FineFT: Efficient and Risk-Aware Ensemble Reinforcement Learning for Futures Trading

Molei Qin
Nanyang Technological University
Singapore
molei001@e.ntu.edu.sg

Xinyu Cai
Nanyang Technological University
Singapore
xinyu009@e.ntu.edu.sg

Yewen Li
Nanyang Technological University
Singapore
yewen001@e.ntu.edu.sg

Haochong Xia
Nanyang Technological University
Singapore
haochong001@e.ntu.edu.sg

Chuqiao Zong
Nanyang Technological University
Singapore
zong0005@e.ntu.edu.sg

Shuo Sun
Hong Kong University of Science and Technology (Guangzhou)
China
shuosun@hkust-gz.edu.cn

Xinrun Wang*
Singapore Management University
Singapore
xrwang@smu.edu.sg

Bo An
Nanyang Technological University
Singapore
boan@ntu.edu.sg

Abstract

Futures are contracts obligating the exchange of an asset at a predetermined date and price, notable for their high leverage (e.g., 5-fold) and liquidity (e.g., trillions of dollars) and, therefore, thrive in the Crypto market. Reinforcement learning (RL) has been widely applied in various quantitative tasks. However, most methods focus on the spot (e.g., stock) and could not be directly applied to the futures market with high leverage because of 2 key challenges. First, high leverage amplifies reward fluctuations, making RL training highly stochastic and difficult to converge. Second, prior works lacked self-awareness of capability boundaries, exposing them to the risk of significant capital loss when encountering previously unseen market state representations (e.g., during a black swan event like COVID-19). To tackle these challenges, we propose the e**F**ficient and r**I**sk-aware e**N**semble r**E**inforcement learning for **F**utures **T**rading (FineFT), a novel three-stage ensemble RL framework with stable training and proper risk management. In stage I, ensemble Q learners are selectively updated by ensemble temporal difference (TD) errors, i.e., TD errors across different learners, to improve convergence and performance. In stage II, we filter the Q-learners based on their profitabilities under different market dynamics and train variational autoencoders (VAEs) on market representations of each dynamic to identify the capability boundaries of the filtered learners. In stage III, we dynamically choose from the filtered ensemble and a conservative policy, guided by trained VAEs, to maintain profitability and mitigate risk with new market states. Through extensive experiments on crypto futures in a high-frequency trading environment with high fidelity and 5× leverage, we demonstrate that FineFT significantly outperforms 12 state-of-the-art baselines in 6 widely-used financial metrics, reducing risk by more than 40% while achieving superior profitability compared to the runner-up. Visualization of the selective update mechanism shows that different agents specialize in distinct market dynamics, and ablation studies certify routing with VAEs reduces maximum drawdown effectively, and selective update improves convergence and performance.

CCS Concepts

• **Computing methodologies** → *Artificial intelligence*; *Dynamic programming for Markov decision processes.*

Keywords

Ensemble Reinforcement Learning, Selective Update, High-frequency Future Trading, Variational Autoencoder

ACM Reference Format:
Molei Qin, Xinyu Cai, Yewen Li, Haochong Xia, Chuqiao Zong, Shuo Sun, Xinrun Wang, and Bo An. 2026. FineFT: Efficient and Risk-Aware Ensemble Reinforcement Learning for Futures Trading. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3770854.3780187

1 Introduction

Futures trading, as standardized financial contracts obligating the purchase or sale of an asset at a predetermined price and date, accounts for over 60% of the trading volume in the cryptocurrency market [1]. This dominance is driven by its flexibility to take long or short positions and its high leverage, which attract risk-seeking traders and foster a highly liquid market. With a market capacity of more than 60 trillion dollars, the cryptocurrency futures market represents a crucial domain for quantitative trading.

As quantitative trading evolves, deep learning (DL) has become popular for modeling complex dependencies using large-scale historical data. In high-frequency trading, reinforcement learning (RL)

*Corresponding authors.

KDD '26, August 09–13, 2026, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2258-5/2026/08
https://doi.org/10.1145/3770854.3780187

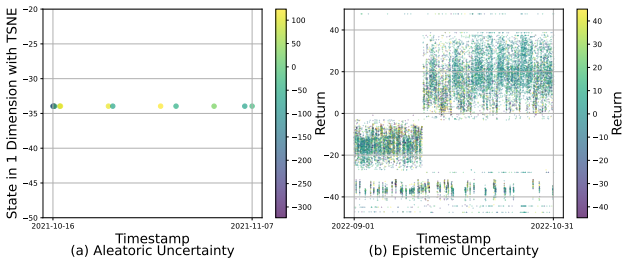

Figure 1: Two challenges RL faces in future trading. The X-axis is the timestamp; the y-axis represents the 1D t-SNE [22] embedding of market state (technical indicators) y_t, and the colour represents the future 5-min return rate.

stands out among DL approaches for optimizing sequential decision-making processes. Unlike supervised learning (SL), which assumes independent and identically distributed (IID) data, RL can handle scenarios where changes in current positions affect trading costs in subsequent decisions due to transaction costs. Although RL has achieved significant success in various quantitative trading tasks and markets [17, 23, 27, 29], most studies focus on low-leverage markets such as stock trading, and are less applicable to the high-leverage futures market due to 2 challenges as shown in Figure 1: 1) High leverage (e.g., 5×) amplifies the inherent stochasticity [21] of financial markets, meaning that even in similar states, identical actions can lead to vastly different rewards and subsequent states. This phenomenon is illustrated in Figure 1 (a), where the future price movements, represented by the color coding, show significant variability even for nearly identical market state values along the y-axis (1-dimensional market state y_t) at different timestamps (x-axis). Such differences highlight the aleatoric uncertainty inherent in financial data, which undermines the convergence and stability of RL algorithms, reduces data efficiency, and can lead to performance decay [19]. 2) Previous works have overlooked the importance of RL agents' self-awareness of their capability boundaries, i.e., the market states it can handle. Due to RL's tendency to overfit the training dataset, the policy becomes unreliable when confronted with unseen market states during training, exposing it to substantial capital loss [12]. For example, a black swan event in late 2022, triggered by the US Federal Reserve's aggressive interest rate hikes, caused significant shifts in market states. As shown in Figure 1 (b), a cluster near -20 in the 1-dimensional state representation suddenly jumped to 20. Such dramatic changes increase the risk of liquidation for RL agents which lack awareness of capability boundaries.

