Agents could hardly fully explore the fluctuate trading
environments through maximizing cumulative rewards without additional knowledge, which further influences the performance of the learned policy.

To overcome this challenge, we propose MetaTrader, a novel RL-based portfolio management approach incorporating with imitation learning techniques. The proposed approach is inspired by the following facts: (1) Beyond learning from interactions with the environment, extracting trading knowledge from seasoned experts is also an appealing way to mine profitable patterns in quantitative trading [6]. (2) Leveraging expert data is a favorable approach to improve the exploration ability of RL algorithms [2, 25, 30, 40]. (3) Financial companies employ multiple portfolio managers to diversify risks, and those managers are adept at dealing with different market conditions [21]. To summarize, the key insight of MetaTrader is that utilizing knowledge from various trading strategies promotes the agent’s adaptability to the changing markets.

Motivated by the above intuitions, MetaTrader decomposes the portfolio optimization process into two phases. The first is a diverse policy learning phase, which aims to obtain a set of base policies capable of dealing with various market conditions. To achieve this goal and promote efficient exploration of the complex trading environment, we develop a novel learning objective that combines RL framework, and MetaTrader learns to score the base policies from interactions with the environments and through extracting knowledge from expert datasets as well. The second is a meta-policy learning phase, where a meta-policy is learned to recognize the market conditions and make decisions on which learned base policy to follow in each holding period. This phase is modeled with the RL framework, and MetaTrader learns to score the base policies obtained in the first phase while freezing their parameters.

The contribution of this paper can be summarized as follows:

- To the best of our knowledge, MetaTrader is the first attempt to utilize trading knowledge from multiple experts to acquire diverse policies in the portfolio optimization literature.
- MetaTrader learns to select the reasonable base policy to follow by recognizing the market conditions. The scores given by the meta-policy reflect its confidence in the profitability of those base policies, which enhances the interpretability of the learned portfolio management decisions.
- Extensive experiments on three well-known stock indexes show that our portfolio management agent learns to adapt to the changing markets properly and achieves superior performance in profitability and risk-return balancing compared to state-of-the-art baselines. Further ablation studies verify the effectiveness of meta-policy learning.

This paper is organized as follows: In Section 2, we formulate the portfolio management problem as a Markov decision process and introduce the trading procedure. Afterward, we describe the proposed approach MetaTrader in Section 3, including the detailed objective, structure, and workflow. Next, in Section 4, we conduct extensive experiments to illustrate the effectiveness of MetaTrader. Then, in Section 5, we review and discuss the related works in portfolio management. Finally, Section 6 concludes and points out some possible future directions.

2 PROBLEM FORMULATION

In this section, we introduce some notations and describe the problem formulation in this work for portfolio management.

2.1 Problem Setup

A financial portfolio is a collection of assets comprised of stocks, bonds, cash, or other forms of assets. We consider the investment in N risky assets in a market, e.g., stocks1.

Portfolio management is a sequential decision making problem involving periodically reallocating the asset weights. Suppose an investor enters the market at time 0 with an initial wealth ACC0. He/she reallocates the asset weights at time t×Δt for t = 0, 1, ..., T−1, where Δt is a minimum time unit for investment, for example, one day or one month. The t-th holding period lasts from time t×Δt to time (t + 1)×Δt. The starting time of the t-th holding period is called timestep t, and the ending time of it is called timestep t + 1. The accumulated capital at timestep t is denoted by ACCt. Similarly, the subscript t denotes timestep t in the following.

This sequential decision making problem is formulated as a Markov decision process (MDP) M = (S, A, P, R, γ), where S is the state space, A is the 2N-dimensional action space, P : S×A×S → [0, 1] is the state transition function specifying the conditional transition probabilities between states, R : S×A×S → R is the bounded reward function and γ ∈ (0, 1) is the discount factor.

The details of the MDP environment are set as follows:

- State space S: A state st ∈ S consists of three components st = (x1:t, xN:t, w1), which are generated from different data sources. Asset features x1:t describe the information of the assets, including historical asset prices and some classical technical indicators2. Market features xN:t reflect current market conditions using macro indicators such as index trends and market sentiment. Account feature w1 is the portfolio weight at timestep t − 1, x1:t = st−1.
- Action space A: Both long and short operations can be performed in this formulation. An action at ∈ A is the re-allocated portfolio weight at timestep t. Specifically, at = [w1:t+1 − w1:t, ..., wN−1:t+1 − wN−1:t, wN−1:t, wN−1:t, ..., wN−1:t].
- Transition function P: Note that a state consists of three parts: x1:t, xN:t, and w1. Since we assume that the agent’s behaviors could hardly influence the market, the transitions of asset features x1:t and market features xN:t are dominated by the price movements in the market. In contrast, the transition of account feature w1 is affected by actions.

