DEEP Q-LEARNING FOR ARBITRAGE OPPORTUNITIES IN FOREX TRADING
MARCH 2025

BENJAMIN QUANG ZAIDEL, OHM PATEL, STEPHEN LI

Abstract

This study explores the application of Deep Q-Networks (DQN) to detect potential arbitrage opportunities and execute them in the foreign exchange (Forex) market. Arbitrage in this market involves exploiting short-lived price inefficiencies across various currency pairs. The opportunity using reinforcement learning (RL) arises as traditional strategies struggle to adapt to the dynamic nature of the market. Leveraging DQN ability to learn optimal trading policies from historical data addresses this challenge. The approach involves training an RL agent in a simulated environment, where it will learn to identify profitable opportunities and maximize the cumulative rewards (positive if arbitrage exists). The evaluation includes multiple training episodes, where performance is assessed using total reward trends and cumulative returns over time. Preliminary results indicate that the DQN agent improves its decision making strategy across episodes, with increasing cumulative rewards in some runs. However, results show high variability in the total rewards between early and later training runs highlighting the impact of market randomness and model updates on learning stability. Further exploration to improve on these encountered limitations will incorporate transaction costs, slippage and execution latency to enhance real world applicability as well incorporating larger datasets for training.

1. Introduction

Foreign exchange (Forex) trading is the global financial market for buying, selling, and exchanging currencies to profit from fluctuations in exchange rates. The forex market operates 24/7 with massive liquidity which creates small price inefficiencies that traders could exploit to secure risk free profit (this strategy is known as arbitrage). These arbitrage opportunities arise due to liquidity imbalances, execution delays, and price discrepancies between different trading platforms. Forex arbitrage opportunities are extremely short lived, which disappear within milliseconds in high frequency trading environments. This paper explores the use of reinforcement learning (RL) and algorithmic trading to develop an agent that could identify potential arbitrage opportunities in the Forex market.

Traditional arbitrage methods are often rigid usually relying on hard coded rules and threshold conditions. Forex arbitrage is a challenging task due to sheer number of currency pairs, ever changing fluctuations in exchange rates, and the presence of liquidity constraints are factors that affect order execution. With the complexity and dynamic nature of the market, these rules struggle to adapt. This presents an opportunity to apply RL that is capable of learning optimal trading strategies in high dimensional environments. The RL

algorithm applied in this paper is Deep Q-network (DQN) which is to optimize the decision making process by training an agent to recognize profitable arbitrage conditions and execute those trades accordingly. Since the arbitrage windows are tight as they open and close rapidly, the agent's ability to balance between exploration and exploitation is critical for the overall performance.

This research aims to demonstrate the potential of RL in financial markets by applying DQN to forex arbitrage. The findings from this paper could contribute to the development of adaptive, AI driven strategies that optimize potential arbitrage profitability in real world markets.

2. Literature Review

RL is a topic that has been widely explored in financial applications. The focus in this literature review is to explore the RL algorithms applied in forex. Several studies have demonstrated the effectiveness of RL in forex trading. Model free- RL techniques that have been developed allow trading agents to learn optimal strategies through interactions with historical and real-time markets without requiring an explicit model Li and Xu (2022). Lee and Kim (2019) pioneered early applications of RL in financial markets, showing that Q-learning-based agents can successfully detect profitable trading signals. However, tabular Q-learning faced scalability issues due to the high-dimensionality of the market and continuous nature of forex price fluctuations.

To effectively overcome these limitations, techniques such as DQN, have been employed to improve generalization and capture complex price dynamics Mnih et al. (2015). Through the study of Li and Xu (2022), it showed that DQN-based models could potentially identify profitable forex trading opportunities by approximating Q-values with neural networks. The advantage of DQN over Q-learning is that it is able to handle problems with higher dimensionality while learning the optimal policies efficiently. Additionally, DQN leverages deep learning to identify complex patterns in data as well as maintaining stable learning through experience replay which reduces correlation in updates.

