On Adaptive Reinforcement Learning in High-Frequency Trading

Chi-Lin Li

1 ABSTRACT

This research introduces a novel approach that applies an adaptive Reinforcement Learning Model to High Frequency Trading. We integrate the Double Q-Network (DQN) framework with the Gated Recurrent Unit (GRU) for the purpose of data encoding. Specifically, we propose an innovative dynamic sliding window algorithm, which is designed to adapt to the constantly changing market conditions. This adaptation is facilitated by utilizing *Sortino ratio* as a key metric to guide trading decisions. Through comprehensive empirical studies, which take into account turnover and transaction costs, we show that this adaptive GRU-DQN approach yields computational benefits and promising trading performances.

2 INTRODUCTION

High Frequency Trading (HFT) represents a significant paradigm shift in financial markets, driven by advancements in technology and algorithmic trading. Characterized by its rapid execution speed and sophisticated algorithms, HFT has become a dominant component of modern financial ecosystems, contributing substantially to market liquidity and efficiency [1, 2]. However, alongside its benefits, HFT poses challenges such as increased market volatility and concerns about market fairness [3]. These intricacies necessitate advanced analytical approaches, with an emerging focus on leveraging machine learning techniques to navigate the complexities of this high-speed trading domain [4]. The incorporation of such technologies is critical for developing effective trading strategies that can respond dynamically to the fast-paced changes in market conditions [5].

Machine learning's integration into HFT machine learning enhances HFT by rapidly deciphering complex data. Random Forests [6] reduce overfitting and manage large datasets , while LSTM [7] networks capture critical temporal patterns. LightGBM and XGBoost [8, 9] offer fast, scalable, and accurate modeling suited for the HFT's intense data environment [10]. Reinforcement learning (RL) [11] is integral to HFT due to its ability to adapt and make strategic decisions in dynamic markets [12]. Its model-free nature enables learning from historical data, crucial in the unpredictable trading environment.

Q-learning, a cornerstone in reinforcement learning, offers a framework for learning optimal policies in environments with stochastic transitions and rewards. It focuses on learning a value function that can predict the quality of actions in given states [13]. Building upon Q-learning, Deep Q-Networks (DQN) integrate deep neural networks to handle complex, high-dimensional spaces typical in HFT [14]. DQN's architecture, featuring experience replay and fixed Q-targets, addresses stability and convergence challenges in traditional Q-learning, making it highly effective for the dynamic and data-rich domain of financial markets [15].

However, the properties of financial data haven't been fully exploited. Gated Recurrent Unit (GRU) are pivotal in HFT for feature extraction from time-series data, as they adeptly capture temporal dependencies critical for market prediction [16]. Their gating mechanisms effectively mitigate the vanishing gradient problem, allowing them to maintain relevant information over extended sequences, which is essential in the fast-paced HFT environment [17]. The high-level features extracted by GRUs enhance the state representations for reinforcement learning models like DQN, thereby improving the decision-making process in trading systems [14].

In this research, we employed 5-second interval high-frequency data to fine-tune the GRU-DQN model for stock trading, a choice that leverages the GRU's proficiency in capturing rapid market movements. This data granularity enhances the model's responsiveness and accuracy in developing trading strategies. Importantly, our approach accounts for transaction costs, ensuring a more realistic assessment of trading profitability. We have also incorporated the Sortino ratio as the reward function in our DQN model, emphasizing the optimization of risk-adjusted returns while focusing specifically on downside risk. Additionally, the implementation of sliding windows in data processing allows for dynamic updating of the model, ensuring that it adapts to new market information and remains relevant over time. These combined elements —short-interval data, transaction cost consideration, Sortino ratio-based rewards, and sliding windows —are crucial in crafting precise and adaptive trading strategies suited to the volatile nature of the stock market.