1For ease of explanation, we may refer to assets as stocks in the following. The proposed approach is applicable to other kinds of risky assets as well.
2The details for the classical technical indicators are specified in the experiment section, Section 4.
• Reward function R: We adopt the Differential Sharpe Ratio (DSR) [27] as rewards. Recall that the Sharpe Ratio (SR) is the average return in excess of the risk-free return per unit of volatility. Considering an investment process that contains $T$ holding periods, its Sharpe ratio can be calculated as:

$$SR_T = \left( \frac{1}{T} \sum_{t=1}^{T} RoR_t - RoR^f \right) / \sqrt{V_T},$$  

where $RoR_t$ is the rate of return for the $t$-th holding period:

$$RoR_t = (AC_{t+1} - AC_t) / AC_t;$$

$RoR^f$ is the risk-free return rate per period; and $V_t$ is the volatility:

$$V_T = \sqrt{\frac{1}{T} \sum_{t=1}^{T} (RoR_t - \frac{1}{T} \sum_{t=1}^{T} RoR_t)^2}. $$

Note that directly taking the Sharpe ratio as the reward induces a sparse-reward challenge for RL algorithms since it can only be obtained at the end of the trading process. Therefore, to leverage dense rewards and achieve a balance between profits and risks, we adopt the DSR as rewards. Let $SR_t$ denote the Sharpe ratio at timestep $t$. The differential Sharpe ratio $DSR_t$ is then obtained by expanding $SR_t$ with decay rate $\eta$, and considering the moving average of the return rates and the standard deviation of return rates in equation (1),

$$DSR_t := \frac{dSR_t}{d\eta} = \beta_{t-1} \Delta a_t - \frac{1}{2} \alpha_{t-1} \Delta \beta_t \left( \beta_{t-1} - \frac{1}{2} \right),$$

where $a_t$ and $\beta_t$ are exponential moving estimates of the first and second moments of $R_t$:

$$a_t = a_{t-1} + \eta \Delta a_t, \quad a_{t-1} = a_{t-1} + \eta (R_t - a_{t-1}),$$

$$\beta_t = \beta_{t-1} + \eta \Delta \beta_t, \quad \beta_{t-1} = \beta_{t-1} + \eta (R_t^2 - \beta_{t-1}).$$

With attractive properties such as weights recent returns more and facilitating recursive updating, the DSR is a proven effective reward form as it enables efficient online optimization [27].

In the above MDP environment, after taking action $a_t$ at state $s_t$, a trading agent switches to the next state $s_{t+1}$ according to the transition function $P(s_{t+1}|s_t, a_t)$ and obtains a reward $r_t = R(s_t, a_t, s_{t+1})$. To solve the portfolio management problem, we need to optimize a policy $\pi(a_t|s_t)$ which outputs an action $a_t$ for a given state $s_t$. To balance profits and risks, we adopt the differential Sharpe ratio as rewards. The learning objective is to obtain a policy $\pi(a_t|s_t)$ to maximize the cumulative rewards $\mathbb{E} \left[ \sum_{t=0}^{T} r_t \right]$, where $T \times \tau$ is the ending time of the whole trading process.

### 2.2 Trading Procedure

Consider the investment process on a portfolio that contains both long position and short position. At the end of the $(t-1)$-th holding period, an investor holds $b_{t-1} = \{b_{t-1,1}^+, b_{t-1,2}^+, ..., b_{t-1,n}^+\} \in \mathbb{R}^n$ volume of stocks on a long position. Besides, the investor has borrowed $\mathbf{b}_{t-1}^- \in \mathbb{R}^n$ volume of stocks at timestep $t-1$ and sold them as a short position. Given the close price of assets $P_{close} \in \mathbb{R}^n$ and portfolio vector $\mathbf{w}_t; \mathbf{w}_t^-$, the investor follows the following steps to conduct wealth repositioning: (1) At the end of the $(t-1)$-th holding period, the investor sells all long position ($b_{t-1}^+$) and gets cash; (2) Meanwhile, he/she buys the borrowed stocks ($\mathbf{b}_{t-1}^-$) and returns them to the stock brokerage, and at that moment the wealth is denoted by $AC_t$. (3) At the beginning of the $t$-th period, the investor borrows stocks from a broker according to short proportion $\mathbf{w}_t^-$ and wealth $AC_t$, and sell them immediately to get cash. At that moment the total value of the cash is denoted by $TC_t$; (4) Afterwards, he/she purchases stocks using the cash $TC_t$ according to long proportion $\mathbf{w}_t^+$. 

### 3 APPROACH

Due to the complexity and volatility of trading markets, portfolio management has long been a challenging problem [27, 44, 46]. A portfolio manager who follows a fixed trading strategy all the time can hardly always gain profits in the changing markets [21]. In contrast, timely identifying market conditions and switching to appropriate trading strategies potentially enable more competitive trading decisions. Inspired by this, we propose a novel RL-based
portfolio management approach named MetaTrader, which conducts the learning by two phases, i.e., a diverse policy learning phase and a meta-policy learning phase. In the first phase, MetaTrader learns multiple diverse trading policies via combining a reinforcement learning (RL) objective and an imitation learning objective. In the second phase, MetaTrader learns a meta-policy to select suitable policies to execute from the learned trading policy set, and the meta-policy is conditioned on the current market condition. Figure 1 depicts the learning scheme of those two stages.

