Hierarchical PPO-Based Deep Reinforcement Learning for Multi-Timescale Financial Decision-Making

Group 05

Xiaoying Mo 223020261

Yuyang Shen, 224020343

Junfeng Chen 224020329

Ruopu Li 224020253

Hang Zeng 223040119

Abstract

Deep Reinforcement Learning (DRL) models have been shown outstanding in stock trading tasks, which could perform well in long-term trading environment measured by trading indexes like total return, max drawdown and sharp ratio, etc. This research tries directly to adopt a DRL model-PPO in short-term stock return prediction tasks, and innovatively builds a Hierarchical-PPO (HPPO) structure which can capture both short-term return and long-term price trend, thus enhancing the prediction power. In the experiment, the single PPO model demonstrates an acceptable predictive ability. However, the Hierarchical-PPO (HPPO) significantly enhances the return prediction performance of PPO. The effectiveness of HPPO is validated through multiple historical stock datasets, showing substantial improvements in predictive accuracy.

Key Word: PPO, Hierarchical, Stock Return, IC, RIC, RMSE

I Introduction

1.1 Review of Preliminary Research

In the field of quantitative trading within securities markets, artificial intelligence (AI) demonstrates significant core advantages. AI excels in processing high-dimensional data, encompassing the integration of market microstructure signals alongside unstructured data sources such as news articles and social media content. It also achieves dynamic optimization and risk management; for example, reinforcement learning (RL) systems are capable of dynamically rebalancing portfolios with adaptive regularization techniques to reduce the risk of overfitting. Additionally, AI integrates diverse methodologies, exemplified by the fusion of gradient-boosted decision trees with deep learning agents. This combination effectively addresses the 'curse of dimensionality' while maintaining a high level of predictive accuracy and detail.

Among the numerous AI models, Deep Reinforcement Learning (DRL) exhibits unique adaptability in automated trading:

Sequential decision optimization: Mimics investor "reward-maximization" behavior, addressing delayed rewards (e.g., holding-period returns) and partial observability.

Hierarchical DRL architecture (Chen & Velikov, 2022): Decomposes complex tasks (e.g., balancing immediate execution costs vs. long-term portfolio goals).

Empirical superiority: DRL agents achieve 12–18% higher risk-adjusted returns than traditional methods in markets with nonlinear momentum decay (Feng et al., 2024).

Additionally, in practice, many quantitative fund companies have achieved remarkable returns by utilizing DRL models. A notable example is the Medallion Fund, which has reported an annualized return of 35%.

While studying this, we had a question:Does the superior performance of machine learning models in quantitative trading stem from their accurate prediction of short-term stock price trends? Or:Are the long-term gains of DRL strategies driven by short-term price trend prediction?

In the preliminary experiments, we developed a short-term DRL model aimed at predicting short-term returns. The results are as follows:

Model Comparison Results:

Model	Direction Acc	RMSE	IC	RIC	R^2
PPO	51.31%	1.0176	0.0446	0.0160	0.0019
GRPO	50.44%	1.0182	0.0305	0.0173	0.0009
DQN	53.78%	1.0180	0.0484	0.0360	0.0012
A2C	53.34%	1.0250	0.0253	0.0364	-0.0126

Fig 1.1 Previous DRL Models In Prediction Tasks

The results were disappointing:

Directional Accuracy (Direction Acc) was around 50%, meaning the model's predictions were no better than a coin toss.

Root Mean Squared Error (RMSE) exceeded 1, indicating large prediction errors.

Information Coefficient (IC/RIC) values were close to 0, showing weak predictive power.

R-squared (R^2) was near 0, meaning the model explained almost none of the variance in returns.

By analyzing the model's performance and reviewing existing literature,we have synthesized several key factors that help explain the observed outcomes.One significant limitation of the current model lies in its reliance on single-step predictions. This approach inherently neglects the cumulative effects and repercussions of multi-step decisions. Another critical factor affecting the performance of our model is the significant presence of price noise in financial markets, which poses substantial challenges for stably modeling short-term signals.

1.2 Directions for Research Improvement

In our final project, we advanced the initial model by introducing a Hierarchical Deep Reinforcement Learning (DRL) Framework featuring Multi-Time Horizon Coordination. This sophisticated architecture distinguishes between a Short-Term Agent (Tactical Layer) and a Long-Term Agent (Strategic Layer), each designed to address specific aspects of market dynamics.

Short-Term Agent (Tactical Layer): This component utilizes high-frequency data, such as minute-level price movements and order book depths, to identify and exploit fleeting arbitrage opportunities like mean reversion and liquidity gaps. The tactical focus here is on capturing immediate market inefficiencies for rapid profit-taking.

Long-Term Agent (Strategic Layer): In contrast, this layer incorporates low-

frequency data, including weekly or monthly trends and macroeconomic indicators, to inform decisions aimed at optimizing the portfolio's risk-return profile and managing overall market exposure. The strategic objective is to align investment strategies with broader economic cycles and long-term market trends for sustained performance.

Our hierarchical DRL framework enables dynamic coordination between the Strategic Layer (long-term agent) and the Tactical Layer (short-term agent). This coordination is achieved through two primary mechanisms: Signal Fusion and Adaptive Weighting.

Signal Fusion: Short-term recommendations must be validated by long-term trends. For instance, in a bear market, aggressive buy signals from the short-term agent should be suppressed to avoid high-risk investments during unfavorable conditions.

Adaptive Weighting: The decision weights between layers are adjusted based on market volatility. This means that during periods of high market fluctuation, more emphasis might be placed on the strategic decisions made by the long-term agent to ensure stability and risk management, whereas in stable markets, the tactical decisions from the short-term agent may be given more weight to exploit immediate opportunities.

II Literature Review

Progresses in machine learning, deep learning, and reinforcement learning have revolutionized quantitative finance by enabling models to process high-dimensional financial data and uncover latent predictive patterns imperceptible to traditional econometric tools [1][19]. Early value-based DRL methods such as Deep Q-Networks (DQN) demonstrated human-level control in Atari games and inspired initial usages into trading, though they struggled with continuous action spaces and stability [2][7][11]. Subsequent frameworks tailored to finance—such as the deep portfolio management system of Jiang et al. [3] and the decision-support architecture of Ansari et al. [4]—extended DQN's concepts to multi-asset allocation, yet often encounter short-term noise with long-term trends.

Actor-critic algorithms, particularly Proximal Policy Optimization (PPO), introduced clipped objectives that stabilize policy updates and accommodate continuous position sizing [5]. Practical implementations for stock trading by Liu et al. [6] and Xiong et al. [18] showcased PPO's robustness in noisy markets, while dueling network architectures improved value estimation by disentangling state and advantage functions [7]. At the same time, risk-sensitive approaches like Mihatsch and Neuneier's framework [8] and recurrent-dynamic RL models [9] incorporated drawdown penalties and stacked recurrent layers to balance profit and risk. DeepTrader further embedded market-condition features to achieve risk-return balance in portfolio management [10].

Reward engineering remains crucial: hybrid volatility forecasting models (e.g. LSTM–GARCH) highlight the challenges of predicting noisy price movements [12], and adaptive reward schemes optimizing risk-adjusted metrics—such as those in Almahdi & Yang [17]—mitigate over-trading. Theoretical analyses underscore that DQN variants can suffer from instability without proper reward shaping [11], motivating more sophisticated ensemble techniques that combine gradient-boosted trees with DRL agents for robustness against the curse of dimensionality [13].

Despite these successes, most DRL traders optimize a single decision horizon, thereby missing the distinct temporal scales of financial signals. Hierarchical reinforcement learning offers a remedy: meta-controllers can coordinate sub-policies specialized in different tasks, improving adaptability across market regimes [16][9].