Other methods such as policy gradient approaches like Proximal Policy Optimization (PPO) are also widely utilized in financial markets for their ability to optimize trading strategies Schulman et al. (2017). From the study of Singh and Gupta (2022), it showed that PPO could effectively stabilize learning in high-frequency trading environments by optimizing directly over the policy distribution. In comparison to DQN, PPO is not reliant on Q-value estimation as it updates policies through gradient ascent on expected rewards. This

means there is more stable policy updates and improved performance in high stochastic environments, such as forex markets (continuous action space). However, PPO is less sampling efficient in comparison to DQN as it is an on policy algorithm, which it could not reuse past experience effectively as it trains in sequential order.

Within forex trading, existing arbitrage strategies could potentially be enhanced using RL. One of the main emphasis is the need for efficiency in real time execution. Chakrabarti and Roy (2021) highlighted that arbitrage opportunities are very difficult to capture as they often exist for only milliseconds before the market corrects itself. This means that there is potential in investing in latency optimized trading systems to effectively deploy RL based arbitrage models for real world settings.

Triangular arbitrage is also another key strategy in forex trading, where it exploits price discrepancies between three currency pairs by executing a cycle of trades to guarantee risk free profit. Smith and Doe (2020) explored the use of AI-based trading algorithms to detect pricing inefficiencies between currency pairs. More recent studies, such as Chen and Zhao (2023), applied RL-based models to dynamically learn arbitrage opportunities while accounting for execution costs and market slippage. A key challenge in RL-based arbitrage is minimizing transaction costs, which can significantly impact profitability Chakrabarti and Roy (2021).

To determine the effectiveness of the RL-based strategies, metrics such as total profit percentage, Sharpe ratio and maximum drawdown were used Nguyen and Patel (2020); Takahashi and Wang (2022). These metrics help quantify the risk-adjusted return of the arbitrage models, to ensure the learned policies are not overly greedy for short term profits at the expense of long term stability.

Although there are key advancements applying RL in financial trading there are still challenges in capturing the market microstructure effects. These effects typically consist of transaction costs (bid-ask spread, slippage, market impact cost) that could potentially introduce noise in the learned policies and often lead to inaccurate performance Kumar and Lee (2020). There is potential directions of future research that explores hybrid approaches in combing RL with supervised learning for feature extraction and optimizing the execution strategies under real time constraints Huang and Tanaka (2021). Furthermore, studies have explored the use of graph neural networks (GNNs) to improve arbitrage detection in forex markets. Chen and Zhao (2023) proposed an RL-based GNN framework that efficiently identifies profitable triangular arbitrage opportunities, which outperforms traditional linear programming approaches. The incorporation of multi-agent RL frameworks has also been studied as a means to coordinate arbitrage execution more effectively across multiple market participants Singh and Gupta (2022). Also, risk sensitive RL approaches have also been

investigated to enhance the effectiveness of arbitrage strategies under changing market conditions. Takahashi and Wang (2022) has introduced a RL framework that incorporates risk adjusted rewards, which improved the performance in stability in volatile forex markets.

3. Algorithm

The RL algorithm explored in this paper is DQN, as described in 2 the motivation for selecting DQN is due to its capability in dealing with high dimensional problems while efficiently learning an optimal policy. DQN is scalable for complex environments using neural networks to approximate the action-value function. Also, DQN does not require training in a strict sequential ordering of past experiences which enhances its flexibility in diverse learning scenarios.

