MetaTrader: Learning to Generalize RL Trading Policies Beyond Offline Data

Haochen Yuan[1*], **Minting Pan**[1*], **Yunbo Wang**[1†], **Siyu Gao**[2], **Xiaokang Yang**[1]

[1]MoE Key Lab of Artificial Intelligence, AI Institute, School of Computer Science, Shanghai Jiao Tong University
[2]China International Capital Corporation Limited
{yuanhaochen, panmt53, yunbow ,xkyang}@sjtu.edu.cn, siyu.gao@cicc.com.cn

Abstract

Reinforcement learning (RL) has shown significant promise in sequential portfolio optimization. A typical solution involves optimizing cumulative returns using historical offline data. However, it may produce less generalizable policies that merely "*memorize*" optimal buying and selling actions from the offline data while neglecting the non-stationary nature of the financial market. We frame portfolio optimization of stock data as a specific type of offline RL problem. Our method, **MetaTrader**, presents two key contributions. First, it introduces a novel bilevel RL algorithm that operates on both the original stock data and its transformations. The core idea is that a robust policy should generalize effectively to out-of-distribution data. Second, we propose a new temporal difference (TD) method that leverages a transformation-based conservative TD target to address value overestimation under limited offline data. Empirical results on two publicly available datasets demonstrate that MetaTrader outperforms existing methods, including both traditional stock prediction models and RL-based trading approaches.

Extended version — https://arxiv.org/abs/2505.12759

Introduction

Sequential portfolio optimization aims to allocate assets over time to maximize returns while managing risk, considering investor preferences and market dynamics. A common and simple strategy in algorithmic trading combines stock prediction models (Li, Shen, and Zhu 2018; Xu and Cohen 2018; Wang et al. 2021; Zheng, Liu, and Zhu 2023) with fixed trading rules, such as buying top-ranked stocks and holding them for a set period. Recent reinforcement learning (RL)-based methods (Deng et al. 2016; Ye et al. 2020; Briola et al. 2021; Liu et al. 2021; Kumar 2023; Gao, Wang, and Yang 2023) outperform these approaches by employing end-to-end deep models to extract representations from noisy financial data and directly optimize trading policies for maximizing expected returns on historical datasets.

However, most existing RL-for-finance methods rely on standard RL algorithms applied to static historical datasets, which introduces challenges due to value function approximation errors on out-of-distribution (OOD) data (see Figure 1a). Without the ability to actively explore dynamic financial markets, agents tend to overfit and memorize the highest-reward actions within the dataset, resulting in policies that lack generalizability. This raises a key question: *How can we learn trading policies that balance in-domain optimality*[1] *and out-of-domain generalization?* We refer to this challenge as the **generalization-optimality dilemma**. Generalization allows agents to apply learned policies to unseen market conditions, which is essential in non-stationary or incomplete datasets. However, excessive generalization may lead to unreliable value estimates in OOD regions. On the other hand, focusing solely on in-domain optimality ensures strong performance on familiar data but limits adaptability, potentially resulting in suboptimal decisions in evolving environments.

This paper introduces MetaTrader, an early study on bilevel optimization of actor-critic methods in financial trading, formulated as a *partial-offline* RL problem with decoupled state branches. The core idea extends beyond maximizing rewards on current trajectories by learning policies that also generalize to OOD financial data. As shown in Figure 1b, we enhance stock trading policies with a bilevel RL framework and a novel temporal difference (TD) method.

Our main contribution is improving the policy's generalization capability from both data and algorithmic perspectives. From a data perspective, we introduce data transformation techniques simulating OOD samples, focusing on short-term randomness, long-term trends, and multi-scale correlations. From an algorithmic perspective, we propose a **bilevel actor-critic** method that prevents overfitting by explicitly evaluating the hypothetical policy on the transformed OOD market data, ensuring the model does not simply memorize the optimal policy based on specific patterns in the training data.

Additionally, we propose a novel TD learning approach that makes stable state-action value estimations by approximating **worst-case TD targets from diverse data transformations**. Compared to existing conservative offline RL methods (Kostrikov, Nair, and Levine 2021; Kumar et al. 2020), our approach more effectively mitigates value overes-

[*]These authors contributed equally.
[†]Corresponding author.

[1]*In-domain optimality* refers to maximizing financial returns, measured by maximum rewards, within the historical data used for training.

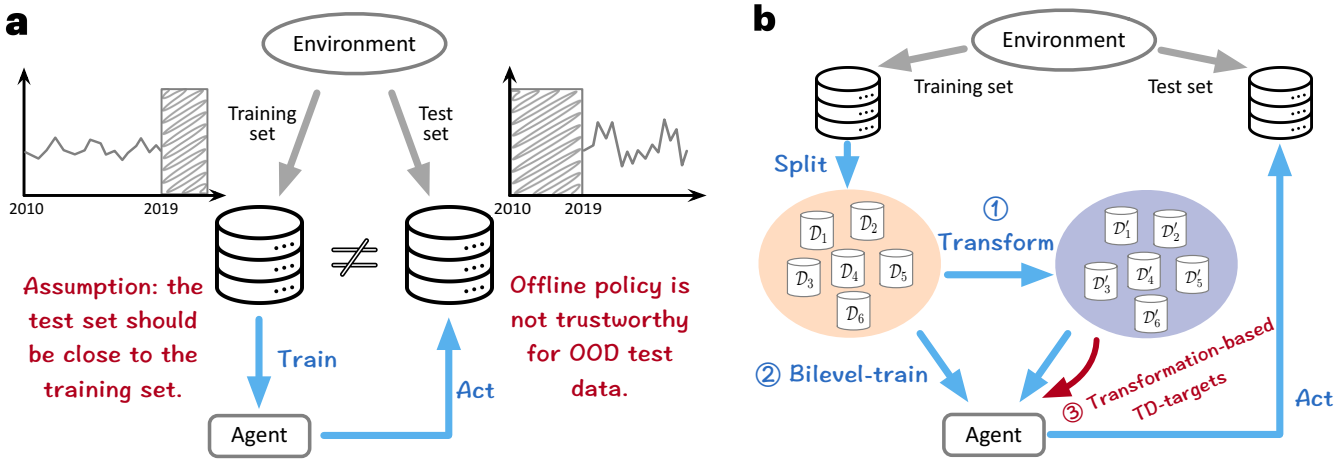


Figure 1: *MetaTrader vs. existing RL-based trading methods:* **a,** Existing RL-for-finance methods typically adopt an offline training setup rather than online RL, causing them to struggle with the *generalization-optimality dilemma* for non-stationary financial markets. **b,** MetaTrader tackles this dilemma through: (i) specialized data transformations to simulate OOD financial data, (ii) a bilevel RL framework that explicitly optimizes both in- and out-of-domain performance across diverse transformations, and (iii) a novel TD method that makes stable state-action value estimations.

timation, particularly under significant distributional shifts between training and test data.

