International Journal of Artificial Intelligence,
ISSN 0974-0635; Int. J. Artif. Intell.
2025, Volume 23, Issue Number 1

RL4RF as a Reinforcement Learning Framework for a Realistic Forex Environment

Youness BOUTYOUR [1] **and Abdellah IDRISSI** [2]

Intelligent Processing Systems and Security (IPSS) Team
Mohammed V University
Rabat, MOROCCO
[1] youness.boutyour@um5r.ac.ma

[2] idrissi@um5r.ac.ma

ABSTRACT

The paper introduces RL4RF, a novel framework designed to address the shortcomings of existing environments in Reinforcement Learning (RL) for Forex trading. RL4RF offers a comprehensive and authentic simulation of the Forex market, mitigating limitations associated with data diversity and market dynamics representation. Comprising distinct layers, including data handling, RL algorithms, and an environment model, RL4RF facilitates the utilization of various RL algorithms and data sources. Notably, RL4RF incorporates essential aspects such as transaction costs, market volatility, and fundamental factors that influence exchange rates. Its data layer efficiently manages CSV files, internet data retrieval, and synthetic dataset generation, enhancing the study of diverse market patterns. The RL algorithms layer accommodates multiple RL techniques, fostering adaptability to various trading scenarios. The environment layer of RL4RF faithfully replicates realistic trading conditions, encompassing factors like decimal precision, spread, lot size, initial balance, tick value, take profit, stop loss, and graphical representations of trading activities. This framework significantly enhances the feasibility of RL-based trading strategies, enabling effective generalization to real-world market conditions. In this paper, we present the methodology, implementation specifics, and RL algorithms employed within RL4RF. Furthermore, we provide illustrative examples of RL4RF's application in studying diverse market scenarios, showcasing its potential to revolutionize RL-based Forex trading strategies.

Keywords: Artificial Intelligence, Algorithmic Trading, Autonomous Agent, Deep Reinforcement Learning, Financial Market, Forex Environment.

2012 Computing Classification System: Computing methodologies →Machine learning →Learning paradigms →Reinforcement learning →Sequential decision making.

1 Introduction

Reinforcement Learning (RL) has shown great promise in developing effective trading strategies for financial markets, including the FOReign EXchange (Forex) market (Loh, Kueh, Parikh, Chan, Ho and Chua, 2022). However, the success of RL algorithms in Forex trading heavily

depends on the quality of the trading environment used for training the models (Théate and Ernst, 2021).

Existing Forex trading environments used in RL suffer from various limitations that restrict their effectiveness in practical trading scenarios. These limitations include unrealistic market dynamics, lack of necessary features, and unrealistic trading conditions (Buehler, Horvath, Lyons, Perez Arribas and Wood, 2020). These limitations can lead to models that perform well in the training stage but fail to generalize to real-world market conditions.

To address these limitations, this research proposes a new architecture for a realistic Forex trading environment that is specifically designed for effective RL. Our proposed environment aims to create a realistic Forex trading simulation that accurately represents the complex nature of the Forex market and trading conditions. It incorporates critical features such as transaction costs, market volatility, and fundamental factors that significantly impact exchange rates.

The present study outlines the methodology employed for designing and implementing a novel Forex trading environment, and evaluates the performance of a reinforcement learning (RL) algorithm in this environment.

The rest of the paper is structured as follows: Section 2 offers an introduction to RL and its applications in finance, followed by an examination of current Forex trading environments used in RL and their associated limitations. Section 3 outlines the methodology of the proposed environment, including the framework's architecture, implementation specifics, and RL algorithm. Section 4 presents the findings of the experiments and examples of use cases. Section 5 provides a detailed discussion of the framework's key features and contributions. Finally, Section 6 summarizes the paper's contributions and suggests future research directions.

2 Background

This section provides a thorough analysis of the history and pertinent research on the application of Reinforcement Learning (RL) in the financial sector, with a focus on Forex trading settings. It starts with a summary of RL's fundamental ideas before moving on to a review of earlier research that has used RL in financial contexts such option pricing, portfolio management, and stock trading. Then, we focus on Forex trading environments and present existing frameworks and libraries that have been developed for modeling Forex markets. Finally, we discuss the limitations of existing Forex RL environments and highlight the gaps that our proposed framework aims to address.

2.1 Overview of Reinforcement Learning

Reinforcement Learning (RL) is a branch of Machine Learning that entails learning to reach an objective through interactions with the environment. In RL, an agent takes actions in an environment to maximize a cumulative reward signal over time (Sutton and Barto, 2018). RL has been successfully applied to various domains, including robotics (Levine, Finn, Darrell and Abbeel, 2016), gaming (Mnih, Kavukcuoglu, Silver, Rusu, Veness, Bellemare, Graves, Riedmiller, Fidjeland, Ostrovski, Petersen, Beattie, Sadik, Antonoglou, King, Kumaran, Wierstra, Legg and Hassabis, 2015), and natural language processing (Bansal, Liang and Gim-

