Proceedings of the URECA@NTU 2020-21

# Reinforcement Learning in FX Trading

Wong Wei Jie
School of Computer Science and Engineering

Zhang Zeyu
School of Electrical and Electronic Engineering

Tng Wei Xuan, Daryl
School of Physical and Mathematical Sciences

Qin Ruochen
Gideon Kharista
Nanyang Business School

Teoh Teik Toe
Nanyang Business School

***Abstract -*** Reinforcement learning (RL) is an area of machine learning where intelligent agents learn to take actions in an environment to maximize total rewards. Over the last few decades, there were successes in applying reinforcement learning techniques in games, including several Atari games and board game Go, by modelling the game into a Markov Decision Process (MDP). In such learning problems, the value of an action depends on future actions and states, which is difficult for conventional supervised learning methods. While supervised and unsupervised learning techniques have been commonly used in financial trading, applying reinforcement learning in the domain of financial trading has been relatively new. In this paper, we will evaluate the effectiveness of reinforcement learning on foreign exchange (FX) trading by treating it as a partially observable MDP (POMDP) with the notion of maximizing total profits. We will be using model-free algorithm Advantage Actor Critic (A2C) and evaluating their performance as a trading strategy.

**Keywords –** Reinforcement learning, On-policy algorithm, Off-policy algorithm, FX trading

**Source Code:**
https://github.com/wowweijie/rl_forex

## 1 INTRODUCTION

The foreign exchange (FX) market is a global marketplace for trading national currencies and is the biggest and most liquid financial market, over the stocks, bonds, and commodity markets, with a daily trading volume of USD$6.6 trillion [1]. The high trading volume is mainly due to market participants exchanging currencies to hedge against currency and interest rates risk by through asset diversification and speculative trades. With the high intraday transaction volume, the volatility of the currency pair prices has ushered in a rise in adoption of technical analysis to capture market psychology by analysing price trends and indicators to formulate trading rules. Over the last 4 decades, surveys have shown that a stronger preference towards capturing irrational market behaviour in Fx trading over fundamental approaches in analysing economic changes, have resulted in modest returns. [1] [2] In recent times, artificial intelligence was widely perceived to help optimize those trading rules. [3]

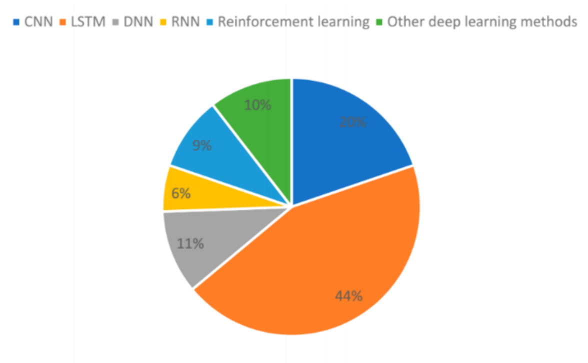


Figure 1: Distribution of methods found literary works covering FX and stock price prediction, from [4]

Most research studies in applying artificial intelligence had delved into most into supervised and unsupervised learning methods, including Long Short Term Memory (LSTM) and Convolutionary Neural Networks (CNN) which accounts for 44% and 20% of literary works of this topic, whereas reinforcement learning methods only accounted for 9%. [4] LSTM and CNN have generally shown successes in time-series application. It is also worthy to note that RL techniques could also involve the use of LSTM and CNN as a policy network.

However, reinforcement learning does provide some advantages over traditional supervised and unsupervised learning methods. First, the experience replay mechanism used in RL helps to smooth the training distribution over many past behaviours and thus, alleviate the problems of correlated data and non-stationary distributions. [5] Second, the future value of taking an action can be better modelled as a reward function that the agent can learn, rather than expensing resources to label the data effectively in convertional supervised learning. [6] Third, reinfrocement learning provides a more effective means for training trading streategies when transaction costs are included, as compared to standard supervised approaches. [7]

In this study, we aim to model the FX trading strategy on a portfolio of 4 major currency pairs – EURUSD, GBPUSD, USDJPY and USDCHF, into a POMDP by modelling technical indicators and 15min OHLC prices as the states and gains in net open position, i.e. profits, as the rewards. We broke down the timeseries problem into smaller tasks that spans across one month and then apply on-policy algorithm, A2C to train our policy in a continual learning approach.