MetaTrader outperforms existing RL-for-finance methods on the CSI-300 and NASDAQ-100 datasets in cumulative returns and Sharpe ratios, demonstrating its ability to balance profits and risks effectively. By addressing policy overfitting and value overestimation, it provides a more reliable and adaptive solution for trading in non-stationary environments.

Problem Formulation: Partial-Offline Reinforcement Learning

We introduce a novel formulation of sequential portfolio optimization as a *partial-offline* RL problem. As illustrated in Figure 2, the Markov decision process (MDP) in the context of stock trading can be described as an 8-tuple $(\mathcal{O}, \mathcal{A}, \mathcal{H}, \mathcal{Z}, P_h, P_z, R, \gamma)$:

Observation space ($\mathcal{O}$). The raw data includes: (1) $o_t^{\text{price}} \in \mathbb{R}^{T\times|S|\times 5}$: Daily *open, close, high, low* stock prices, and trading *volumes* for the previous T days. $|S|$ is the total number of stocks. (2) $o_t^{\text{stat}} \in \mathbb{R}^{|S|\times I}$: I represents technical indicators that reflect the temporal trends of stock prices. (3) A covariance matrix o_t^{cov} that measures the correlations between historical daily closing prices of all stocks. In our partial-offline RL setup, only a finite set of data is accessible.

Action space ($\mathcal{A}$). We use a continuous action space $a_t \in \mathbb{R}^{|S|}$, where each component represents the number of shares to buy, hold, or sell for each asset. To simulate real-world trading, we discretize a_t into several intervals, such as $100, 200, \ldots$ shares when deploying the agent for testing.

Decoupled state space ($\mathcal{H}$, $\mathcal{Z}$). We decouple the state space into two components: $\mathcal{S} = (\mathcal{H}, \mathcal{Z})$. The *market state* h_t is formed by concatenating three types of latent representations h_t^{relat}, h_t^{long}, and h_t^{short}, which are encoded from the observed financial data o_t^{price}, o_t^{stat}, and o_t^{cov}. Our approach

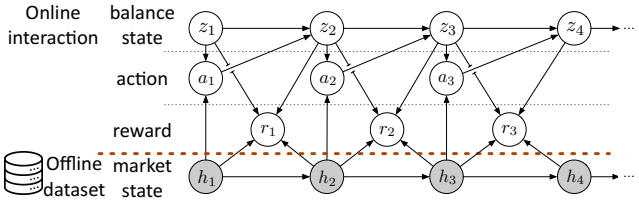


Figure 2: *The partial-offline RL formulation:* The MDP consists of decoupled pairs of action-free *market states* (offline) and action-dependent *balance states* (online).

follows the StockFormer encoder to obtain h_t. The *balance state* $z_t \in \mathbb{R}^{|S|+1}$ represents the total account balance and holding amount of each trading asset.

State transitions (P_h, P_z). Since individual buying and selling actions typically have minimal impact on market dynamics, market state transitions are largely *action-free* whereas balance state transitions are *action-dependent*. Therefore, we define the state transition probabilities as $P_h(h_{t+1}|h_t)$ for market states and $P_z(z_{t+1}|z_t, a_t)$ for balance states. In partial-offline RL, market states are limited to the offline training set, while the agent can explore various actions that lead to new balance states.

Reward function (R) and discount factor (γ). The immediate reward is defined as the daily portfolio return ratios: $r_t = R(h_{t:t+1}, z_{t:t+1})$, where z_{t+1} is dependent on a_t. γ is the reward discount factor that determines how much the RL agents care about rewards in the distant future.

Distinctions from Conventional Offline RL

In standard offline learning, value overestimation arises because agents are trained on limited observable states, historical actions, and rewards, unable to assess OOD actions and future states. This leads to inaccurate value estimates and

overly optimistic TD learning due to bootstrapping. In partial-offline RL, agents can explore actions within a fixed training set and evaluate new policies with online reward feedback. Unlike conventional offline RL, where future rewards cannot be directly estimated, partial-offline RL uses decoupled state spaces to compute next-step rewards, even for out-of-domain actions. While accurately estimating the next market state remains challenging, our approach approximates the worst-case reward using Monte Carlo sampling over transformed states. This transformation-based TD method mitigates value overestimation and outperforms existing conservative offline RL methods, such as CQL (Kumar et al. 2020) and IQL (Kostrikov, Nair, and Levine 2021), in portfolio optimization tasks.

MetaTrader

Overview

To improve policy generalization under distribution shifts, we augment training with diverse OOD market data. As shown in Appendix Figure 1, we partition the offline training set into subsets $\{\mathcal{D}_m\}_{m=1}^{M}$, each segmented into sequences of $T = 64$ time steps (approximating one quarter of trading days in a year). The transformation techniques are specialized to simulate OOD financial data. The training procedure includes two stages: (i) *OOD policy learning* on the first $M - M'$ subsets using both transformed and original data, and (ii) *in-domain finetuning* on the most recent M' subsets using only original data to better align with test conditions. The core of our bilevel RL algorithm is a novel TD learning strategy that estimates worst-case TD values by aggregating transformed TD targets from a batch of augmented data.

Out-of-Distribution Data Transformations

To design effective data transformation methods, we treat market data as multivariate time series, whose dynamic patterns can typically be viewed as a combination of three components: short-term randomness, long-term trends, and multi-scale dynamics. We introduce three data transformation methods, $F_{1:3}$, to simulate OOD yet plausible market changes that have not been included in the training set, with each method focusing on one of the three dynamic components:

- F_1: At each time step, we select the top $\alpha\%$ assets with the highest price gains and invert their growth rates to simulate unexpected short-term disruptions.
- F_2: We reverse the overall trends in each training subset to simulate the long-term impact of sudden market events, aiming to assess the policy's robustness in such scenarios.
- F_3: We downsample the training data by Δ time steps, squeezing the original temporal dynamics. This enables the model to capture multi-scale temporal correlations with greater flexibility.

