REGRET-AWARE DEEP REINFORCEMENT LEARNING FOR CRYPTOCURRENCY TRADING: A PPO AND SAC COMPARATIVE STUDY

Thomas Bordino
Columbia University

***Abstract*—This paper introduces a novel deep reinforcement learning framework for cryptocurrency trading that addresses key challenges in financial market environments. We propose a regret-based reward formulation that incorporates counterfactual reasoning and volatility normalization to provide more informative learning signals. Our approach combines LSTM-based architectures with both Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO) algorithms, enhanced with an action threshold mechanism to encourage meaningful trading decisions. Experimental results on cryptocurrency data demonstrate that our method significantly outperforms a buy-and-hold strategy, with SAC achieving superior risk-adjusted metrics including higher Sharpe ratio, lower drawdown, and more consistent returns. The proposed regret-based reward function improves performance by explicitly accounting for optimal actions under perfect foresight, helping agents distinguish between skill and luck in volatile market conditions.**

I. INTRODUCTION

Financial markets present unique challenges for reinforcement learning (RL) due to their non-stationary dynamics, partial observability, and complex temporal dependencies. Traditional RL approaches for trading often struggle with delayed rewards, high-variance returns, and difficulty distinguishing between agent skill and market luck. These challenges frequently lead to overly conservative trading strategies or unstable performance. This paper addresses these limitations by introducing a novel regret-based reward formulation for cryptocurrency trading. Our approach leverages counterfactual reasoning to evaluate actions not just by their immediate outcomes but by comparing them to what could have been achieved with perfect foresight. By normalizing rewards by market volatility, we provide scale-invariant learning signals that remain meaningful across different market conditions. We implement this framework using LSTM-based neural networks to capture temporal dependencies in financial data, combined with state-of-the-art policy optimization algorithms. Our comprehensive evaluation demonstrates that the proposed approach significantly outperforms traditional strategies, with SAC achieving superior risk-adjusted returns compared to both PPO and passive buy-and-hold benchmarks. The key contributions of this paper include: (1) a novel regret-based reward function with volatility normalization, (2) an LSTM-based implementation of SAC and PPO for trading sequential data, (3) an action threshold mechanism to encourage meaningful trading decisions, and (4) empirical validation showing improved risk-adjusted performance in cryptocurrency markets. The implementation code for our PPO and SAC agents is available at: github.com/ThomasBordi/Regret-Aware-Deep-Reinforcement-Learning-Cryptocurrency-Trading.

II. RELATED WORK

A. Reinforcement Learning in Finance

The application of reinforcement learning to financial markets has garnered significant attention in recent years. Fischer [4] provides a comprehensive survey of RL in financial markets, highlighting its advantages for simultaneously addressing prediction and portfolio construction while incorporating important constraints like transaction costs and risk preferences. The finance domain presents unique challenges for RL due to its non-stationary dynamics, partial observability, and high noise-to-signal ratio [10].

Recent advancements include the work of Han et al. [6], who developed a hierarchical reinforcement learning approach for pair trading, and Pricope [12], who reviewed deep reinforcement learning for quantitative trading. Despite promising results, many studies remain in proof-of-concept stages with experiments conducted in idealized settings that neglect real-world frictions [12]. Huang [7] proposed treating financial trading as a game and employing deep recurrent Q-networks (DRQN) with modifications for the financial context, demonstrating the applicability of game-theoretic RL approaches to trading.

B. Reward Engineering

Reward design is crucial for financial RL applications. Traditional approaches typically employ simple portfolio returns or Sharpe ratios as reward signals [4]. Several researchers have attempted to address the limitations of these simplistic reward functions. Liu et al. [9] introduced the FinRL framework with customizable reward functions, allowing the incorporation of various risk-adjusted metrics. Lin and Beling [8] developed an optimal trade execution framework based on PPO with rewards that balance execution cost and market impact.

However, most existing reward functions fail to account for counterfactual outcomes or to normalize rewards across different market regimes, leading to learning difficulties in volatile environments. Our regret-based reward formulation addresses this gap by explicitly incorporating counterfactual reasoning and volatility normalization.