## 2 BACKGROUND AND RELATED WORK

### 2.1 Reinforcement Learning

Reinforcement learning is a type of machine learning technique whereby an agent learns in an interactive environment by trials and errors using feedback from its own actions and experiences.

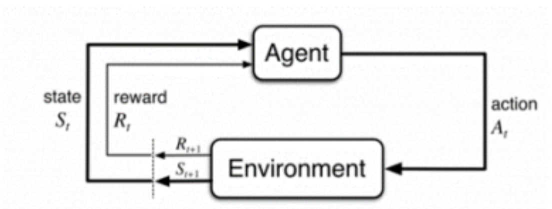


Figure 2: An overview of reinforcement learning

The agent learns by updating its policy which comprises of suggested actions that the agent should take for every possible state in pursuit of maximizing total rewards.

Generally, there are two approaches to RL that defines characteristics of various RL algorithms, namely, on-policy and off-policy methods (i.e., direct and indirect respectively).

#### 2.1.1 On-Policy methods

On-policy approach involves parameterizing the policy and optimize the policy directly through gradient descent in order the maximize total expected rewards.

$$\nabla_\theta J(\theta) = \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) G_t]$$

Figure 3: Monte-Carlo policy gradient

Some popular on-policy algorithms include Monte-Carlo policy gradient, Advantage Actor-Critic (A2C). Off-policy approach, on the other hand, solves the optimal policy by directly maximizing an objective function using gradient descent methods. Q-learning is one of the most popular on-policy algorithms used.

In this study, we have used the A2C algorithm only due to its convergence properties and ability to cover continuous action spaces. [2]

### 2.2 Advantage Actor-Critic (A2C) Algorithm

The A2C algorithm is a type of Actor-Critic algorithm. The Actor-Critic method is decomposed into two components, the policy model and the value function. The "Critic" updates the objective value function parameters and depending on the algorithm it could be action-value, Q, or state-value, V. The "Actor" updates the policy parameters in the direction suggested by the critic. From the Monte-Carlo policy gradient, the Q-value is used to estimate the expected reward.

$$\mathbb{E}_{r_{t+1}, s_{t+1}, \ldots, r_T, s_T}[G_t] = Q(s_t, a_t)$$

$$\begin{aligned} \nabla_\theta J(\theta) &= \mathbb{E}_{s_0, a_0, \ldots, s_t, a_t}[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)] Q_w(s_t, a_t) \\ &= \mathbb{E}_\tau[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) Q_w(s_t, a_t)] \end{aligned}$$

In removing the variance from the gradient, a baseline is used to calculate how much of an advantage would an action give above the state-value.

$$A(s_t, a_t) = Q_w(s_t, a_t) - V_v(s_t)$$

$$Q(s_t, a_t) = \mathbb{E}[r_{t+1} + \gamma V(s_{t+1})]$$

$$A(s_t, a_t) = r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)$$

Hence, the advantage value can be used to approximate the policy gradient descent.

$$\begin{aligned} \nabla_\theta J(\theta) &\sim \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t)(r_{t+1} + \gamma V_v(s_{t+1}) - V_v(s_t)) \\ &= \sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t|s_t) A(s_t, a_t) \end{aligned}$$

```
Advantage actor critic algorithm
Initialize parameters $\theta, s$
Repeat
    Sample $a \sim \pi_\theta$
    Take action $a$, get reward $r_t$ and next state $s'$
    Compute TD error:
        $\delta_{t_k} = r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t)$
    Update the Actor parameters:
    $\theta \leftarrow \theta + \omega(\frac{1}{N}\sum_{i=1}^{N}[\sum_{t=0}^{T}\gamma^t \nabla_\theta \log \pi_\theta(a_{i,t} | s_{i,t})(r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t)))$
    Update $V(s_t)$ using target $r(s_t, a_t) + \gamma V(s_{t+1})$
Until $s$ is the expected state
```

Figure 4: Pseudo code of A2C, obtained from [3]

## 2.3 Bayesian Optimization

Bayesian optimization is a hyperparameter tuning technique that is used to optimize the performance of the underlying model.

$$P(\mathbf{\Theta}|data) = \frac{P(data|\mathbf{\Theta}) \times P(\mathbf{\Theta})}{P(data)}.$$

Figure 5: Bayes Theorem

It uses Bayes Theorem to search for combinations of hyperparameters to form a probabilistic model mapping the hyperparameters to a probability of the metric used in the underlying model. This is done by treating the training iteration with a set of hyperparameters as a surrogate function.