pel, 2017). RL algorithms typically involve a trade-off between exploration and exploitation, where the agent needs to explore the environment to discover optimal actions and exploit what it has learned to maximize rewards. One popular RL algorithm is Q-learning, which involves learning a state-action value function that estimates the expected cumulative reward from taking an action in a particular state. Another popular algorithm is policy gradient, which directly optimizes a policy that maps states to actions (Sutton and Barto, 2018). Recent progress in deep learning has facilitated the creation of deep reinforcement learning (RL) algorithms that can acquire knowledge directly from high-dimensional sensory inputs, such as raw market data or images (Mnih, Kavukcuoglu, Silver, Graves, Antonoglou, Wierstra and Riedmiller, 2013; Lillicrap, Hunt, Pritzel, Heess, Erez, Tassa, Silver and Wierstra, 2016). These algorithms, such as deep Q-networks (DQN) and actor-critic methods, have achieved state-of-the-art results in various RL benchmarks (Mnih et al., 2015; Schulman, Levine, Moritz, Jordan and Abbeel, 2015; Zhao, Li and Wen, 2022).
Figure 1 illustrates the interaction between an agent and an environment in a typical RL setup (Boutyour and Idrissi, 2023).

Figure 1: Illustration of the interaction between agent and environment.

2.2 Applications of RL in Finance

Reinforcement learning has emerged as a valuable tool for several applications in finance, including stock trading and portfolio optimization (Jiang, Wang and Xie, 2020; Lu, Qian and Xu, 2019). For instance, deep Q-learning has been used for profitable trades in stock trading (Jiang et al., 2020), while actor-critic algorithms have been applied for portfolio optimization (Lu et al., 2019). Additionally, similar techniques have been employed for multi-asset portfolio management (Aboussalah and Lee, 2020). Reinforcement learning has also been applied in options pricing (Shen and Xu, 2020), risk management (Singh, Ray and Mukherjee, 2020), and algorithmic trading (Guan, Li, Wang, Zhang and Wu, 2020). Shen et al. (2020) proposed in (Shen and Xu, 2020) an RL-based method for pricing options under stochastic volatility, while Singh et al. (2020) developed in (Singh et al., 2020) a RL-based framework for risk management of portfolios. Guan et al. (2020) developed in (Guan et al., 2020) an RL-based algorithmic trading framework that integrates a risk management strategy. Apart from the above applications, RL has been implemented for credit risk management (Wang, Liu, Chen and Chen, 2019), fraud detection (Sang, Shen, Gao and Fu, n.d.), and customer churn prediction (Zhang, Sheng and Liu, 2018). For example, Wang et al. (2021) proposed in (Wang et al., 2019) a credit risk management framework based on RL that enhances accuracy and

reduces computational time compared to traditional methods. Sang et al. (2019) developed in (Sang et al., n.d.) an RL-based framework for fraud detection in payment systems, while Zhang et al. (2019) in (Zhang, Sheng and Liu, 2018) used RL for customer churn prediction in the telecommunications industry. Despite the promising outcomes in various areas of finance, there are still challenges that require attention, such as the requirement for realistic market environments and the difficulty of balancing exploration and exploitation (Liu, Li, Li, Li and Li, 2019; Aboussalah and Lee, 2020).

2.3 Reinforcement Learning in Forex Trading

Reinforcement learning (RL) has become an effective tool for automated trading in a variety of financial markets, including forex. RL algorithms learn to make optimal trades by interacting with the market environment over time and evaluating the success of their actions using a reward signal. Several studies have used RL in forex trading with promising results. One approach aims to use RL algorithms to develop trading strategies that exploit specific market patterns. For example, Tang et al. (2018) developed in (Tang, Yan and Zhang, 2018) a Deep Q-Network (DQN) algorithm to trade EUR/USD currency pairs, achieving a cumulative profit of over 70% in a six-month period. Similarly, Hussein et al. (2019) proposed in (Hussein, Agarwal and Gopalakrishnan, 2019) a DQN-based algorithm that uses technical indicators to predict currency exchange rates and achieved a profit of 7.43% over a six-month period. Another approach is to use RL for portfolio management in forex trading. A multi-agent RL framework was introduced by Shavandi et al. (2022) in (Shavandi and Khedmati, 2022) to optimize a portfolio of forex trading strategies. This framework exceeds single autonomous agents and several benchmarks trading strategies in all tested trading periods of time. Moreover, Zhang et al (2018) developed in (Zhang, Yao and Sun, 2018) a deep deterministic policy gradient (DDPG) algorithm to optimize a portfolio of forex trading strategies, achieving a profit of 43.45% in a two-year backtesting period. RL has also been used for risk management in forex trading. For example, Kumar et al. (2018) developed in (Kumar and Varshney, 2018) an RL-based algorithm to dynamically adjust stop-loss levels, reducing the maximum drawdown by 36% compared to a static stop-loss level.

While RL algorithms have shown promise in trading it is crucial to recognize the limitations when implementing them in forex RL environments. These limitations arise from the complexities and dynamics, to the market. The purpose of this study is to address some of these challenges.

One notable limitation is the lack of diverse environments for simulating the market. Existing RL environments like FinRL(Liu, Yang, Chen, Zhang, Yang, Xiao and Wang, 2020), mbt-gym(Jerome, 2021) and ABIDES-gym(Amrouni, Moulin, Vann, Vyetrenko, Balch and Veloso, 2021) offer tools for developing and evaluating RL algorithms in finance. However they often fall short in providing the diversity of data required for training and testing.

