Hierarchical Reinforced Trader

Gaurav Agarwal, Govind Singh, Vaisakh M. Menon

1 Problem Statement

Financial markets are characterized by their complexity, volatility, and the influence of a wide range of interdependent factors, making the development of effective trading and portfolio management strategies a persistent challenge. Traditional quantitative models, such as Modern Portfolio Theory (MPT), have provided valuable frameworks for balancing risk and return. However, these models often assume static conditions and linear relationships, which limit their scalability and effectiveness in dynamically evolving market environments.

In contrast, Deep Reinforcement Learning (DRL) has gained attention for its ability to learn adaptive policies directly from market data, enabling automated decision-making in complex, high-dimensional spaces. However, applying DRL to multi-stock portfolio management presents challenges:

- **Curse of Dimensionality:** State and action spaces scale exponentially with asset count.
- **Inertia/Momentum Effect:** Agents over-rely on past rewards, leading to suboptimal clustering.
- **Insufficient Diversification:** Agents concentrate on a few high-reward assets, increasing risk.

To address these, we implement the Hierarchical Reinforced Trader (HRT) agent based on the framework proposed by Zhao and Welsch [1]. HRT decomposes the trading problem into two levels: a High-Level Controller (HLC) for strategic stock selection and a Low-Level Controller (LLC) for trade execution. This report covers implementation details, methodology, experimen- tal setup, training results, and performance evaluation.

2 RL Problem Formulation

2.1 High Level Controller

- **States:**
  s = [fr,ss]
  fr represents **predicted forward returns**, which is percentage change of opening prices of a stock from day t to day t+1 from historical price and volume data. **ss** represents **sentiment scores** from text data, such as news or tweets (4-day Relative Strength Index (RSI14) was calculated and used as a proxy for sentiment scores).

- **Actions:**
  a = [-1,0,1] , representing **selling, holding, and buying** respectively.

- **Rewards:**

  The alignment reward for stock i, $r_i^{\text{align}}(s, a_i, s')$, correlates with action a_i and the actual price change ΔP_i. The sgn$(\cdot)$ function, returning -1 for negative inputs, $+1$ for positive inputs, and 0 for neutral input, is employed to determine rewards. A reward of 1 is granted if the action aligns with the real stock return, -1 if it contradicts the stock return, and no reward or penalty is applied if the action is to hold.

$$r_i^{\text{align}}(s, a_i, s') = \begin{cases} \text{sign}(a_i) \cdot \text{sign}(\Delta P_i) & \text{if } a_i \neq 0 \\ 0 & \text{if } a_i = 0 \end{cases} \tag{1}$$

  The comprehensive reward for the HLC, $r^h(s, \mathbf{a}, s')$, is a linear mix of the sum of r_i^{align} for all n stocks and the LLC reward. The weighting factor α_t starts near 1, emphasizing the HLC's alignment with stock price movements, and gradually decreases to focus more on the LLC's reward, following an exponential decay $\alpha_t = \alpha_0 e^{-\lambda t}$, where we set $\alpha_0 = 1$ and $\lambda = 0.001$.

$$r^h(s, \mathbf{a}, s') = \frac{1}{n} \sum_{i=1}^{n} r_i^{\text{align}}(s, a_i, s') + (1 - \alpha_t) r^l(s, \mathbf{a}, s') \tag{2}$$

2.2 Low Level Controller

- **States:**
  The state at each time step is represented as a vector combining normalized stock prices, normalized holdings, normalized cash, and the HLC (Buy, Sell, Hold) decisions. Specifically:
  s = [pt, ht, bt, ah]
  pt, ht, bt represents **stock prices, holdings and cash balance**, with augmented **ah** - decisions from HLC.

- **Actions:**
  The action space is continuous, representing the proportion of the portfolio to allocate to each stock. For one stock, 0,1...k where $k <= hmax$ (how much of the stock to be traded). Normalized to [-1,1] (continuous space)

- **Rewards:**
  Change in portfolio value from trades executed at **t to t+1**.
  Value = p$^T h + b$

3 Methodology

3.1 Data Preprocessing

We collected historical stock data using the YahooFinance API [2], obtaining daily Open, High, Low, Close, and Volume (OHLCV) values for a universe of large-cap U.S. equities. The dataset initially included the tickers AAPL, MSFT, GOOGL, AMZN, and META. However, due to persistent null values in the META data, this ticker was excluded from further analysis to ensure data integrity, resulting in a final set of four stocks.

From the raw OHLCV data, we derived the Volume Weighted Average Price (VWAP) on a daily basis to capture the average trading price weighted by volume, which is widely used for optimal execution strategies.