Adaptive position-sizing algorithms in multi-agent systems further demonstrate how strategic and tactical objectives can be balanced [15]. However, concrete applications of hierarchical DRL to explicit multi-timescale trading remain sparse.

Building on foundational principles from Sutton & Barto [14] and informed by ensemble strategies in automated trading [16], we propose a Hierarchical-PPO framework that decouples short-term signal extraction from long-term trend management. Our system employs a high-frequency PPO agent to capture transient anomalies and a slower PPO agent to enforce strategic asset allocation. Through a learned arbitration mechanism, the framework dynamically blends tactical and strategic recommendations, enhancing both immediate trade accuracy and sustained portfolio growth. This multi-timescale approach addresses the limitations of single-horizon models and aligns DRL trading more closely with human portfolio management practices.

Besides, previous research seldom focused on short-term stock return predictions, which left a fresh area where several characteristics need to be figured out. This research aims at analyzing the performance of DRL models in return prediction, and uses PPO as an example model, to discover the prediction power of DRL models. Beyond that, it also innovatively creates a hierarchical structure combing long-short term horizons in time series to enhance the ability of the model.

III Methodology

3.1 Single-Layer Reinforcement Learning Agent

This research first tests the performance of single DRL agent in stock market, specifically targeting its short-term yield prediction performance.

3.1.1 State Space

The short-term state space is formed similar to previous relative research, which organized daily stock trading indices into observants. Traditionally, daily observants can be written as

$$\overrightarrow{S_t} = [p_t^O, p_t^H, p_t^L, p_t^C]$$

, where S_t represents state space observants at time t, $p_t^O, p_t^H, p_t^L, p_t^C$ means the open price, high price, low price and close price at time t.

While this formula only uses daily prices as features, actually indicating the assumption of EMH (Efficient Market Hypothesis), which suggests that the stock price has fully reflected the historical information. Thus, analyzing the prices is adequate.

This research contends that asset prices fail to comprehensively encapsulate all available market information, owing to the persistence of private information and the limited rationality exhibited by investors. Consequently, the market does not achieve full informational efficiency. Accordingly, beyond mere price-based metrics, the state space must integrate additional critical observables to more accurately represent the underlying market dynamics:

$$\overrightarrow{S_t} = \{p_t^O, p_t^H, p_t^L, p_t^C, E_t\}$$

. Where E_t represents other important market factors. Previous studies have discovered many significant factors which can used in predicting returns, and this paper select part of them in observants. Besides, factors may show different performances in different markets, so this research actually adopt different factors when experimenting different markets' stocks. Table 4.1 shows the usage of factors in different stocks.

While daily observation may not be enough, adding time window into observation, the state space can store more information, which is:

$$\boldsymbol{S_t} = \begin{bmatrix} S_{t-\tau} \\ S_{t-\tau+1} \\ \vdots \\ S_t \end{bmatrix} = \begin{bmatrix} p^O_{t-\tau}, p^H_{t-\tau}, p^L_{t-\tau}, p^C_{t-\tau}, E_{t-\tau} \\ p^O_{t-\tau+1}, p^H_{t-\tau+1}, p^L_{t-\tau+1}, p^C_{t-\tau+1}, E_{t-\tau+1} \\ \vdots \\ p^O_t, p^H_t, p^L_t, p^C_t, E_t \end{bmatrix} \in R^{\tau \times n}$$

, which suggests the agent observes the state from time $t-\tau$ to t, and the window is τ.

3.1.2 Action

This research first tests the performance of DRL agents in yield predicting process. Thus, the actions are directly formed as predicted return:

$$A_t = \widehat{r_{t+1}}$$

, where $\widehat{r_{t+1}}$ is predicted value of daily stock return, and $r_{t+1} = \frac{P_{t+1}}{P_t} - 1$.

In the real experiments process, the predicted value should of course be limited into a specific range, in case the huge volatility of actions. In this work, we construct the action space based on the agent's historical records of a stock's upward or downward movement. Let

$$A_t \in [L, U]$$

, where L and U are the lower bound and upper bound. The specific value of L and U can be formed using several methods. This research constructs the action space by scaling the historically observed maximum gain and maximum loss of the stock. Specifically, the upper bound of the action space is determined by multiplying the highest recorded return by 1.2 (representing a 20% increase), while the lower bound is established by multiplying the largest recorded loss by 1.2 (representing a 120% amplification). This scaled range delineates the permissible set of actions or decisions within the study's framework. It should also be noted that the observed history only among the training set excluding test set to prevent the information leak.

3.1.3 Reward Function

The reward is naturally the prediction accuracy. While this research addresses importance in direction prediction, the reward can be formed as gap between predicted value and direction correctness bonus:

$$R(t) = \alpha \cdot accuracy + \beta \cdot direction_{bonus}$$

$$= \alpha \cdot \frac{1}{|r_{t+1} - \widehat{r_{t+1}}| + 1} + \beta \cdot I\big(sgn(r_{t+1}) = sgn(\widehat{r_{t+1}})\big)$$

, in which α and β are two parameters, and $I(\cdot)$ is an indicator function. The sign

function, denoted $sgn(x)$, is defined as follows: it returns 1 if $x > 0$, -1 if $x < 0$, and 0 if $x = 0$. The condition is used to determine whether the actual return r_t and the predicted return r_t share the same direction, i.e., both are positive, both are negative, or both are zero.

3.1.4 PPO Model Structure

For modeling the trading process and solution algorithms, in this research, we employ PPO algorithm. Below, we briefly describe why the algorithms is selected and how it is applied in this context, followed by its respective pseudocode.

PPO (Proximal Policy Optimization). PPO is a policy gradient method that uses a clipped objective function to update the policy in a way that avoids making excessively large updates. This clipping mechanism helps maintain relatively small yet effective updates, improving the stability of the training process.

In PPO, the policy $\pi_\theta(a_t|s_t)$ is parameterized by θ, and the objective function is designed to penalize large policy updates by clipping the probability ratio. The key idea is to maintain the balance between exploration and exploitation while preventing catastrophic policy changes.

The objective function in PPO can be written as:

$$L^{clip}(\theta) = E_t[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$$

where $r_t(\theta)$ is the probability ratio between the current policy and the old policy:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

Here, A_t is the advantage at time t, which quantifies how much better an action is compared to the average action at that state. The advantage function is often computed using Generalized Advantage Estimation (GAE) to balance bias and variance:

$$A_t = \sum_{t=0}^{T-t} (\gamma\lambda)^t \delta_t^l$$

where $\delta_t^l = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$ is the temporal difference (TD) error.

The clipping term $clip(r_t(\theta), 1-\epsilon, 1+\epsilon)$ ensures that the objective function becomes less sensitive to the probability ratio when the ratio moves too far away from 1. This helps to constrain the policy update size, preventing large steps that could destabilize the learning process.

The value function $V_\phi(s_t)$, parameterized by ϕ, is updated to minimize the squared error between the predicted value and the target value:

$$L_v(\phi) = E_t\left[\left(V_\phi(s_t) - V_t^{target}\right)^2\right]$$

where $V_t^{target} = r_t + \gamma V_\phi(s_{t+1})$ is the target value derived from the rewards and the next state's value.