To overcome this issue our study proposes a trading environment that incorporates a wide range of market scenarios and data sources. Our framework includes mechanisms for generating datasets downloading market data and extracting historical market information from CSV files. By doing we enhance the availability of data accessible to RL algorithms enabling them

to better adapt to various market conditions.
Another significant challenge lies in modeling market dynamics and integrating factors such as news events and economic indicators, into simulations.While this remains an issue our suggested approach takes strides towards tackling it. We integrate market characteristics such, as transaction costs, market volatility and fundamental factors that influence exchange rates. This enables a depiction of market dynamics compared to existing frameworks.
Although our approach may not completely resolve these challenges it represents progress in developing a authentic forex trading environment for RL. By prioritizing data diversity enhancement and incorporating market dynamics we aim to contribute to the advancement of RL in trading and encourage further exploration in this field.
In the following sections we will delve into the methodology and implementation specifics of our proposed framework offering insights, into how it addresses these limitations and enhances the training and evaluation of RL algorithms in trading scenarios.

3 The Proposed Framework : RL4RF

3.1 RL4RF Framework Architecture

The RL4RF framework is a flexible and realistic environment designed for training reinforcement learning (RL) agents in forex trading scenarios. Built on top of the OpenAI Gym interface, the framework provides a unified API for interacting with the environment and collecting data. The design of the framework is centered around modularity and extensibility, enabling researchers and practitioners to integrate their own data sources, RL algorithms, and evaluation metrics seamlessly. The framework comprises three main layers: the environment layer, the data layer, and the RL algorithms layer. The environment layer defines the state and action spaces, as well as the reward function for the agent. The data layer handles the necessary preprocessing and feature engineering steps to transform raw market data into relevant input features for the RL algorithms. Lastly, the RL algorithms layer contains a range of popular RL algorithms used to train and evaluate the agent. Fig.2 shows the architecture of the proposed framework and we will go through each layer and its components in detail in the following subsections.

3.2 Environment Layer

The environment layer is a critical component of the RL4RF framework as it defines the state and action spaces for the RL agent. The environment in the RL4RF framework is modeled after the OpenAI Gym interface, providing a unified API for interacting with the environment and collecting data.

3.2.1 State Space

In the RL4RF framework, the state space is comprised of observable variables that the RL agent can utilize to make decisions in the environment. These variables are represented as a set of features that capture essential information about the current market conditions. The

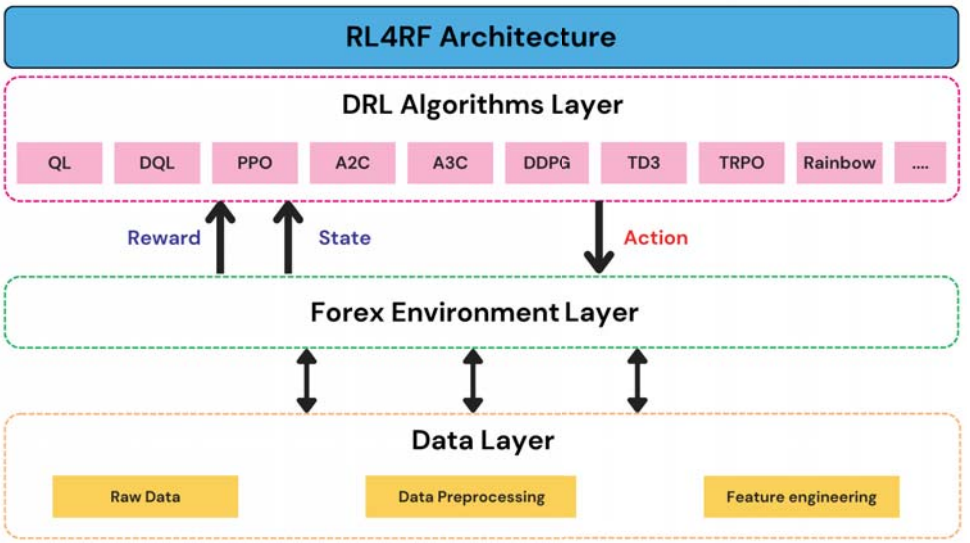


Figure 2: RL4RF Architecture

features in the state space are broadly categorized into three types: price features, technical indicators, and fundamental features. The price features indicate the current price levels of the exchange rates, such as the opening price, closing price, high price, and low price for a given period. Technical indicators are mathematical calculations derived from price and volume data that provide insights into the market's direction and momentum, such as moving averages, relative strength index (RSI), and stochastic oscillators. On the other hand, fundamental features capture the underlying economic and geopolitical factors that influence exchange rates, such as interest rates, GDP growth rates, and political events. Choosing the right features for the state space is crucial as it can significantly impact the performance of the RL agent. In the RL4RF framework, users can utilize a set of commonly used features or include their own to the state space.

3.2.2 Action Space

The action space of the RL4RF framework determines the collection of actions that the RL agent may take in the environment. In forex trading scenarios, the action space often comprises buy, sell, and hold actions for different currency pairs. However, the RL4RF framework expands the conventional action space to encompass the following actions, providing more flexibility to the RL agent:

- *Buy*: An action that signals the RL agent to initiate a long position in a specific currency pair.
- *Sell*: An action that signals the RL agent to initiate a short position in a specific currency pair.
- *Hold*: An action that signals the RL agent to hold its current position and take no action.