For constructing the High-Level Controller (HLC) state space, we computed the daily forward return for each stock as the percentage change in its opening price from day T to day $T+1$, using historical data. This forward return serves both as a predictive feature and as a reward signal for the reinforcement learning agent.

To provide a comprehensive and information-rich feature set, we incorporated 158 engineered technical and fundamental factors from the Alpha158 library in Qlib [3]. Additionally, the 14-day Relative Strength Index (RSI14) was calculated and used as a proxy for sentiment scores, capturing aspects of market momentum and potential overbought or oversold conditions. All features were computed using a rolling lookback window of five days, which balances the need to capture short-term market dynamics with the stability required for robust model training.

The resulting dataset was filtered and aligned by trading date, and all features were normalized to have zero mean and unit variance using running estimates from the training set. This normalization was applied to both the HLC and low-level controller (LLC) state inputs, including account balances, prices, and holdings, ensuring stable and efficient learning. The final preprocessed data was structured as multi-dimensional arrays compatible with the hierarchical reinforcement learning framework.

3.2 Implementation

3.2.1 Framework

The Hierarchical Reinforced Trader (HRT) employs a bi-level HRL structure to optimize the stock trading process, dividing it into strategic selection and tactical execution. This decomposition aims to manage complexity and enhance decision-making quality. Figure 2 (from the original paper, not reproduced here) provides a schematic overview.

- **High-Level Controller (HLC):** Positioned at the strategic level, the HLC analyzes market conditions (represented by features derived from price/volume data) and external signals (like sentiment scores) to determine the trading direction for each stock in the portfolio. Its output is a discrete action for each stock: Buy $(+1)$, Sell (-1), or Hold (0). Given its role in making crucial discrete choices, Proximal Policy Optimization (PPO) is selected for its stability and effectiveness in discrete action spaces.
  Algorithm (PPO): PPO is used with separate Actor and Critic MLPs (2 layers, 128 neurons). PPO updates are done using clipped objectives with entropy regularization (coef $= 0.01$), GAE ($\lambda = 0.95$), and value loss coefficient (0.5), using AdamW optimizer (lr $= 3e-4$).
  For the **HLC's** architecture, we select Proximal Policy Optimization (**PPO**), which moderates policy updates to prevent detrimental performance shifts. This process calculates the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ between new and old policies, incorporating a clipping mechanism in the objective function:

$$L^{\text{CLIP}}(\theta) = \hat{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \tag{1}$$

  Here, $\hat{A}(s_t, a_t)$ denotes the advantage function estimate, and the clipping limits the ratio $r_t(\theta)$ to $[1-\epsilon, 1+\epsilon]$, opting for the minimum value to restrict excessive policy changes and ensure training stability.

- **Low-Level Controller (LLC):** Operating under the HLC's directives, the LLC focuses on the execution aspect. For stocks flagged for buying or selling by the HLC, the LLC determines the precise quantity of shares to transact, constrained by a predefined maximum (h_{max}). For stocks designated as 'Hold' by the HLC, the LLC is bypassed. Deep Deterministic Policy Gradient (DDPG) is chosen for the LLC due to its ability to handle continuous action spaces effectively and maintain stable learning, suitable for optimizing the nuanced task of trade sizing.

Algorithm (DDPG): DDPG with separate Actor (2x256 MLP, tanh output) and Critic (state/action input). Uses Replay Buffer, target networks with $\tau = 0.005$, exploration noise $\mathcal{N}(0, 0.1)$, and AdamW ($\text{lr} = 1e - 3$).
The **LLC** employs the Deep Deterministic Policy Gradient (**DDPG**) framework for implementation. At each timestep, the LLC executes an action a in state s, earns a reward r, and transitions to a new state s'. These transactions (s, a, s', r) are stored in the replay buffer R. A batch of N transitions is drawn from R to update the Q-value y_i as follows:

$$y_i = r_i + \gamma Q' \left(s_{i+1}, \mu' \left(s_{i+1} \middle| \theta^{\mu'} \right) \right), \quad i = 1, \cdots, N \tag{2}$$

Next, the critic network is updated by minimizing the loss function $L(w^l)$, which calculates the expected difference between the target critic network Q' and the actual critic network Q:

$$L(w^l) = E_{s,a,r,s' \sim \text{buffer}} \left[\left(y_i - Q(s, a|w^l) \right)^2 \right]. \tag{3}$$

The two controllers work synergistically. The HLC's directional decisions (a^h) are incorporated as part of the LLC's state (s^l), directly influencing its execution strategy. The outcome of the LLC's trades, quantified by the change in portfolio value (r^l), serves as a crucial feedback signal for the HLC, particularly in the later stages of training.