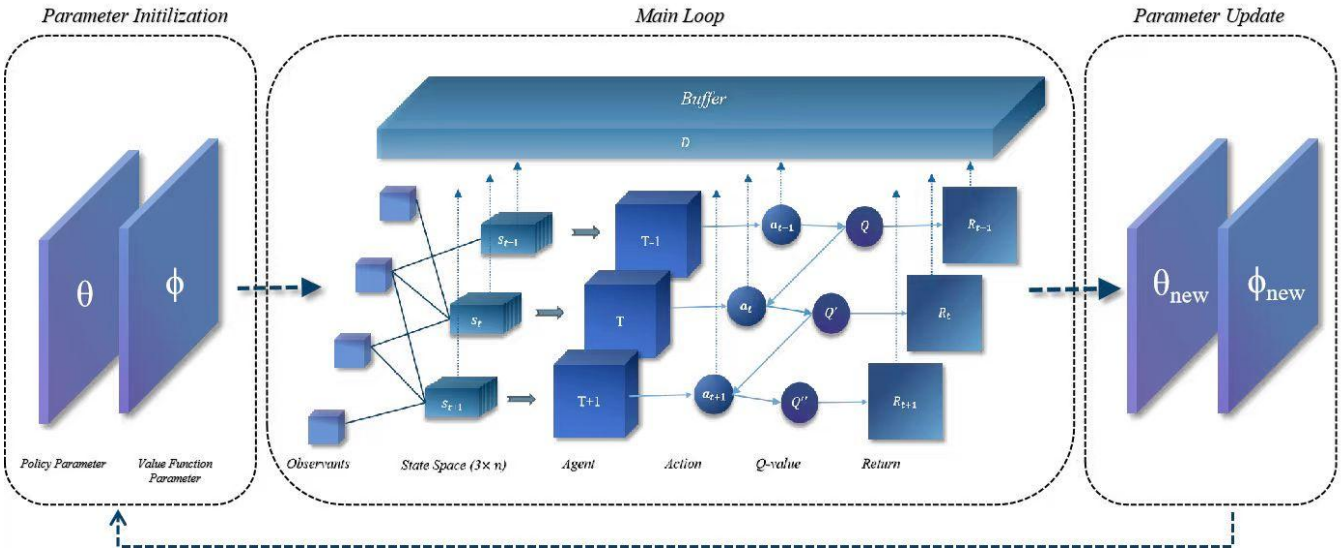

Fig 3.1 Structure of PPO

The overall algorithm proceeds as follows:

Algorithm 1 PPO Pseudocode

1: Initialize policy parameters θ, value function parameters ϕ.

2: **for** iteration = 1 to M **do**

3: **for** t = 1 to T **do**

4: Observe state s_t, select action $a\_t \sim \pi\_\theta\ (a\_t \mid s\_t)$.

5: Execute action a_t, observe reward r_t and next state s_{t+1}.

6: Store transition (s_t, a_t, r_t, s_{t+1}).

7: **end for**

8: Compute advantages A_t using value function V_ϕ.

9: Update policy θ by maximizing the clipped objective:

$$L^{clip}(\theta) = E_t[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$

10: Update value function parameters ϕ by minimizing

$$V_\phi(s_t) - V_t^{target}$$

11: Update $\theta_{old} \leftarrow \theta$.

12: **end for**

3.2 Hierarchical PPO Agent

In this study, we propose a hierarchical multi-layer agent framework that integrates a short-term agent and a long-term agent to enhance both short-term return prediction and long-term trading performance in stock market environments. Building upon the evaluation of single-layer models, this multi-layer architecture leverages the complementary strengths of the two agents through a structured interaction mechanism. The short-term agent is tasked with monitoring short-term market states, predicting near-term returns, and dynamically adjusting its actions based on prediction accuracy. These actions are subsequently incorporated as part of the long-term state representation and relayed to the long-term agent. Conversely, the long-term agent focuses on executing trading decisions, deriving rewards from the outcomes of these trades. The rewards obtained by the long-term agent are then fed back to the short-term agent, enabling it to refine its predictive strategies. This bidirectional feedback loop fosters a synergistic relationship between the two agents, optimizing short-term forecasting precision while simultaneously improving long-term trading outcomes. By employing this hierarchical approach, the proposed framework effectively addresses the dual challenges of capturing short-term market dynamics and achieving sustained profitability over extended trading horizons.

It's necessary to clarify some basic issues related to the long-short structure. The shot-term agent will take actions daily and calculate rewards at the same time and same frequency, while the long-term agent will act (trade) daily but calculate reward each m days to capture the long-term trading performence.

3.2.1 Short-term Agent

The short-term agent is basically the same as single-layer agent. The only difference lies in state space and reward function.

State space of shotr-term agent can be written as:

$$\overrightarrow{S_t^{short}} = \left[p_t^O, p_t^H, p_t^L, p_t^C, E_t^{short}, \frac{R^{long}(last)}{m} \right]$$

,where $\overrightarrow{S_t^{short}}$ means the observation state of short-term agent at time t,

$[p_t^O, p_t^H, p_t^L, p_t^C, E_t]$ is the same as previous, as a part of the state space. $\frac{R^{long}(last)}{m}$ is the reward feedback from long-term trading, where $R^{long}(last)$ is the reward calculated in the long-term agent last time and m stands for the frequency of long-term reward calculated. Thus,

$$\boldsymbol{S_t}^{\boldsymbol{short}} = \begin{bmatrix} S_{t-\tau}{}^{short} \\ S_{t-\tau+1}{}^{short} \\ \vdots \\ S_t{}^{short} \end{bmatrix} = \in R^{\tau \times n}$$

Reward function of short-term agent can be written as:

$$R^{short}(t) = \alpha \cdot \frac{1}{|r_t - \hat{r_t}| + 1} + \beta \cdot I\big(sgn(r_t) = sgn(\hat{r_t})\big) + \frac{\gamma}{m} \cdot R^{long}(last)$$

, which is the original single-layer reward function plus feedbacks from long-term agent, where γ is a weight coefficient of long-term performance and $R\hat{}long\ (last)/m$ is the long-term reward.

3.2.2 Long-term Agent

The long-term agent is responsible for catching long-term trend and directly trading.

The state space of long-term agent is the single-layer state space with a prediction from shor-term agent's action:

$$\overrightarrow{S_t^{long}} = \left[p_t^O, p_t^H, p_t^L, p_t^C, E_t^{long}, A_t^{short}\right]$$

, where $E\_t\hat{}long$ is the long-term signals in the stock market and A_t^{short}

The action of long-term agent is to trade the stock, just as traditional research in this area did:

$$\begin{cases} Sell\ N_t = int_{feasible}\left(\left|A_t^{long}\right| \cdot H_t, u\right) shares\ at\ the\ end\ of\ time\ t, & if\ A_t^{long} < 0 \\ Buy\ N_t = int_{feasible}\left(\dfrac{A_t^{long} \cdot C_t}{P_t}, u\right) shares\ at\ the\ end\ of\ time\ t, & if\ A_t^{long} < 0 \\ Hold\ (no\ action) at\ the\ end\ of\ time\ t, & if\ A_t^{long} = 0 \end{cases}$$

, where $A_t^{long} \in [-1,1]$ is the action range, H_t is the shares in hand at time t, C_t is the available cash at time t, P_t is the stock price at time t. And $int_{feasible}(x,u) = u \cdot \left\lfloor \frac{x}{u} \right\rfloor$, in which x stands for a number, and u is the minimum shares the buyers must buy in one hand. In the formula, $int_{feasible}\left(\left|A_t^{long}\right| \cdot H_t, u\right)$ and

$int_{feasible}\left(\frac{A_t^{long} \cdot C_t}{P_t}, u\right)$ are the possible units of shares that an investor can buy. For instance, in A stock market, $u = 100$ which means the investor must buy 100 shares a hand.