To tackle these challenges, we propose an e**F**ficient and r**I**sk-aware e**N**semble r**E**inforcement learning for **F**utures **T**rading (FineFT) In stage I, we initiate N deep Q-networks [16] (DQN) as ensemble learners and pre-train them with transitions from demonstrations. Then, an ensemble temporal difference (TD) error matrix, i.e., TD errors across the learners for each transition, is computed. This matrix selectively updates the learner with the smallest TD error with its neighbors, enhancing training stability, performance, and convergence. In Stage II, we back-test each ensemble element on various market dynamics and then filter the ensemble based on their profitability on each dynamic to reduce the size of the strategy pool. Then, variational autoencoders (VAEs) for state representations from each market dynamic are also trained to capture the capability boundary for each learner in the ensemble and detect out-of-distribution (OOD) state representations during the test phase. In stage III, guided by VAE losses of a rolling window of recent market states representations, we dynamically select the optimal agent from the filtered ensemble for in-distribution (ID) market states to maximize profit while using a conservative policy for OOD market states to mitigate risk. The main contributions are 4-fold:

- We introduce selective updates with neighbors, which enables agents to segment non-stationary environments into stationary sub-environments and learn optimal policies for each, significantly improving training convergence and overall performance.
- To the best of our knowledge, FineFT is the first to use VAEs for out-of-distribution (OOD) detection in RL for quantitative trading to reduce risks from unseen market state representations.
- Through extensive experiments on 4 Crypto with diverse market dynamics, evaluated at a minute-level scale, FineFT demonstrates high profitability, significantly reduces risk, and outperforms 12 state-of-the-art baseline methods. The ablation studies show that the selective update segments the market into meaningful financial dynamics reduces the convergence steps, and boosts the performance of the ensemble learners. VAEs effectively route learners and significantly reduce the maximum drawdown.
- We develop a high-fidelity trading environment with adjustable leverage, transaction costs, slippage and funding fees. The code is at https://github.com/qinmoelei/FineFT_code_space.

2 Related Works

We first introduce technical analysis and traditional routing methods, followed by the application of reinforcement learning (RL) in quantitative trading, with a focus on their respective limitations.

2.1 Technical Analysis & Traditional Routing

Technical analysis has been a fundamental tool in financial markets, relying on historical data to predict price movements [8]. For instance, imbalance volume (IV) [6], indicating current buying and selling pressures in the limit order book, is useful for short-term price movement prediction, while moving average convergence divergence (MACD) [11] which shows price trends in multiple time windows, is commonly used in a single-dynamic market. Routing methods are commonly used in Fintech to combine results from multiple predictors, allowing for more robust decision-making. These methods dynamically select the best-performing model based on historical performance. For instance, the Winnow algorithm [25] adjusts weights iteratively, giving more importance to predictors with better performance and improving decision accuracy.

However, technical analysis often experiences significant performance volatility in non-stationary markets like Crypto, which has been repeatedly criticized in recent studies [14, 17]. Moreover, because Winnow relies on historical performance to select models, it tends to perform poorly in sudden trend-reversal situations.

2.2 RL for Quantitative Trading

RL has demonstrated stellar performance under multiple quantitative trading tasks. EHFT [17] and MHFT [29] use a hierarchical

framework and optimal action value supervisor to train efficient RL on multiple dynamics for high-frequency trading. DRA [4] utilizes LSTM to capture trends with different time window lengths and PPO [18] in high-frequency trading. CRP [28] incorporates random perturbation to stabilize the training of a convolutional DQN [16]. RAQR [3] trains a QRDQN to handle the aleatoric uncertainty in the RL and utilizes a risk-averse policy to avoid the worst case. EDQN [5] utilizes different DQNs to handle different markets. SUNRISE [13] views the deviation of learners as the aleatoric uncertainty and decreases the learning rate of the uncertain samples.

However, these studies focus on non-leveraged markets, overlooking risk management and convergence issues. Consequently, they lack sensitivity to black swan events, rendering them unreliable when applied directly to leveraged futures in Crypto.

3 Problem Formulation

This section first presents several finance concepts used to simulate the trading process and then models this futures trading process as a dynamic parametric Markov decision process (DP-MDP).

3.1 Financial Foundations for Futures Trading

We first introduce some fundamental financial concepts used to describe state, reward, and action in the DP-MDP framework and present the objective of futures trading using these concepts.

Limit Order book (LOB) refers to unfilled orders aggregated by price and side, reflecting the micro-structure [15]. An m-level LOB at time t is denoted as $B_t = (p_t^{b_1}, p_t^{a_1}, q_t^{b_1}, q_t^{a_1}, ..., p_t^{b_m}, p_t^{a_m}, q_t^{b_m}, q_t^{a_m})$, where $p_t^{b_i}$ ($p_t^{a_i}$) is level i's bid (ask) price, $q_t^{b_i}$ ($q_t^{a_i}$) is the quantity.

Trades refers to executed orders, denoted as $T_{\tau_0:\tau_t}$ from time τ_0 to time τ_t, where an executed order $T_t = (p_t, q_t, c_t)$ at time t is defined by the price p_t, quantity q_t and side c_t respectively.

Mark Price refers to the reference price used to calculate the fair value of the holding position. It is denoted as M_t at timestamp t.

Technical Indicators indicate features calculated by a function of the past trades and LOB to predict future price movements, denoted as $y_t = \phi(B_t, M_t, ..., B_{t-h}, M_{t-h}, T_{t-h:t})$, where ϕ is a function.

Position is the amount of asset a trader holds, denoted as H_t at time t. Relative to 0, it indicates having a long or short position.

Leverage refers to the ratio of the value of the wanted asset to the capital required (margin), denoted as L_t at time t, and it magnifies the maximum position a trader can hold under the same capital.

Market Order Loss refers to the loss caused by the difference between the executed price and the mark price when executing market orders. It is denoted as O_t and calculated as $O_t = c_t(\sum_i (p_t^{c_i} \times \min(q_t^{c_i}, \Delta_{i-1})) \times (1+\kappa) - QM_t)$, where c_t is 1/ask for buying and -1/bid for selling, $p_t^{c_i}, q_t^{c_i}$ refer to the price and quantity in level i LOB B_t, Q is the total trading quantity, Δ_{i-1} is the remaining quantity after level i in B_t, κ refers to the commission rate.

Funding fee refers to interests required to borrow capital for leveraging a position. Denoted as $F_{ft} = F_{rt} \times H_t$, it is determined by a funding rate F_{rt}, which is given by the exchange and determined by the price difference between spot and futures, and the position.

Margin Balance V_t refers to the sum of cash (wallet balance) and position value, where the realized PnL, such as transaction cost or funding fee, is directly deducted from the cash, and the unrealized PnL, such as position value increments due to non-zero position and mark price fluctuations, is included in the total position value.

Our Objective is to maximize the margin balance via market orders with leverage on a single asset at a minute-level time scale.