In the rest of this section, we provide the details of the proposed approach. Section 3.1 elaborates on the methods of learning multiple policies which act as diversely as possible, and Section 3.2 presents the meta-policy learning, which is in charge of selecting proper policies to execute based on the market conditions.

3.1 Diverse Policy Learning

To address Challenge (1), we develop a novel learning objective, which combines an imitation learning objective that encourages mimicking the behaviors in expert datasets, with an RL objective of maximizing the cumulative rewards. The data flow of the diverse policy learning phase is shown in Figure 1(a). Here we explain the motivation of this proposed learning objective. On the one hand, despite the RL objective could help balance profits and risks, the full exploration in the complex trading environment is challenging for RL algorithms. Furthermore, learning via optimizing a single RL objective could not achieve the goal of generating multiple policies that act diversely. On the other hand, extracting trading knowledge from various expert datasets might help diversify the learned policies.

To acquire diverse and profitable trading policies, MetaTrader integrates the knowledge from a set of expert datasets \( \{D_k | 1 \leq k \leq K \} \) with the policy learned online by RL. The various datasets ensure the diversity of the learned policies, and the RL objective enables the maximization of profits while balancing risks.

Specifically, the loss function for this mixed learning objective is formulated as

\[
L_{\theta_k} = \mathbb{E}_{T \sim \pi} \left[ - \sum_{t=0}^{T} \Psi_t \log \pi(a_t | s_t; \theta_k) + \lambda \mathbb{E}_{(x, a) \sim D_k} [\pi(a_t | s_t; \theta_k) - a_t]^2 \right],
\]

where \( \Psi_t = \sum_{t'=T}^T \gamma^{t'-t} r_{t'} \) is the discounted return, \( r \in \{s_0, a_0, ..., s_{T+1}\} \) is a trajectory rolled out by policy \( \pi(a_t | s_t; \theta_k) \) in the trading environment, \( D_k \) is the \( k \)-th expert dataset and \( \lambda \) is a scaling factor.

3.1.2 Network Structure. To tackle Challenge (2), we provide a well-designed neural network structure to abstract the time-axis and data-axis correlations among the assets. This structure is shown in Figure 2 as a detailed supplement for Figure 1(a).

**Temporal Convolution Block.** As demonstrated in Figure 2, MetaTrader first utilizes a temporal convolution network (TCN) [50] block to extract the time-axis relations in the asset features data \( x^a \). Compared to recurrent neural networks, TCN has several appealing properties including facilitating parallel computation, alleviating the gradient exploration/vanishing issue, and demonstrating longer effective memory [46, 50]. After conducting TCN operations on \( x^a \) along the time axis, we obtain an output tensor denoted by \( \tilde{H} \in \mathbb{R}^{N \times F \times T} \), where \( F \) is the dimension of hidden features, \( N \) is the number of risky assets, and \( T \) is the temporal dimension.

**Spatial Attention Layer.** Afterward, MetaTrader adopts an attention mechanism [42] to handle the spatial relationships among different assets. Given the output vector of TCN, we calculate the spatial attention weight as

\[
\hat{S} = V_s \cdot \text{sigmoid}(\langle HW_s \rangle W_2 (W_3 \tilde{H}^T \rangle)^T + b_s),
\]

Figure 2: The proposed network structure for portfolio generation in the diverse policy learning phase.
where $W_1 \in \mathbb{R}^{T \times T}$, $W_2 \in \mathbb{R}^{F \times T}$, $W_3 \in \mathbb{R}^{F}$, and $V_k \in \mathbb{R}^{N \times N}$ are parameters to learn, $b_k \in \mathbb{R}^{N \times N}$ is the bias vector, and the superscript $f_t^2$ denotes transposing the first two dimensions of $H$. The matrix $S \in \mathbb{R}^{N \times N}$ is then normalized by rows to represent the correlation among stocks:

$$S_{i,j} = \frac{\exp(\hat{S}_{i,j})}{\sum_{i=1}^{N} \exp(\hat{S}_{i,j})}, \forall 1 \leq i \leq N. \quad (6)$$

**Residual Connections.** We adopt the ResNet [11] structure to alleviate the vanishing gradient problem in deep learning. To be precise, the final representation abstracted from $x^a$ is denoted by $H = S \times H + x^a$, and it is then translated to a vector with dim N using a fully connected layer: $\hat{\nu} = W_4 H + b_k$. The representation $\hat{\nu}$ is concatenated with the account feature $x^u$ to obtain a vector $\nu$ with dim N: $\nu = \text{sigmoid}(W_7 \cdot [\hat{\nu}; x^u] + b)$, which implies the price rising potential of the N risky assets.