More technical details are provided in Appendix B.2. By applying the above transformation methods, we expand the subset collections to $\{\mathcal{D}_{m,n}\}_{m=1,n=0}^{M,N}$ during OOD policy learning, where $n = 0$ denotes the original data split. N is scalable in MetaTrader by adjusting hyperparameters (*e.g.*, α, T, and Δ) in the data transformation functions. As N increases, we can achieve (1) a broader expansion of the data distribution for policy learning, increasing the likelihood of covering OOD market dynamics, and (2) more accurate Monte Carlo estimations for the expected worst-case future payoffs during TD learning, which we will discuss later. For simplicity and without loss of generality, we use $N = 3$ with $\alpha = 10$, $T = 64$, and $\Delta = 4$ in our experiments.

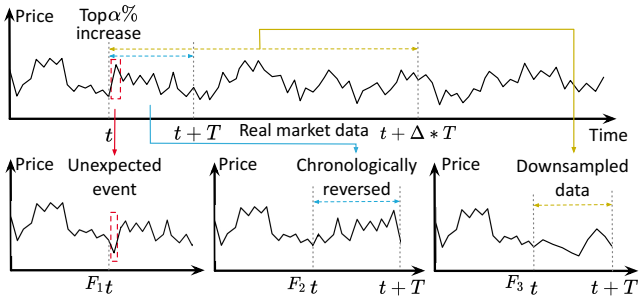


Figure 3: *An example of market data transformations.*

Algorithm 1: OOD Policy Learning

Input: Expanded datasets $\{\mathcal{D}_{m,n}\}_{m=1,n=0}^{M,N}$
Parameters: α_1, α_2, η_1, η_2

Randomly initialize model parameters θ, ϕ_1, ϕ_2
for T_1 steps **do**
  Sample datasets $\{\mathcal{D}_i\}_{i=1}^{K} \sim \{\mathcal{D}_{m,n}\}$
  for each $\mathcal{D}_i \in \{\mathcal{D}_i\}_{i=1}^{K}$ **do**
    Sample a batch of data $\mathcal{B}_i \sim \mathcal{D}_i$
    $\phi'_{1,i} \leftarrow \phi_1 - \eta_1 \nabla_{\phi_1} \mathcal{L}_Q\left(\mathcal{B}_i;\ \phi_1\right)$
    $\phi'_{2,i} \leftarrow \phi_2 - \eta_1 \nabla_{\phi_2} \mathcal{L}_Q\left(\mathcal{B}_i;\ \phi_2\right)$
    $\theta'_i \leftarrow \theta - \alpha_1 \nabla_\theta \mathcal{L}_\pi\left(\mathcal{B}_i;\ \theta, \phi'_{1,i}\right)$
  end for
  $\phi_1 \leftarrow \phi_1 - \eta_2 \sum_i \sum_j \left[\nabla_{\phi_1} \mathcal{L}'_Q\left(\mathcal{B}_i;\ \phi'_{1,j}\right)\right]$
  $\phi_2 \leftarrow \phi_2 - \eta_2 \sum_i \sum_j \left[\nabla_{\phi_2} \mathcal{L}'_Q\left(\mathcal{B}_i;\ \phi'_{2,j}\right)\right]$
  $\theta \leftarrow \theta - \alpha_2 \sum_i \sum_j \left[\nabla_\theta \mathcal{L}_\pi\left(\mathcal{B}_i;\ \theta'_j, \phi'_{1,j}\right)\right]$
end for

Algorithm 2: In-Distribution Model Finetuning

Input: Datasets $\{\mathcal{D}_{m,n=0}\}_{m=M+1}^{M'}$ of real stock data
Parameters: α_1, α_2, η_1, η_2

Obtain the pretrained θ, ϕ_1, ϕ_2 from Alg. 1
for T_2 steps **do**
  Sample datasets $\{\mathcal{D}_i\}_{i=1}^{K} \sim \{\mathcal{D}_{m,0}\}_{m=M+1}^{M'}$
  for each $\mathcal{D}_i \in \{\mathcal{D}_i\}_{i=1}^{K}$ **do**
    Sample disjoint data batches $\mathcal{B}_i^{\text{tr}}, \mathcal{B}_i^{\text{ts}} \sim \mathcal{D}_i$
    $\phi'_{1,i} \leftarrow \phi_1 - \eta_1 \nabla_{\phi_1} \mathcal{L}_Q\left(\mathcal{B}_i^{\text{tr}};\ \phi_1\right)$
    $\phi'_{2,i} \leftarrow \phi_2 - \eta_1 \nabla_{\phi_2} \mathcal{L}_Q\left(\mathcal{B}_i^{\text{tr}};\ \phi_2\right)$
    $\theta'_i \leftarrow \theta - \alpha_1 \nabla_\theta \mathcal{L}_\pi\left(\mathcal{B}_i^{\text{tr}};\ \theta, \phi'_{1,i}\right)$
  end for
  $\phi_1 \leftarrow \phi_1 - \eta_2 \sum_i \left[\nabla_{\phi_1} \mathcal{L}_Q\left(\mathcal{B}_i^{\text{ts}};\ \phi'_{1,i}\right)\right]$
  $\phi_2 \leftarrow \phi_2 - \eta_2 \sum_i \left[\nabla_{\phi_2} \mathcal{L}_Q\left(\mathcal{B}_i^{\text{ts}};\ \phi'_{2,i}\right)\right]$
  $\theta \leftarrow \theta - \alpha_2 \sum_i \left[\nabla_\theta \mathcal{L}_\pi\left(\mathcal{B}_i^{\text{ts}};\ \theta'_i, \phi'_{1,i}\right)\right]$
end for

Bilevel RL with Data Transformations

OOD policy learning. As shown in Alg. 1, we first sample training subsets randomly, $\{\mathcal{D}^{(i)}\}_{i=1}^{K} \sim \{\mathcal{D}_{m,n}\}_{m=1,n=0}^{M,N}$.

We then perform an *inner-loop* optimization step to derive K sets of hypothetical model parameters for the actor and critics, denoted by $\theta^{(i)}$ and $\phi_k^{(i)}$ for each data subset. Here, we employ double Q-networks, parameterized by $\phi_{1,2}$, with k representing their index. The objective of inner-loop optimization is to maximize in-domain rewards for each subset. We proceed with *outer-loop* optimization to compute second-order derivatives by evaluating the *inner-loop* parameters $\theta^{(j)}$ and $\phi_k^{(j)}$, learned from subset j, on the data split $\mathcal{B}^{(i)}$ from distinct subsets (as shown in Appendix Figure 1). This approach distinguishes itself from most existing meta-RL methods by conducting bilevel gradient updates across distinct pairs of subsets. The aim is to update model parameters to improve the policy's robustness to OOD trajectories. The actor objective function $\mathcal{L}_\pi$ is as follows: $\min_\theta \mathbb{E}_{s_t}\big[D_{\mathrm{KL}}(\pi_\theta(a_t \mid s_t) \parallel \exp(Q_{\phi_1}(s_t, a_t))/Z_{\phi_1}(s_t))\big]$, where $s_t = [h_t^{\text{relat}}, h_t^{\text{long}}, h_t^{\text{short}}, z_t]$. Z_{ϕ_1} is for normalization.