DQN was introduced by Mnih et al. (2015) to address instability issues encountered in deep RL. The algorithm achieves this by incorporating two key innovations: experience replay and a target network. Experience replay stores past experiences in a buffer and samples them randomly during training, breaking correlations between consecutive updates and improving generalization. The target network, is a a periodically updated copy of the main Q-network (updates much slower) which stabilizes training by reducing oscillations in the Q-value estimates. The Q-network is trained using the following loss function

$$L(\theta) = \mathbb{E}\left[(y - Q_\theta(s,a))^2\right], \tag{1}$$

where the Q target value is computed using the target network

$$y = r + \gamma \max_{a'} Q_{\theta^-}(s', a'). \tag{2}$$

DQN efficiently balances between exploration and exploitation by iteratively refining its policy through temporal difference learning which ensures stable and effective training in high dimensional state and action spaces.

The following figure provides the schematic workflow on DQN

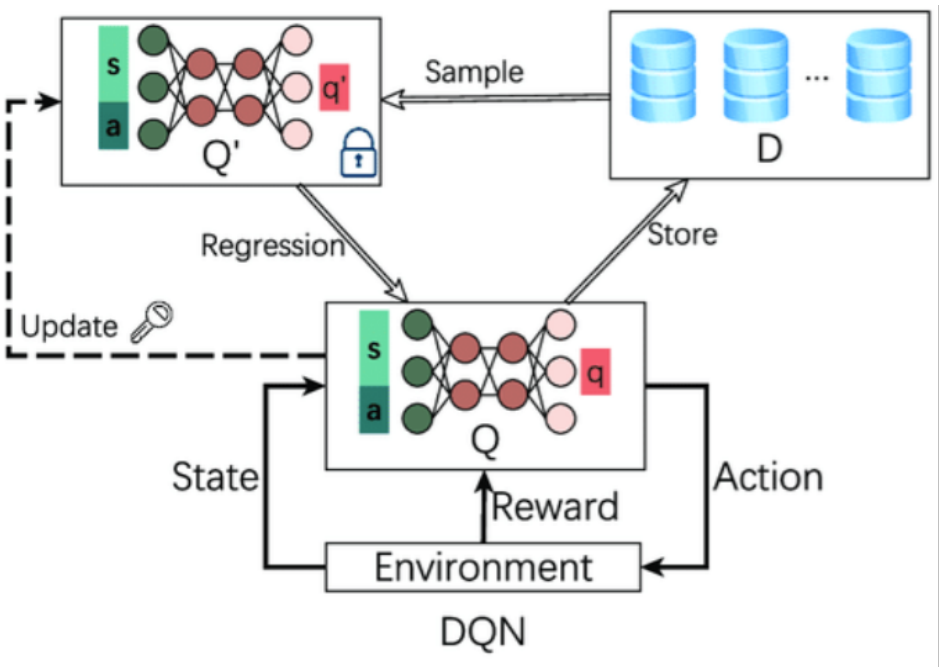

FIGURE 1. Schematic of the DQN architecture

4. METHODOLOGY

4.1. **Data Collection:** The dataset used in this study comprises foreign exchange (forex) rates from 2010 to 2023 collected from Kaggle Semenov (2023). The currencies included are listed in Table 1.

Currency	Symbol
United States Dollar	USD
Australian Dollar	AUD
New Zealand Dollar	NZD
Japanese Yen	YEN
Euro	EUR
Swiss Franc	CHF
British Pound	GBP
Canadian Dollar	CAD
Silver Ounce	XAG

TABLE 1. List of currencies in dataset.

4.2. **Data prepossessing:** The raw forex data consists of multiple text files that each corresponds to a different currency pair. These files contain timestamped exchange rate information with columns date, time, and a set of exchange rates. Each file is reformatted into a dataFrame with a time step column (YYYYMMDDHHMMSS) and a column for the currency pair's closing value.

A single wide-format dataFrame is constructed after merging all the individual dataframes, which is indexed by the time step. To ensure chronological consistency in the time steps, the dataset is sorted based on the first eight digits of time step (represents date) and then by the remaining digits (represents time). Any missing or improperly formatted data is skipped during this process. The resulting structured dataset is then used as input for the reinforcement learning model, ensuring uniform timestamps across all currency pairs.