3.2 Dynamic Parametric MDP Framework

In this subsection, we formulate futures trading as a DP-MDP [24]. A DP-MDP is defined by the tuple: (S, A, P, r, γ, T,Z,P_z), where S is the state space and A is the action space. Z is the dynamic space, which determines the transition function $P : S \times Z \times A \times S \to [0, 1]$. Z also has its transition function $P_z : Z \times Z \to [0, 1]$, where the dynamics evolve much slower than state transitions. This indicates that within a short time horizon, the experienced transitions can be viewed as resulting from a static MDP. $r : S \times A \times S \to R$ is the reward function, $\gamma \in (0, 1]$ is the discount factor and T is the time horizon. In a DP-MDP, the agent receives the current state $s_t \in S$ from the environment, performs an action $a_t \in A$, and moves to the next state $s_{t+1} \in S$ with a reward r_t, but the current dynamic z is not explicitly revealed to the agent. An agent's policy is defined by π_θ: $S \times A \to [0, 1]$, which is parameterized by θ. A router's policy is π_{θ_r}: $S \times A \to Z$, parameterized by θ_r. Our goal is to learn a set of agent policies $\Pi_A = (\pi_{\theta_1}, \cdots, \pi_{\theta_m})$, and a router π_{θ_r} to composite an optimal strategy $\pi^* = \arg\max_{A,\theta_r} E_{\Pi_A,\pi_{\theta_r}}[\sum_{t=0}^{T} \gamma^t r_t | S_0, Z_0]$, where S_0 is the initial state and Z_0 is the initial dynamic.

State S_t consists of 2 parts: market state, consisting of around 300 technical indicators y_t and personal state, which is the position H_t.

Dynamic Z_t refers to the dependence of market state and price movements, which cannot be reflected in the current market state.

Reward r_t denotes margin balance change, including valuation and order loss [17, 29], and is computed in Equation 1.

$$r_t = V_{t+1} - V_t = H_t(M_{t+1} - M_t) - O_t \tag{1}$$

Action a_t at time t refers to the target position with corresponding leverage choice from a finite pre-defined pool $A = \{-H_{l_1}, \cdots, H_{l_n}\}$, where $\{-H, \cdots, H\}$ represents the position pool and $\{l_1, \cdots, l_n\}$ is the leverage pool. We conduct the market order instantly to fill the gap between the current position and leverage and the target.

4 Method

In this section, we demonstrate three stages of FineFT as shown in Figure 2. In stage I, we selectively update the ensemble deep Q-networks (EDQN) based on the ensemble TD errors to enhance training efficiency. In stage II, we filter the ensemble based on profitability under various dynamics to reduce the strategy pool size and train VAEs for each dynamic to identify the filtered learners' capability boundaries regarding market state representation. In stage III, we dynamically choose from the filtered ensemble and a conservative policy, guided by VAEs' losses, to ensure stable performance in the non-stationary environment and mitigate risk when confronting unseen market state representations.

4.1 Efficient Ensemble with Selective Update

While a mixture of experts (MoE) has shown exceptional results in quantitative trading tasks [21], it comes with a significantly increased computational burden. Unlike previous approaches aiming to improve each learner's training efficiency [17], our method first

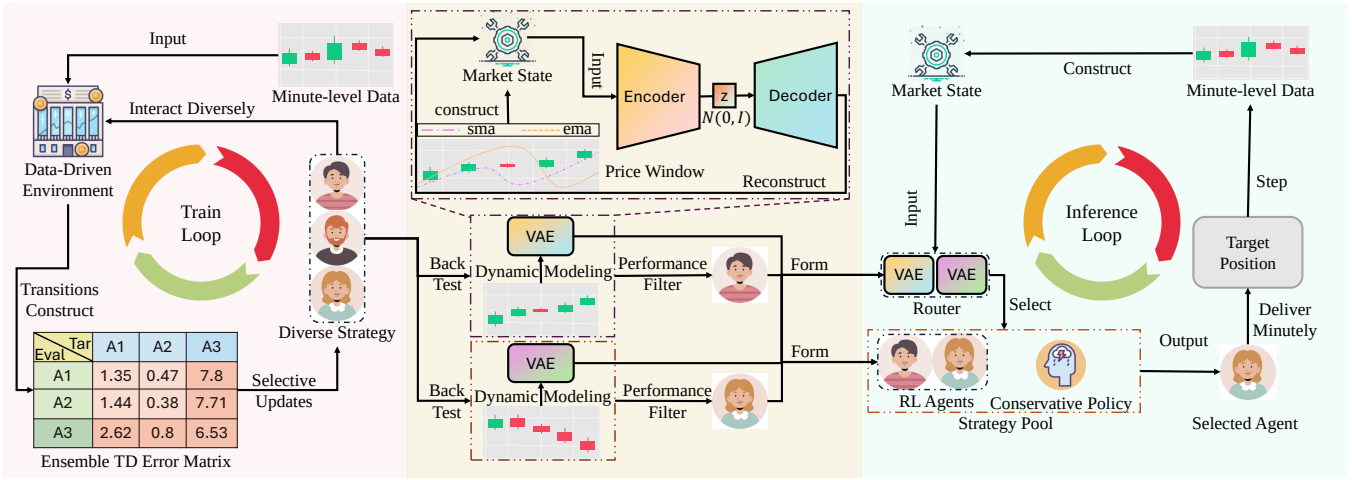

Figure 2: The overview of FineFT. First, we compute ensemble TD errors for selective updates to enhance efficiency. Then, we filter the ensemble based on performance across different dynamics and train VAEs for each dynamic to identify the ensemble's capability boundaries. Finally, VAEs route the ensemble and a conservative policy for stable performance and risk reduction.

Algorithm 1 Constrained Optimal Action Value
(Line 4: Dynamic Programming with Eq 1; Line 6: Constraint Masking)

Input: LOB Series B, Markprice Series M with Length N, Action Space A, Mask Punishment P, Capital C.
Output: Tabular Q^* of Optimal Action Value at Time t, Capital C, Position p, and Action a.

```
Initialize Q*[N, |A|, |A|] and all elements 0.
for t ← N − 1 to 1 do
    for p, a ← 1 to |A| do
        Q*[t, p, a] ← max_{a'} Q*[t + 1, a, a'] + r_t      // Future LOB
        if V_t ≤ C then
            Q*[t, p, a] ← Q*[t, p, a] − P                  // Apply Masking
        end if
    end for                                                // Rollout All Position-Action Pairs
end for                                                    // Rollout Each Timestamp
return Q*
```

equally pre-trains each learner to establish common knowledge, then employs a selective update strategy that prioritizes updates for the learner with the best estimation, i.e., the smallest TD error, of the current dynamic transitions, along with its neighbors to reduce the computational cost. Our goal in this stage is to train an ensemble DQN where each individual agent within the ensemble learns an optimal policy tailored to a specific dynamic Z in the DP-MDP.

First, each learner in the ensemble is initialized with the identical multilayer perception (MLP) structure but a different random seed. Then, we individually assign a unique index from 1 to N to each learner, where N is the number of learners, before equally pre-training the ensemble and selectively updating with neighbors.