Next, MetaTrader generates the portfolio weights according to vector $\nu$: Taking the top $M$ elements of $\nu$ and normalizing them with a SoftMax activation function, we obtain the weights $\nu^+$ for long position; Note that $\nu \in [-1, 1]^N$, the weights $\nu^-$ are acquired by a similar procedure using SoftMax activation for $-\nu$, and these elements are multiplied by the total short ratio $\rho$. Let $G^+$ and $G^-$ denote the set of selected stocks for long position and short position respectively. Thus, the portfolio weights can be calculated by

$$w^+_i = \left\{ \begin{array}{ll} \frac{\exp(\nu_i)}{\sum_{j \in G^+} \exp(\nu_j)} & i \in G^+ \\ 0 & i \notin G^+ \end{array} \right., \quad w^-_i = \left\{ \begin{array}{ll} \rho \frac{\exp(-\nu_i)}{\sum_{j \in G^-} \exp(-\nu_j)} & i \in G^- \\ 0 & i \notin G^- \end{array} \right.. \quad (7)$$

As shown in Figure 2, we condition the short ratio $\rho$ on market features $x^m$. Similar to DeepTrader [46] and DA-RNN [33], MetaTrader employs an LSTM neural network [12] and a temporal attention structure to extract representation for market features:

$$h_t = \text{LSTM}(h_{t-1}, x^m_t), 1 \leq t \leq T,$$

$$e_t = \frac{V_e^T \text{tanh}(W_3 [h_t; h_T]) + W_6 x^m_t}{\sum_{j=1}^{T} \exp(e_j)},$$

$$\alpha_t = \frac{\exp(e_t)}{\sum_{j=1}^{T} \exp(e_j)},$$

$$\rho_t = \frac{1}{2} \text{sigmoid}(W_7 \cdot \{\sum_{t=1}^{T} \alpha_t \cdot h_t\} + b_m) + \frac{1}{2}, \quad (8)$$

where $V_e \in \mathbb{R}^F$, $W_3 \in \mathbb{R}^{F \times 2F}$, $W_6 \in \mathbb{R}^{F \times F}$, $W_7 \in \mathbb{R}^F$ are free parameters. $h_t$ denotes the hidden embedding encoded by LSTM at step $t$, $\alpha_t$ is the normalized attention weight output by the temporal attention module, and $\rho_t$ is the short ratio at step $t$. It is worth mentioning that compared to the DeepTrader method, MetaTrader does not rely on a separate portfolio generator module. The short ratio $\rho$ serves as a latent variable in the neural network, which is trained jointly with the portfolio vector $[\nu^+, \nu^-]$ with the learning objective introduced in Section 3.1.1.

### 3.2 Meta-Policy Learning

With the diverse policies learned in Section 3.1, there are generally two methods for policy ensemble: weighted sum of the pretrained policies, or selecting one policy to follow per timestep. We adopt the latter one in this paper, as illustrated in Figure 1(b).

Dynamical policy selection is also a challenging sequential decision making problem. In the second phase, DeepTrader trains a meta-policy to accomplish the automatic policy selection. The meta-policy makes decisions in a meta MDP environment. The meta-policy recognizes the market conditions mainly based on several macro indicators such as the index trend and market sentiment, and the historical performance of base policies. Therefore, the state at timestep $t$ in the meta-MDP is defined by $s_t^m = [x^m_t; x^p_t]$, where the superscript $p$ denotes the past performance of the frozen diverse policies. And the action space for the meta-MDP is the policy identity space $\{1, ..., K\}$. Since the action space of the meta-policy is discrete, we optimize the meta-policy with the deep Q-learning algorithm [26]. For the Q network structure, we adopt an LSTM network and the temporal attention mechanism to extract a latent embedding from the market features $x^m_t$. The latent embedding of $x^m_t$ is concatenated with the past performance vector $x^p_t$. The concatenated embedding is passed through two fully connected layers to output the Q values.

**Algorithm 1 Meta-Policy Learning**

**Input:** A trading policy set $\{\pi_a(s|\theta_k), k = 1,..., K\}$

**Initialize:** Q-function $Q(s^m, k)$, replay buffer $B$

1: for episode $= 1..N_e$ do
2: for $t = 1..T$ do
3: With a probability $\epsilon$, randomly sample $k_t$ from $[1,...,K]$.
4: otherwise $k_t = \arg\max_k Q(s^m_t, k)$
5: $a_t \sim \pi_1(s_t|\theta_{k_t})$
6: Execute the action $a_t$ and observe reward $r_t$ and new states $s^m_{t+1}$ and $s^m_{t+1}$ in B
7: Store transition $(s_t^m, k_t, r_t, s^m_{t+1})$ in B
8: Perform one step of optimization using Eq. (9)
9: end for
10: end for
11: Return: $Q(s^m, k)$