3 PRELIMINARIES

This section introduces fundamental financial metrics and concepts essential to understanding our research's framework and analysis. Consider a stock with a time series of prices, containing n total number of observations. The price at each time point is represented as S_i, for $i = 1, 2, 3, \dots, n$. The total return r^p is defined as: $\left(\frac{S_n - S_1}{S_1}\right) \times 100$, where S_n represents the stock price at the final time point. S_i is the stock price at the initial time point. The annualized Sharpe ratio SR is defined as: $\frac{\bar{r} - r_f}{\sigma}$, where $\bar{r} - r_f$ represents the excess mean of the returns and σ is the volatility.

Moreover, to research the downside risks, we take d^* to be the maximum percentage drawdown. Consider a time series of an investment's value represented by V_i for $i = 1, 2, \dots, n$. The maximum drawdown d^* is defined as $\max_{1 \le t \le n} \left(\max_{1 \le k \le t} \left(\frac{V_t - V_k}{V_k} \right) \right)$.

4 PROBLEM FORMULATION

Q-learning, a foundational algorithm in reinforcement learning (RL), is well-regarded for its simplicity and versatility [13]. However, it faces challenges in high-dimensional or continuous spaces, suffering from slow convergence and limited state generalization [18, 19]. To address these drawbacks, Deep Q-Networks (DQN) were developed, integrating deep learning with Q-learning to efficiently handle complex environments [14]. DQN's advanced techniques, like experience replay and fixed Q-targets, enhance learning stability and convergence, making it more suitable for dynamic scenarios such as high-frequency trading [20].

DQN leverages neural networks to approximate the Q-function, enabling the handling of the multi-dimensional state spaces prevalent in HFT. Key features of DQN include experience replay and fixed Q-targets, which enhance stability and convergence in environments with high variance and uncertainty:

$$Q(s,a;\theta) \leftarrow Q(s,a;\theta) + \alpha \left[r + \gamma \max_{a'} Q(s',a';\theta') - Q(s,a;\theta) \right] \tag{1}$$

where θ represents the parameters of the neural network, and θ' are the parameters of a target network used to stabilize training. The DQN algorithm learns the approximated action-value function $Q(s,a;\theta)$ The learning process involves iteratively updating θ to minimize the loss function $L(\theta)$, which is defined as:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim U(D)} \left[\left(r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta) \right)^2 \right] \tag{2}$$

where (s,a,r,s') represents a transition tuple in the replay buffer D, consisting of the current state s, action a, reward r, and next state s'. γ is the discount factor, determining the importance of future rewards. θ^- represents the parameters of a target network, which is a copy of the DQN network and is updated less frequently. $U(D)$ denotes a uniform distribution over the replay buffer D. The use of a replay buffer and a target network are key innovations in DQN, helping to stabilize the learning process. The replay buffer stores transitions that are then sampled randomly, breaking the correlations in the observation sequence. The target network provides a fixed target for the Q-value updates, reducing oscillations and divergence in the learning process.

In the application of DQN to HFT, the choice of the reward function is critical. Employing the Sortino ratio as the reward function provides a risk-adjusted measure of return, its focus on downside risk, aligning with the primary objective of maximizing returns while minimizing losses [21]. This approach is advantageous in HFT environments, where managing downside risk is crucial [22]. The Sortino Ratio enables the DQN to prioritize strategies that balance higher risk-adjusted returns with the inherent volatility of financial markets [14], making it a practical and effective measure for financial trading algorithms [23]. The Sortino ratio-based reward is formulated as:

$$STR = \frac{E(r) - rf}{D(r)} \tag{3}$$

where $E(r)$ is the expected return, r_f is the risk-free rate, and $D(r)$ is the downside deviation for $r < 0$. Thus, DQN, augmented with a Sortino ratio-based reward function, presents a robust framework for developing effective trading strategies in the dynamic and complex environment of HFT.

To enhance the processing of sequential time-series data essential in high-frequency trading, Gated Recurrent Units (GRU) are integrated into the DQN framework. This integration capitalizes on GRU's ability to capture temporal patterns and address vanishing gradients, crucial for decoding complex market dynamics [17]. The synergy of GRU with DQN not only facilitates informed trading decisions but also adapts to changing market trends, creating a robust and adaptive model [14].