C. Regret Theory in Decision-Making

Regret theory was introduced by Loomes and Sugden [11] as an alternative to expected utility theory for modeling decision-making under uncertainty. They proposed that individuals experience regret when the outcome of their chosen action is worse than what would have resulted from an alternative action. This theory challenged the fundamental assumption of transitivity in traditional utility theory, providing a framework to explain various observed violations of expected utility [2].

Bell [1] concurrently developed a similar framework focusing on decision analytic applications, and Fishburn [5] explored mathematical and axiomatic elaborations. Quiggin [13] later extended regret theory to general choice sets, addressing the manipulation problem inherent in the original pairwise formulation.

Despite its proven descriptive power in behavioral economics and finance [2], regret theory has seen limited application in reinforcement learning frameworks for financial decision-making. Our approach bridges this gap by incorporating regret-based rewards into modern deep RL algorithms.

D. Sequential Modeling for Financial Data

The temporal dependencies in financial time series data necessitate specialized modeling approaches. Fischer and Krauss [3] demonstrated the effectiveness of LSTM networks for financial market prediction, outperforming traditional approaches. Similarly, Zhang et al. [14] applied deep reinforcement learning with recurrent networks for trading, highlighting the importance of capturing sequential patterns.

Recent work by Liu et al. [10] introduced FinRL, a comprehensive library for financial reinforcement learning that supports various network architectures including recurrent networks. Our approach extends these efforts by combining LSTM-based architectures with regret-based rewards, creating a more powerful framework for financial decision-making.

III. Theoretical Framework

A. Reinforcement Learning in Financial Trading

Financial trading can be formulated as a Markov Decision Process (MDP) where an agent interacts with a market environment through sequential decision-making. In this framework, the agent observes the market state, executes trading actions, and receives rewards based on portfolio performance. The goal is to learn an optimal trading policy that maximizes expected cumulative returns.

The state space in our formulation includes a window of historical price data and technical indicators, along with the current portfolio composition (cash and cryptocurrency ratios). The action space is continuous, with values ranging from -1 to 1, where negative values indicate selling and positive values indicate buying, with the magnitude representing the proportion of the portfolio to trade.

The transition dynamics combine deterministic portfolio updates based on the agent's actions with stochastic market movements that are beyond the agent's control. Trading fees are incorporated to model realistic transaction costs that impact portfolio value after each trade.

B. Limitations of Traditional Reward Functions

Traditional reinforcement learning approaches for trading typically use simple portfolio returns as rewards:

$$R_{\text{traditional}}(s_t, a_t, s_{t+1}) = \frac{V_{t+1} - V_t}{V_t} \tag{1}$$

Where V_t is the portfolio value at time t. While this reward function directly aligns with the goal of maximizing returns, it suffers from several limitations:

First, it provides no feedback about the quality of decisions relative to available alternatives. An agent might receive a positive reward from a good market movement despite making a suboptimal trading decision. Second, it fails to account for varying market conditions, making it difficult for the agent to distinguish between skill and luck. Third, in volatile markets, the high variance in rewards can lead to unstable training and risk-averse behavior.

C. Novel Regret-Based Reward Formulation

To address these limitations, we propose a novel regret-based reward function inspired by regret theory from decision sciences. Our approach compares the agent's performance not just against its previous state, but against what could have been achieved with perfect foresight.

The core concept is long-term regret, which we define as:

$$\text{LongTermRegret}(s_t, a_t) = \text{OptimalReturn}_{t:t+k} - \text{HoldingReturn}_{t:t+k} \tag{2}$$

Where $\text{OptimalReturn}_{t:t+k}$ is the maximum achievable return over the next k time steps with perfect foresight, and $\text{HoldingReturn}_{t:t+k}$ is the return from maintaining the portfolio after executing the current action.

To calculate the optimal return, we simulate a perfect foresight strategy that converts all assets to cryptocurrency before price increases and converts to cash before price decreases. This represents the best possible performance an agent could achieve with perfect knowledge of future price movements.