The reward function of long-term agent is calculated each m days. The reward function can be written as:

$$R^{long}(t) = \delta \cdot r_{t-m}^{t} - \varepsilon \cdot s + \epsilon \cdot k$$

, where δ, ε and ϵ are weight parameters and $r\_(t-m)\hat{}t$ is the cumulative return from time $t-m$ to time t. s is the standard deviation of the stock in m days and k is the total cost of transaction. Specifically,

$$R^{long}(t) = \delta \cdot \frac{P_t - P_{t-m}}{P_{t-m}} - \varepsilon \cdot \sqrt{\frac{1}{m-1} \cdot \sum_{i=t-m+1}^{t} (r_i - \bar{r})^2} + \epsilon \cdot k$$

$$= \delta \cdot \frac{P_t - P_{t-m}}{P_{t-m}} - \varepsilon \cdot \sqrt{\frac{1}{m-1} \cdot \sum_{i=t-m+1}^{t} \left(\frac{P_i - P_{i-1}}{P_{i-1}} - \frac{1}{m} \cdot \sum_{i=t-m+1}^{t} \frac{P_i - P_{i-1}}{P_{i-1}}\right)^2} + \epsilon \cdot k$$

The short-term agent and long-term agent corporates together to combine as a strong agent, the whole algorithm can be represented as:

3.2.3 Hierarchical Structure Combining Long-Short Term

The Hierarchical PPO (HPPO) model combines long-term agent and shot-term agent by designing a interactive mechanism to address their advantages. All in all, Short-term agent takes actions and calculate rewards each step. Its action will transfer to long-term agent as state. Long-term agent takes actions each step while calculate rewards each $m(\Delta t)$ steps. Its reward in the last time will be transferred to short-term agent.

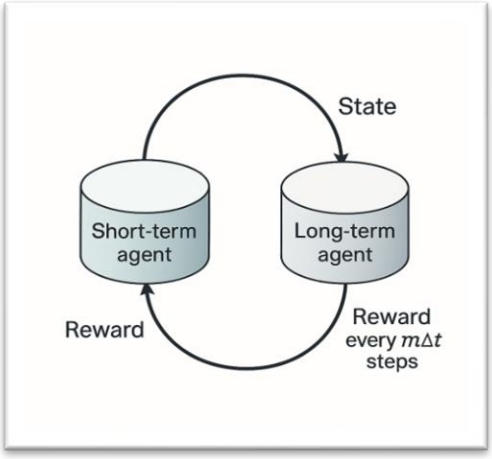

Fig 3.2 Long-Short Term Interaction

This hierarchical design effectively captures both short-term market fluctuations and long-term strategic patterns. By allowing the short-term agent to respond quickly to immediate signals and feeding its behavior into the long-term agent's decision-making, the model achieves a dynamic balance between responsiveness and stability. The mutual feedback loop enhances temporal consistency in actions and improves the precision of return prediction across multiple time horizons.

Algorithm 2 Hierarchical Proximal Policy Optimization (H-PPO) Pseudocode

1: Initialize short-term policy parameters θ_{short} and value-function parameters ϕ_{short}

2: Initialize long-term policy parameters θ_{long} and value-function parameters ϕ_{long}

3: Set long-term feedback $R_{long_{feedback}} \leftarrow 0$

4: for episode = 1 to M do

5: Reset short-term state ${s_0}^{short}$ and long-term state ${s_0}^{long}$

6: Clear short-term trajectory Γ_{short} and long-term trajectory Γ_{long}

7: for t = 0 to T–1 do

8: // Short-horizon agent

9: Observe $s_t^{short} = \left[p_t^O,\ p_t^H,\ p_t^L,\ p_t^C,\ E_t^{short},\ R_{long_{feedback}}\right]$

" 10: Sample $a_t^{short} \sim \pi_{\{\theta_{short}\}}\left(\cdot \middle| s_t^{short}\right)$

11: Receive actual next-day return $r_{\{t+1\}}$

12: Compute

$$R_t^{short} = \alpha \cdot \left(\frac{1}{\left|r_{\{t+1\}} - a_t^{short}\right| + 1}\right) + \beta \cdot \mathbb{1}\left[sign\left(r_{\{t+1\}}\right) = sign\left(a_t^{short}\right)\right] + \frac{\gamma}{m} \cdot R_{long_{feedback}}$$

13: Append $\left(s_t^{short},\ a_t^{short},\ R_t^{short},\ s_{\{t+1\}}^{short}\right)$ $to\ \Gamma_{short}$

14: // Long-horizon agent

15: Observe $s_t^{long} = \left[p_t^O,\ p_t^H,\ p_t^L,\ p_t^C,\ E_t^{long},\ a_t^{short}\right]$

16: Sample $a_t^{long} \sim \pi_{\{\theta_{long}\}}\left(\cdot \middle| s_t^{long}\right)$

17: Execute action a_t^{long}

18: Every reward_freq steps, compute R_t^{long} and set $R_{long_{feedback}} \leftarrow R_t^{long}$

19: Observe next state $s_{\{t+1\}}^{long}$

20: Append $\left(s_t^{long},\ a_t^{long},\ R_t^{long},\ s_{\{t+1\}}^{long}\right)$ $to\ \Gamma_{long}$

21: end for

22: Compute short-term advantages A_t^{short} from Γ_{short} and long-term advantages A_t^{long} from Γ_{long}

23: Perform PPO update on θ_{short} using A_t^{short}

24: Update ϕ_{short} via value-loss on Γ_{short}

25: Perform PPO update on θ_{long} using A_t^{long}

26: Update ϕ_{long} via value-loss on Γ_{long}

27: end for

IV Experiment

4.1 Dataset

We mainly used data from five types of stocks for testing. We also tested some Hong Kong stocks and A-shares, and the performance improvement was satisfactory in all cases.

The stocks tested include:

IXIC: Nasdaq Composite Index, a market-capitalization-weighted index that covers over 2,500 companies listed on the Nasdaq Stock Exchange.

DJI: Dow Jones Industrial Average, a price-weighted index composed of 30 blue-chip companies traded on the New York Stock Exchange and Nasdaq.

7203T: Toyota Motor Corporation, one of Japan's largest automakers, with the ticker symbol 7203.T.

6758T: Sony Group Corporation, a Japanese multinational conglomerate primarily engaged in electronics, gaming, music, and film businesses, with the ticker symbol 6758.T.

9983T: Fast Retailing Co., Ltd. (the parent company of Uniqlo), a major Japanese clothing retailer, with the ticker symbol 9983.T.

4.2 Test Model

Before making specific yield predictions, let’s first examine the training effectiveness of the model.