Pretrain with Demonstration. In real trading companies, most new traders are trained together before trading separately. Inspired by this, we first train all the agents with the same transitions equally before the selective update. Inspired by [17], a constrained optimal action value is constructed as shown in Algorithm 1 to create demonstration transitions as an optimal actor by choosing the action with the largest action value given the state and, as an optimal value supervisor, to help update the EDQN. The computational complexity of this process is $O(T)$, where T denotes the number of timestamps. We further utilize 3 additional policies to generate more diverse transitions for better generalization, which include 1) emptying the position, 2) longing the maximum position, and 3) shorting the maximum position. We update each agent using Huber TD error to avoid the influence from outliers and the optimal value supervisor to boost the learning of advantage value across different actions for the same state as shown in Equation (2),

$$L(\theta_i) = L_{td_i} + \alpha KL(Q(s,\cdot;\theta_i)||Q^*(s,\cdot)) \tag{2}$$

where $KL(Q(s,\cdot;\theta_i)||Q^*(s,\cdot))$ is the KL-divergence of the agent with index i's action value and the optimal action value. L_{td_i} is the Huber temporal difference error and is defined as

$$L_{td_i} = \mathcal{H}\left(r + \gamma \max Q(s,\cdot;\theta_i') - Q(s,a;\theta_i)\right) \tag{3}$$

γ is the decay coefficient, and $\mathcal{H}(x)$ is the Huber loss [9], preventing influence from outliers by using mean absolute error for large error and mean square error for small scale errors. It is defined as

$$\mathcal{H}(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \le \delta \\ \delta(|x| - \frac{1}{2}\delta) & \text{otherwise} \end{cases}. \tag{4}$$

During pre-training, each learner is updated equally regarding each transition, and the total loss for all learners could be described as

$$L(E)_{prt} = \sum_{i}^{N} L(\theta_i). \tag{5}$$

Pretraining with demonstration enables the ensemble to rapidly acquire fundamental trading principles (e.g., not trading too fast or recklessly), thereby expediting the convergence process.

Algorithm 2 Construction of Weight Matrix
(Line 3-5: Diagonal Element; Line 6-9: Non-Diagonal Element)

Input: ETD Error Matrix $\mathcal{L}$, Neighbor Number m
Output: $\mathcal{W}$ indicating the Weights assigned to Learners

Initialize $\mathcal{W}$ with shape (N, N) and all elements 0
$i^* \leftarrow \arg\min_i L_{ii}$; $i_{\min} \leftarrow \max(i^* - m, 1)$;
$i_{\max} \leftarrow \min(i^* + m, N)$ *// Confirm Update Agent Index Range*
for $i = i_{\min}$ to $i_{\max}$ **do**
  $\mathcal{W}_{ii} \leftarrow 1 - \frac{|i-i^*|}{i_{\max}-i_{\min}}$ *// Diagonal Element Decay*
end for
for $i, j \leftarrow 1$ to N **do**
  if $i \neq j$ **then**
    $\mathcal{W}_{ij} \leftarrow \min(\mathcal{W}_{ii}, \mathcal{W}_{jj}) \times \left(1 - \frac{|i-j|}{i_{\max}-i_{\min}}\right)$
  end if *// Non-Diagonal Decay*
end for

Selective Update with Neighbors. Though various dynamics share some fundamental trading principles, the most profitable strategies under different market dynamics often conflict with each other, highlighting the importance of a diverse strategy pool. Previous works [17, 21] use preference sampling to update the learners independently for the ensemble's diversity, causing a huge computational burden. To address this problem, we demonstrate a more efficient way of training ensembles: assigning transitions with different weights to different agents. Specifically, we first collect enough transitions generated by ϵ-greedy policy from each learner and the optimal actor from the optimal action value. Then, for each transition, we assign larger weights to the learner with a more accurate Q-value estimation and its neighbors (learners whose index is close to the accurate learner's index), i.e., we prioritize updating the agents already with strong performance and their neighbors in the given dynamic while keeping those with poor performance untouched, because those learners with poor performance under this given dynamic may perform well under a different dynamic and using the experience under the given dynamic will poison their performance on the dynamic where they excel. This method is effective because the agent's updates are tied to transitions from a specific dynamic, preventing random updates and ensuring more focused learning. When the agent is updated with transitions from a particular dynamic, its Q-value estimates improve within that context. As these estimates become more accurate, the TD error from the same dynamic decreases, making it more likely that future transitions from the same dynamic will be used for subsequent updates for the same agent. This creates a positive feedback loop, progressively reinforcing the agent's specialization within that dynamic.

To determine the specific weights assigned to different learners for each transition, an ensemble temporal difference (ETD) error matrix for each transition is firstly calculated in Equation 6.

$$L = \begin{pmatrix} L_{11} & \cdots & L_{1N} \\ \vdots & \ddots & \vdots \\ L_{N1} & \cdots & L_{NN} \end{pmatrix} \tag{6}$$

L_{ij} is the Huber TD error across learner i and j, defined as Equation (7), where L_{ii} is just the Huber TD error of agent i.

$$L_{ij} = \mathcal{H}\left(r + \gamma \max Q(s', \cdot; \theta'_j) - Q(s, a; \theta_i)\right) \tag{7}$$

Then, we compute the weight matrix, whose shape is the same as the ETD error matrix, using Algorithm 2, where we first find out the index of the learner with the least Huber TD error i^*, then expand the index range according to the predefined neighbor size m. Then, we compute the weights for the diagonal elements, where i^{*th} element's weight is 1, and the rest of the weights decay to $\frac{1}{2}$ for the boundary element i_{min} and i_{max}. Those indices not covered in the range remain 0. Finally, we calculate the non-diagonal element as the minimum diagonal value at its row and column times a decay based on the absolute value of the subtraction between its row and column indices. After constructing the weight matrix for the given transition (s, a, r, s') with $O(N + m^2)$ complexity, where N is the ensemble size and m is the neighbor size, we further take its diagonal elements to form a weight vector $\mathbf{d}_{\mathcal{W}} = \begin{pmatrix} \mathcal{W}_{11} & \cdots & \mathcal{W}_{NN} \end{pmatrix}$ as a weight for each learner with the optimal supervisor. The total loss for selective update given the transition (s, a, r, s') is defined as

$$L = \sum_{i=1}^{n} \sum_{j=1}^{n} \mathcal{W}_{ij} L_{ij} + \mathbf{d}_{\mathcal{W}} O_{\mathrm{KL}}^T \tag{8}$$

where $O_{\mathrm{KL}} = \begin{pmatrix} \mathrm{KL}_1 & \cdots & \mathrm{KL}_N \end{pmatrix}$ is the optimal supervisor loss vector and $\mathrm{KL}_i = \mathrm{KL}(Q(s, \cdot; \theta_i) || Q^*(s, \cdot))$ is the optimal supervisor's loss for agent i. The final loss could be viewed as a weighted version of Equation (5) with some cross-learner regulation for similar dynamics, improving the training efficiency and performance upper bond by sharing transitions among learners for similar dynamics and preventing experience poisoning from extremely different dynamics.

4.2 Ensemble Filter & Boundary Identification

To effectively route the ensemble during the testing phase, it is crucial to identify the market conditions under which each learner performs well or poorly. While previous research has back-tested low-level RL agents under different dynamics to minimize the strategy pool size for improving routing efficiency, we are the first to propose not only filtering the ensemble based on their profitability under various dynamics to reduce the ensemble size but also training VAEs for each market dynamic to capture market state representations, thereby gaining a deeper understanding of the capability boundaries within which learners perform well.