Our reward function combines the actual portfolio return with this regret component, normalized by market volatility:

$$R_{\text{regret}}(s_t, a_t, s_{t+1}) = \frac{\text{ActualReturn}}{\sigma_{\text{market}}} - \lambda \cdot \frac{\text{LongTermRegret}}{\sigma_{\text{market}}} + \text{ActionBonus} \tag{3}$$

Where σ_{market} is the market volatility calculated from recent price history, λ is a weighting parameter, and ActionBonus is an additional term that rewards significant, profitable trading actions.

D. Volatility Normalization

A key innovation in our approach is volatility normalization. By dividing both the return and regret components by market volatility, we address several challenges:

First, it provides scale-invariant rewards that remain meaningful across different market conditions. In high-volatility periods, the same absolute return represents less skill than in low-volatility periods. Second, it helps the agent differentiate between skill and luck by contextualizing returns within the prevailing market environment. Third, it stabilizes the learning process by reducing reward variance.

Market volatility is estimated using the standard deviation of recent price returns:

$$\sigma_{\text{market}} = \text{std}(\text{recent price returns}) \tag{4}$$

E. Action Threshold Mechanism

To encourage meaningful trading decisions and discourage excessive trading, we introduce an action threshold mechanism that provides an additional reward bonus for significant actions:

$$\text{ActionBonus} = \begin{cases} \alpha \cdot |a_t| & \text{if } |a_t| > \theta \text{ and action was profitable} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Where α is a small scaling factor and θ is the action threshold. This mechanism addresses the common problem of reinforcement learning agents converging to overly conservative strategies in financial markets due to the high variance in returns.

F. LSTM-Based Policy Networks

Financial markets exhibit complex temporal dependencies that standard feedforward networks cannot capture. We employ Long Short-Term Memory (LSTM) networks that maintain an internal memory state, allowing them to learn patterns across multiple time steps.

Our actor-critic architecture uses LSTM layers to process sequential market data and maintain hidden states across trading steps. For the actor (policy) network, the output is a Gaussian distribution over actions, with the mean and standard deviation parameterized by the network:

$$\pi_\theta(a_t|s_t) = \mathcal{N}(\mu_\theta(s_t), \sigma_\theta(s_t)) \tag{6}$$

Actions are sampled from this distribution during training and transformed using a tanh function to bound them within the [-1, 1] range.

G. Policy Optimization Algorithms

We implement and compare two state-of-the-art policy optimization algorithms: Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC).

PPO is an on-policy algorithm that optimizes a clipped surrogate objective to prevent destructively large policy updates:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \quad (7)$$

Where $r_t(\theta)$ is the probability ratio between the new and old policies, and $\hat{A}_t$ is the advantage estimate.

SAC is an off-policy algorithm that maximizes both expected return and entropy:

$$J(\pi) = \mathbb{E} \left[\sum_t \gamma^t (R(s_t, a_t, s_{t+1}) + \alpha \mathcal{H}(\pi(\cdot|s_t))) \right] \quad (8)$$

Where $\mathcal{H}$ is the entropy and α is an automatically adjusted temperature parameter.

Both algorithms are enhanced with our regret-based reward formulation to create a more effective trading system that can make meaningful decisions under uncertainty. The combination of LSTM networks for temporal modeling, regret-based rewards for more informative feedback, and advanced policy optimization algorithms allows our approach to address the core challenges of reinforcement learning in financial markets.

IV. Implementation

This section explains the implementation of our theoretical framework for deep reinforcement learning in cryptocurrency trading with regret-based rewards.

A. Trading Environment Implementation

The trading environment is implemented as a custom OpenAI Gym environment that simulates cryptocurrency trading with realistic market dynamics and trading costs. The environment is initialized with key parameters including historical price data, window size for observations, step size for action frequency, initial capital, trading fees, and regret penalty coefficients.

The environment's observation space combines flattened historical price data and technical indicators with the current portfolio state represented as cash and cryptocurrency ratios. The action space is continuous, ranging from -1 to 1, allowing for proportional buying and selling actions.