The GRU is an architecture within recurrent neural networks designed to process sequential data more effectively. The GRU addresses the vanishing gradient problem commonly found in standard RNNs by using gating mechanisms. These mechanisms control the flow of information within the unit and are defined mathematically as follows: Given a sequence of input vectors $\mathbf{X} = \{x_1, x_2, \ldots, x_T\}$, where T denotes the sequence length, the GRU updates its hidden state h_t at each time step t using:

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{4}$$

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{5}$$

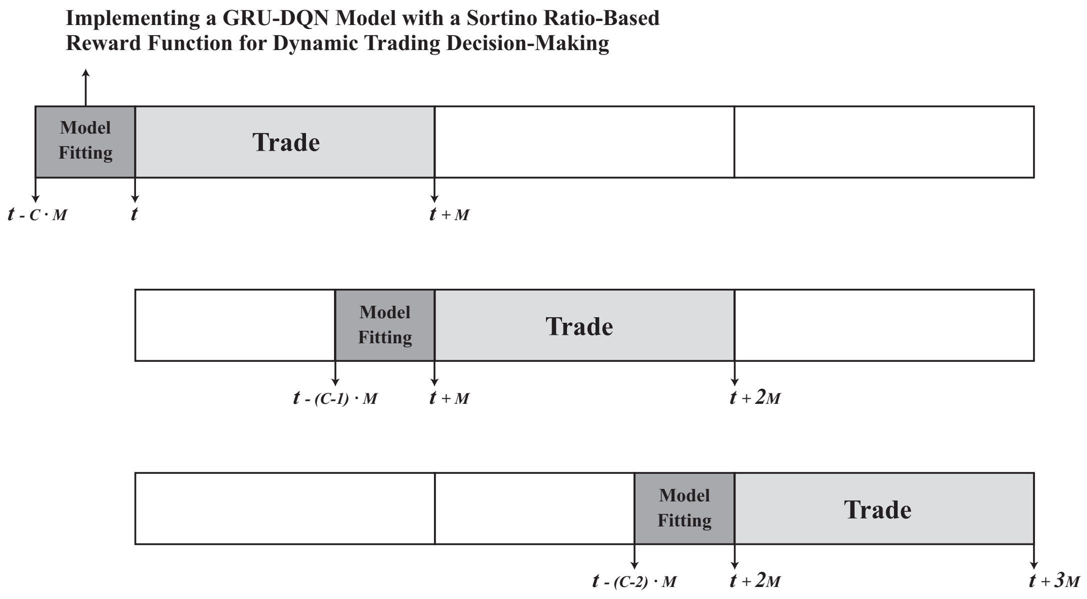
$$\tilde{h}_t = \tanh(W_h x_t + r_t \odot (U_h h_{t-1}) + b_h) \tag{6}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{7}$$

Here, r_t and z_t are the reset and update gates, respectively, which are vectors controlling how much of the past information to forget and to pass to the future. W_r, W_z, W_h, U_r, U_z, and U_h are weight matrices, b_r, b_z, and b_h are bias vectors, and σ denotes the sigmoid function. The operator $\odot$ represents element-wise multiplication. This formulation allows the GRU to effectively capture dependencies in sequential data, addressing challenges like varying time lags between important events in a time series.

To address the estimation error, we adopt the sliding window techniques, which can be shown in Figure 1. The sliding window technique was employed to analyze time-series data, enabling dynamic model updating and enhanced decision-making accuracy in the fast-paced environment of high-frequency trading [24]. The total pipeline can be shown in Algorithm 1.

Figure 1: Idea of Generating Time-Varying Trading Actions: Dynamic Sliding Window