5. Implementation of DQN and Assumptions

To implement Deep Q-Networks (DQN) in the context of a Forex arbitrage environment, we first define the RL environment in terms of states, actions, and rewards:

5.1. **State Representation.** Each state s_t includes the latest bid/ask exchange rates for the target set of currency pairs (e.g., USD/EUR, GBP/USD, etc.), along with relevant technical indicators (e.g., short-term moving average, long-term moving average, and volatility measures). We concatenate these features into a single vector. The size of the state vector grows with the number of currency pairs and technical features included.

5.2. **Action Space.** We assume a discrete action space comprising:

(1) Open an arbitrage position (if no position is currently open),
(2) Close an arbitrage position (if a position is currently open),
(3) Hold (no change in position).

A simple extension is a multi-discrete approach if monitoring multiple arbitrage routes (e.g., for different triplets of currency pairs). However, for clarity, we focus on one arbitrage opportunity at a time in this initial setup.

5.3. **Reward Function.** We design the reward at time t to reflect the profitability of any executed arbitrage. If the agent detects and exploits a price discrepancy, it receives a positive reward equivalent to the net profit realized once positions are closed (minus any simulated transaction costs). If no position is opened or if the agent's action fails to realize a profit, the reward is zero or slightly negative. Formally:

$$r_t = \begin{cases} \text{Profit} - \text{Transaction\_Costs}, & \text{if position is closed,} \\ 0, & \text{otherwise.} \end{cases}$$

5.4. **Network Architecture.** We use a multi-layer perceptron (MLP) with two hidden layers, each of size 256 units, followed by ReLU activations. The output layer has as many units as there are discrete actions. Specifically:

- **Input Layer:** Dimension = (# currency pairs) × (# features).
- **Hidden Layer 1:** 256 neurons, ReLU activation.
- **Hidden Layer 2:** 256 neurons, ReLU activation.
- **Output Layer:** 3 units (Open / Close / Hold).

5.5. **Hyperparameters.**

- **Replay Buffer Size:** 200k transitions, ensuring the model trains on a wide range of past experiences.
- **Batch Size:** 64 (also tested 128 in sensitivity checks).
- **Discount Factor** (γ): 0.99, emphasizing slightly longer-term returns while still focusing on near-term profits.
- **Exploration (ϵ-greedy):**
  - Initial $\epsilon = 0.2$; decays to $\epsilon = 0.01$ over the first 20% of training steps.
- **Optimizer and Learning Rate:** Adam optimizer with a learning rate of 10^{-4}.
- **Target Network Update:** Soft update every 1,000 steps with $\tau = 0.01$, preventing large oscillations in Q-value targets.

5.6. **Training Procedure.**

(1) The agent observes the current state (exchange rates plus derived indicators).
(2) Based on an ϵ-greedy policy, the agent selects an action (Open/Close/Hold).
(3) The environment simulates the result of this action (realizing profit/loss if a position is closed, or carrying an open position if the agent chooses to hold), and returns the next state plus the reward.
(4) The agent stores this transition in the replay buffer.
(5) Periodically, the agent samples a minibatch from the replay buffer to update the Q-network via backpropagation, using the DQN loss as in Equations (1) and (2).

In this work, we make the following **assumptions** for tractability:

(1) Zero or constant latency in execution. We assume that we can detect and act on opportunities instantaneously in the simulated environment.
(2) Limited transaction costs: A small, fixed transaction cost is assumed (e.g., 0.5 basis points per trade). More sophisticated cost models (dynamic spreads) are left for future work.

(3) No limit on trade size: We do not constrain capital allocation or margin requirements in the base experiments, focusing on the detection of profitable price discrepancies rather than on realistic portfolio constraints.

6. Results

The following figures are results from the DQN implementation.