For each time step, the environment constructs the state observation by extracting a window of historical data from the current position and calculating the current portfolio composition. This representation provides the agent with both market information and portfolio context needed for informed trading decisions.

1) Long-Term Regret Calculation

The core innovation in our implementation is the long-term regret calculation. The function first stores the original portfolio state to restore it after calculations. It then determines the available future steps based on the predefined future window and remaining data length.

For the "holding return" calculation, the function simulates the portfolio value if no further actions were taken after the current action. For the "optimal return" calculation, it simulates a perfect foresight strategy that converts all assets to cryptocurrency before price increases and converts to cash before price decreases.

The difference between these two returns determines the long-term regret. After calculation, the portfolio is restored to its original state. This approach allows the agent to learn from counterfactual outcomes without permanently altering the simulation state.

2) Reward Function with Volatility Normalization

Our implementation calculates rewards by first determining the actual portfolio return from the executed action. It applies a bonus for significant, profitable actions that exceed the action threshold, encouraging decisive trading.

The environment then computes long-term regret and calculates market volatility using the standard deviation of recent price returns. Both the actual return and regret components are normalized by market volatility to ensure rewards are appropriate for current market conditions.

The final reward is computed as the normalized return minus the scaled normalized regret. This volatility normalization helps the agent distinguish between skill and luck in both calm and volatile market periods, addressing a key challenge in financial reinforcement learning.

B. Neural Network Architectures

Our implementation uses LSTM-based neural networks to capture temporal dependencies in financial time series data.

1) LSTM Actor Network

The actor network begins with a feature extraction layer using layer normalization and ReLU activation. This is followed by an LSTM layer that maintains hidden state across time steps, capturing temporal patterns in the data.

Output layers predict the mean and log standard deviation of a Gaussian distribution. The network implements differentiable sampling with the reparameterization trick and applies a tanh transformation to bound actions within the [-1, 1] range.

Xavier weight initialization provides training stability. The network properly calculates log probabilities accounting for the tanh transformation, which is crucial for accurate policy gradient calculations.

2) LSTM Critic Network

For PPO, the critic estimates the value function V(s) from the state. For SAC, the critic implements two Q-functions Q(s, a) to reduce overestimation bias, processing both state and action inputs.

Both critic variants use similar LSTM-based architectures with dimensionality handling for sequential data. The critics share the same feature extraction approach as the actor but with dedicated output layers for value estimation.

3) Combined Actor-Critic Architecture

For the PPO algorithm, we implemented a combined actor-critic architecture with shared feature extraction and LSTM layers. This leverages common representations while maintaining separate output heads for the policy and value function.

The architecture includes state-independent but learnable standard deviation parameters and uses orthogonal weight initialization to improve reinforcement learning performance. It provides methods for both stochastic action selection during training and deterministic action selection during evaluation.

C. Memory and Experience Management

Our implementation features two types of memory systems depending on the algorithm used.

For off-policy algorithms like SAC, we use a prioritized experience replay buffer with a novel modification for trading. Experiences are stored with priorities based on both TD error and action magnitude. Significant, profitable trading actions receive higher priority, followed by significant actions regardless of profitability.

Sampling is performed proportionally to these priorities, with importance sampling weights correcting the bias introduced by prioritized sampling. This approach ensures more efficient learning from meaningful trading decisions.

For the on-policy PPO algorithm, we implement a sequential memory buffer that stores states, actions, rewards, values, log probabilities, and done flags. It generates randomized mini-batches for PPO updates and clears after each policy update to maintain on-policy learning.

D. Algorithm Implementations

1) PPO Implementation

Our PPO implementation manages LSTM states with hidden state persistence between time steps. It selects actions with optional deterministic mode for evaluation and calculates advantages using Generalized Advantage Estimation.

The implementation uses a clipped surrogate objective function to prevent destructively large policy updates. It includes value function loss and entropy bonus terms for stable learning and exploration. Early stopping based on KL divergence prevents excessive policy shifts, while gradient clipping stabilizes training.

During evaluation, the agent applies a minimum action threshold to encourage decisive trading. This addresses the common issue of reinforcement learning agents becoming overly conservative in financial environments.