Algorithm 1 Adaptive GRU-DQN Algorithm

```
Require: Consider an initial window size $M \geq 1$, parameters $\phi$ of GRU model, deep Q-network
    with parameters $\theta$ and $\theta'$, and Sortino ratio threshold $\tau$.
Ensure: Action $a_t$.
 1: At initial time stamp $t \geq 0$, collect $M$ historical stock prices' data
        $$S := (s(t), s(t-1), \dots, s(t-(M-1)))$$
 2: Add transformation features and technical indicator into data $S$.
 3: Encode data $S$ using GRU model by layers 4 to 7, obtaining the hidden state $h_t$ and $h_{t+1}$.
 4: Use data $S$ to calculate Sortino ratio $STR$ as reward and generate action $a_t$ according to the
    relationship between $STR$ and Sortino ratio threshold $\tau$.
 5: Employ the hidden state $h_t$ to calculate the DQN model using parameter $\theta$, resulting in $Q(s, a; \theta)$.
    Subsequently, use the next hidden state $h_{t+1}$ to assess the target DQN model characterized by
    parameter $\theta'$, thereby deriving $Q(s, a; \theta')$. This process informs the subsequent trading decision.
 6: if $t < t + M$ then
 7:     Set $t := t + M$ and go back to Step 1.
 8: else
 9:     Get the previous portfolio volatility $\sigma^*$ from $t - c \cdot M$ to $t$, where constant $c \in (0, 1)$
10:     if $\sigma \geq \sigma^*$ then
11:         $\tau \leftarrow c_+ \cdot \tau$ where constant $c_+ \geq 1$
12:     else
13:         $\tau \leftarrow c_- \cdot \tau$ where constant $c_- \in (0, 1)$;
14:     end if
15:     Set $t := t + M$ and go back to Step 1.
16: end if
```

5 EMPIRICAL STUDIES

The empirical analysis commenced with a comprehensive dataset[1], comprising high-frequency trading records of three prominent equities: Apple, JPMorgan Chase, and Sony. The dataset from September 1, 2023, to October 31, 2023, with granular data captured at 5-second intervals. This fine resolution provided a detailed temporal sequence of price movements, allowing for an intricate examination of market behavior and the efficacy of algorithmic trading strategies. Additionally, the analysis incorporated a realistic transaction cost of 1%[2], reflecting the maximum costs encountered in live trading environments. This inclusion ensures that the results of the trading strategies are not only theoretically sound but also practically viable when considering the cost implications of

[1]The data, encompassing various types such as midpoint, bid, and ask prices, as well as the bid-ask spread and trading volume, is retrieved from Interactive Brokers.

[2]Some brokerage services, such as Interactive Brokers, impose a maximum transaction fee rate of 1% per order on the trade value. The fee structure is outlined on their pricing page, see https://www.interactivebrokers.com/en/pricing/commissions-stocks.php.

executing trades in the real world.

In the feature engineering stage of this analysis, a strategic approach was adopted, implementing a subset of transformational techniques to enhance data's analytical utility. Logarithmic transformations, serving to stabilize variance and address positive skewness, brought the data closer to a normal distribution [25]. Square transformations were utilized to mitigate left skewness, improving data symmetry and interpretability [26]. Additionally, the Modified Box-Cox transformation was employed for its adaptability in normalizing data across various skewness types, ensuring robustness in subsequent modeling [27]. These targeted transformations were pivotal in refining data attributes, aligning them with the underlying assumptions of the analytical models.

In Figure 5, the Adaptive GRU-DQN model shows a distinct outperformance in capital balance over time compared to the other strategies, which exhibit more fluctuations and generally lower returns. And in the Table 1, The Adaptive GRU-DQN model significantly outperforms other strategies in AAPL performance metrics, with a positive average return of 7.51% and the highest Sharpe Ratio of 2.26, indicating better risk-adjusted returns, despite a slightly lower maximum drawdown percentage d^* compared to some models.