In-domain finetuning. Due to the non-stationary nature of the time-evolving market, fine-tuning MetaTrader on recent training data, close to the test set, can enhance its final performance. In Alg. 2, we employ the bilevel optimization scheme *within* each training subset. We first draw subsets from the buffer of raw data, such that $\{\mathcal{D}^{(i)}\}_{i=1}^{K} \sim \{\mathcal{D}_{m,n=0}\}_{m=M-M'+1}^{M}$. Importantly, we exclusively use the original data to eliminate the unexpected noise introduced by the transformed data. It is essential to note that during the fine-tuning phase, we perform the inner-loop and outer-loop gradient steps on separate data batches, $\mathcal{B}_{\text{tr}}^{(i)}$ and $\mathcal{B}_{\text{ts}}^{(i)}$, sampled from the same subset $\mathcal{D}^{(i)}$. This approach aims to facilitate model adaptation to recent market dynamics.

Transformation-Based TD Learning with Worst-Case Bootstrapping

We propose a novel TD method for training the critic model during the OOD policy learning phase. In Alg. 1, the training objectives of $Q_{\phi_{1,2}}$, including the inner-loop $\mathcal{L}_Q$ and the outer-loop $\mathcal{L}_Q^{\text{ens}}$, can be formulated as $\min_{\phi_k} \mathbb{E}_{(s_t,a_t)}\big[Q_{\phi_k}(s_t, a_t) - \mathrm{sg}\big(\widehat{Q}(s_t, a_t)\big)\big]^2$, where $Q_{\phi_k}(\cdot)$ represents the TD estimate of the critic k at timestamp t, $\widehat{Q}(\cdot)$ represents the corresponding TD target, and $\mathrm{sg}(\cdot)$ denotes the stopping of gradient backpropagation. We here denote $s_t = [h_t, z_t]$ and $h_t = [h_t^{\text{relat}}, h_t^{\text{long}}, h_t^{\text{short}}]$. In the inner-loop optimization step, we formulate the TD target $\widehat{Q}(\cdot)$ as

$$\widehat{Q}(s_t, a_t) = R(s_t, s_{t+1}) + \gamma\big[-\lambda \log \pi_\theta(a_{t+1} \mid s_{t+1}) + \min_{k=1,2} Q_{\bar{\phi}_k}(s_{t+1}, a_{t+1})\big], \tag{1}$$

where $R(\cdot)$ is the pre-defined reward function and a_{t+1} is generated by the policy $\pi_\theta(\cdot \mid s_{t+1})$. We incorporate double target Q-networks $Q_{\bar{\phi}_{1,2}}$, which are updated using the moving-average parameters from corresponding Q-networks $Q_{\phi_{1,2}}$. $Q_{\bar{\phi}_k}$ is the next-step Q-value from each target Q-network.

In the outer-loop gradient update step, as shown in Appendix Figure 2, we use a new form of TD target derived from a batch of transformed data. We denote $\widehat{Q}(\cdot)$ as

$$\widehat{Q}^{\text{ens}}(s_t, a_t) = \min_{n=0:N}\Big[R(s_t, s_{t+1}^{(n)}) + \gamma\big(-\lambda \log \pi_\theta(a_{t+1}^{(n)} \mid s_{t+1}^{(n)}) + \min_{k=1,2} Q_{\bar{\phi}_k}(s_{t+1}^{(n)}, a_{t+1}^{(n)})\big)\Big], \tag{2}$$

where $\{s_{t+1}^{(n)}\}_{n=1}^{N}$ represent simulated next-step market states transformed by $F_{1:3}$, and $a_{t+1}^{(n)}$ is generated by the policy $\pi_\theta(\cdot \mid s_{t+1}^{(n)})$. Notably, s_{t+1} is referred to as $s_{t+1}^{(0)}$, denoting the next-step market state encoded from the original data.

Eq. (2) is feasible in our partial-offline RL formulation, where $h_{t+1} \sim P_h(h_{t+1} \mid h_t)$ and $z_{t+1} \sim P_z(z_{t+1} \mid z_t, a_t)$. This decoupled state transition approach allows us to directly evaluate the value of a_t given h_t and z_t, since in $R(h_t, h_{t+1}, z_t, z_{t+1})$, the only missing component is the next market state h_{t+1}, which is independent of the action. As previously described, while accurately estimating the distribution of h_{t+1} is intractable, we perform Monte Carlo sampling over diverse transformations of the original data at $t+1$ and approximate the *worst-case* TD target using $\{h_{t+1}^{(n)}\}_{n=1}^{N}$.

It is important to note that existing ensemble-based TD methods (An et al. 2021; Lee et al. 2022; Wu et al. 2021b) typically train multiple target Q-networks with separate model parameters and compute ensemble value regularization by exploiting the implicit diversity among these Q-networks. In contrast, our approach uses a single pair of target Q-networks and derives the worst-case TD target by leveraging the explicit diversity introduced by transformed data.

Results

Experimental Setups

We evaluate MetaTrader using the CSI-300 and NASDAQ-100 datasets, both adopted from StockFormer (Gao, Wang, and Yang 2023). The CSI dataset is sourced from the CSI-300 Composite Index, which includes 88 stocks. It spans from 01/17/2011 to 04/01/2022 and is split into training and test sets containing 1,936 and 785 trading days, respectively. The NASDAQ dataset contains 86 NASDAQ stocks, collected from Yahoo Finance. It covers the period from 01/17/2011 to 04/01/2022, with a training set of 2,002 trading days and a test set of 819 trading days. The data used in our experiments is sourced from the public, fully preprocessed dataset released by StockFormer. Notably, our evaluation period includes the original test set of StockFormer and extends it to a longer horizon. We provide more details in Appendix C.