4.2.1 Single-Layer PPO Model

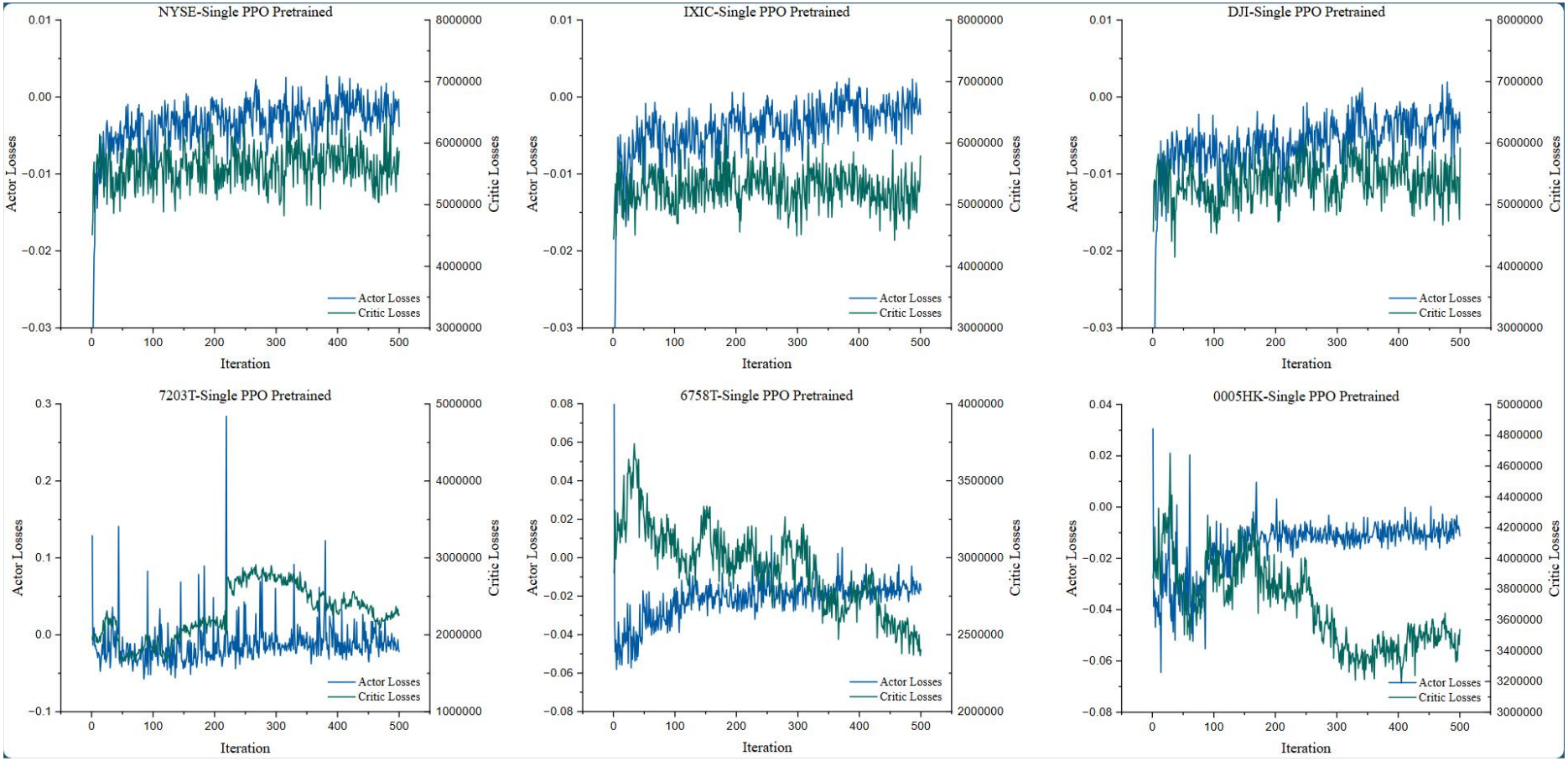

Fig 4.1 Loss Convergence Figure

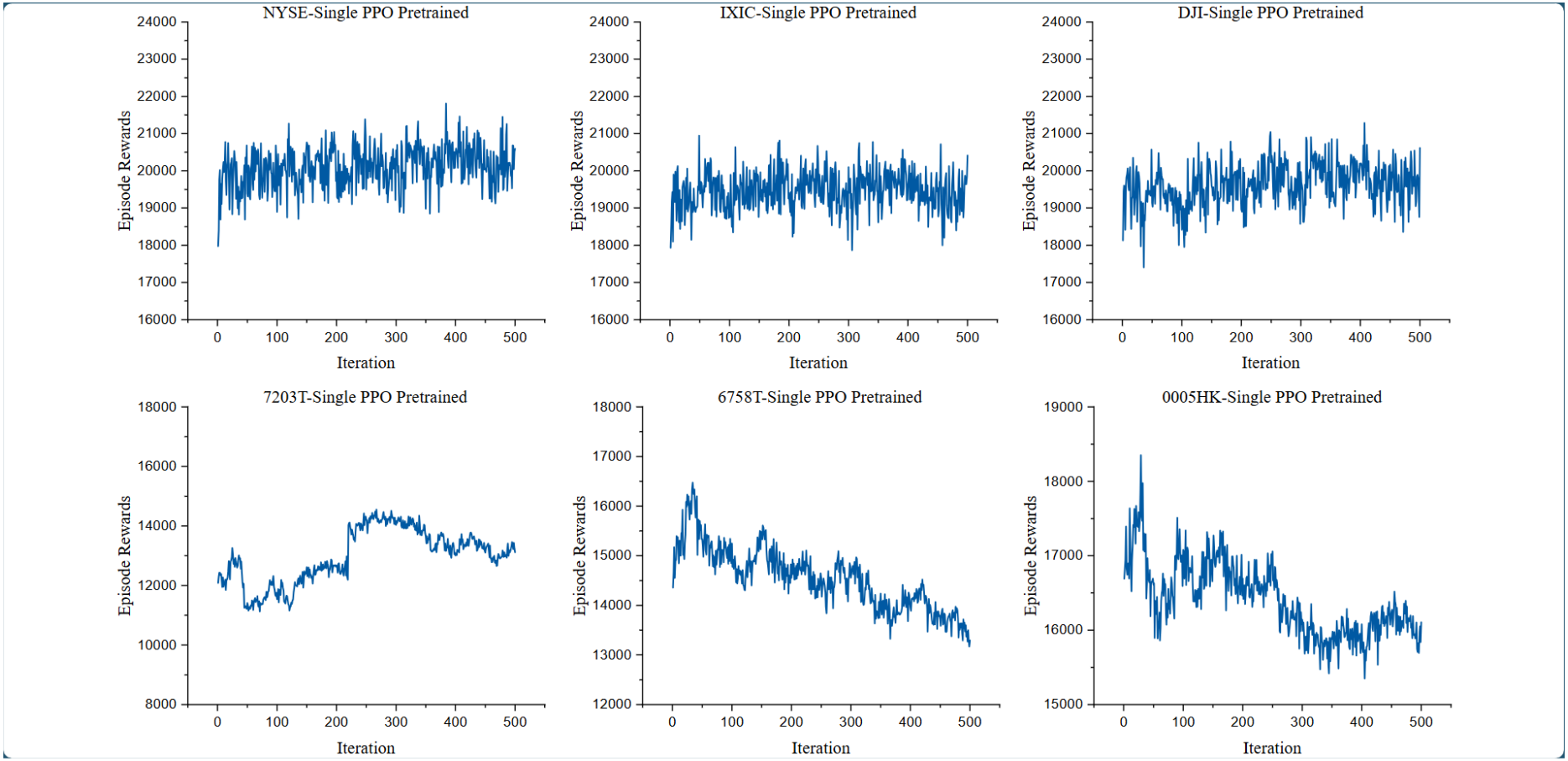

Fig 4.2 Reward Convergence Figure

The figure represents the loss and rewards. From the Fig 4.1 and Fig 4.2, we can generally see that the graphs gradually approach a certain value from left to right, indicating that the changes become smaller and smaller over time and the model becomes more stable. This suggests that the model has "converged," which means the training has been effective.Therefore, we proceeded to predict the returns.

4.2.2 HPPO

Here is the long-term and short-term combined model—HPPO. Before making specific return predictions, we first examined the training effectiveness of the model. The results presented here reflect the model's training performance across three different types of stocks: DJI, IXIC, and NYSE. In terms of both loss and rewards, we can clearly observe from Figure 4.3 to Figure 4.5 that the graphs gradually approach a certain value as they progress from left to right. This indicates that the fluctuations in loss and rewards decrease over time, suggesting a stabilization of the model's performance. As the training progresses, these values converge, demonstrating a reduction in volatility and an overall increase in the model's stability. This behavior is a strong indication that the model has "converged," signifying that the training process has been effective and the model is now capable of providing reliable predictions for return forecasts.