Ensemble Filter based on Profitability Performance. After training an ensemble with diverse strategies, it is essential to identify the specific market dynamics for each strategy best suited. Many empirical methods divide a long market time series into various dynamics. Here, we follow the market dynamics division based on slope [17], where the sequence is first passed through a low-pass filter to remove high-frequency noise, and local extrema points are identified to segment the sequence into smaller chunks. If the slope differences between adjacent chunks are insignificant, these chunks are merged, and we continue the merging process until there are no chunks left to be merged. By applying this algorithm, we can obtain m distinct market dynamics, where each dynamic comprises multiple segments of continuous time series data. We

Algorithm 3 Risk-Aware Heuristic Routing
(Line 1-4: EMA Reference Score Computing; Line 5-9: Routing)

Input: Selected Strategies $\Pi = (\pi^1 \cdots \pi^m)$, VAEs M^v, Reference Scores R, State Window Y_T^u, time decay γ, reject threshold τ
Output: Current Strategy π

```
for Dynamic i from 1 to m do
    Obtain state score R_t^i from VAE M^{v_i}, Y_T^u and R^i
    Compute the EMA using Eq. 10        // Compute EMA Score
end for
Compute i = argmax_i Re_T^{iu}
if h ≤ τ then
    Return conservative policy π^c      // Choose Conservative strategy
else
    Return π^i                          // Choose Heuristic Strategy
end if
```

then back-test each learner in our ensemble with different initial positions. We then pick the agents that gain the highest average profitability across all initial positions on each dynamic to form an efficient, diverse, profitable strategy pool fit for routing.

Boundary Identification for State Representation. Picking profitable learners under different market dynamics divided empirically is insufficient for safe routing. We also should know the capability boundary of each learner within which it can perform optimally. To better understand these dynamics, we train VAEs [10] for the market state representations from each market dynamic. A VAE consists of an encoder compressing the input into a low-dimensional distribution, regularized with a normal distribution, and a decoder reconstructing the original input based on the sample drawn from the low-dimensional distribution. Unlike standard autoencoders, VAEs provide a probabilistic measure of reconstruction that accounts for the variability in data distribution, making them a more principled and objective anomaly detection method and, therefore, is often used in OOD detection. Here, for a total of m market dynamics, we first initialise m VAEs with identical MLP structures and random seeds. Then, for each market dynamic z_i, the market states y_t for t belonging to dynamic i are collected to make the dataset $Y_i = \begin{pmatrix} y_{i1} & \cdots & y_{in_i} \end{pmatrix}$ to update VAE i. The loss function for each VAE i is defined as

$$L^i(y \in Y_i) = N_{LL}(\mu_d, 0.5\log(\sigma_d^2), y) + KLD(\mu, \log(\sigma^2)), \quad (9)$$

where the negative likelihood loss $N_{LL}(\mu, \log(\sigma^2), y) = \frac{1}{2}\left(\frac{(y-\mu)^2}{\exp(\log\sigma)^2}\right) + \log\sigma + \frac{1}{2}\log(2\pi)$ refers to the reconstruction loss and $KLD(\mu, \log(\sigma^2)) = KL(N(\mu, \sigma^2)|N(0, I))$ refers to normal distribution regulation. After training 2000 epochs, we record $-L^i(y)$ of each VAE M^{v_i} for every market from the corresponding dynamic z_i, denoted as reference scores $R^i = \begin{pmatrix} -L^i(y_{i1}) & \cdots & -L^i(y_{in}) \end{pmatrix}$. So, in conclusion, we have a set of filtered policies $\Pi = \begin{pmatrix} \pi^1 & \cdots & \pi^m \end{pmatrix}$, a set of trained VAEs $M^v = \begin{pmatrix} M^{v_1} & \cdots & M^{v_m} \end{pmatrix}$, and their corresponding reference scores set $R = \begin{pmatrix} R^1 & \cdots & R^m \end{pmatrix}$, which will be used in routing.

4.3 Risk-Aware Heuristic Routing

In Stage III, we construct a conservative policy for our strategy pool to handle unseen markets in train or valid datasets. Then, we design a risk-aware heuristic routing mechanism to choose the proper agent to gain stable performance under a rapidly changing dynamic and avoid risks in the new market state representations.

Conservative Policy Design. We need a conservative strategy to handle market state representations we have never encountered during the training or valid phase. We design the conservative policy as maintaining the position if the maximum drawdown of this single trade is not over 5% or closing the position otherwise. This conservative policy only focuses on closing the position instead of opening positions, reducing the frequency of position changes. It also ensures that the position is closed to mitigate further losses once a stop-loss is triggered under unseen market representations.

Routing with VAEs' Losses. Though a few previous works [17, 29] have constructed hierarchical MDP and let the high-level agent serve as the router, it is time-consuming to train such a meta-policy because of RL's poor data efficiency. Therefore, we utilize a heuristic method based on scores from recent market states and trained VAEs from stage II to determine which agent to conduct trading. Given a sequence of recent market states $Y_T^u = \begin{pmatrix} y_{T-u}, \cdots, y_T \end{pmatrix}$, each market state's score R_t^i is computed via empirical cumulative distribution function (ecdf) from reference scores R^i, followed by an EMA smoothing, described in Equation (10),

$$R_t^i = F_{R^i}(-L^i(y_t)), Re_{T_t}^{iu} = \gamma Re_{T_{t-1}}^{iu} + R_t^{iu}, Re_T^{iu} = Re_{T_T}^{iu} \quad (10)$$

where $Re_{T_{T-u}}^{iu} = R_{T-u}^i$ for each market dynamic i. We pick the dynamic $i = argmax_i Re_T^{iu}$ and compare it with a risk threshold τ. If Re_T^{iu} is smaller than τ, we are unfamiliar with the recent market representations and choose the conservative policy for safe consideration. Otherwise, we choose the learner corresponding to dynamic i to conduct trading, as concluded in Algorithm 3.

Table 1: Dataset statistics detailing market, dynamics, step size, and chronological period, where the dates are in the YY/MM/DD formats, 22/01/01 indicating Jan 1st of 2022.

Dataset	Test Dynamics	Size	From	To
BNB/USDT	Bear	210241	22/01/01	24/01/01
BTC/USDT	Bull	210241	22/01/01	24/01/01
DOT/USDT	Sideways	210241	22/01/01	24/01/01
ETH/USDT	Sideways	210241	22/01/01	24/01/01

5 Experiment

5.1 Experiments Setup

Datasets. To comprehensively evaluate the algorithm, testing is conducted on 4 most mainstream cryptocurrencies over a period exceeding 2 years, covering a variety of market conditions, including bull, bear, and volatile markets, as summarized in Table 1. For the dataset split, we use data from the last 6 months for testing, the penultimate 6 months for validation, and the remaining 1 year for training on all 4 datasets. We first train the EDQN on the training dataset, then segment and label the validation dataset for performance filter & boundary identification (VAE training) and tune the hyper-parameter (risk threshold τ, time decay γ, and window

Table 2: Performance comparison on 4 Crypto markets with 12 baselines, including 4 plain, 3 hierarchical, 2 ensemble, 1 distributional RL, and 2 rule-based methods. Pink, green, and blue results show the best, second-best, and third-best results.