$\mathrm{P}(\Theta)$ is the prior distribution, representing the initial belief of the true values of the parameters. This is instantiated with a Gaussian Process distribution.

$\mathrm{P}(\mathrm{data}|\ \Theta)$ is the likelihood distribution, which can be estimated from the past searches of hyperparameters and the return values of the surrogate function.

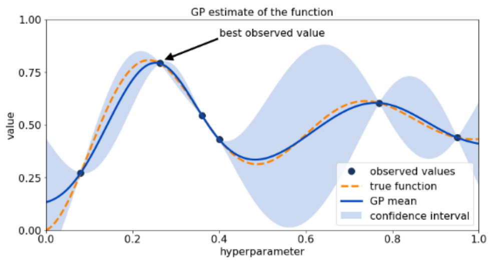


Figure 5: Gaussian posterior distribution in Bayesian Optimization, obtained from [4]

Hence, we can obtain the posterior distribution, $\mathrm{P}(\Theta|\mathrm{data})$ of hyperparameters and use it to efficiently explore hyperparameter space and return optima set of hyperparameters to be used for training the agent.

Hyperparameter tuning is even more necessary for policy gradient methods, where the performance is sensitive to changes in learning rate and significant performance gains can be obtained by tuning this hyperparameter.

Since tuning hyperparameters of RL algorithms through grid search is sample inefficient computationally expensive, using bayesian optimization to tune the parameters have been significantly reduced effort and time and result in better performance. [11]

# 3 METHODOLOGY

## 3.1 Experiment Set Up

### 3.1.1 Overview

We modelled the diversification trading strategy across the 4 currency pairs as a MDP and trained the agent using the A2C algorithm only. The agent can learn how to take actions of buying and selling with different order size (in units of the base currency) across the 4 currency pairs at the same time. In doing so, the trading strategy is like a portfolio diversification FX strategy that seeks to minimize the impact of currency risk.

In the experiment, we performed training on 4 environments, each environment representing the trading problem across the span of one month, from January 2017 to April 2017 and with a new agent. We then evaluate the performance of each strategy (agent) from January 2018 to April 2018. Each environment has a total 4 training iteration for hyperparameter tuning, and each training iteration has 200 episodes.

### 3.1.1 Data Pipeline

We used tick data from Dukascopy broker to generate 15min Open-High-Low-Close data. This allows us to include tick volume, which is the number of tick changes within the 15min interval, which might be an important indication of price movements. [4].

## 3.2 MDP Design

Our reinforcement learning FX trading strategy focuses on diversifying an initial USD balance into 5 different currency balances through buying and selling across 4 major currency pairs – EURUSD, GBPUSD, USDCHF, USDJPY. We implemented a simulated trading environment with historical FX price data and an initial account balance of USD$100, 000.

| | Upper bound | Lower bound |
|---|---|---|
| Learning Rate | $2 \text{ X } 10^{-4}$ | $1.0 \text{ X } 10^{-5}$ |
| Epsilon | $2 \text{ X } 10^{-5}$ | $1.0 \text{ X } 10^{-6}$ |

### 3.2.1 State Design

The states are derived from the agent's observation of the bid and ask close prices of the currency pairs in the 15min OHLC data. With bid and ask prices, transactional cost is incurred as spreads.

In addition to the prices, we abstracted features by calculating several popular momentum and pattern technical indicators that traders use to trade FX.

### 3.2.2 Action Space Design

The action space is a continuous 4-dimensional signal with buy and sell orders. The magnitude of the signal corresponds to the size of the order, in units of the base currency, i.e., EUR in EURUSD pair. A positive signal indicates a buy order while a negative signal indicates a sell order.

### 3.2.3 Reward Function

We used the change in net open position value, expressed in USD using the prevailing market prices to convert non-USD balance (i.e., position), as the reward function.

$$\boldsymbol{Reward} = \Delta\, \boldsymbol{Net\ Open\ Position\ Value_{USD}}$$

$$= \Delta \begin{pmatrix} \boldsymbol{Balance_{USD}} + \\ \boldsymbol{Balance_{EUR} * Price_{EURUSD}} + \\ \boldsymbol{Balance_{GBP} * Price_{GBPUSD}} + \\ \boldsymbol{Balance_{JPY} / Price_{USDJPY}} + \\ \boldsymbol{Balance_{CHF} / Price_{USDCHF}} + \end{pmatrix}$$

## 3.3 Training

In our previous attempts to train the agent in an environment that spans across several months, we found difficulty in converging the agent policy with high losses incurred when evaluated on the training period itself, even with a bayesian search of hyperparameters. It is possible that non-stationarity of the environment may have cause the neural network to suffer from catastrophic forgetting, where the training on new data quickly erases past learned experiences. [5]. Thus, we chose to break down the task into monthly segments and perform training onto each month.