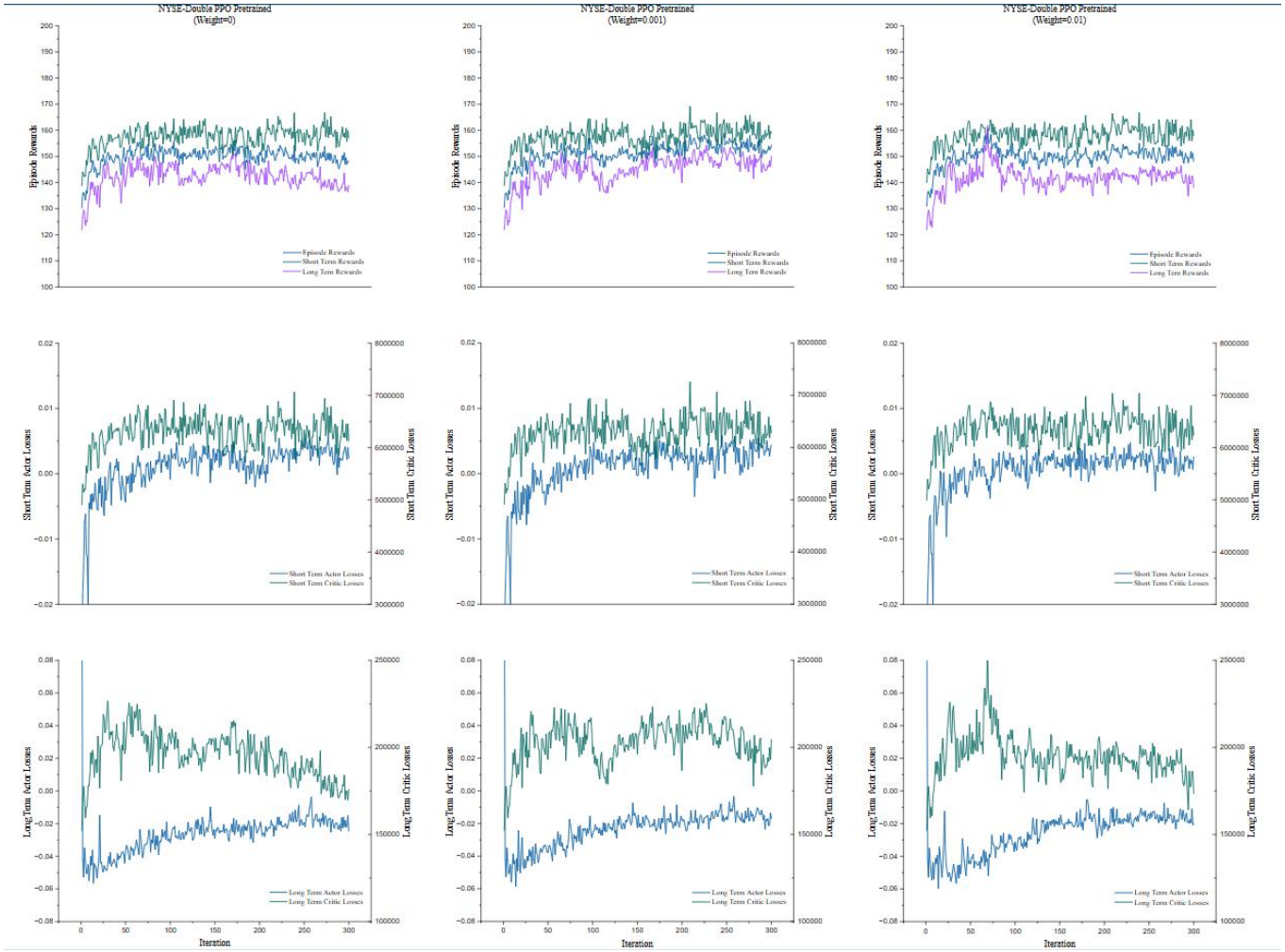

Fig 4.3 HPPO Trained on NYSE

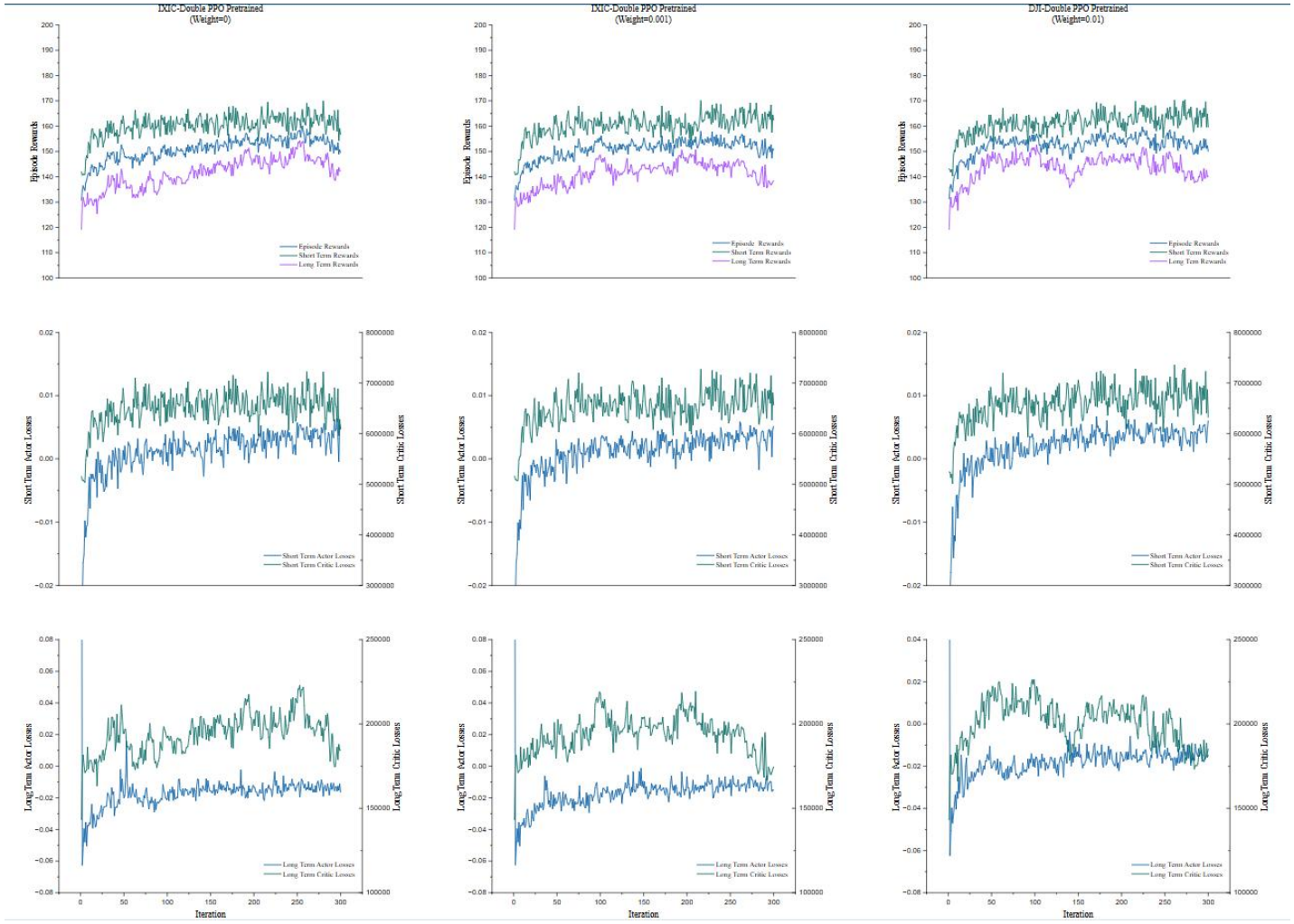

Fig 4.4 HPPO Trained on IXIC

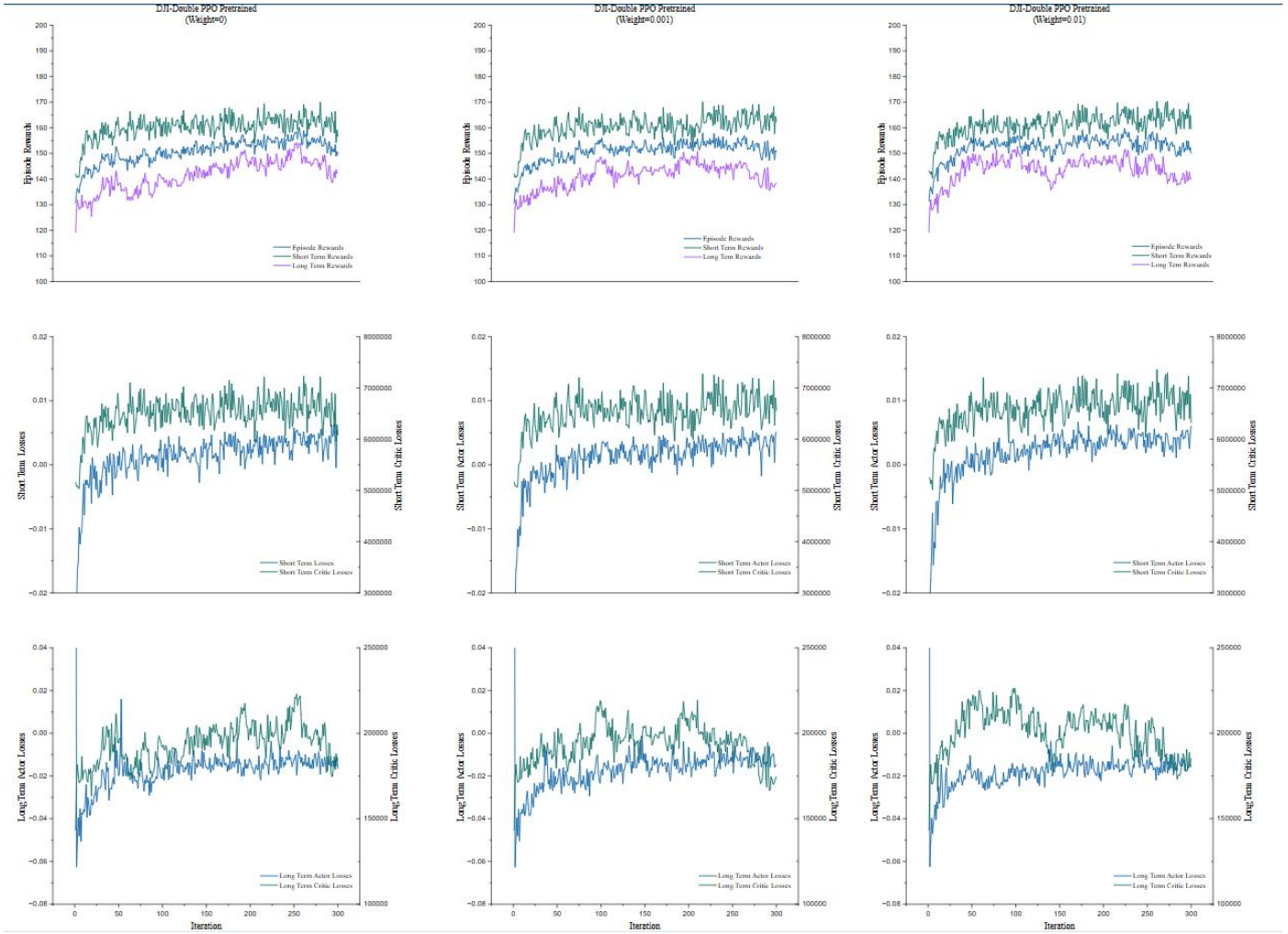

Fig 4.5 HPPO Trained on DJI

4.3 Training Results

4.3.1 Single-Layer PPO Model

Model Comparison Results:

Model	Direction Acc	RMSE	IC	RIC	R^2
PPO	51.31%	1.0176	0.0446	0.0160	0.0019
GRPO	50.44%	1.0182	0.0305	0.0173	0.0009
DQN	53.78%	1.0180	0.0484	0.0360	0.0012
A2C	53.34%	1.0250	0.0253	0.0364	-0.0126

Training Time Statistics:

Model	Total Time (s)	Avg Time per Iter (s)	Iterations	Total Iters
PPO	142.26	0.4742	300	300
GRPO	7846.31	26.1544	300	300
DQN	14422.09	48.0736	300	300
A2C	4009.75	13.3658	300	300

Fig 4.6 Performance Descriptions of Single-Layer PPO Model

After conducting the experiments, we found that PPO is the fastest, completing 300 iterations in just 142.26 seconds, with an average iteration time of 0.4742 seconds, demonstrating high efficiency. GRPO took 7846.31 seconds, with an average iteration time of 26.1544 seconds, but it did not show clear advantages, which raises questions about its efficiency. DQN and A2C were slower, with DQN taking 14422.09 seconds.

A2C was faster than both DQN and GRPO but still lagged behind PPO, further highlighting the practicality of PPO. If only price metrics are observed, the results are even worse.

4.3.2 HPPO Model

Table 4.1 Test Results

Stock	Weight	IC↑	RIC↑	RMSE↓	Win↑(%)
NYSE	-	0.2997	0.2823	0.0901	61.3372
	0.0000	0.2864	0.2902	0.0789	61.3470
	0.0010	0.3469	**0.3695**	0.0788	**65.1537**
	0.0100	**0.3471**	0.3421	**0.0779**	64.2753
Largest Improvement		***15.83%***	***30.91%***	***-13.55%***	***3.82%***
IXIC	-	0.1793	0.1728	0.0922	59.5930
	0.0000	**0.4072**	**0.4248**	**0.0698**	64.8609
	0.0010	0.3891	0.3970	0.0707	**66.3250**
	0.0100	0.3026	0.3247	0.0707	60.7613
Largest Improvement		***127.13%***	***145.85%***	***-24.34%***	***6.73%***
DJI	-	0.2953	0.2661	0.0985	59.8837
	0.0000	**0.4286**	0.4331	**0.0691**	**66.0433**
	0.0010	0.4103	**0.4405**	0.0771	65.3124
	0.0100	0.4007	0.4102	0.0781	62.0923