Algorithm 1 demonstrates the learning process of the meta-policy. To explore the diverse policies, MetaTrader randomly selects a policy in the policy set probabilistically (Line 3). As the Q-learning converges, the probability $\epsilon$ of the uniform random exploration decreases. After making decisions using the meta-policy, the portfolio weights given by the selected policy are executed in the simulated trading environment, and the agent receives a reward and transits to the next state (Line 5, 6). The transitions are stored in a replay buffer B (Line 7). Q-values of the meta-policy are updated using the minibatches sampled from $B$ with the following loss function:

$$L_Q = \mathbb{E}_{(s_t,k_t,r_t,s_{t+1})} \left[ (y_j - Q(s_t^m, k_t))^2 \right], \quad (9)$$

where $y_j = r_j + \gamma \max_{k_j} Q(s_{t+1}^m, k_{j+1})$. The target Q-function $Q_{target}$ is synchronized from the optimized Q-function every $C$ steps. As $Q(s^m, k)$ is the expected future cumulative reward of selecting the $k$-th policy at state $s^m$, the learned meta-policy $\pi^m(k|s^m)$ is determined by $\arg\max_k Q(s^m, k)$. The portfolio weights are generated with a joint inference of the Q-network and the selected policy network. In addition, to stabilize the learning process, those diverse trading policies are not finetuned in this learning stage.
4 EXPERIMENTS

In this section, we conduct a series of experiments to evaluate the proposed approach. First, we demonstrate that our approach achieves better risk-return balancing in three real financial markets and substantially outperforms the baseline methods. Next, we dig into the reason for the effectiveness of the proposed method and analyze the learned portfolio management strategy in detail. Finally, we conduct ablation studies and show the importance of the meta-policy learning.

4.1 Experimental Setup

This subsection provides a description about the market data for training and test, the expert datasets for imitation learning, the compared baseline methods and the evaluation metrics.

Financial Data. Aiming at enhancing the indexes as lots of real-world funds, we conduct experiments on the constituent stocks of three well-known stock indexes, including Dow Jones Industrial Average (DJIA) in the U.S. market, Hang Seng Index (HSI) in Hong Kong market, and CSI100 Index in Chinese A-share market. Note that Chinese A-share market does not allow short positions, the indices shown here are widely-used ensemble model of decision trees for classification and regression. The DA-RNN method utilizes an LSTM neural network with temporal attention mechanism for stock price prediction.

Baseline Methods. We compare our method with the three categories of portfolio management methods summarized in Section 5. (1) Traditional investment strategies: Market, CSM and BLSW. Market is a strategy replicating the index. CSM and BLSW are the strategies to generate the expert datasets. (2) Supervised learning approaches: LightGBM [20] and DA-RNN [33]. LightGBM is a widely-used ensemble model of decision trees for classification and regression. The DA-RNN method utilizes an LSTM neural network with temporal attention mechanism for stock price prediction. (3) State-of-the-art RL-based methods: AlphaStock [44] and DeepTrader [46].

Evaluation Metrics. We adopt six evaluation metrics to meet different risk appetites of the investors.

(1) Annualized Rate of Return (ARR) is the annualized average of the return rate of one holding period, calculated as

$$ARR_T = \frac{\sum_{t=1}^{T} R_{i,T}}{T} \times \frac{1}{N_T}$$

where $R_{i,T}$ is the average return rate of a holding period, and $N_T$ is the number of holding periods in a year.

(2) Annualized Variability (AVol) is an annualized average of volatility (Vol) that reflects risks of a strategy

$$AVol_T = \sqrt{\frac{1}{N_T}}$$

(3) Maximum DrawDown (MDD) measures the loss under the worst case during the investment. It can be defined as

$$MDD_T = \max_{1 \leq i < j \leq T} \left( AC_i - AC_j \right) / AC_i$$

where $AC_i$ and $AC_j$ denote the accumulated capital at timestep $i$ and $j$ respectively.

(4) Annualized Sharpe Ratio (ASR) is the risk-adjusted ARR based on Annualized Volatility, calculated as

$$ASR_T = \frac{ARR}{AVol}$$

(5) Calmar Ratio (CR) is the risk-adjusted ARR based on MDD, formulated by

$$CR_T = APR / MDD$$

(6) Sortino Ratio (SoR) is also a risk-adjustment metric, which implies the additional return for each unit of downside risk, defined as

$$SoR = \frac{\sum_{t=1}^{T} \min(\frac{R_{i,t}}{\sigma_i}, 0)}{\sum_{t=1}^{T} \min(\frac{R_{i,t}}{\sigma_i}^2, 0)/T}$$

where $\sigma_i$ is the standard deviation of returns for asset $i$.

Table 1: Training/Test splitting of the datasets

| Index   | # stocks | Training          | Test               |
|---------|----------|-------------------|--------------------|
| DJIA    | 23       | 1987.08-2007.08   | 2007.09-2018.12    |
| HSI     | 38       | 2005.06-2015.04   | 2015.05-2021.12    |
| CSI100  | 82       | 2006.01-2013.12   | 2014.01-2020.12    |

More indicators can be find in https://github.com/dventimiglia/StockStats.

7, 12, 18 stocks are dropped from the index constitutes of DJIA, HSI, and CSI100, respectively.

The implementation of those supervised learning methods is based on Qlib (https://github.com/microsoft/qlib).
Among these metrics, ARR is a profit criteria; AVol and MDD are risk criterion for which the lower values are preferred; ASR, CR, and SoR are risk-return criterion.

### 4.2 Comparison Results on DJIA, HSI, and CSI100

Table 2 summarises the comparison results of our approach and the baselines, and Figure 3 depicts the corresponding accumulated capital in the three markets.