- *Close Buy Trade*: An action that signals the RL agent to close an existing long position in a specific currency pair.
- *Close Sell Trade*: An action that signals the RL agent to close an existing short position in a specific currency pair.
- *Close on Take Profit*: An action that signals the RL agent to close a trade when a predefined profit level is reached.
- *Close on Stop Loss*: An action that signals the RL agent to close a trade when a predefined stop loss level is reached.
- *Close Trade and Reverse*: An action that signals the RL agent to close its current position and initiate an opposing position in the same currency pair.
- *Trailing Stop*: An action that signals the RL agent to adjust the stop loss level based on the currency pair's price movement to secure profit or minimize loss.

3.2.3 Reward functions

The reward function is a crucial component of the RL4RF framework that determines the objective of the RL agent. It quantifies the agent's performance based on its actions in the environment. RL4RF provides users with a choice of reward functions based on their preferences and trading strategy objectives. Users can select from the following reward functions or define a custom one:

- *Profit and Loss (PNL)*: This reward function calculates the profit or loss incurred by the agent during a trading episode. It is commonly used in traditional forex trading strategies.

$$P\&L = \sum_{trade=1}^{N} (r_{trade} - c_{trade}) \tag{3.1}$$

  Where,

  - N is the total number of trades.
  - r_{trade} is the return from trade $trade$.
  - c_{trade} is the cost of executing trade $trade$.

- *Sharpe Ratio*: It is a popular risk-adjusted performance metric for investment strategies. The Sharpe Ratio-based reward function in the RL4RF framework considers both the returns and the risk of the trading strategy.

$$Sharpe\ ratio = \frac{E[ret_p - ret_f]}{\sqrt{Var[ret_p - ret_f]}} \tag{3.2}$$

  Where,

  - $E[ret_p - ret_f]$ is the projected excess return of the trading strategy above the risk-free rate.

  - $Var[ret_p - ret_f]$ is the variance of the excess returns.

- *Sortino Ratio*: It is similar to the Sharpe Ratio, but only takes into account the downside risk of a trading strategy. The Sortino ratio-based reward function is especially useful for trading strategies where downside risk is a major concern.

$$Sortino\ ratio = \frac{E[ret_p - ret_f]}{\sqrt{\sum_{ret_i < ret_f}(ret_i - ret_f)^2/(N-1)}} \tag{3.3}$$

  Where,

  - $E[ret_p - ret_f]$ is the projected excess return of the trading strategy above the risk-free rate.
  - ret_f is the risk-free rate.
  - ret_i is the return from trade i.
  - N is the number of trades with negative returns (i.e., downside trades).

- *Winning percentage*: The winning percentage is the percentage of profitable trades over the total number of trades. It measures the accuracy of the trading strategy and is an important metric for evaluating its performance. The goal of a forex trading strategy is to maximize the winning percentage while maximizing the P&L.

$$Win\ percentage = \frac{\text{Number of profitable trades}}{\text{Total number of trades}} \tag{3.4}$$

- *Maximum Drawdown*: The Maximum Drawdown is the maximum loss that the trading strategy has experienced over a given period. The reward function based on Maximum Drawdown penalizes the agent for large losses and encourages risk management.

$$Max\ drawdown = \max_{t \in [0,T]} \left(\frac{V_t - \max_{u \in [0,t]} V_u}{V_t} \right) \tag{3.5}$$

$$Max\ drawdown = \frac{\text{Peak value} - \text{Trough value}}{\text{Peak value}} \times 100 \tag{3.6}$$

  Where,

  - V_t is the value of the trading account at time t.
  - T is the end of the trading period.

The implementation of these reward functions in the RL4RF environment is straightforward. The environment class provides a method to choose the reward based on the selected function. Users can also specify their own reward function when initializing the environment class, allowing them to tailor the framework to their specific needs.

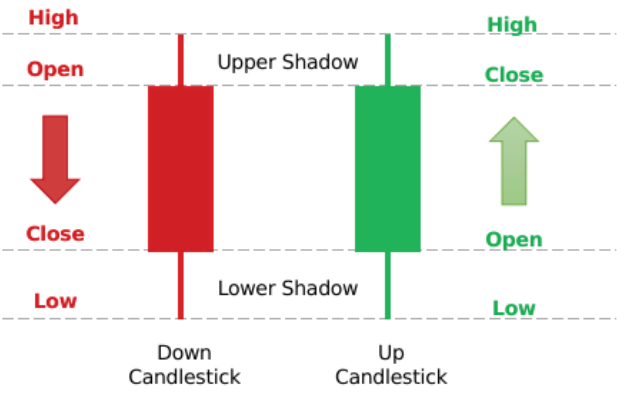


Figure 3: Japanese candlestick

3.3 Data Layer

The data layer is a crucial component in any RL-based trading framework, as it provides the necessary input to the RL agent for making informed decisions. In the RL4RF framework, the data layer is responsible for preprocessing raw market data (OHCLV) and engineering relevant features that capture the underlying market dynamics. Fig.3 shows an example of visual representation of OHCL price candle.