Largest Improvement		***45.12%***	***65.56%***	***-29.81%***	***6.16%***
7203T	-	0.0229	0.0817	2.5035	54.8701
	0.0000	0.1346	**0.2869**	**2.1876**	57.4257
	0.0010	0.0789	0.1489	2.3852	54.4554
	0.0100	**0.1440**	0.2596	2.2310	**58.4158**
Largest Improvement		***528.50%***	***251.30%***	***-12.62%***	***3.55%***
6758T	-	-0.0034	-0.0379	0.9988	47.7273
	0.0000	**0.0621**	0.0500	0.9063	49.8350
	0.0010	0.0072	**0.0796**	**0.8561**	**52.1452**
	0.0100	0.0072	**0.0796**	**0.8561**	**52.1452**
Largest Improvement		***0.07****	***0.12****	***-14.29%***	***4.42%***
9983T	-	0.0749	0.0046	0.8653	48.3766
	0.0000	0.0779	0.0511	0.8743	50.8251
	0.0010	**0.2058**	**0.1400**	**0.8473**	**56.7657**
	0.0100	0.1346	0.1387	0.9114	53.1353
Largest Improvement		***174.64%***	***2959.33%***	***-2.09%***	***8.39%***

From results, weight represents the reward weight of the long-term PPO in the entire model. The weight here is not the actual value, because the reward function does not directly use this weight. However, an increase in weight does indicate an increase in the weight of the long-term PPO. The table tests the results of a certain stock as the weight of the long-term PPO increases continuously;

For example, assume that in a certain use of HPPO (where the model requires collaboration between long-term and short-term PPO), the long-term weight is 0.001. When testing NYSE stocks, the IC value is 0.3469, the RIC is 0.3695, the RMSE is 0.0788, and the Win (the accuracy of predicting the direction of price movement) is 65.1537%. The bold entries indicate that the IC and Win of this model test are the best among all models. The red text below indicates how much better the performance is compared to the single-layer PPO (the first row marked with "-"). The arrows indicate whether a higher value (pointing up) or a lower value (pointing down) is better for the metric;

We also found that it is easier to predict index movements than individual stocks, and the inherent volatility of individual stocks can greatly affect model performance.