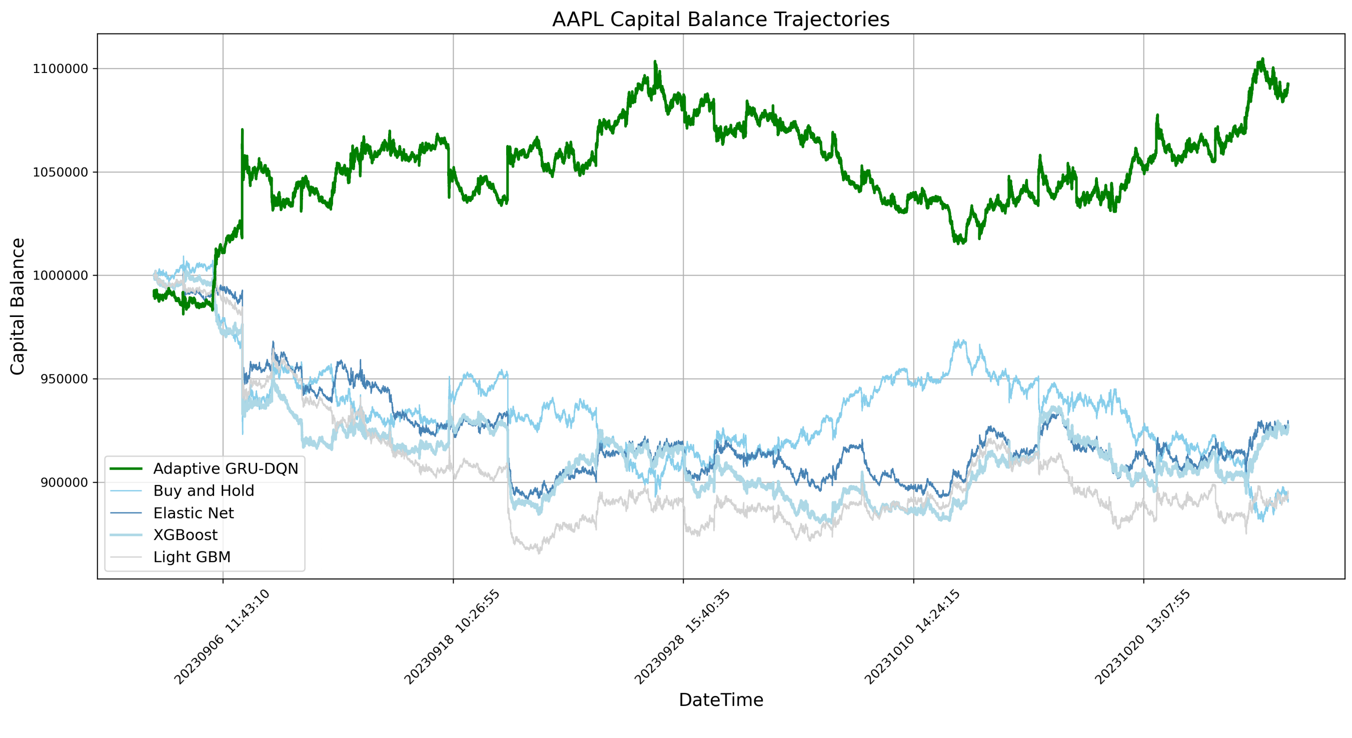


Figure 2: AAPL Capital Balance Trajectories

In Figure 5, despite the Adaptive GRU-DQN Model showing underwhelming performance, it achieves the highest total return r^p and Sharpe Ratio SR, coupled with the lowest maximum drawdown, indicating a strong risk-adjusted return profile.

The JPM Capital Balance Trajectories 5 illustrates that, at the conclusion of the simulation, the Adaptive GRU-DQN Model outperforms other strategies, culminating in victory and boasting the highest Sharpe Ratioo SR 1.28, signifying its superior risk-adjusted performance.

	r^p (%)	σ	SR	d^* (%)
Adaptive GRU-DQN	**7.51**	**0.00022**	**2.26**	**11.19**
Buy and Hold	-10.58	0.00022	-2.85	12.73
Elastic Net	-6.98	0.00019	-2.16	11.02
XGBoost	-7.30	0.00020	-2.18	12.18
Light GBM	-10.38	0.00019	-3.28	13.66