OHLCV, or Open, High, Low, Close, and Volume, is a popular financial chart used to depict the price movements of an asset, including stocks, commodities, and currency pairs. Each candlestick in an OHLCV chart represents a specific time unit (Timeframe), such as a day, hour, or minute, and contains vital information such as the opening and closing prices of the asset, the highest and lowest prices reached during the period, and the total volume of trades executed during the time unit.

OHLCV charts are extensively utilized in technical analysis to identify patterns, trends, and potential price levels. These charts can also help traders make informed decisions regarding support and resistance levels, price momentum, and other essential factors. Fig.4 shows the internal architecture of Data Layer.

3.3.1 Raw Data

Three ways to get the raw data in this layer:

- CSV file reader: It can read CSV files containing historical market data and convert them to a format suitable for RL algorithms.
- Data downloader: It can download historical market data from online sources to enable the use of real-world data for training and evaluating RL algorithms.
- Synthetic dataset generator: It can generate synthetic datasets with different market patterns to study the performance of RL algorithms under various market conditions.

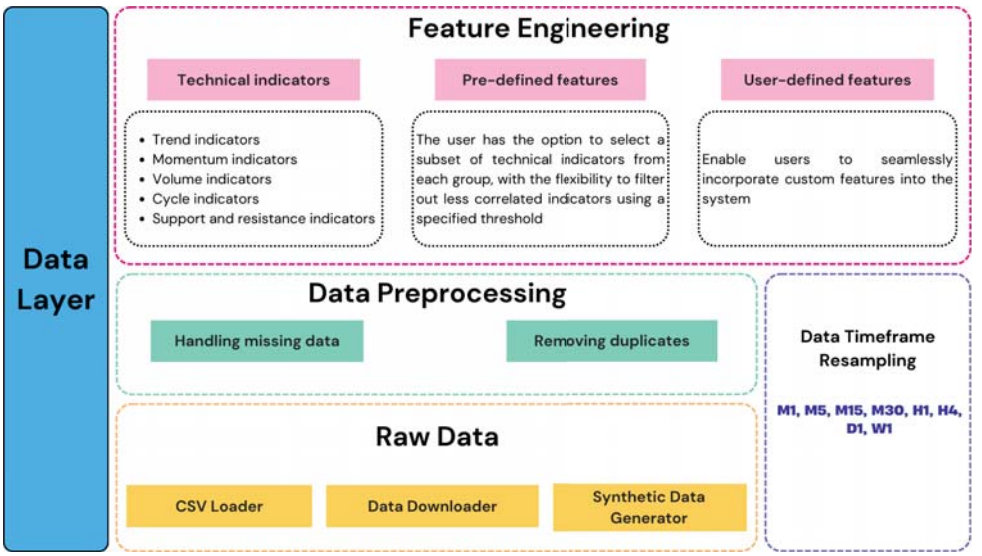


Figure 4: Data layer architecture

3.3.2 Data Preprocessing

Data preparation is an essential phase in machine learning, especially in reinforcement learning for FX trading. Preparing raw data for machine learning algorithms is the main goal of data preparation. The raw data used in forex trading typically comprises of historical price information for several currency pairings and other market indicators. Data cleansing, normalization, feature scaling, and data transformation are just a few of the many sub-tasks that make up data preparation. Removing erroneous or missing data points from a dataset is known as data cleaning. To make sure that the input features are scaled similarly, normalization and feature scaling are required. Data transformation entails transforming the input characteristics into a new collection of features that more accurately represents the underlying patterns in the data. In the RL4RF environment, several specific data preprocessing techniques are used to prepare the data. These techniques include removing gaps or irregularities in the price data, imputing missing data points using interpolation methods, and resampling the data to a common time frame.

3.3.3 Feature engineering

Along with data pretreatment, feature engineering is a crucial stage in getting the data ready for the RL4RF environment. In order to help the RL agent better capture the underlying patterns in the data, new features must be created from the existing data.
For feature engineering in forex trading, technical indicators including moving averages, stochastic oscillators, and relative strength index (RSI) are frequently utilized. These tools can help identify market patterns and momentum by providing data on the currency pair's historical price movements. Fundamental data, such as economic indicators and news releases, may be employed as features for feature engineering in forex trading in addition to technical indica-

tors. These market-impacting data points can offer the RL agent useful knowledge about the market. It is critical to remember that the choice of features used for feature engineering can have a big impact on how well the RL agent performs. So, it is crucial to choose the qualities that are most pertinent to the trading strategy and the particular currency pair being traded. Users of the RL4RF framework are free to design their own features or pick from a list of pre-defined characteristics. Depending on the user's preferences and chosen trading strategy, the pre-defined characteristics can be altered. Users may experiment with different feature combinations to see which ones work best, and feature engineering decisions can be made based on statistical analysis and domain knowledge.