		Profit	Risk-Adjusted Profit			Risk Metrics				Profit	Risk-Adjusted Profit			Risk Metrics	
Market	Model	TR(%)↑	ASR↑	ACR↑	ASoR↑	AVOL(%)↓	MDD(%)↓	Market	Model	TR(%)↑	ASR↑	ACR↑	ASoR↑	AVOL(%)↓	MDD(%)↓
BNB	DRA	-64.79	-1.51	-2.58	-13.43	6.53	72.88	BTC	DRA	21.93	0.94	2.12	4.90	6.90	58.28
	PPO	-59.58	-1.64	-2.59	-14.12	5.55	67.37		PPO	19.68	0.90	2.01	4.39	6.83	58.30
	CRP	-64.79	-1.43	-2.59	-12.98	6.92	72.88		CRP	19.61	0.85	1.79	4.62	6.44	58.30
	DQN	-55.85	-1.21	-2.14	-13.72	6.09	66.09		DQN	-82.13	-12.03	-4.94	-51.81	1.77	82.37
	MACD	-82.85	-3.04	-4.12	-22.67	6.28	88.70		MACD	-26.28	-0.17	-0.31	-9.44	5.91	60.17
	IV	-71.00	-1.59	-2.73	-14.31	6.59	73.21		IV	-17.26	0.01	0.02	-4.26	5.80	63.25
	EDQN	-64.79	-1.51	-2.58	-13.43	6.53	72.88		EDQN	25.11	0.98	2.15	5.88	6.80	59.28
	SUNRISE	-66.75	-1.74	-2.80	-14.45	6.27	74.32		SUNRISE	-7.91	0.37	0.76	-2.06	6.87	63.36
	RAQR	-68.88	-1.84	-3.10	-14.22	6.47	73.27		RAQR	2.84	0.57	1.10	0.71	5.19	51.19
	WINOW	86.65	2.01	5.99	31.71	5.46	34.93		WINOW	-34.03	-0.23	-0.56	-8.62	7.09	55.30
	EHFT	-69.32	-2.08	-3.26	-17.31	6.05	73.61		EHFT	33.44	1.42	2.73	10.78	4.29	42.61
	MHFT	-50.42	-0.81	-1.48	-9.10	6.47	67.95		MHFT	-25.85	-0.32	-0.57	-7.27	4.46	47.97
	FineFT	83.70	2.55	11.19	101.17	3.57	15.49		FineFT	76.90	2.13	7.44	39.51	4.29	23.51
DOT	DRA	-65.15	-1.63	-2.63	-15.21	6.20	73.36	ETH	DRA	-46.56	-0.77	-1.44	-10.86	6.18	63.20
	PPO	-65.65	-1.63	-2.64	-15.18	6.27	73.80		PPO	-46.08	-0.74	-1.39	-10.68	6.23	63.01
	CRP	-73.13	-4.12	-4.09	-27.08	3.93	75.58		CRP	-46.06	-0.95	-1.69	-11.36	5.88	62.96
	DQN	-64.89	-1.35	-2.40	-13.66	6.84	73.42		DQN	-73.15	-3.94	-3.88	-27.31	3.89	75.48
	MACD	26.07	1.10	2.72	6.24	7.74	60.01		MACD	-73.93	-2.78	-3.49	-21.52	5.30	80.85
	IV	-66.24	-1.67	-2.70	-15.32	6.26	73.80		IV	-50.53	-1.04	-1.82	-12.29	5.87	63.87
	EDQN	-65.65	-1.63	-2.64	-15.18	6.27	73.80		EDQN	-46.00	-0.73	-1.39	-10.65	6.23	62.96
	SUNRISE	-59.18	-1.02	-1.84	-12.51	6.94	73.60		SUNRISE	-81.22	-4.45	-4.57	-23.85	4.48	83.36
	RAQR	-4.45	-4.47	-1.76	-45.61	0.10	4.72		RAQR	-5.94	-4.10	-2.65	-13.75	0.21	6.08
	WINOW	-81.86	-4.17	-4.63	-26.72	4.85	83.35		WINOW	26.36	1.06	3.55	6.98	6.59	37.55
	EHFT	89.00	1.95	4.09	19.78	6.28	57.18		EHFT	48.72	1.49	3.33	10.66	5.65	48.33
	MHFT	220.44	3.48	10.47	65.02	5.10	32.44		MHFT	-75.16	-2.21	-3.26	-15.37	6.32	81.79
	FineFT	18.39	1.08	2.50	14.46	2.64	21.87		FineFT	39.05	1.74	5.78	14.45	2.96	16.96

length L) for heuristic routing on the validation dataset. The final model is tested in the test dataset.

Evaluation Metrics. We compare FineFT with 12 state-of-the-art baseline models using six widely-used financial metrics, as outlined in previous works [17, 20, 26, 29]. These metrics include one profit metric—total return rate (TR)—three risk-adjusted profit metrics: annual Sharpe ratio (ASR), annual Calmar ratio (CR), and annual Sortino ratio (ASoR), as well as two risk metrics: maximum drawdown (MDD) and annual volatility (AVOL).

Training Setup. All experiments were conducted on a server equipped with four NVIDIA RTX 4090 GPUs and an AMD Ryzen Threadripper PRO 5995WX CPU, with a total computation time of 6 hours. For a fair comparison, we applied the same hyperparameters used in FineFT (if included) to the baseline models. The transaction cost was set at 0.02%, aligned with Binance's regulations, and a leverage factor of 5 was utilized.

Baselines To evaluate FineFT comprehensively, we selected 12 baselines to compare with FineFT: 2 policy-based RL algorithms, PPO [18] and DRA [4], 2 value-based RL algorithms, DQN [16] and CRP [28], 2 widely used rule-based trading strategies, MACD [11] and IV [6], 2 ensemble RL algorithms: ensemble DQN (EDQN) [2] and SUNRISE [13], 1 distributional RL algorithm: RAQR [7], and 3 hierarchical RL algorithms: WINOW [25], EHFT [17] and MHFT [29].

Table 3: Convergence efficiency and performance analysis comparison of the efficient ensemble with selective updates. Pink and green demonstrate the best and second-best results.