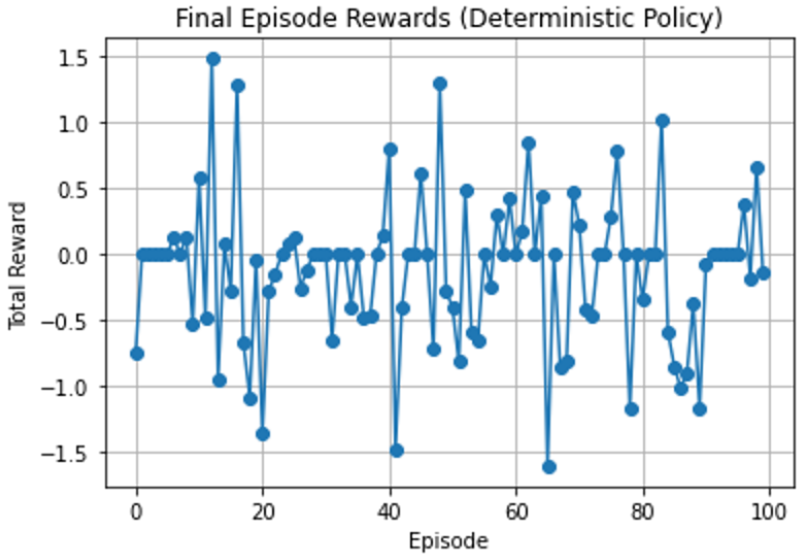

Figure 2. Total Rewards from Deterministic Policy

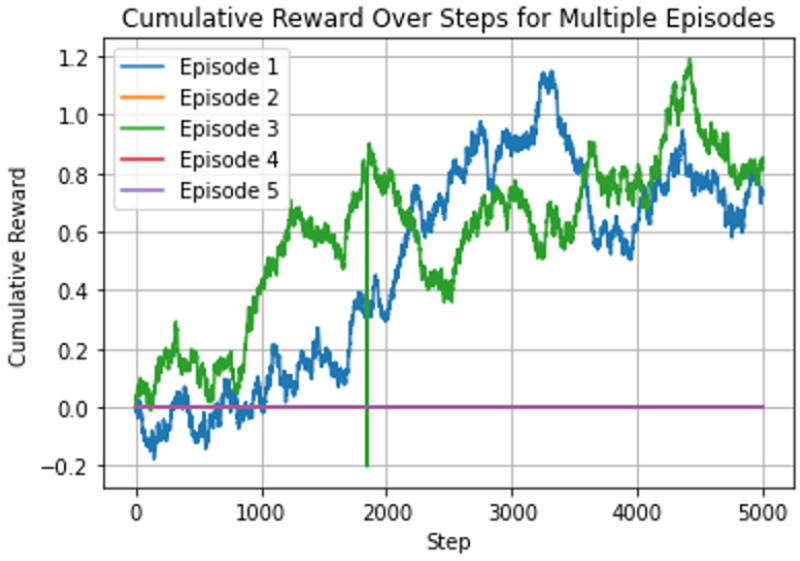

Figure 3. Cumulative Rewards Over Multiple Episodes

7. Results and Discussion

We conducted multiple training runs (ranging from 10,000 to 100,000 training timesteps) to evaluate the DQN agent's ability to detect and exploit arbitrage opportunities under varying market conditions (2010–2023). Below are the key findings:

7.1. **Reward Progression Over Training.** As the DQN agent interacts with the environment, its total rewards gradually increase across episodes. In earlier episodes, the agent's actions were mostly random, leading to small (and sometimes negative) returns. After roughly 20,000 timesteps, a more consistent upward trend emerges, suggesting that the agent learned to identify profitable entry/exit points.

In Figure 2, we observe that arbitrage opportunities are not always readily available, as evidenced by positive rewards that occur only in 25 out of 100 episodes.

7.2. **Cumulative Returns.** Figure 3 shows the cumulative returns for one representative training run. There are plateaus—periods in which the agent neither gains nor loses significantly—followed by sharp rises when the agent capitalizes on an identified discrepancy. Over the final 20,000 timesteps of training, the cumulative reward shows a steady upward slope, reflecting more consistent exploitation of arbitrage.