Table 1: AAPL Performance Metrics

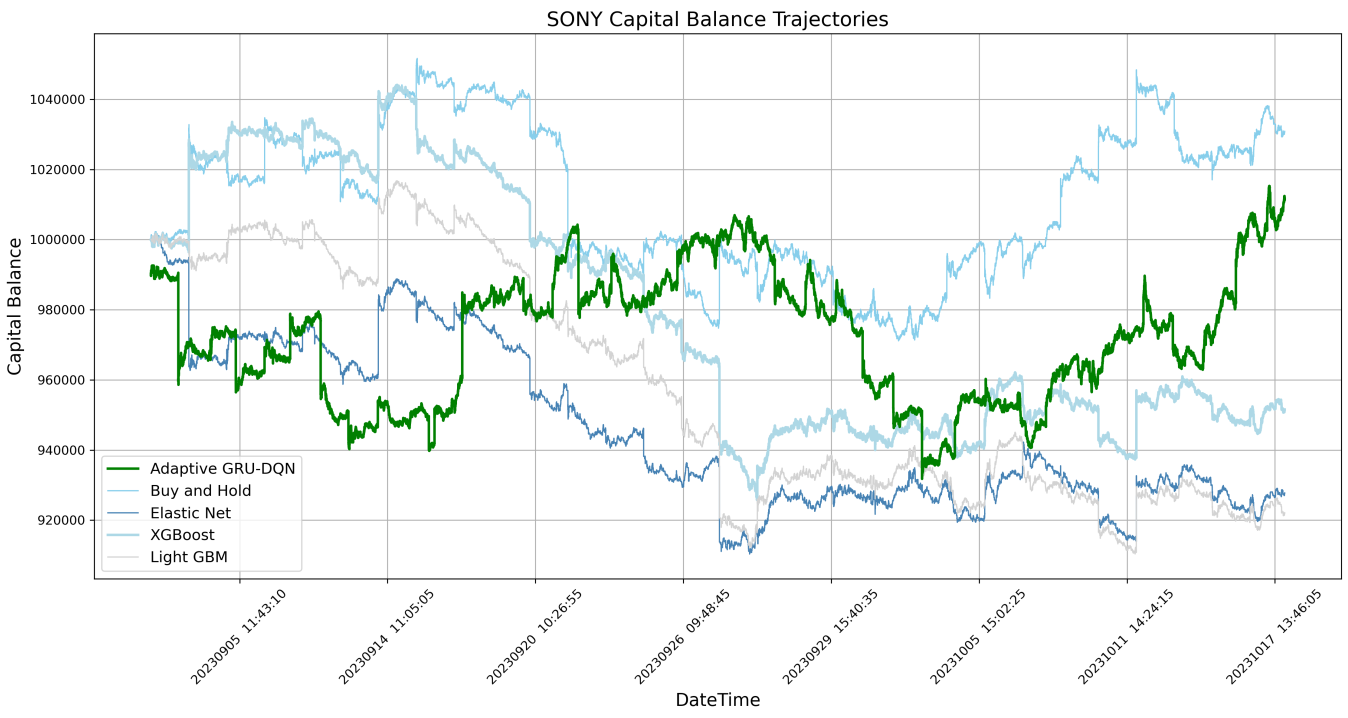


Figure 3: SONY Capital Balance Trajectories

Overall, the Adaptive GRU-DQN Model demonstrates consistent stability across various equity portfolios. Its superior performance evidences a robust adaptability to diverse market conditions.

6 CONCLUDING REMARKS

The development of the adaptive GRU-DQN model marks a notable advancement in algorithmic trading. Outperforming traditional machine learning methods in analyzing high-frequency data, our Algorithm 1 is enhanced by incorporating the Sortino ratio as the reward and including sliding window algorithm in evaluations. Proven effective in complex market navigation, the model shows potential for wider financial applications.

The future trajectory for the GRU-DQN model in high-frequency trading involves several key areas of enhancement. Initially, delving into higher frequency data, such as milliseconds, can offer more granular insights into market dynamics [4]. Additionally, broadening the scope to include a diverse range of financial instruments, from global stocks to commodities and foreign exchange, will

	r^p (%)	σ	SR	d^* (%)
Adaptive GRU-DQN	**1.41**	**0.00022**	**0.96**	**8.60**
Buy and Hold	-3.81	0.00019	-1.10	9.00
Elastic Net	-7.45	0.00016	-2.83	9.49
XGBoost	-5.80	0.00016	-2.12	11.93
Light GBM	-9.74	0.00014	-4.19	13.48