Meth	CS↓	Dyn	RS↑	Meth	CS↓	Dyn	RS↑
EP	71712	1	2.86	ER	112320	1	2.86
		2	2.14			2	2.15
		3	0.50			3	0.48
		4	4.98			4	4.98
		5	9.14			5	9.14
FP	66528	1	45.29	FwoP	79488	1	37.34
		2	18.21			2	16.18
		3	4.77			3	6.39
		4	24.48			4	25.53
		5	40.23			5	46.17

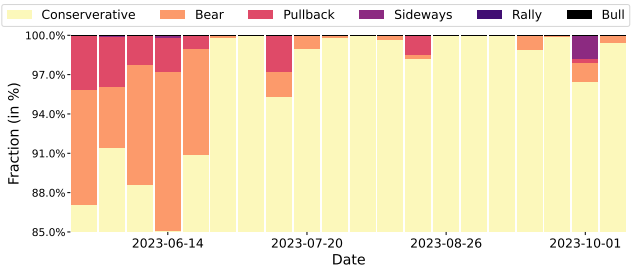

Figure 3: Routing result of VAEs along time during the test phase. The x-axis represents the timestamp. The y-axis represents the proportion of different market dynamics.

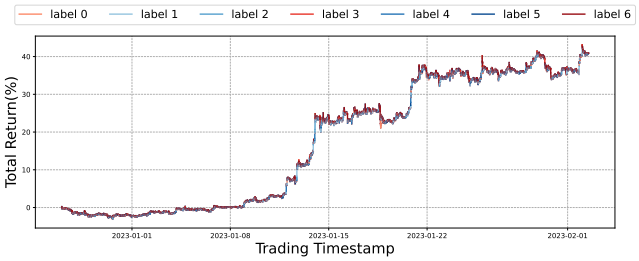

Figure 4: BTC Return Curve: Highlighting the relationship between updated agent indices and market dynamics, with the x-axis representing time, the y-axis showing cumulative returns, and colors indicating updated agent indices.

5.2 Comparison with Baselines

According to Table 2, our method achieves the lowest risk in all 4 datasets, the highest risk-adjusted profit in 3 datasets, and the highest profit in 2 datasets. Value-based methods (CRP and DQN) perform well when the market trend is stable and the gap between the validation and the test dataset is small. CRP performs better than DQN, demonstrating the importance of training stability in a high-stochastic environment. The policy-based methods (DRA and PPO) generally outperform DQN. However, they fail to surpass the Constant Rebalanced Portfolio (CRP), suggesting that policy-based approaches may converge prematurely to suboptimal policies in non-stationary environments, thereby failing to identify the global optimum. Yet the difference between the DRA and PPO is not evident; demonstrating LSTM does not help much in state representation under a stochastic environment. Rule-based methods (MACD and IV) rely heavily on the similarity between the validation dataset and test dataset since we pick the hyperparameter using the result from the validation dataset, and the overall performance is extremely sensitive to those hyperparameters. Ensemble RL methods (EDQN and SUNRISE) and distributional RL (RAQR) demonstrate stable performance while effectively mitigating risk due to reduced variance. Hierarchical RL (WINOW, EHFT and MHFT) achieve significantly higher profit performance compared to the aforementioned methods. Moreover, their performance is less dependent on the similarity between validation and test datasets, highlighting their robustness in diverse scenarios. However, these hierarchical methods exhibit high maximum drawdowns, attributed to a lack of clear understanding of their capability boundaries. FineFT retains its performance under familiar market states, profiting like the previous hierarchical RL methods, while it becomes conservative when it confronts unfamiliar market states and, therefore, avoids huge losses.

5.3 Ablation

In this section, we conduct ablation experiments to demonstrate the effectiveness of the efficient ensemble with selective updates and risk-aware heuristic routing on BTCUSDT.

The Effectiveness of Efficient Ensemble with Selective Update. To demonstrate the improved efficiency and performance in the training strategy pool given by the efficient ensemble with selective update, we conduct an ablation study on whether to use ensemble learning or pre-train on 5 dynamics. We utilize the convergence steps (CS) to demonstrate the training efficiency and reward sum (RS) to demonstrate the converged performance. Besides FineFT with pre-training (the original training method for strategy pool in FineFT, FP), there are other 3 baselines to compare the performance: 1) EHFT with prioritized episode selection (EP), where each agent is trained independently and does not share any transitions but data chunk for each agent's training is sampled with different preference based on market trend for each agent. 2) EHFT with random dataset selection (ER), where each agent is trained independently, and data is randomly sampled for every agent. 3) FineFT without pre-training (FwoP), where agents are not trained equally before selective updates. The result shows that FP and FwoP demonstrate the best converged-result while EP and FP converge within the fewest steps. Overall, the FineFT with pre-training (FP) demonstrates the best performance with the least number of steps required to converge.

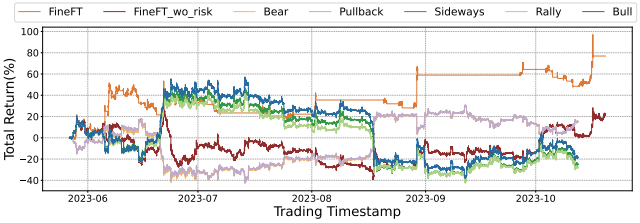

Figure 5: Risk-aware routing performance. The x-axis is the timestamp; the y-axis shows FineFT and subagents' returns.

The Effectiveness of Risk-Aware Heuristic Routing. We examine the effectiveness of the risk-aware routing mechanism by analyzing how the routing results evolve and the performance comparison among the FineFT, FineFT without the risk threshold (FineFT_wo_risk), and each agent from the strategy pool. Figure 3 shows how the routing result evolves. In most periods, the conservative policy dominates the routing outcome, and we can see an increase in the proportion of conservative policy as the timestamp is further away from the train and valid dataset. Figure 5 is the performance comparison of the routing. It shows that FineFT is the best, while FineFT_wo_risk beats any other single agent. Agents for a bear market and pullback market are very similar, making a profit in July when the price dropped, while the sideways, rally, and

bull market made profits from August to October when the price increased. The FineFT takes advantage of both sides and closes its position in a volatile dynamic, where market state representations are not met in the validation dataset to avoid potential loss.

5.4 Case Study

In this section, we first visualize the relationship between market dynamics and the corresponding agent being updated based on selective updates. Then, we present a scenario where the VAE model successfully detects a significant change in market conditions through its rejection mechanism. This rejection coincided with a fundamental event, leading to a sudden shift in price dynamics.

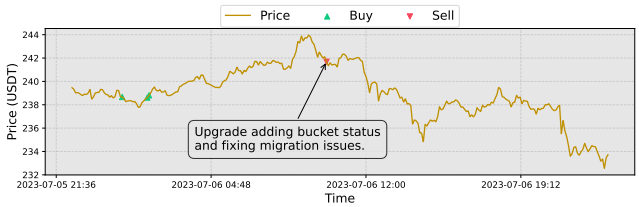

Figure 6: OOD detection with VAE. The X-axis represents the timestamp, and the y-axis is the price of BNB. The red and green symbol shows the buy and sell signal from FineFT.