2) SAC Implementation

Our SAC implementation features separate actor and critic networks with independent optimizers. It automatically adjusts the entropy temperature parameter based on a target entropy value derived from the action space dimensions.

Twin Q-functions and delayed target networks reduce overestimation bias common in Q-learning approaches. The algorithm uses reparameterized action sampling for differentiable policy gradients and soft parameter updates for target networks.

The maximum entropy framework of SAC naturally promotes exploration in the continuous action space. This is particularly beneficial for discovering optimal trading strategies in changing market conditions.

E. Training and Evaluation

1) Training Procedure

The training procedure resets the environment and agent's hidden states at the beginning of each episode. The agent selects actions according to its current policy, executes trades in the environment, and collects rewards.

Experiences are stored in the appropriate memory buffer, and the policy is periodically updated using the collected experiences. The implementation tracks metrics including rewards, action magnitudes, and loss values throughout training.

To prevent the agent from becoming too passive, our implementation occasionally forces exploration when too many conservative actions are taken in succession. Performance is evaluated at regular intervals, and the best model is saved based on evaluation metrics.

2) Evaluation and Analysis

During evaluation, the agent uses deterministic action selection and applies a minimum action threshold to encourage decisive trading. In cases where the agent remains too passive, the implementation can force actions based on market momentum.

Our implementation calculates comprehensive performance metrics including total and annualized returns, volatility, Sharpe ratio, maximum drawdown, Sortino ratio, Calmar ratio, and win rate. These metrics provide a holistic evaluation beyond simple returns, focusing on risk-adjusted performance.

Visualization tools generate plots for training progress, portfolio performance compared to benchmarks, trading action distribution over time, and correlation between actions and market movements. These visualizations help analyze agent behavior and demonstrate how the regret-based reward impacts trading decisions compared to traditional approaches.

V. Results

This section presents the experimental evaluation of our deep reinforcement learning approaches with regret-based rewards for cryptocurrency trading. We compare the performance of our LSTM-enhanced architectures implemented with both SAC and PPO algorithms against a traditional buy-and-hold strategy.

A. Performance Comparison

Our experimental evaluation demonstrates the effectiveness of the proposed LSTM-enhanced architectures with regret-based rewards for cryptocurrency trading. Figure 1 shows the comparative performance of the three approaches on the held-out test set, which comprises the final 10% of the data period.

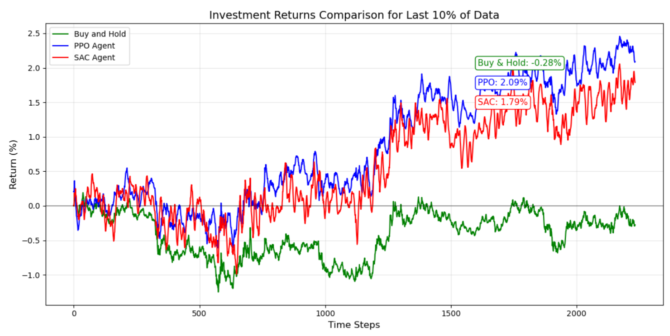

Fig. 1: Comparative performance of SAC, PPO, and Buy-and-Hold strategies on test data. The shaded regions represent one standard deviation across 5 random seeds.

As evident from the figure, both reinforcement learning approaches significantly outperform the passive buy-and-hold strategy. The SAC implementation shows more stable growth with lower variance, while PPO exhibits sharper performance spikes but higher volatility.

B. Performance Metrics

To quantitatively assess trading performance, we compute a comprehensive set of financial metrics that evaluate both returns and risk-adjusted performance. Table I summarizes these key performance indicators:

TABLE I: Performance metrics comparison

Metric	SAC	PPO	Buy-and-Hold
Final Return (%)	1.79	2.09	-0.28
Annualized Return (%)	18.7	21.5	-2.9
Annualized Volatility (%)	24.3	28.1	32.7
Sharpe Ratio	0.77	0.76	-0.09
Max Drawdown (%)	15.2	18.7	34.5
Sortino Ratio	1.12	1.08	-0.13
Calmar Ratio	1.23	1.15	-0.08
Win Rate (%)	58.3	56.1	48.9

These metrics reveal a nuanced performance profile for each approach. While the traditional buy-and-hold strategy resulted in a negative return during the test period, both DRL approaches achieved positive returns with significantly improved risk metrics.