**Performance on DJIA.** Our method achieves the best performance in terms of profit criteria (ARR) and profit-risk criteria (ASR, SoR, and CR). It gains a much higher return rate while keeping the risk at a relatively lower level compared to other methods. For the traditional investment strategies, the CSM strategy achieves a higher average return rate than the Market. However, it also suffers from high volatility and MDD. As shown in Figure 3(a), the CSM strategy performs well from 2009 to 2014, but it performs poorly from 2015 to 2018. While the BLSW strategy performs well during the period of 2009-2012, 2013-2014, and 2015-2017, and fails to gain profits in 2012 and 2014. These results verify that the traditional investment strategies only perform well in certain cases. The supervised learning-based baselines (LightGBM and DA-RNN) fail in reducing risks, since they concentrate more on price movement prediction, ignoring the expensive cost of failed predictions. We find that RL-based methods, i.e., AlphaStock, DeepTrader, and our method, outperform traditional strategies and supervised learning-based methods in risk-return balancing. In Figure 3(a), the wealth curves of RL-based methods rise steadily. Correspondingly, as shown in the DJIA columns of Table 2, these methods have high risk-adjusted return rates and low risks. In particular, DeepTrader achieves the lowest MDD (13.22%) possibly because MDD is 22%.

**Performance on HSI.** As shown in the middle columns of Table 2, the ARR of Market on HSI in the back-test period is negative, which indicates a grim economic situation. Even in this situation, our method still substantially outperforms the baselines in terms of profits and profit-risk criteria, verifying the effectiveness of our approach. As illustrated in Figure 3(b), the traditional investment strategies and the supervised learning baselines perform no better than the Market. The two RL baselines remain competitive methods in this bad economic situation.

**Performance on CSI100.** In the market of CSI100, CSM achieves a fascinating ARR of 32.49%. However, as demonstrated in the right columns of Table 2, all the traditional methods and supervised learning methods have poor performance regarding MDD and AVol. These results are consistent with the declining wealth curves during the bear market in 2015 shown by Figure 3(c). In comparison, RL-based methods maintain relatively steady performance during the whole back-test period. Particularly, our method outperforms the baselines in obtaining higher profits with low risks.

### 4.3 Analysis of Diverse Trading policies

We visualize the trading policies trained with the learning objective proposed in Section 3.1 to answer the following questions: (1) Do the learned policies have diverse trading styles? (2) Is there a single policy performing consistently better than others from the beginning to the end?

To answer Question (1), we depict the portfolio weights given by the four trading policies on May 18th, 2010 in the U.S. market in Figure 4(a). The heights of the bars represent the price rising rates in the future trading period. The colors denote the portfolio weights of the candidate stocks: green for long operations, and red for short operations. As shown in Figure 4(a), the behaviors of those policies are quite different from each other. For example, Policy 1 and 2 hold opposite views on the allocation for stock AXP and CAT. According to the price rising rates, the decision of Policy 2 is the most reasonable on that day. That is, the stocks with higher price rising rates are selected for long operations by Policy 2.

To answer Question (2), we rank the four policies based on their monthly profits from Oct. 2010 to Sep. 2013 in the U.S. market in Figure 4(c), where higher ranks indicate higher profits. Figure 4(c) demonstrates that no single policy always performs better than the others. Instead, their rankings fluctuate with time. Besides, Figure 4(b) depicts the monthly return of the four base policies from Oct. 2007 to Sep. 2010. According to Figure 4(b), in the vast majority of cases, there are at least one base policy that performs not bad even when encountering the 2008 crisis. This confirms the effectiveness of the first phase of DeepTrader. Therefore, switching to appropriate policies according to the current market conditions is crucial to improving the performance of portfolio management as demonstrated in the next subsection.