Table 2: SONY Performance Metrics

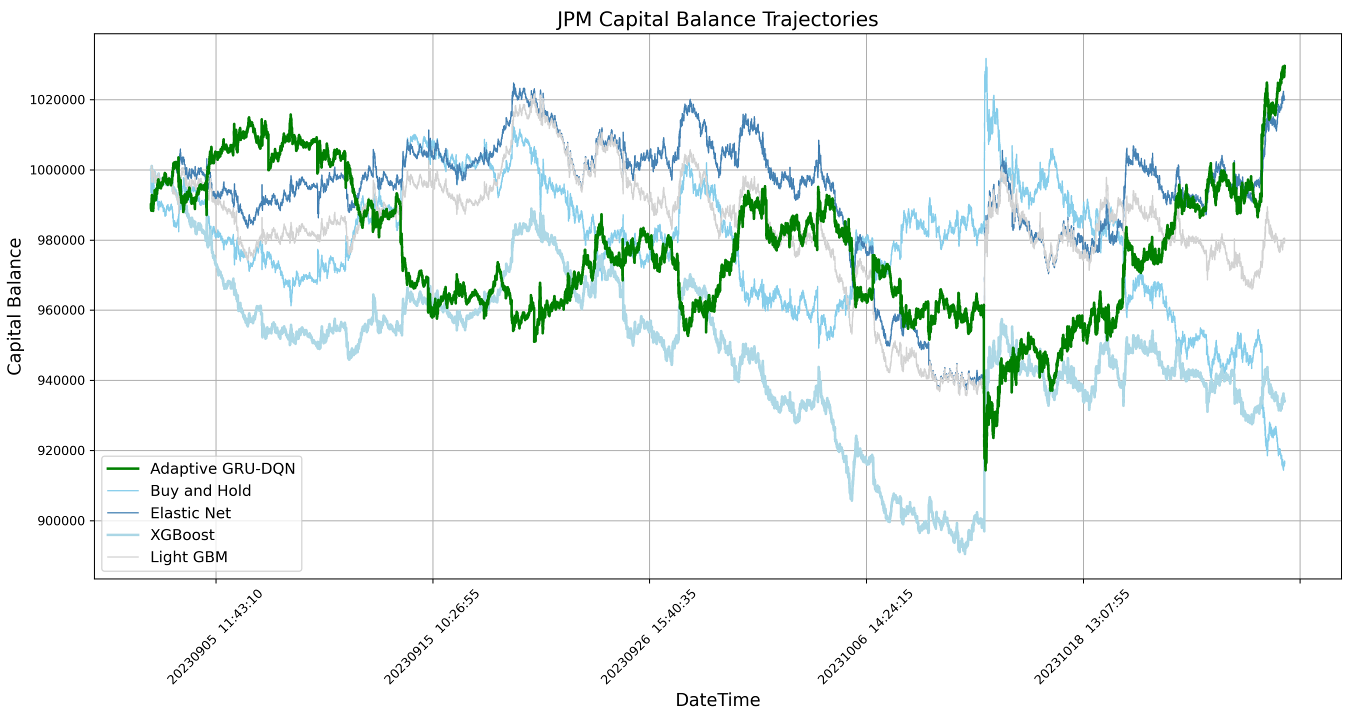


Figure 4: JPM Capital Balance Trajectories

test and potentially enhance the model's adaptability [28]. Extending training periods to encapsulate longer market cycles can provide a richer understanding of enduring trends and fluctuations [29]. Advancing the model could involve exploring newer reinforcement learning algorithms such as Proximal Policy Optimization (PPO) [30] and Trust Region Policy Optimization (TRPO) [31]. These techniques, known for effectively managing large action spaces, could enhance the model's stability and responsiveness, offering refined trading strategies in the dynamic financial market.

REFERENCES

[1] Joel Hasbrouck and Gideon Saar. "Low-latency trading". In: *Journal of Financial Markets* 16.4 (2013), pp. 646–679.