Table 4.2 HPPO Compared to Other Models

Method	IXIC			NYSE			DJI		
	RMSE↓	IC↑	RIC↑	RMSE↓	IC↑	RIC↑	RMSE↓	IC↑	RIC↑
LSTM	0.0187	0.0644	0.0674	0.0144	0.08	0.0649	0.0121	0.0926	0.0918
Transformer	0.0147	0.244	0.2542	0.0141	0.3189	0.3293	0.0166	0.282	0.3063
FreTS	0.0143	0.2722	0.2644	0.0143	0.2324	0.2208	0.0116	0.2658	0.2827
StockMixer	0.0149	0.2123	0.193	0.0159	0.2847	0.26	0.0117	0.2286	0.2253
AGCRN	0.0147	0.0948	0.0932	0.0142	0.0757	0.0772	0.0169	0.0607	0.0742
FourierGNN	0.0152	0.1395	0.1295	0.0144	0.1169	0.1199	0.0119	0.1451	0.1281
MambaStock	0.0145	0.2488	0.2059	0.0139	0.3697	0.3125	0.0141	0.3355	0.2776
HPPO	**0.0698**	**0.4072**	**0.4248**	**0.0779**	**0.3471**	**0.3695**	**0.0691**	**0.4286**	**0.4405**
SAMBA	0.0128	0.5046	0.4767	0.0125	0.5044	0.495	0.0108	0.4483	0.4703

Source of comparison data: https://doi.org/10.48550/arXiv.2410.03707

This table 4.2 compares HPPO with other models, with HPPO highlighted in black as our model. The comparison is based on three representative stocks: IXIC, NYSE, and DJI;

In terms of results, our model has a higher RMSE compared to other models, indicating that the prediction results have larger fluctuations. This may be due to the need for a smaller action space or the inherent nature of reinforcement learning;

Apart from RMSE, our model's IC and RIC far exceed those of general deep learning models, only trailing behind the SAMBA model;

However, our model's training speed is much faster than that of SAMBA. All other models in the table are trained using V100 (really strong hardware), while HPPO used RTX4070 and reached 0.86 sec/epoch. Therefore, overall, HPPO has significant advantages.

V Conclusion

5.1 Key Findings

The Hierarchical PPO model has demonstrated a strong ability to generate returns, indicating its ability as an effective trading strategy. Also, the weight allocation between the long-term and short-term models significantly affects the training results. Finding the optimal balance is crucial for maximizing performance. Model performance exhibited variability across different stocks, suggesting the need for adaptive strategies tailored to individual stock characteristics.

5.2 Model Advantages

Multi-Timescale Information Capture: The model successfully integrated both long-term and short-term market signals, enabling comprehensive analysis of price trends and volatility.

Reduced Overfitting Risk: By hierarchically separating objectives, the framework minimized overfitting and enhanced generalization across diverse market conditions.

Stability-Flexibility Balance: The architecture achieved an optimal trade-off between stability (long-term consistency) and flexibility (short-term responsiveness).

Cross-Stock Adaptability: Improved adaptability to different stock types and market inefficiencies, allowing for scalable deployment in heterogeneous portfolios.

5.3 Implications for Market Behavior

Investors need to consider both long-term and short-term factors for better decision-making, switch between long-term strategies and short-term tactics. Also, the hierarchical model's success indicates that there are inefficiencies and exploitable patterns at different time scales. Long-term investors identify undervalued stocks via fundamental analysis, while short-term traders exploit price anomalies using technical analysis. Combining these approaches enhances effectiveness in imperfect markets. Besides, distinguish between Value Stocks, Momentum Stocks, and Growth Stocks to optimize strategies and enhance overall performance.

5.4 Future Improvements

Meta-RL for Weight Optimization: Implement meta-reinforcement learning (meta-RL) to automatically optimize the weights between the long-term and short-term models can further enhance performance by adapting to real-time market conditions.

Integration of Multi-Source Data: Incorporating additional data sources, such as news sentiment, social media trends, and alternative data, can provide a more comprehensive understanding of market dynamics and improve the model's predictive power.

Visualization of Factor Contributions: Visualize the contributions of long-term and short-term factors to provide transparency into the model's decision-making process, helping users better understand and adjust its behavior.

Personalization Based on Risk Preferences: Tailoring the model's factors according to users' risk preferences can make it more user-friendly and aligned with individual investment goals.

Optimization of Risk-Aware Strategies: Enhance the model's risk management capabilities for further improvement of its robustness by balancing return generation with risk minimization.

Reference

[1] Sahu, S. K., Mokhade, A., & Bokde, N. D. (2023). An overview of machine learning, deep learning, and reinforcement learning-based techniques in quantitative finance: recent progress and challenges. *Applied Sciences*, *13*(3), 1956.

[2] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.

[3] Jiang, Z., Xu, D., & Liang, J. (2017). A deep reinforcement learning framework for the financial portfolio management problem. arXiv preprint arXiv:1706.10059.

[4] Ansari, Y., Yasmin, S., Naz, S., Zaffar, H., Ali, Z., Moon, J., & Rho, S. (2022). A deep reinforcement learning-based decision support system for automated stock market trading. IEEE Access, 10, 127469-127501.

[5] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[6] Liu, X. Y., Xiong, Z., Zhong, S., Yang, H., & Walid, A. (2018). Practical deep reinforcement learning approach for stock trading. arXiv preprint arXiv:1811.07522.

[7] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, June). Dueling network architectures for deep reinforcement learning. In International conference on machine learning (pp. 1995-2003). PMLR.

[8] Mihatsch, O., & Neuneier, R. (2002). Risk-sensitive reinforcement learning. Machine learning, 49, 267-290.

[9] Aboussalah, A. M., & Lee, C. G. (2020). Continuous control with stacked deep dynamic recurrent reinforcement learning for portfolio optimization. Expert Systems with Applications, 140, 112891.

[10] Wang, Z., Huang, B., Tu, S., Zhang, K., & Xu, L. (2021, May). Deeptrader: a deep reinforcement learning approach for risk-return balanced portfolio management with market conditions embedding. In Proceedings of the AAAI conference on artificial intelligence (Vol. 35, No. 1, pp. 643-650).

[11] Fan, J., Wang, Z., Xie, Y., & Yang, Z. (2020, July). A theoretical analysis of deep Q-learning. In Learning for dynamics and control (pp. 486-489). PMLR.

[12] Kim, H. Y., & Won, C. H. (2018). Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models. Expert Systems with Applications, 103, 25-37.

[13] Feng, X., et al. (2024). Ensemble models combining gradient-boosted trees with deep reinforcement learning for robust portfolio management. Expert Systems with Applications, 238, 121456.

[14] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.

[15] Chen, L., & Velikov, M. (2022). Adaptive position-sizing algorithms in multi-agent reinforcement learning systems. Quantitative Finance, 22(7), 1235–1250.

[16] Yang, H., Liu, X., Zhong, S., & Walid, A. (2020). Deep reinforcement learning for automated stock trading: An ensemble strategy. In Proceedings of the ACM International Conference on AI in Finance (ICAIF '20) (8 pages). ACM. https://doi.org/10.1145/3383455.3422540

[17] Almahdi, S., & Yang, S. Y. (2018). An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning. Expert Systems with Applications, 95, 262-271 arxiv.org.

[18] Xiong, Y., et al. (2018). Practical Deep Reinforcement Learning Approach for Stock Trading. ArXiv preprint arXiv:1811.07522 openfin.engineering.columbia.edu.

[19] Zhang, Y., Zohren, S., & Roberts, S. (2020). Deep Reinforcement Learning for Trading. Journal of Financial Data Science, 2(2), 25-40mdpi.com. (Survey)