| Data       | ARR (%) | AVol | ASR | SoR | MDD (%) | CR   | ARR (%) | AVol | ASR | SoR | MDD (%) | CR   | ARR (%) | AVol | ASR | SoR | MDD (%) | CR   |
|------------|---------|------|-----|-----|---------|------|---------|------|-----|-----|---------|------|---------|------|-----|-----|---------|------|
| DJIA       |         |      |     |     |         |      |         |      |     |     |         |      |         |      |     |     |         |      |
| Market     | 8.66    | 0.176 | 0.491 | 7.560 | 53.78     | 0.161 | -1.19 | 0.190 | -0.065 | -0.934 | 35.15   | 0.034 | 16.80   | 0.234 | 0.717 | 0.966 | 43.75   | 0.384 |
| CSM        | 12.84   | 0.537 | 0.239 | 2.073 | 55.30     | 0.231 | -1.24 | 0.195 | 0.013 | 0.426 | 45.52   | 0.109 | 32.49   | 0.510 | 0.638 | 2.945 | 66.78   | 0.487 |
| BLSW       | 18.78   | 0.239 | 0.452 | 1.757 | 49.73     | 0.217 | -0.35 | 0.345 | -0.010 | -0.035 | 67.01   | -0.005 | 11.21   | 0.416 | 0.270 | 1.002 | 75.01   | 0.149 |
| LightGBM   | 8.05    | 0.251 | 0.330 | 1.951 | 62.96     | 0.128 | 0.364 | 0.313 | 0.180 | 0.611 | 48.20   | 0.117 | 14.48   | 0.464 | 0.312 | 1.103 | 72.99   | 0.198 |
| DA-RNN     | 12.09   | 0.230 | 0.525 | 1.841 | 47.58     | 0.254 | -6.96 | -0.281 | -0.248 | -0.723 | 60.48   | -0.115 | 13.93   | 0.305 | 0.456 | 1.630 | 42.92   | 0.325 |
| AlphaStock | 17.86   | 0.150 | 1.190 | 4.333 | 24.51     | 0.729 | 18.75 | 0.193 | 0.969 | 4.169 | 28.90   | 0.649 | 27.51   | 0.250 | 1.099 | 4.501 | 19.08   | 1.442 |
| DeepTrader | 19.63   | 0.122 | 1.112 | 4.617 | 13.22     | 1.127 | 21.68 | 0.189 | 1.145 | 4.322 | 23.24   | 0.933 | 35.55   | 0.280 | 1.197 | 4.380 | 22.70   | 1.478 |
| Ours       | 15.99   | 0.196 | 1.310 | 5.662 | 19.91     | 1.287 | 44.10 | 0.320 | 1.379 | 7.759 | 27.46   | 1.607 | 43.21   | 0.249 | 1.733 | 6.609 | 22.45   | 1.924 |

**Table 2: The comparison results on DJIA, HSI, and CSI100.**
Accumulated Capital ($1M)

Figure 3: The accumulated capital of the proposed method and baselines on DJIA, HSI and CSI100.

Figure 4: Visualization of the diverse policies learned in the first phase on DJIA constituent stocks.

4.4 Visualization of the Learned Meta-Policy
In this subsection, we visualize and analyze the decisions of the learned meta-policy on DJIA during the test period. Figure 5 compares the accumulated capital of the four base trading policies and the method of integrating them together. The color of the integration curve represents the policy identity selected by the meta-policy. As we can see, the wealth of the base policies all significantly outperforms that of the Market index, which verifies the utility of the novel learning objective proposed in Section 3.1. With the second training stage described in Section 3.2, the meta-policy has learned to select the most profitable trading policy to follow. For example, after Sep. 2011, the policy selection switches from the red policy (Policy 2) to the light blue policy (Policy 3). Similarly, from Sep. 2013 to Sep. 2014, the selection of the red policy is switched back to the light blue policy for a faster increase of wealth. We calculate the profit rankings of the selected policies as well, and find that for about 95% of the trading dates, the meta-policy selects the most profitable or the second profitable policy.

Figure 5: The accumulated capital of the meta-policy and the base policies on DJIA during the test period.

4.5 Ablation Study of Meta-Policy Learning
To investigate the effectiveness of the meta-policy learning, we compare the proposed approach with its three variations including Single-Best, Random-Pick and Average-Weight. Single-Best selects the policy with the largest ARR from the policy set. Random-Pick denotes using a random meta-policy instead of training it. Average-Weight means taking the average of outputs of the four base policies as the portfolio weights, which is the same as the ensemble method in MAPS [21].

The results of the ablation study are shown in Table 3. The performance of our method is superior than all the three variant methods. This phenomenon empirically verifies that it is critical to effectively integrate the learned base policies based on market conditions via learning the meta-policy. Note that the Single-Best method outperforms the other two variant methods on HSI and CSI100, indicating that integrating different policies by averaging or random selection is unhelpful to spread risks.


Table 3: Ablation study of the meta-policy learning.

| Data     | Methods         | ARRI(%) | AVol | ASR  | SoR  | MDD(%) | CR  | ARRI(%) | AVol | ASR  | SoR  | MDD(%) | CR  | ARRI(%) | AVol | ASR  | SoR  | MDD(%) | CR  |
|----------|-----------------|---------|------|------|------|--------|-----|---------|------|------|------|--------|-----|---------|------|------|------|--------|-----|
|          | Single-Best     | 16.95   | 0.133| 1.210| 5.210| 13.10  | 1.226| 22.31   | 0.223| 0.199| 5.222| 19.08  | 1.169| 19.11   | 0.221| 1.236| 5.207| 20.40  | 1.363|
|          | Random-Pick     | 21.64   | 0.182| 1.189| 4.637| 24.04  | 0.906| 19.54   | 0.217| 1.191| 3.437| 31.44  | 0.622| 22.15   | 0.444| 0.499| 2.000| 63.55  | 0.348|
|          | Average-Weight  | 20.08   | 0.174| 1.152| 3.844| 28.01  | 0.717| 18.59   | 0.177| 0.901| 3.699| 32.36  | 0.575| 24.89   | 0.369| 0.805| 2.965| 42.77  | 0.382|
|          | Ours            | 23.61   | 0.196| 1.310| 5.662| 19.91  | 1.287| 44.12   | 0.320| 1.379| 7.759| 27.46  | 1.607| 43.21   | 0.249| 1.733| 6.409| 22.45  | 1.924|

5 RELATED WORK

Quantitative portfolio management has received much attention in financial domain. We classify the related work into three categories, and provide a discussion about them as follows.