We compare MetaTrader with the following models:

- *Market benchmarks:* the CSI-300 Index and the NASDAQ-100 Index.
- *General time series forecasting methods:* AutoFormer (Wu et al. 2021a).
- *Stock prediction methods:* HATR (Wang et al. 2021), Relational Ranking (Feng et al. 2019), and FactorVAE (Duan et al. 2022).
- *RL-based stock trading methods:* FinRL (Liu et al. 2021), SARL (Ye et al. 2020), and StockFormer (Gao, Wang, and Yang 2023).

Method	CSI				NASDAQ			
	CR↑	AR↑	SR↑	MDD↓	CR↑	AR↑	SR↑	MDD↓
Market benchmark	0.08	0.02	0.23	0.31	0.99	0.26	0.98	0.28
HATR	-0.05	-0.02	0.06	0.51	0.10	0.03	0.25	0.35
Relational Ranking	0.79	0.22	0.75	0.37	-0.13	-0.05	-0.05	0.37
AutoFormer	-0.08	-0.03	0.02	0.58	-0.28	-0.10	-0.27	0.41
FactorVAE	0.96	0.25	1.25	**0.17**	0.90	0.24	0.77	**0.26**
FinRL-SAC	0.83±0.05	0.22±0.01	0.92±0.04	0.30±0.01	0.37±0.05	0.11±0.01	0.54±0.04	0.32±0.01
FinRL-DDPG	0.58±0.15	0.16±0.04	0.73±0.12	0.34±0.03	0.91±0.11	0.24±0.02	0.75±0.05	0.41±0.01
CQL	0.64±0.07	0.18±0.02	0.75±0.05	0.33±0.02	0.77±0.10	0.21±0.02	0.76±0.06	0.35±0.01
IQL	1.02±0.10	0.26±0.02	0.94±0.06	0.32±0.02	0.92±0.09	0.24±0.02	0.87±0.04	0.36±0.01
SARL	1.06±0.14	0.27±0.03	0.98±0.08	0.36±0.02	1.03±0.20	0.27±0.04	0.80±0.09	0.40±0.01
StockFormer	1.24±0.10	0.31±0.02	1.20±0.06	0.31±0.02	0.98±0.07	0.26±0.02	0.93±0.04	0.32±0.01
MetaTrader	**1.44±0.07**	**0.35±0.03**	**1.35±0.08**	0.28±0.02	**1.30±0.08**	**0.32±0.02**	**1.11±0.04**	0.31±0.02

Table 1: *Offline evaluation on CSI and NASDAQ:* We use *cumulative return*, *annualized return*, *Sharpe ratio*, and *maximum drawdown* as metrics. Given the inherent instability of RL, we present the results of RL models from 10 random training seeds.

Method	CSI-300			NASDAQ-100		
	CR↑	AR↑	SR↑	CR↑	AR↑	SR↑
Market benchmark	0.08	0.02	0.23	0.99	0.26	0.98
FactorVAE-Finetune	1.07	0.27	1.32	1.02	0.26	0.84
StockFormer-Finetune	1.46±0.05	0.35±0.01	1.37±0.05	1.26±0.08	0.31±0.02	1.03±0.09
MetaTrader	**1.84±0.03**	**0.42±0.01**	**1.61±0.03**	**1.58±0.03**	**0.37±0.01**	**1.47±0.04**

Table 2: *Online adaptation results.* We divide the entire test set into 3 equal-length splits. Throughout the streaming test phase, we progressively finetune the models on the **previous test split** before evaluating them on the current one.

- *Offline RL methods:* CQL (Kumar et al. 2020) and IQL (Kostrikov, Nair, and Levine 2021).

For the stock prediction methods, we apply the *buy-and-hold* strategy, *i.e.*, buying the stock with the highest predicted 5-days return and selling it 5 days later. The results for the RL-based methods are averaged across at least 3 random training seeds. All models are tested with transaction costs.

Standard Offline Evaluation

For both datasets, we perform OOD policy learning using training data from 01/17/2011 to 12/31/2018. Next, we conduct in-domain finetuning on the last-year training data, specifically from 01/04/2018 to 12/31/2018. To ensure no overlap between the test and training sets, we set the input data of the test sequences to start from January 2019, covering the trading days from 04/01/2019 to 04/01/2022.

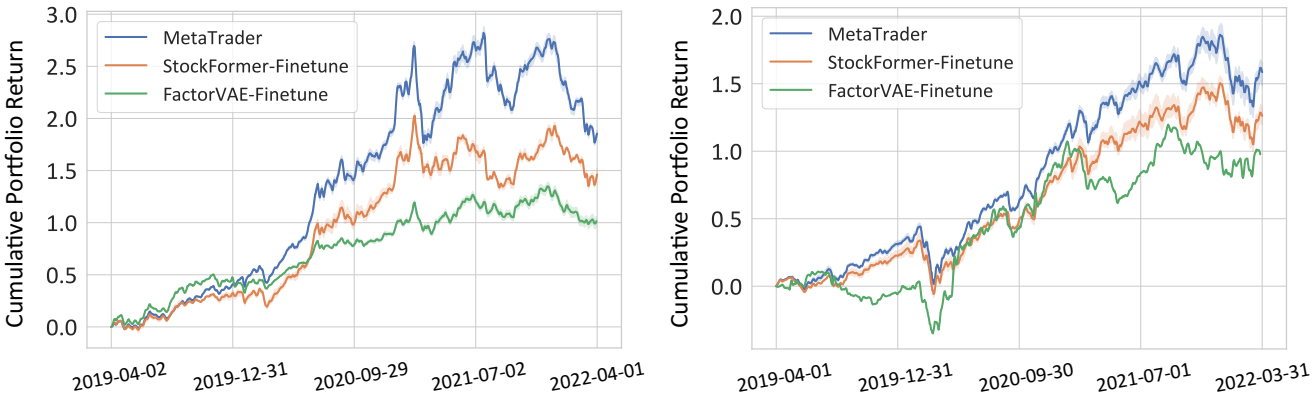
Table 1 presents the quantitative results of MetaTrader in *cumulative return* (CR), *annualized return* (AR), *Sharpe ratio* (SR), and *maximum drawdown* (MDD). Please refer to Appendix D for detailed definitions of the metrics. Notably, MetaTrader outperforms all stock prediction methods by substantial margins. For example, our approach outperforms FactorVAE by **50%** in cumulative return (1.44 vs. 0.96) on the CSI dataset and by **44.4%** on the NASDAQ dataset (1.30 vs. 0.90). The Sharpe ratio measures the additional return that an investor receives per unit of risk. It is defined as the difference between the returns of the investment and the risk-free return, divided by the standard deviation of the investment returns. MetaTrader outperforms FactorVAE by **44.1%** in Sharpe ratio (1.11 vs. 0.77) on the NASDAQ dataset, showing a strong balance between portfolio returns and risk control.