To effectively train the interconnected HLC and LLC, we implement the Phased Alternating Training method described in Algorithm 3. This structured approach ensures stability and promotes synergistic learning.

- **Phase 1 - HLC Pre-training (Iterations 1 to 30):**
  - *Goal:* Establish a solid foundation for strategic direction finding.
  - *Procedure:* Only the HLC (PPO) is trained. The LLC acts (potentially with noise depending on implementation detail, though ideally deterministic here) but its parameters are not updated. Crucially, the HLC is trained using *only* the alignment reward component ($\sum r_i^{align}$), effectively setting $\alpha_t = 1$ for the reward calculation during this phase. This focuses the HLC purely on predicting short-term price movements correctly.
- **Phase 2 - LLC Pre-training (Iterations 31 to 60):**
  - *Goal:* Optimize trade execution based on fixed strategic directions.
  - *Procedure:* The HLC's parameters are frozen. Only the LLC (DDPG) is trained. The HLC continues to provide directional actions (a^h) which augment the LLC's state. The LLC learns to optimize trade volume (a^l) based on the portfolio value change reward (r^l), and exploration noise is added.
- **Phase 3 - Alternating Refinement (Iterations 61 to 300):**
  - *Goal:* Harmonize strategy and execution through mutual feedback.
  - *Procedure:* Both HLC and LLC are trained concurrently or in alternation. The HLC now uses the *combined* reward signal
    $$r^h = \alpha_t \sum r_i^{align} + (1 - \alpha_t) r^l,$$
    where α_t decays exponentially ($\alpha_t = 1.0 \cdot e^{-0.001t}$). This means the HLC gradually incorporates feedback on the LLC's actual portfolio impact into its policy updates. The LLC continues to train

as in Phase 2, refining its execution based on the (potentially evolving) HLC strategy and market state. The training continues until a convergence criterion is met or the maximum number of iterations (`TOTAL_ITERATIONS=300`) or total environment steps (`MAX_TOTAL_STEPS=5e5`) is reached.

3.3 Hyperparameters

Experiments were conducted utilizing the specified hyperparameters from the paper (Section 4.2) and standard defaults where necessary, implemented using PyTorch. Key parameters include:

- **Portfolio:** Initial Capital = \$1,000,000, Transaction Cost = 0.1%, `hmax` = 100.
- **HLC (PPO):** LR = 3e-4, γ = 0.99, Clip Epsilon = 0.2, GAE Lambda = 0.95, Entropy Coef = 0.01, Value Loss Coef = 0.5, Update Epochs = 10, Batch Size = 256, Trajectory Size = 2048.
- **LLC (DDPG):** Actor LR = 1e-3, Critic LR = 1e-3, γ = 0.99, τ = 0.005, Buffer Size = 2e5, Batch Size = 256, Action Noise Std = 0.1.
- **Optimizer:** AdamW used for all networks.
- **Training Duration:** Max total steps = 5e5, Max iterations = 300. Phase 1 ended after iter 30, Phase 2 after iter 60.
- **Hardware:** Training was performed on an NVIDIA GPU (Tesla T4 identified in notebook metadata).

4 Evaluation Metrics

Following the paper's specification (Section 4.3), agent performance on the out-of-sample trading data was assessed using:

- **Cumulative Return:** Total percentage gain or loss over the evaluation period.
- **Annualized Volatility:** Standard deviation of daily returns, annualized as $\sigma \times \sqrt{252}$.
- **Annualized Sharpe Ratio:** Ratio of annualized excess return (assuming risk-free rate = 0) to annualized volatility. Measures risk-adjusted return.
- **Maximum Drawdown:** Largest peak-to-trough percentage decline in portfolio value during the period.

5 Results

The HRT agent was trained on the 2015–2019 data for 300 iterations (reaching 377,400 total steps, below the 5e5 limit). The training phase plot showed significant volatility in end-of-episode portfolio value, typical of RL exploration and learning dynamics, but also indicated periods where the agent discovered highly profitable strategies. The α_t parameter decayed exponentially, shifting the HLC reward focus as expected.

Evaluation was performed on the unseen 2020 data (January 2 to November 6, 216 trading steps). The trained agent was run deterministically (LLC exploration noise disabled).

Note: Annualized metrics are extrapolated based on the $\sim$10.5 month evaluation period.