Traditional Investment Strategies. Traditional portfolio management strategies comprise momentum trading, mean reversion, and multi-factor models. The momentum trading strategies suggest following the current trends, e.g., buying the winners and selling the losers [9], cross-sectional momentum [18], time series momentum [28], and so on. The mean reversion strategy [32] purchases assets whose prices are lower than their historical mean and sells those whose prices are higher than the mean, e.g., buying the losers and selling the winners [17]. Multi-factor models [7] do asset selection according to various factors correlated with the returns of the assets. Although based on solid financial theories, most traditional investment strategies only perform well in particular scenarios, and fail to adapt to the complex markets.

Supervised Learning in Portfolio Management. With an exceptional ability to extract useful representations, deep neural networks have been applied to solve the portfolio management problem in recent years. The supervised learning methods firstly train a deep predictive model of asset prices, and then use the predictive model and a rule to generate portfolio weights. Previous supervised learning methods mainly focus on two aspects to improve the prediction accuracy: using advanced neural network structures [4, 33–35, 49], and proposing new representation learning objectives [8, 13, 51].

The DA-RNN method [33] adopts a temporal attention structure to learn the correlations among historical prices, and the DTML approach [49] utilizes a transformer encoder to learn inter-stock correlations. Graph neural networks [36] are also widely used to detect stock relations by extracting knowledge from extra data resources [4, 15, 34, 35, 47]. As for the new representation learning objectives, Feng et al. [8] proposed an adversarial training objective to achieve more robust predictions, and Hou et al. [13] leveraged a contrastive learning objective to process the multi-granularity data in markets. The prediction models in those supervised learning methods are trained with deep learning techniques, but the way of generating portfolio weights is based on specific rules. In contrast, our approach trains the portfolio allocation framework in an end-to-end manner, which has obtained superior performance in markets as demonstrated in Section 4.

Reinforcement Learning in Portfolio Management. Compared to the supervised learning methods, RL provides a seamless and flexible framework for portfolio management [39]. With different risk appetite, previous RL-based portfolio management algorithms adopt various reward functions including the Sharpe ratio [44], the maximum drawdown [1, 46], and the total profits [19, 45, 48]. The performance of the methods utilizing a pure RL objective is constrained by not fully exploring the fluctuate markets. Via imitating the behaviors of real investors, the Investor-Imitator method [6] achieves better exploration in the complex trading environment. However, this method has not solved the problem of integrating different human trading styles to accomplish a superior trading performance. Besides the portfolio management methods using the single-agent RL framework, some related work employs multi-agent techniques to learn investment decisions [3, 16, 21]. MAPS [21] is a multi-agent portfolio management method, which diversifies the multiple trading agents with a global reward based on correlations between agents. Nevertheless, MAPS naively takes the average of those agents’ outputs as the final portfolio weights. The performance of this averaging method is much worse than our method of learning a meta-policy, as demonstrated in Section 4.5. MPSM [16] is also a multi-agent portfolio management method, and it aims to obtain a scalable and reusable trading system, which is different from our motivation of training policies with varying trading styles to succeed in various market conditions.

6 CONCLUSION

In this paper, we propose MetaTrader, a two-phase deep RL-based portfolio management approach incorporated with imitation learning techniques to deal with the challenge of changing markets. Through abstracting trading knowledge from multiple experts meanwhile interacting with the environments, MetaTrader learns a set of base policies that act diversely in the first phase. Visualization results confirm the diversity of the learned base policies. In the second phase, these learned base policies are then integrated by a meta-policy optimized by DQN based on market conditions. Experiments on three markets demonstrate that MetaTrader outperforms the existing baselines in terms of risk-adjusted returns. And further ablation studies verify the effectiveness of the components in MetaTrader. Since it is important for investors to avoid risks in financial scenarios, we expect the meta-policy could conduct policy ensemble while providing the confidence for its decisions in the future work.

7 ACKNOWLEDGEMENT

Niu and Li are supported in part by the National Natural Science Foundation of China Grant 62161146004, Turing AI Institute of Nanjing and Xi’an Institute for Interdisciplinary Information Core Technology.

REFERENCES

[1] Saud Almahdi and Steve Y Yang. 2017. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with
[49] 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 &amp; Data Mining (Virtual Event, Singapore) (KDD ’21). Association for Computing Machinery, New York, NY, USA, 2037–2045. https://doi.org/10.1145/3447548.3467297

[50] Fisher Yu and Vladlen Koltun. 2015. Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015).

[51] Liang Zeng, Lei Wang, Hui Niu, Jian Li, Ruchen Zhang, Zhonghao Dai, Dewei Zhu, and Ling Wang. 2021. Trade When Opportunity Comes: Price Movement Forecasting via Locality-Aware Attention and Iterative Refinement Labeling. https://doi.org/10.48550/ARXIV.2107.11972