Compared to other RL-based trading methods, MetaTrader delivers the best performance across all evaluation metrics. It improves upon the state-of-the-art StockFormer method in terms of cumulative return by **16.1%** on the CSI dataset and by **32.7%** on NASDAQ. Furthermore, we implement baseline models using the same neural network architecture as MetaTrader, but trained with other conservative offline RL techniques such as CQL and IQL. As observed, existing offline RL methods struggle with RL-for-finance tasks due to fluctuations in data distributions, leading to significant shifts between the training and testing domains. In contrast, MetaTrader achieves significant performance gains through bilevel policy learning, which effectively prevents the policy from overfitting to the offline data.

Online Adaptation on Streaming Data

We employ another experimental setup that closely aligns with dynamic financial decision-making applications, where we finetune the model on-the-fly using streaming test data. The entire test set is divided into 3 equal-length periods: 04/01/2019—04/01/2020, 04/02/2020—04/01/2021, and 04/02/2021—04/01/2022. Each period is followed by an in-domain finetuning phase before testing. For instance, for the test period of 04/02/2020—04/01/2021, we perform in-domain finetuning using data in 04/01/2019—04/01/2020 prior to testing.

In this setup, our primary comparison is between Meta-



(a) Results on the CSI dataset

(b) Results on the NASDAQ dataset

Figure 4: *Cumulative returns under the online adaptation setup:* We partition the entire test set into 3 equally sized splits and incrementally finetune the models using each preceding split sequentially. All results are obtained from 10 random seeds.

Method	CR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$
Original	1.68±0.04	0.39±0.01	1.49±0.03	0.22±0.01
Minimum value	1.49±0.06	0.36±0.01	1.38±0.06	0.30±0.02
Mean value	1.39±0.08	0.34±0.02	1.29±0.07	0.33±0.01
Ours	**1.84±0.03**	**0.42±0.01**	**1.61±0.03**	**0.20±0.01**

Table 3: Ablation studies of the ensemble-based TD target in the *OOD policy learning* phase (Alg. 1). The experiments are conducted under the online adaptation setup on CSI.

Bilevel	Transform	CSI CR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$
✗	✗	1.78±0.03	0.41±0.01	1.57±0.03	0.23±0.01
✓	✗	**1.84±0.03**	**0.42±0.01**	**1.61±0.03**	**0.20**±0.01
✓	✓	0.84±0.04	0.23±0.01	0.94±0.05	0.33±0.02
Bilevel	Transform	NASDAQ CR$^{\uparrow}$	AR$^{\uparrow}$	SR$^{\uparrow}$	MDD$^{\downarrow}$
✗	✗	1.41±0.04	0.34±0.01	1.34±0.04	0.31±0.01
✓	✗	**1.58±0.03**	**0.37±0.01**	**1.47±0.04**	**0.30±0.01**
✓	✓	1.24±0.03	0.31±0.01	1.03±0.05	0.33±0.01

Table 4: *Ablation studies of alternative designs of in-domain finetuning (Alg. 2):* The experiments are conducted within the online adaptation setup.

Trader, *FactorVAE-Finetune*, and *StockFormer-Finetune*, all of which are continuously finetuned using the streaming test data. As shown in Table 2 and Figure 4, MetaTrader offers a significant advantage over the other approaches, including the state-of-the-art stock prediction model (FactorVAE) and the RL-based stock trading method (StockFormer). On the CSI dataset, it improves StockFormer-Finetune by **26%** in cumulative return (1.84 vs. 1.46) and by approximately **18%** in Sharpe ratio (1.61 vs. 1.37). On the NASDAQ dataset, MetaTrader improves StockFormer-Finetune by over **25%** in cumulative return (1.58 vs. 1.26) and by around **43%** in Sharpe ratio (1.47 vs. 1.03).

Ablation Studies

Transformation-based TD ensembles. To assess the effectiveness of the transformation-based TD method used in the proposed bilevel RL framework, we implement a baseline model that adopts the original TD method from SAC. Figure 3 demonstrates the improvements achieved by our proposed TD method, with a significant increase of **9.5%** in cumulative return on the CSI dataset. Furthermore, we experiment with other TD ensemble approaches (An et al. 2021; Lee et al. 2022), *i.e.*, such as using the minimum and mean values from 5 parallel Q-networks as the Bellman target. As shown in Table 3, our approach, which computes TD targets using transformed data and a single pair of target Q-networks, demonstrates a significant advantage over other ensemble-based TD learning alternatives. These results highlight the effectiveness of using diverse future data transformations to approximate worst-case future returns, guiding the agent toward safer and more reliable trading behaviors.

In-domain finetuning. We perform model finetuning on real data from the most recent year, using bilevel gradient updates (see Alg. 2). In Table 4, we investigate the necessity of bilevel optimization and explain why the transformed data is excluded during the finetuning phase. Compared to directly using the inner-loop gradients to update the model, bilevel optimization results in a 3.4% improvement in the cumulative return on the CSI dataset (1.84 vs. 1.78) and a 12.1% improvement on NASDAQ (1.58 vs. 1.41). Furthermore, incorporating data transformations during the finetuning phase leads to a noticeable performance drop. This is expected, as the transformed data may not align with the recent dynamic patterns close to the test set.

The impact of data transformations. We compare baselines that (i) use no transformed data and (ii) use only partial transformations during training. Table 5 reveals two key findings: First, applying data transformation during OOD policy learning consistently improves performance across all metrics. Second, combining multiple transformations yields further gains, with a notable **10.8%** increase in cumulative return (1.66 → 1.84) on CSI for online adaptation.

Transformation	CR$^\uparrow$	AR$^\uparrow$	SR$^\uparrow$	Transformation	CR$^\uparrow$	AR$^\uparrow$	SR$^\uparrow$
w/o Data Transform	1.66	0.385	1.44	F_3	1.73	0.398	1.53
F_1	1.69	0.390	1.53	$F_1 + F_2$	1.77	0.404	1.59
F_2	1.67	0.389	1.50	$F_1 + F_2 + F_3$	**1.84**	**0.417**	**1.61**

Table 5: Analyses of data transformation techniques used in the *out-of-distribution policy learning* phase (Alg. 1). We report the mean results on CSI over 3 random seeds.