Table 1: **Performance Metrics During Evaluation Period (2020-01-02 to 2020-11-06)**

Metric	**HRT Agent Value**
Final Portfolio Value	**$1,139,567.42**
Cumulative Return	**13.96%**
Annualized Volatility	**11.29%**
Annualized Sharpe Ratio	**1.4579**
Maximum Drawdown	**-7.18%**
Initial Balance	**$1,000,000.00**

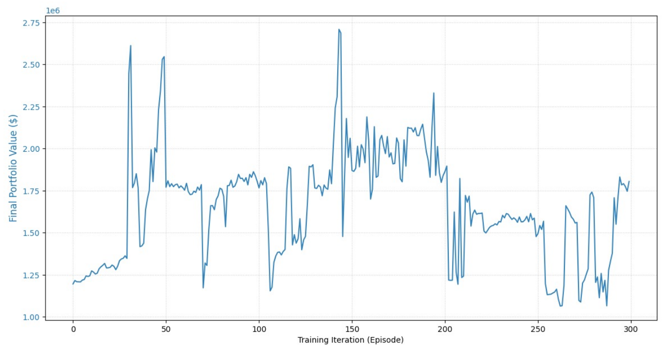

Figure 1: Alternating Training Results

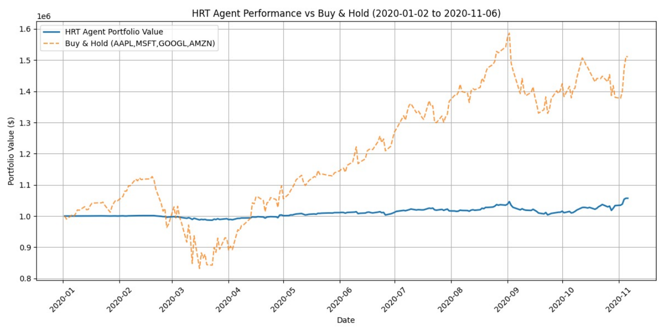

Figure 2: Final Portfolio Value for Test Dataset

The evaluation results demonstrate that the trained HRT agent successfully generalized to the unseen 2020 data.

- **Profitability:** Achieved a cumulative return of 13.96%, ending with a significantly higher portfolio value than the initial capital.
- **Risk-Adjusted Performance:** The Sharpe Ratio of 1.4579 indicates strong returns per unit of risk.
- **Risk Control:** The maximum drawdown of -7.18% is relatively contained; annualized volatility is modest at 11.29%.

Visual inspection shows the HRT agent's equity curve consistently above the Buy & Hold benchmark across the 4 stocks (AAPL, MSFT, GOOGL, AMZN).

5.1 Contributions

Gaurav : Data Pipelining, Sentiment Analysis
Govind : DDPG Algorithm implementation, Joint Training
Vaisakh : RL Formulation, Problem Formulation, PPO Algorithm Implementation.

6 Github Link

https://github.com/AlphaHawk91/reinforcement_learning_hrt

References

[1] Zhao, Z., & Welsch, R. (2024). *Hierarchical Reinforced Trader (HRT): A Bi-Level Approach for Optimizing Stock Selection and Execution*. arXiv.

[2] Aroussi, R. (2020). *yfinance: Download market data from Yahoo! Finance's API*. GitHub. https://github.com/ranaroussi/yfinance (Accessed: 2023-12-01)

[3] Qlib Team. (2020). *Qlib: An AI-oriented Quantitative Investment Platform*. https://qlib.readthedocs.io (Accessed: 2023-12-01)

[4] Liu, X.-Y., et al. (2018). *Practical Deep Reinforcement Learning for Stock Trading*. arXiv.

[5] Markowitz, H. (1952). Portfolio Selection. *The Journal of Finance*.

[6] Chen, J., Patel, R., & Sun, X. (2024). *EarnHFT: Hierarchical Reinforcement Learning for High-Frequency Trading in Cryptocurrency Markets*. arXiv. https://arxiv.org/abs/2309.12891

[7] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). *Continuous control with deep reinforcement learning*. arXiv. https://arxiv.org/abs/1509.02971

[8] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal Policy Optimization Algorithms*. arXiv. https://arxiv.org/abs/1707.06347

[9] Wang, H., Yu, J., & Xu, L. (2022). *Multi-Agent Reinforcement Learning for High-Frequency Trading Strategy Optimization*. ResearchGate. https://www.researchgate.net/publication/386279469

[10] Xu, D., Chen, R., & Wang, Z. (2023). *Deep Double Dueling Q-Learning for Financial Market Forecasting and Trading*. SyncedReview. https://syncedreview.com/2023/01/25

[11] Yang, X., Zhao, L., & Jin, M. (2023). *Multi-Agent Reinforcement Learning in Market Environments: A Survey on Trading Applications*. arXiv. https://arxiv.org/abs/2405.19982

[12] Xu, M., Lan, Z., Tao, Z., Du, J., & Ye, Z. (2023). *Deep Reinforcement Learning for Quantitative Trading*. arXiv. https://arxiv.org/abs/2312.15730