[2] Albert J Menkveld. "The economics of high-frequency trading: Taking stock". In: *Annual Review of Financial Economics* 8 (2016), pp. 1–24.

	r^p (%)	σ	SR	d^* (%)
Adaptive GRU-DQN	**2.93**	**0.00019**	**1.28**	**11.50**
Buy and Hold	-8.54	0.00018	-2.68	11.70
Elastic Net	2.30	0.00015	0.93	8.65
XGBoost	-6.47	0.00015	-2.52	11.04
Light GBM	-2.55	0.00015	-0.90	8.41

Table 3: JPM Performance Metrics

[3] Albert S Kyle, M Samadi, and T Tuzun. "The flash crash: The impact of high frequency trading on an electronic market". In: *Manuscript, U of Maryland* (2011).

[4] Irene Aldridge. *High-frequency trading: a practical guide to algorithmic strategies and trading systems.* Vol. 604. John Wiley & Sons, 2013.

[5] Bruno Biais, Thierry Foucault, and Sophie Moinas. "Equilibrium fast trading". In: *Journal of Financial economics* 116.2 (2015), pp. 292–313.

[6] Leo Breiman. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32.

[7] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[8] Guolin Ke et al. "Lightgbm: A highly efficient gradient boosting decision tree". In: *Advances in neural information processing systems* 30 (2017).

[9] Hongge Chen et al. "Robust decision trees against adversarial examples". In: *International Conference on Machine Learning.* PMLR. 2019, pp. 1122–1131.

[10] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.* 2016, pp. 785–794.

[11] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[12] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[13] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[14] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *nature* 518.7540 (2015), pp. 529–533.

[15] Guillaume Lample and Devendra Singh Chaplot. "Playing FPS games with deep reinforcement learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence.* Vol. 31. 1. 2017.

[16] Kyunghyun Cho et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation". In: *arXiv preprint arXiv:1406.1078* (2014).

[17] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[18] Thomas G Dietterich. “Ensemble methods in machine learning”. In: *International workshop on multiple classifier systems*. Springer. 2000, pp. 1–15.

[19] Francisco S Melo. “Convergence of Q-learning: A simple proof”. In: *Institute Of Systems and Robotics, Tech. Rep* (2001), pp. 1–4.

[20] Hado Van Hasselt, Arthur Guez, and David Silver. “Deep reinforcement learning with double q-learning”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 30. 1. 2016.

[21] Frank A Sortino and Lee N Price. “Performance measurement in a downside risk framework”. In: *the Journal of Investing* 3.3 (1994), pp. 59–64.

[22] Con Keating and William F Shadwick. “A universal performance measure”. In: *Journal of performance measurement* 6.3 (2002), pp. 59–84.

[23] Bruce Vanstone and Gavin Finnie. “An empirical methodology for developing stockmarket trading systems using artificial neural networks”. In: *Expert systems with Applications* 36.3 (2009), pp. 6668–6680.

[24] João Gama et al. “A survey on concept drift adaptation”. In: *ACM computing surveys (CSUR)* 46.4 (2014), pp. 1–37.

[25] Jason Osborne. “Improving your data transformations: Applying the Box-Cox transformation”. In: *Practical Assessment, Research, and Evaluation* 15.1 (2010), p. 12.

[26] Gary F Templeton. “A two-step approach for transforming continuous variables to normal: implications and recommendations for IS research”. In: *Communications of the association for information systems* 28.1 (2011), p. 4.

[27] Remi M Sakia. “The Box-Cox transformation technique: a review”. In: *Journal of the Royal Statistical Society Series D: The Statistician* 41.2 (1992), pp. 169–178.

[28] Matthew F Dixon, Igor Halperin, and Paul Bilokon. *Machine learning in finance*. Vol. 1170. Springer, 2020.

[29] Iqbal H Sarker. “Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions”. In: *SN Computer Science* 2.6 (2021), p. 420.

[30] John Schulman et al. “Proximal policy optimization algorithms”. In: *arXiv preprint arXiv:1707.06347* (2017).

[31] John Schulman et al. “Trust region policy optimization”. In: *International conference on machine learning*. PMLR. 2015, pp. 1889–1897.