Method	CR$^\uparrow$	AR$^\uparrow$	SR$^\uparrow$	MDD$^\downarrow$
Market benchmark	0.15	0.05	0.30	0.29
SARL	0.18	0.06	0.32	0.46
FinRL-SAC	-0.11	-0.04	-0.03	0.46
StockFormer	0.24	0.07	0.37	0.40
MetaTrader	**0.52**	**0.15**	**0.74**	**0.37**

Table 6: *Online evaluation results on the expanded dataset with* 587 *stocks.*

Results on Larger Markets

Existing stock trading methods typically operate on small-scale datasets. This limitation arises from two main factors. From the data side, real-world trading suspensions are common, so prior work often filters stocks with high data availability (*e.g.*, $> 98\%$ in StockFormer) to minimize interpolation noise. From the algorithm side, expanding the stock pool drastically enlarges the action space, making it harder for RL methods to scale. It potentially results in more action dimensions than training sequences when trading thousands of stocks. We conduct experiments on a larger dataset by expanding CSI to 587 stocks and show the results in Table 6.

Related Work

Learning-based portfolio optimization. There are two primary categories of deep learning-based approaches for portfolio optimization. The first category leverages the temporal modeling capabilities of existing models to forecast asset prices (Li, Shen, and Zhu 2018; Xu and Cohen 2018; Feng et al. 2019; Wang et al. 2021; Duan et al. 2022; Zheng, Liu, and Zhu 2023). For stock trading, these methods are typically combined with relatively simple trading policies, such as buying stocks predicted to yield the highest returns and selling them at a predetermined time. The second line of work employs deep RL, framing portfolio optimization as an MDP to make dynamic decisions about the timing and quantity of investments (Deng et al. 2016; Briola et al. 2021; Jeong and Kim 2019; Liu et al. 2021; Kumar 2023; Liu et al. 2022; Gao, Wang, and Yang 2023). In this paper, we show that these methods often overfit to the optimal yet unrealistic behaviors specific to the offline dataset. This limits the agent's ability to generalize to OOD data scenarios.

Bilevel RL. Bilevel optimization-based meta-learning has emerged as a powerful tool for addressing various machine learning problems, such as few-shot learning (Antoniou, Edwards, and Storkey 2019; Li et al. 2019; Triantafillou et al. 2020; Day et al. 2022; Cheng et al. 2023) and domain adaptation (Schmidhuber 1987; Finn 2018; Hospedales et al. 2021). In the context of RL, it has been applied to learning dynamics models (Sæmundsson, Hofmann, and Deisenroth 2018; Nagabandi et al. 2019) or directly optimizing policies (Duan et al. 2017; Mishra et al. 2018; Finn, Abbeel, and Levine 2017; Nagabandi et al. 2019; Gupta et al. 2018; Humplik et al. 2019; Mitchell et al. 2021; Pong et al. 2022; Tang 2022; Greenberg et al. 2023; Gao et al. 2023; Ma et al. 2023; Wang et al. 2023). These optimization-based meta-learning models have demonstrated the potential to improve the generalizability of RL policies. In contrast to prior work, we focus on the challenges posed by limited and non-stationary financial data in policy learning. To address these challenges, we introduce a novel bilevel RL approach that enhances policy generalizability while mitigating the issue of value overestimation.

Ensemble RL. Ensemble methods are useful in uncertain environments or when the goal is to learn an optimal policy from a static dataset previously collected, particularly when applied to OOD data points. A common approach builds on *model diversity* (An et al. 2021; Lee et al. 2022; Wu et al. 2021b; Zhao et al. 2023), using multiple Q-networks and aggregating their outputs to estimate the Bellman target. In contrast, our method exploits *data diversity* by applying bootstrapped data transformations to the Q-function, capturing broader variability while maintaining efficiency.

Conclusions and Limitations

This paper presents MetaTrader, an RL approach that formulates sequential trading as a partial-offline RL problem with decoupled market states and balance states. MetaTrader improves generalization to non-stationary stock data by integrating bilevel RL and specific data augmentation techniques. We further propose a novel Q-learning method with a transformation-based TD bootstrapping method, which learns stable policies in highly dynamic scenarios with limited training data. Experiments on two public stock datasets demonstrate the effectiveness of MetaTrader compared to existing RL-for-finance methods, showcasing its great potential in handling rapidly changing financial markets.

An unresolved problem in this study is training instability, as RL-based methods (including SARL, StockFormer, and ours) show higher performance variance than stock prediction models like FactorVAE. Future work will explore more robust initialization and regularization to further enhance stability. Additionally, since our method is currently limited to daily-level data with a per-sequence inference time of approximately 20ms, we aim to improve efficiency for potential use in high-frequency trading.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant 62250062), the Smart Grid National Science and Technology Major Project (Grant 2024ZD0801200), the Shanghai Municipal Science and Technology Major Project (Grant 2021SHZDZX0102), and the Fundamental Research Funds for the Central Universities.

References

An, G.; Moon, S.; Kim, J.-H.; and Song, H. O. 2021. Uncertainty-based offline reinforcement learning with diversified q-ensemble. In *NeurIPS*, volume 34, 7436–7447.

Antoniou, A.; Edwards, H.; and Storkey, A. 2019. How to train your MAML. In *ICLR*.

Briola, A.; Turiel, J.; Marcaccioli, R.; Cauderan, A.; and Aste, T. 2021. Deep reinforcement learning for active high frequency trading. *arXiv preprint arXiv:2101.07107*.

Cheng, C.; Song, L.; Xue, R.; Wang, H.; Sun, H.; Ge, Y.; and Shan, Y. 2023. Meta-Adapter: An Online Few-shot Learner for Vision-Language Model. In *NeurIPS*, volume 36, 55361–55374.

Day, B. J.; Torné, R. V.; Simidjievski, N.; and Lio, P. 2022. Attentional Meta-learners for Few-shot Polythetic Classification. In *ICML*, volume 162, 4867–4889.

Deng, Y.; Bao, F.; Kong, Y.; Ren, Z.; and Dai, Q. 2016. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3): 653–664.

Duan, Y.; Schulman, J.; Chen, X.; Bartlett, P. L.; Sutskever, I.; and Abbeel, P. 2017. RI^2: Fast reinforcement learning via slow reinforcement learning. In *ICLR*.

Duan, Y.; Wang, L.; Zhang, Q.; and Li, J. 2022. Factorvae: A probabilistic dynamic factor model based on variational autoencoder for predicting cross-sectional stock returns. In *AAAI*, 4468–4476.