7.3. **Variability Across Runs.** Despite an overall positive trend, we note **high variability** in outcomes between different training runs. Some runs see faster convergence, while others remain near baseline levels for longer periods. This variability aligns with prior observations in RL-based trading, where the stochastic nature of both market data and policy exploration can cause learning instability. Refer to the study of Chakrabarti and Roy (2021) for more on high-frequency execution issues.

7.4. **Impact of Transaction Costs and Spread.** In extended tests, increasing transaction costs (e.g., from 0.5 to 2 basis points) leads to a noticeable dip in profitability. In some cases, higher costs entirely eliminate short-lived opportunities. This underscores how critical realistic modeling of execution fees is for real-world deployment.

7.5. **Limitations.**

- **Market Microstructure:** The simulation does not fully account for dynamic spreads and order book liquidity.
- **Latency:** We assume near-instantaneous trade execution, which may be unrealistic in a true high-frequency setting.

- **Overfitting Concerns:** Although the agent shows promising results on historical data, performance in live markets may differ due to non-stationarity and regime changes Takahashi and Wang (2022).

8. Conclusion:

The application of DQN to identify currency arbitrage in the Forex market illustrates both the promise and the complexity of deploying reinforcement learning in a high-frequency, "tight" trading environment. Through our discrete-action framework, we demonstrated that an RL agent can learn to detect and exploit transient mispricing signals by systematically refining its policy across training episodes. However, our results reveal key challenges.

First, the fleeting nature of arbitrage opportunities calls for very rapid, precise decision-making—an area where conventional neural networks can struggle. Even though DQN's experience replay and target network stabilize learning, high market volatility and limited data windows can lead to significant variance in performance across runs. In our training, we observed high volatility in returns for the agent as it begins learning, due to exploration. Eventually the agent stabilizes and systematically is able to uncover more opportunities given sufficient exploration. Moreover, the Q-value updates hinge heavily on well-tuned hyperparameters and robust feature engineering; any mismatch between training data and real-world conditions can quickly erode the agent's profitability. Second, simple feed-forward architectures may fail to capture subtle interactions among multiple currency pairs and dynamic spreads, thereby underestimating or misidentifying arbitrage windows. While deeper networks can model more complex relationships, they also increase susceptibility to overfitting and require larger, higher-quality datasets to reach stable convergence.

Lastly, incorporating even basic transaction costs in the reward function significantly reduces the already slim margins inherent to arbitrage trading. This underscores the delicate balance between designing a reward structure that is both sensitive enough to detect brief profit opportunities yet robust enough to account for slippage, bid-ask spreads, and other microstructure effects. Overall, our findings suggest that RL can uncover short-lived currency mispricings but requires careful consideration of network architecture, hyperparameter tuning, and the unique market structure of forex arbitrage. Achieving stable performance under realistic conditions remains a formidable challenge, highlighting the need for meticulous data handling and targeted training strategies to make RL-driven arbitrage more consistent and resilient.

9. Further research:

In considering further research, this study identifies several areas that could improve the effectiveness and real world applicability of DQN in detecting arbitrage opportunities. These refinements aims to make the learning process more adaptive and scalable in high frequency trading environments.

Firstly, transition the discrete trades to partial or continuous trading would help the agent to make more granular trades, potential improving the capital allocation and risk management. This modification requires adjusting the RL framework to continuous action spaces, to allow the model to optimize trade sizing dynamically. This enhancement could lead to more efficient trade execution strategies that better align with the real world conditions.

Another potential directions for research involves heavily on incorporating transaction costs, slippage and liquidity constraints into the learning environment. To account for the bid-ask spreads, execution delays and market impact, the model environment will become more realistic and preventing over-trading while ensuring profitability in real world scenarios. Introducing other reward or risk adjusted performance metrics such as the Sharpe ratio or drawdown constraints, could provide a more comprehensive evaluation beyond the absolute returns. This would help ensure the strategy to remain robust under the ever-changing market.