Visualisation of the market dynamic and selected index. To validate the proposed selective update with the neighbour mechanism, a visualisation between the market dynamics and updated index is designed to demonstrate that agents do not learn randomly but instead specialise in handling distinct market dynamics. As shown in Figure 4, we utilize 7 agents for selective update with neighbours. Each shows its speciality in the dynamic: agents corresponding to labels 4 and 5 primarily update during sudden price surges, label 7 handles significant price drops, labels 0 and 1 focus on minor fluctuations, and labels 2 and 3 manage larger market movements. This clear alignment between agents and market dynamics validates the selective update with neighbors mechanism's ability to promote specialization and avoid random learning.

Detecting Market Anomalies with VAEs. Figure 6 illustrates using a VAE to handle unseen market state representations during test time, focusing on BNB's price movements in July 2023. Despite no explicit closing signals from the selected agent, FineFT proactively closed a position upon encountering unfamiliar states, indicated by a large mean VAE loss over a rolling window. This action avoided losses from subsequent price declines, coinciding with a macro event in BNB: an on-chain upgrade aimed at enhancing bucket status management, including introducing a new status classification and resolving previous migration-related issues.

6 Conclusion

This paper proposes FineFT, a novel three-stage ensemble RL approach for handling high stochasticity and risk for unseen markets in futures trading. First, an ETD error is computed to update the learner selectively to improve data efficiency and performance. Then, the ensemble is back-tested on various dynamics modelled by VAEs. Finally, we utilize risk-aware heuristic routing to avoid potential loss caused by epistemic uncertainty. Extensive experiments show FineFT's high profitability and strong risk management. Visualization and trading cases demonstrate that selective updates enable automated market segmentation, and the risk-aware routing mechanism effectively detects macroeconomic events. Ablation studies demonstrate their effectiveness in performance.

7 Acknowledgments

This research is supported by the Joint NTU-WeBank Research Centre on Fintech, Nanyang Technological University, Singapore. This research is also supported by the Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 1 grant (Proposal ID: 23-SIS-SMU-037). Shuo Sun is supported by the Guangzhou-HKUST(GZ) Joint Funding Program (No. 2024A03J0630).

References

[1] Saketh Aleti and Bruce Mizrach. 2021. Bitcoin spot and futures market microstructure. *Journal of Futures Markets* 41, 2 (2021), 194–225.
[2] Oron Anschel, Nir Baram, and Nahum Shimkin. 2017. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 176–185.
[3] Julian Bernhard, Stefan Pollok, and Alois Knoll. 2019. Addressing inherent uncertainty: Risk-sensitive behavior generation for automated driving using distributional reinforcement learning. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2148–2155.
[4] Antonio Briola, Jeremy Turiel, Riccardo Marcaccioli, and Tomaso Aste. 2021. Deep reinforcement learning for active high frequency trading. *arXiv preprint arXiv:2101.07107* (2021).
[5] Salvatore Carta, Anselmo Ferreira, Alessandro Sebastian Podda, Diego Reforgiato Recupero, and Antonio Sanna. 2021. Multi-DQN: An ensemble of Deep Q-learning Agents for Stock Market Forecasting. *Expert Systems with pplications* 164 (2021), 113820.
[6] Tarun Chordia, Richard Roll, and Avanidhar Subrahmanyam. 2002. Order imbalance, liquidity, and market returns. *Journal of Financial Economics* 65, 1 (2002), 111–130.
[7] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. 2018. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.
[8] Robert D Edwards, John Magee, and WH Charles Bassetti. 2018. *Technical Analysis of Stock Trends*. CRC press.
[9] Peter J Huber. 1992. Robust estimation of a location parameter. In *Breakthroughs in Statistics: Methodology and Distribution*. Springer, 492–518.
[10] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).
[11] Thomas Krug, Jürgen Dobaj, and Georg Macher. 2022. Enforcing network safety-margins in industrial process control using MACD indicators. In *European Conference on Software Process Improvement*. 401–413.
[12] Charline Le Lan, Stephen Tu, Adam Oberman, Rishabh Agarwal, and Marc G Bellemare. 2022. On the generalization of representations in reinforcement learning. *arXiv preprint arXiv:2203.00543* (2022).
[13] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. 2021. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*. 6131–6141.
[14] Yang Liu, Qi Liu, Hongke Zhao, Zhen Pan, and Chuanren Liu. 2020. Adaptive quantitative trading: An imitative deep reinforcement learning approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2128–2135.
[15] Ananth Madhavan. 2000. Market microstructure: A survey. *Journal of Financial Markets* 3, 3 (2000), 205–258.
[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.
[17] Molei Qin, Shuo Sun, Wentao Zhang, Haochong Xia, Xinrun Wang, and Bo An. 2024. Earnhft: Efficient hierarchical reinforcement learning for high frequency trading. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 14669–14676.
[18] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[19] Freek Stulp, Jonas Buchli, Alice Ellmer, Michael Mistry, Evangelos A Theodorou, and Stefan Schaal. 2012. Model-free reinforcement learning of impedance control in stochastic environments. *IEEE Transactions on Autonomous Mental Development* 4, 4 (2012), 330–341.

[20] Shuo Sun, Molei Qin, Wentao Zhang, Haochong Xia, Chuqiao Zong, Jie Ying, Yonggang Xie, Lingxuan Zhao, Xinrun Wang, and Bo An. 2024. TradeMaster: A holistic quantitative trading platform empowered by reinforcement learning. In *Advances in Neural Information Processing Systems*.

[21] Shuo Sun, Xinrun Wang, Wanqi Xue, Xiaoxuan Lou, and Bo An. 2023. Mastering stock markets with efficient mixture of diversified trading experts. *Proceedings of the 29th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2023).

[22] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, 11 (2008).

[23] Haochong Xia, Shuo Sun, Xinrun Wang, and Bo An. 2024. Market-GAN: Adding Control to Financial Market Data Generation with Semantic Context. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 15996–16004.
vfill

[24] Annie Xie. 2020. Deep reinforcement learning amidst lifelong non-stationarity. *arXiv preprint arXiv:2006.10701* (2020).

[25] Tong Zhang. 2000. Regularized winnow methods. *Advances in Neural Information Processing Systems* 13 (2000).

[26] Wentao Zhang, Lingxuan Zhao, Haochong Xia, Shuo Sun, Jiaze Sun, Molei Qin, Xinyi Li, Yuqing Zhao, Yilei Zhao, Xinyu Cai, et al. 2024. FinAgent: A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist. *arXiv preprint arXiv:2402.18485* (2024).

[27] Wentao Zhang, Yilei Zhao, Shuo Sun, Jie Ying, Yonggang Xie, Zitao Song, Xinrun Wang, and Bo An. 2024. Reinforcement learning with maskable stock representation for portfolio management in customizable stock pools. In *The Web Conference 2024*.

[28] Tian Zhu and Wei Zhu. 2022. Quantitative trading through random perturbation Q-network with nonlinear transaction costs. *Stats* 5, 2 (2022), 546–560.

[29] Chuqiao Zong, Chaojie Wang, Molei Qin, Lei Feng, Xinrun Wang, and Bo An. 2024. MacroHFT: Memory augmented context-aware reinforcement learning on high frequency trading. *arXiv preprint arXiv:2406.14537* (2024).