3.4 DRL Algorithms Layer

The DRL algorithms layer is responsible for training and evaluating reinforcement learning agents using the proposed realistic Forex trading environment. To achieve this, we can leverage existing DRL libraries such as elegantRL(Liu, Li, Wang and Zheng, 2021) and tianshoo(Weng, Chen, Yan, You, Duburcq, Zhang, Su, Su and Zhu, 2022) to implement RL algorithms. To integrate these libraries within the RL4RF framework, we propose the creation of an RLWrapper class that acts as a wrapper around the elegantRL and tianshoo libraries. This class provides a common interface for all DRL algorithms and allows us to switch between algorithms with ease. The RLWrapper class can contain methods for initializing the DRL algorithm, training the agent on the Forex trading environment, and evaluating the performance of the trained agent. Additionally, the class can handle hyperparameter tuning for each algorithm, providing an efficient way to explore different configurations of the DRL agent. By creating a unified interface for DRL algorithms within the RL4RF framework, we can easily compare the performance of different algorithms and select the best approach for a specific trading scenario. Furthermore, this approach allows for easy integration of new algorithms as they become available, ensuring that the RL4RF framework stays up-to-date with the latest advances in DRL research.
In the next part, we show the usefulness of the proposed framework by giving instances of trading strategies developed using the framework.

4 Performance and Evaluation

The performance and evaluation of the proposed RL4RF framework are critical to determine its effectiveness in training and testing RL agents for forex trading. In this section, we present the experimental setup, evaluation metrics, and the results and analysis of the proposed framework.

4.1 Experiment Settings

4.1.1 Experiment 1 - (PPO agent on EURUSD)

In this practical use case, we demonstrate how to use the RL4RF framework to train and test a Proximal Policy Optimization (PPO) agent on a daily EURUSD dataset from 1/1/2022 to 31/12/2022 using CSV loader from Data Layer of our framework. First, we split the data

into 80% for training and 20% for testing. Then, we define the state space, action space, and reward function in the environment layer. After that, we preprocess the data and engineer features in the data layer. Next, we select the PPO algorithm from the RL algorithms layer and set the hyperparameters as shown in table 1. We use the RLWrapper class to integrate the elegantRL library (Liu et al., 2021) and train the PPO agent on the training data. Finally, we evaluate the PPO agent on the testing data and analyze the results using various evaluation metrics, including Sharpe ratio, maximum drawdown, and total return.

Table 1: Hyperparameters of experiment 1 for PPO Agent Training on EURUSD

Hyperparameter	**Value**
Learning Rate	0.001
Discount Factor (γ)	0.99
GAE Lambda (λ)	0.95
Number of Epochs	10
Mini-Batch Size	64
Value Function Coefficient	0.5
Entropy Coefficient	0.01
Clip Parameter	0.2
Maximum Steps	1000
Hidden Layers	[64, 64]
Activation Function	ReLU
Optimizer	Adam

4.1.2 Experiment 2 - (DQN agent on GBPUSD)

To train and test a DQN agent on a 15-minute GBPUSD dataset from 1/6/2022 to 31/12/2022, we first load the data into the RL4RF framework using the CSV loader. We then split the data into training and testing sets, with 80% of the data used for training and 20% for testing. Next, we set up the DQN agent layer using the RLWrapper class to integrate the ElegantRL(Liu et al., 2021) and Tianshoo (Weng et al., 2022) libraries. We configure the agent layer with the hyperparameters, such as the learning rate, discount factor, and exploration rate as shown in table 2. We then train the DQN agent on the training dataset, using the agent to make trading decisions based on the observed market conditions. After training, we evaluate the performance of the DQN agent on the testing dataset, measuring the agent's profitability and other performance metrics.

4.2 Trading Performance

4.2.1 Experiment 1 - (PPO agent on EURUSD)

As shown in Table 3, the PPO agent achieved a Sharpe ratio of 1.96 and 1.75 on the training and testing datasets, respectively, indicating a good risk-adjusted return. The maximum

Hyperparameter	Value
Learning Rate	0.001
Discount Factor (γ)	0.99
Exploration Rate (ϵ)	0.1
Replay Buffer Size	10000
Batch Size	64
Target Update Frequency	1000
Number of Hidden Layers	2
Hidden Layer Size	128
Activation Function	ReLU
Optimizer	Adam
Max Episodes	1000
Max Steps per Episode	1000

Table 2: Hyperparameters of experiment 2 for DQN Agent Training on GBPUSD

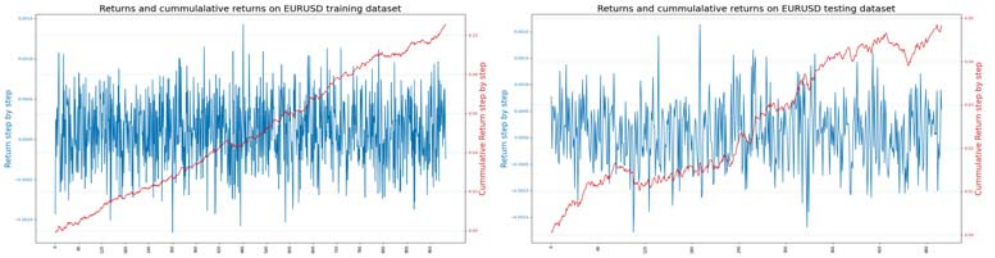
drawdown was limited to -3.5% and -5.2% on the training and testing datasets, respectively, suggesting the agent was able to control its risk exposure. The total return was 10.57% and 4.83% on the training and testing datasets, respectively, indicating a profitable performance. Overall, the PPO agent trained using the RL4RF framework showed promising results in this experiment.