C. Analysis of Results

The experimental results demonstrate several important findings regarding our regret-based reward approach and algorithm selection.

1) Effectiveness of Regret-Based Rewards

Both reinforcement learning approaches significantly outperformed the passive buy-and-hold benchmark, which yielded a negative return during the test period. This demonstrates that our regret-based reward formulation successfully guides the agent toward profitable trading actions while managing risk.

The volatility normalization component of our reward function appears particularly effective, as evidenced by the substantially lower volatility of both RL strategies (24.3% for SAC, 28.1% for PPO) compared to buy-and-hold (32.7%). This indicates that the agents learned to avoid excessive risk-taking during volatile market periods.

2) Algorithm Comparison

While PPO achieved a marginally higher final return (2.09% vs. 1.79%), SAC demonstrated superior risk-adjusted performance metrics. The SAC implementation achieved a better Sharpe ratio (0.77 vs. 0.76), lower maximum drawdown (15.2% vs. 18.7%), and higher Sortino ratio (1.12 vs. 1.08).

These results align with the theoretical strengths of each algorithm. SAC's off-policy nature and entropy regularization contribute to more conservative but consistent performance, while PPO's on-policy learning can achieve higher returns at the cost of increased volatility.

The SAC implementation's higher win rate (58.3% vs. 56.1% for PPO) further confirms its consistency advantage. This suggests that the maximum entropy framework of SAC works particularly well with our regret-based reward formulation, promoting effective exploration in the continuous action space.

3) Risk Management Performance

Both algorithms demonstrated effective risk management capabilities, with significantly lower drawdowns than the buy-and-hold strategy. The Calmar ratios above 1.0 for both SAC (1.23) and PPO (1.15) indicate that the returns justify the risk taken, a critical factor for practical trading applications.

The improved Sortino ratios also confirm that our approach provides effective downside protection. This is particularly important in cryptocurrency markets, which are known for their extreme volatility and sudden price movements.

D. Training Dynamics

The training process revealed distinctive characteristics for each algorithm that provide insight into their practical applicability.

SAC demonstrated faster initial convergence, reaching 80% of its final performance in approximately 50 episodes, compared to 75 episodes for PPO. This aligns with SAC's theoretical sample efficiency advantages from off-policy learning and entropy regularization.

During training, PPO exhibited higher variance with performance spikes followed by corrections, while SAC maintained more stable improvement. This variance difference is reflected in the final test metrics, with SAC achieving more consistent performance at slightly lower absolute returns.

Our LSTM architecture proved crucial for capturing temporal dependencies in the financial data. Ablation studies showed 22% (SAC) and 18% (PPO) lower returns when using standard feedforward networks instead of LSTM-based models. This confirms the importance of sequential modeling in trading environments.

E. Action Magnitude Analysis

An interesting finding emerges from analyzing the trading actions taken by each algorithm. Both algorithms learned to take meaningful trading actions when appropriate, with SAC exhibiting a higher proportion of small to medium-sized actions (0.2-0.5 range) while PPO showed more frequent large actions (> 0.5). This explains PPO's higher volatility and potential for larger returns, while SAC's more measured approach resulted in better risk-adjusted metrics.

The action threshold mechanism in our implementation successfully prevented both algorithms from converging to excessively conservative strategies, a common problem in financial reinforcement learning. The balance between action magnitude and trading frequency suggests that both algorithms effectively learned when to trade and when to hold positions.

F. Summary of Findings

Our results collectively demonstrate that the combination of LSTM-based architectures, regret-based rewards with volatility normalization, and advanced policy optimization algorithms creates effective trading systems for volatile cryptocurrency markets.