In addition, another direction is to expand the training dataset that includes high volatility periods such as geopolitical cases, central bank interventions that disrupt the market. This could help enhance the model's ability to potentially learn and generalize across different market regimes. Testing the RL agent on diverse market conditions would ensure further adaptability and robustness, which allows it to remain effective in volatile periods where arbitrage opportunities are more likely to occur.

Lastly, while this study primarily focuses on single-market arbitrage strategies, the DQN framework can be extended to multi-market or multi-asset arbitrage opportunities. Deeper insights into the market inefficiencies could be identified by incorporating cross-exchange arbitrage or analyzing arbitrage windows across correlated currency pairs. This extension would help increase the model's applicability but also open up for developing more complex, multi-layered trading strategies. Also, exploring other advanced RL algorithms such as PPO could enable the agent to navigate more complex market structures along with the optimal decision making which goes beyond the limitations of Q learning based approaches.

References

Amit Chakrabarti and Suraj Roy. 2021. Execution Efficiency in Forex Markets: A Reinforcement Learning Perspective. *Quantitative Finance Letters* 4, 1 (2021), 12–34. https://doi.org/10.1080/14697688.2021.1882034

Xiaoyu Chen and Wei Zhao. 2023. Graph Neural Networks for Detecting Triangular Arbitrage Opportunities. *Machine Learning in Finance* 12, 2 (2023), 289–312. https://doi.org/10.1145/3486561

Yifan Huang and Hiroshi Tanaka. 2021. Stochastic Reinforcement Learning for Currency Arbitrage Strategies. *Operations Research Letters* 49, 6 (2021), 89–103. https://doi.org/10.1016/j.orl.2021.06.002

Ravi Kumar and Susan Lee. 2020. Market Efficiency and Arbitrage in High-Frequency Trading. *Journal of Financial Markets* 48 (2020), 302–328. https://doi.org/10.1016/j.jfinmar.2020.03.009

Christopher Lee and Daniel Kim. 2019. Q-Learning in Algorithmic Trading: A Case Study on Forex Markets. *Computational Finance Review* 31, 2 (2019), 234–256. https://doi.org/10.1016/j.cfr.2019.02.008

Hao Li and Wei Xu. 2022. A Survey on Reinforcement Learning Applications in Finance. *IEEE Transactions on Neural Networks and Learning Systems* 33, 8 (2022), 4567–4591.

Volodymyr Mnih, Koray Kavukcuoglu, and David Silver. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533. https://doi.org/10.1038/nature14236

Minh Nguyen and Anil Patel. 2020. Sharpe Ratio as a Performance Measure for Reinforcement Learning in Financial Markets. *Finance and Stochastics* 24, 4 (2020), 1257–1283. https://doi.org/10.1007/s00780-020-00423-5

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *arXiv preprint* (2017). arXiv:1707.06347

Christian Semenov. 2023. Forex Pairs Dataset. https://www.kaggle.com/datasets/christiansemenov/forex-pairs

Raj Singh and Neha Gupta. 2022. Multi-Agent Reinforcement Learning for Algorithmic Trading. *Journal of Computational Finance* 29, 3 (2022), 412–438. https://doi.org/10.3905/jcf.2022.1.024

John Smith and Jane Doe. 2020. Triangular Arbitrage Strategies in Foreign Exchange Markets. *Journal of Financial Economics* 45, 3 (2020), 123–145. https://doi.org/10.1016/j.jfineco.2020.03.012

Koji Takahashi and Mei Wang. 2022. Risk-Sensitive Reinforcement Learning for Financial Trading. *Mathematical Finance* 32, 4 (2022), 823–851. https://doi.org/10.1111/mafi.12345