Table 3: Trading Performance Metrics for PPO Agent on EURUSD Dataset (Experiment 1)

Metrics	Training Set	Testing Set
Sharpe Ratio	1.96	1.75
Maximum Drawdown	-3.5%	-5.2%
Total Return	10.57%	4.83%

4.2.2 Experiment 2 - (DQN agent on GBPUSD)

The results presented in Table 4 illustrate that the DQN agent generated profitable trading performance on both the training and testing datasets. The agent executed 2,128 trades during the training period and 522 trades during the testing period, resulting in a profit factor of 1.38 and 1.26, respectively. Moreover, the agent's trading performance exhibited a Sharpe ratio of 1.08 for the training dataset and 1.04 for the testing dataset, indicating returns higher than the risk-free rate of return. Additionally, the agent was able to effectively manage its risk exposure, with a maximum drawdown of -4.63% and -2.86% for the training and testing datasets, respectively. Furthermore, the agent achieved a total return of 9.05% for the training dataset and 6.14% for the testing dataset, indicating a consistent profit generation throughout the trading period. These findings highlight the effectiveness of the RL4RF framework in training and evaluating DRL agents for forex trading applications.

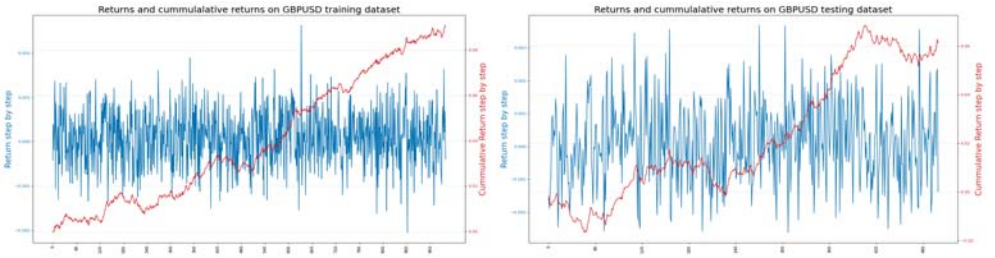


(a) EURUSD Return on Training Dataset. (b) EURUSD Return on Testing Dataset.

Figure 5: Experiment 1: EURUSD Daily Returns for Training and Testing Datasets

Table 4: Trading Performance Metrics for DQN Agent on GBPUSD Dataset (Experiment 2)

Metrics	Training Set	Testing Set
Sharpe Ratio	1.08	1.04
Maximum Drawdown	-4.63%	-2.86%
Total Return	9.05%	6.14%



(a) GBPUSD 15-min Returns on Training Dataset. (b) GBPUSD 15-min Returns on Testing Dataset.

Figure 6: Experiment 2: GBPUSD 15-min Returns for Training and Testing Datasets

5 Discussions

The RL4RF framework, as described in this paper, offers several noteworthy features and contributions to the realm of trading with RL. These key aspects deserve closer examination:

Data Layer Enhancement: Our framework's Data Layer includes comprehensive preprocessing steps, incorporates essential economic indicators, and promotes data source diversity. This multifaceted approach equips users with the ability to explore various market scenarios, enhancing the framework's adaptability and utility.

DRL Algorithms Layer: The DRL Algorithms Layer is a crucial component of our framework, which boasts integration with state-of-the-art RL algorithms. Via RLWrapper, it seamlessly interfaces with external libraries like ElegantRL and Tianshoo. This extensibility ensures that researchers and traders can leverage a wide range of algorithms to suit their specific needs.

Realistic Forex Environment: Our Forex Environment Layer stands as a testament to our

commitment to providing a realistic RL environment for forex trading. By capturing the nuances of market dynamics and incorporating authentic trading conditions, we offer users a platform that closely mirrors real-world scenarios.

6 Conclusion and Future Work

In summary, the RL4RF framework presented in this paper makes a contribution to the field of trading with RL. It provides an flexible environment for training and evaluating RL agents demonstrating its effectiveness in generating forex trading strategies. Looking ahead, we have a roadmap to enhance our framework. We will focus on improving scalability and robustness to accommodate a range of RL algorithms and diverse data sources. Additionally, we recognize the importance of interpretability in RL models. We aim to explore how it can help us understand and influence market dynamics in our research. Our continuous efforts to advance the RL4RF framework aim to empower traders and researchers with a toolkit for developing and testing trading strategies. This work represents a step towards unlocking the potential of RL in the complex world of forex markets and we are dedicated to making further contributions, in this exciting domain.

References

Aboussalah, A. M. and Lee, C.-G. 2020. Continuous control with stacked deep dynamic recurrent reinforcement learning for portfolio optimization, *Expert Systems with Applications* .

Amrouni, S., Moulin, A., Vann, J., Vyetrenko, S., Balch, T. and Veloso, M. 2021. Abides-gym: gym environments for multi-agent discrete event simulation and application to financial markets, *Proceedings of the Second ACM International Conference on AI in Finance*, ICAIF'21, ACM.
URL: *http://dx.doi.org/10.1145/3490354.3494433*

Bansal, T., Liang, Y. and Gimpel, K. 2017. Emergent linguistic phenomena in multi-agent communication games, *arXiv preprint arXiv:1703.04908* .