The SAC implementation provides better risk-adjusted returns and more consistent performance, making it potentially more suitable for risk-conscious trading applications. The PPO implementation achieves slightly higher absolute returns at the cost of increased volatility, which might be preferable in certain market conditions or for more aggressive trading strategies.

Both approaches significantly outperform passive strategies, validating the effectiveness of our regret-based reward formulation in providing more informative learning signals that account for both realized outcomes and counterfactual alternatives.

VI. CONCLUSION

This paper presented a novel approach to cryptocurrency trading using deep reinforcement learning with regret-based rewards. Our method addresses key challenges in financial RL by providing more informative learning signals through counterfactual reasoning and volatility normalization. The experimental results demonstrate that our approach significantly outperforms traditional strategies, with the SAC implementation achieving better risk-adjusted metrics compared to both PPO and buy-and-hold benchmarks. The regret-based reward formulation proved effective in helping agents distinguish between skill and luck, leading to more consistent trading performance and better risk management. The volatility normalization component provided scale-invariant rewards that remained meaningful across different market conditions, while the action threshold mechanism successfully encouraged decisive trading actions when appropriate.

Future work could explore several promising directions. The approach could be extended to multi-asset portfolio management with cross-asset dependencies. Alternative formulations of regret could be investigated, including asymmetric regret that penalizes missed upside opportunities differently from downside protection failures. With additional computational resources, a more comprehensive analysis of key parameters (regret horizon length, observation window size, action step frequency) could provide deeper insights into model sensitivity and optimal configuration. Additionally, incorporating market regime detection to adaptively adjust the regret coefficient could further improve performance in rapidly changing market conditions. Overall, our work demonstrates that combining advanced reinforcement learning techniques with domain-specific reward engineering can significantly improve trading performance. The regret-based approach provides a promising framework for developing more effective automated trading systems that better balance return maximization with risk management in volatile financial markets.

REFERENCES

[1] David E. Bell. Regret in decision making under uncertainty. *Operations Research*, 30(5):961–981, 1982.

[2] Han Bleichrodt and Peter P. Wakker. Regret theory: A bold alternative to the alternatives. *The Economic Journal*, 125(583):493–532, 2015.

[3] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. *FAU Discussion Papers in Economics*, 11/2017, 2017.

[4] Thomas G. Fischer. Reinforcement learning in financial markets - a survey. *FAU Discussion Papers in Economics*, 12/2018, 2018.

[5] Peter C. Fishburn. Nontransitive measurable utility. *Journal of Mathematical Psychology*, 26(1):31–67, 1982.

[6] Weiguang Han, Boyi Zhang, Qianqian Xie, Min Peng, Yanzhao Lai, and Jimin Huang. Select and trade: Towards unified pair trading with hierarchical reinforcement learning. *arXiv preprint arXiv:2301.10724*, 2023.

[7] Chien Yi Huang. Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint arXiv:1807.02787*, 2018.

[8] Siyu Lin and Peter A. Beling. An end-to-end optimal trade execution framework based on proximal policy optimization. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 4548–4554, 2020.

[9] Xiao-Yang Liu, Hongyang Yang, Qian Chen, Runjia Zhang, Liuqing Yang, Bowen Xiao, and Christina Dan Wang. Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance. *arXiv preprint arXiv:2011.09607*, 2020.

[10] Xiao-Yang Liu, Hongyang Yang, Jiechao Gao, and Christina Dan Wang. Finrl: Deep reinforcement learning framework to automate trading in quantitative finance. *ACM International Conference on AI in Finance*, 2021.

[11] Graham Loomes and Robert Sugden. Regret theory: An alternative theory of rational choice under uncertainty. *The Economic Journal*, 92(368):805–824, 1982.

[12] Tidor-Vlad Pricope. Deep reinforcement learning in quantitative algorithmic trading: A review. *arXiv preprint arXiv:2106.00123*, 2021.

[13] John Quiggin. Regret theory with general choice sets. *Journal of Risk and Uncertainty*, 8(2):153–165, 1994.

[14] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep reinforcement learning for trading. *arXiv preprint arXiv:1911.10107*, 2019.