Feng, F.; He, X.; Wang, X.; Luo, C.; Liu, Y.; and Chua, T.-S. 2019. Temporal relational ranking for stock prediction. *ACM Transactions on Information Systems (TOIS)*, 37(2): 1–30.

Finn, C. 2018. *Learning to Learn with Gradients*. Ph.D. thesis, University of California, Berkeley, USA.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 1126–1135.

Gao, S.; Wang, Y.; and Yang, X. 2023. StockFormer: learning hybrid trading machines with predictive coding. In *IJCAI*, 4766–4774.

Gao, Y.; Zhang, R.; Guo, J.; Wu, F.; Yi, Q.; Peng, S.; Lan, S.; Chen, R.; Du, Z.; Hu, X.; et al. 2023. Context Shift Reduction for Offline Meta-Reinforcement Learning. In *NeurIPS*, volume 36, 10775–10795.

Greenberg, I.; Mannor, S.; Chechik, G.; and Meirom, E. 2023. Train Hard, Fight Easy: Robust Meta Reinforcement Learning. In *NeurIPS*, volume 36, 68276–68299.

Gupta, A.; Mendonca, R.; Liu, Y.; Abbeel, P.; and Levine, S. 2018. Meta-reinforcement learning of structured exploration strategies. In *NeurIPS*, volume 31, 5302–5311.

Hospedales, T.; Antoniou, A.; Micaelli, P.; and Storkey, A. 2021. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9): 5149–5169.

Humplik, J.; Galashov, A.; Hasenclever, L.; Ortega, P. A.; Teh, Y. W.; and Heess, N. 2019. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*.

Jeong, G.; and Kim, H. Y. 2019. Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, 117: 125–138.

Kostrikov, I.; Nair, A.; and Levine, S. 2021. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*.

Kumar, A.; Zhou, A.; Tucker, G.; and Levine, S. 2020. Conservative q-learning for offline reinforcement learning. In *NeurIPS*, volume 33, 1179–1191.

Kumar, P. 2023. Deep Reinforcement Learning for High-Frequency Market Making. In *ACML*, volume 189, 531–546.

Lee, S.; Seo, Y.; Lee, K.; Abbeel, P.; and Shin, J. 2022. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *CoRL*, 1702–1712. PMLR.

Li, H.; Shen, Y.; and Zhu, Y. 2018. Stock price prediction using attention-based multi-input LSTM. In *ACML*, volume 95, 454–469.

Li, W.; Wang, L.; Xu, J.; Huo, J.; Gao, Y.; and Luo, J. 2019. Revisiting local descriptor based image-to-class measure for few-shot learning. In *CVPR*, 7260–7268.

Liu, X.-Y.; Xia, Z.; Rui, J.; Gao, J.; Yang, H.; Zhu, M.; Wang, C.; Wang, Z.; and Guo, J. 2022. FinRL-Meta: Market environments and benchmarks for data-driven financial reinforcement learning. In *NeurIPS*, volume 35, 1835–1849.

Liu, X.-Y.; Yang, H.; Gao, J.; and Wang, C. D. 2021. FinRL: Deep reinforcement learning framework to automate trading in quantitative finance. In *ICAIF*.

Ma, Z.; Guo, H.; Chen, J.; Li, Z.; Peng, G.; Gong, Y.-J.; Ma, Y.; and Cao, Z. 2023. MetaBox: A Benchmark Platform for Meta-Black-Box Optimization with Reinforcement Learning. In *NeurIPS*.

Mishra, N.; Rohaninejad, M.; Chen, X.; and Abbeel, P. 2018. A simple neural attentive meta-learner. In *ICLR*.

Mitchell, E.; Rafailov, R.; Peng, X. B.; Levine, S.; and Finn, C. 2021. Offline meta-reinforcement learning with advantage weighting. In *ICML*, 6979–6991.

Nagabandi, A.; Clavera, I.; Liu, S.; Fearing, R. S.; Abbeel, P.; Levine, S.; and Finn, C. 2019. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *ICLR*.

Pong, V. H.; Nair, A. V.; Smith, L. M.; Huang, C.; and Levine, S. 2022. Offline meta-reinforcement learning with online self-supervision. In *ICML*, 17813–17829.

Sæmundsson, S.; Hofmann, K.; and Deisenroth, M. P. 2018. Meta reinforcement learning with latent variable gaussian processes. *arXiv preprint arXiv:1803.07551*.

Schmidhuber, J. 1987. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. Ph.D. thesis, Technische Universität München.

Tang, Y. 2022. Biased Gradient Estimate with Drastic Variance Reduction for Meta Reinforcement Learning. In *ICML*, volume 162, 21050–21075.

Triantafillou, E.; Zhu, T.; Dumoulin, V.; Lamblin, P.; Evci, U.; Xu, K.; Goroshin, R.; Gelada, C.; Swersky, K.; Manzagol, P.-A.; et al. 2020. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *ICLR*.

Wang, H.; Li, S.; Wang, T.; and Zheng, J. 2021. Hierarchical Adaptive Temporal-Relational Modeling for Stock Trend Prediction. In *IJCAI*, 3691–3698.

Wang, J.; Zhang, J.; Jiang, H.; Zhang, J.; Wang, L.; and Zhang, C. 2023. Offline Meta Reinforcement Learning with In-Distribution Online Adaptation. In *ICML*, volume 202, 36626–36669.

Wu, H.; Xu, J.; Wang, J.; and Long, M. 2021a. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *NeurIPS*, volume 34, 22419–22430.

Wu, Y.; Chen, X.; Wang, C.; Zhang, Y.; Zhou, Z.; and Ross, K. W. 2021b. Aggressive Q-Learning with Ensembles: Achieving Both High Sample Efficiency and High Asymptotic Performance. *arXiv preprint arXiv:2111.09159*.

Xu, Y.; and Cohen, S. B. 2018. Stock movement prediction from tweets and historical prices. In *ACL*, 1970–1979.

Ye, Y.; Pei, H.; Wang, B.; Chen, P.-Y.; Zhu, Y.; Xiao, J.; and Li, B. 2020. Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *AAAI*, 1112–1119.

Zhao, K.; Ma, Y.; Liu, J.; Zheng, Y.; and Meng, Z. 2023. Ensemble-based offline-to-online reinforcement learning: From pessimistic learning to optimistic exploration. *arXiv preprint arXiv:2306.06871*.

Zheng, X.; Liu, M.; and Zhu, M. 2023. Deep hashing-based dynamic stock correlation estimation via normalizing flow. In *IJCAI*, 4993–5001.