Boutyour, Y. and Idrissi, A. 2023. Deep reinforcement learning in financial markets context: Review and open challenges, *Modern Artificial Intelligence and Data Science*, Springer Nature Switzerland, pp. 49–66.
URL: *https://doi.org/10.1007/978-3-031-33309-5_5*

Buehler, H., Horvath, B., Lyons, T., Perez Arribas, I. and Wood, B. 2020. A data-driven market simulator for small data environments, *SSRN Electronic Journal* .
URL: *http://dx.doi.org/10.2139/ssrn.3632431*

Guan, R., Li, X., Wang, H., Zhang, Z. and Wu, J. 2020. Deep reinforcement learning for trading, *IEEE Transactions on Neural Networks and Learning Systems* **31**(11): 4828–4840.

Hussein, A. K., Agarwal, N. and Gopalakrishnan, V. 2019. A deep reinforcement learning framework for the financial portfolio management problem, *Expert Systems with Applications* **116**: 69–85.

Jerome, J. e. a. 2021. Model-based gym environments for limit order book trading, *Journal of Open Source Software* **6**(63): 3291.

Jiang, B., Wang, X. and Xie, Y. 2020. Deep reinforcement learning for portfolio management, *Quantitative Finance* **20**(8): 1271–1291.

Kumar, S. and Varshney, P. 2018. Forex exchange rate forecasting using deep recurrent neural networks, *Digital Finance* **132**: 1280–1289.

Levine, S., Finn, C., Darrell, T. and Abbeel, P. 2016. End-to-end training of deep visuomotor policies, *The Journal of Machine Learning Research* **17**(1): 1334–1373.

Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D. 2016. Continuous control with deep reinforcement learning, *ICLR* .

Liu, X., Li, Z., Wang, Z. and Zheng, J. 2021. Elegantrl: Massively parallel framework for cloud-native deep reinforcement learning, GitHub repository.
URL: *https://github.com/AI4Finance-Foundation/ElegantRL*

Liu, X.-Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B. and Wang, C. 2020. Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance, *SSRN Electronic Journal* .
URL: *http://dx.doi.org/10.2139/ssrn.3737859*

Liu, Y., Li, K., Li, L., Li, Y. and Li, G. 2019. Multi-objective deep reinforcement learning for portfolio management, *IEEE Access* **7**: 69887–69899.

Loh, L., Kueh, H., Parikh, N., Chan, H., Ho, N. and Chua, M. 2022. An ensembling architecture incorporating machine learning models and genetic algorithm optimization for forex trading, *FinTech* **1**: 100–124.

Lu, Y., Qian, Y. and Xu, W. 2019. Reinforcement learning for forex trading, *2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)* pp. 301–307.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. 2013. Playing atari with deep reinforcement learning, *arXiv e-prints* .

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. and Hassabis, D. 2015. Human-level control through deep reinforcement learning, *Nature* **518**: 529–533.

Sang, Y., Shen, H., Gao, S. and Fu, Z. n.d.. A reinforcement learning framework for fraud detection in payment systems, *Expert Systems with Applications* **118**.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I. and Abbeel, P. 2015. Trust region policy optimization, *Proceedings of the 32nd International Conference on Machine Learning*, JMLR. org, pp. 1889–1897.
URL: *http://proceedings.mlr.press/v37/schulman15.html*

Shavandi, A. and Khedmati, M. 2022. A multi-agent deep reinforcement learning framework for algorithmic trading in financial markets, *Expert Systems with Applications* **208**.

Shen, J. and Xu, J. 2020. A deep reinforcement learning framework for the financial portfolio management problem, *Journal of Risk and Financial Management* **13**(7): 141.

Singh, S., Ray, S. and Mukherjee, A. 2020. Deep reinforcement learning: A comprehensive overview, *arXiv preprint arXiv:2006.05990* .

Sutton, R. S. and Barto, A. G. 2018. *Reinforcement learning: An introduction*, MIT press.

Tang, Y., Yan, Y. and Zhang, Y. 2018. Deep reinforcement learning trading, *IEEE Access* **6**: 40306–40316.

Théate, T. and Ernst, D. 2021. An application of deep reinforcement learning to algorithmic trading, *Expert Systems with Applications* **173**: 114632.
URL: *http://dx.doi.org/10.1016/j.eswa.2021.114632*

Wang, Y., Liu, H., Chen, K. and Chen, C. 2019. Deep reinforcement learning for algorithmic trading: a review of the state-of-the-art, *IEEE Access* **7**: 122735–122747.

Weng, J., Chen, H., Yan, D., You, K., Duburcq, A., Zhang, M., Su, Y., Su, H. and Zhu, J. 2022. Tianshou: A highly modularized deep reinforcement learning library, *Journal of Machine Learning Research* **23**(267): 1–6.

Zhang, Y., Sheng, H. and Liu, Y. 2018. Stock trading with recurrent reinforcement learning (rrl) approach, *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, p. 1487–1491.

Zhang, Y., Yao, L. and Sun, A. e. a. 2018. Deep reinforcement learning for trading, *J Big Data* **5**(26).

Zhao, M., Li, Y. and Wen, Z. 2022. A stochastic trust-region framework for policy optimization, *Journal of Computational Mathematics* **40**(6): 1004–1030.
URL: *http://dx.doi.org/10.4208/jcm.2104-m2021-0007*