The training done on each month is reiterated 5 times to tune the hyperparameters using bayesian optimization. The following table shows the hyperparameter search space that we have defined.

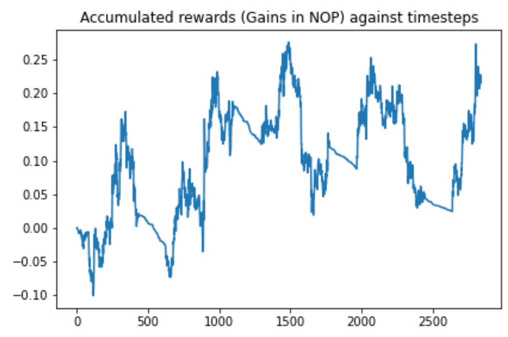
The target value that we have defined is the Sortino ratio of the learned policy when evaluated against the same training environment. The Sortino ratio is an improvement of Sharp ratio, that gauges the rate of return against likelihood of downside risk. We will then evaluate the performance of the learned policy against the same month in the next year.

## 4 RESULTS AND DISCUSSION

We obtained mixed results from the experiment. While the learned policies from January and February 2017 had positive returns when evaluated against January and February 2018 respectively, the policies from March and April 2017 had negative returns when evaluated in their respective months.

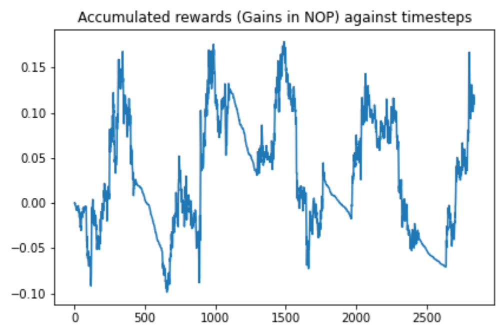
### 4.1 Training Environment Performance

**January 2017**



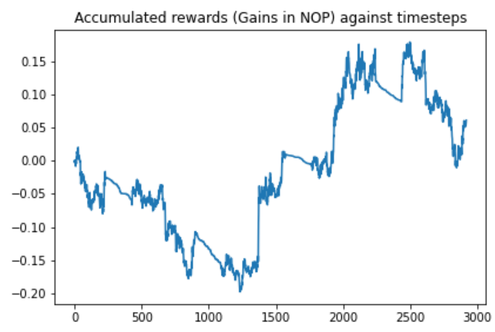
| | |
|---|---|
| Returns: $2660.34 (2.66%) | Sortino Ratio: 2.598 |

**February 2017**



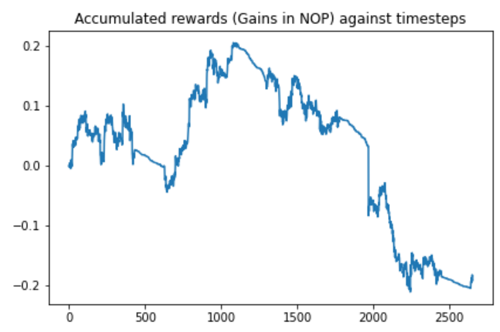
| | |
|---|---|
| Returns: $1687.20 (1.687%) | Sortino Ratio: 1.409 |

**March 2017**



| | |
|---|---|
| Returns: $5013.87 (0.501%) | Sortino Ratio: 1.823 |

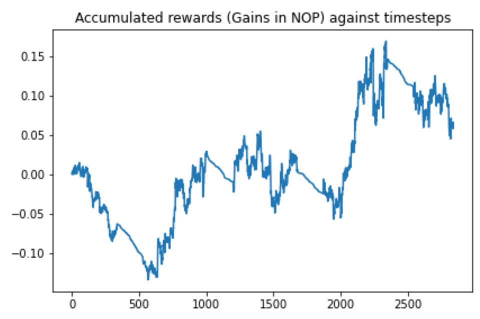
**April 2017**



| | |
|---|---|
| Returns: $-1994.19 (-1.994%) | Sortino Ratio: -4.018 |

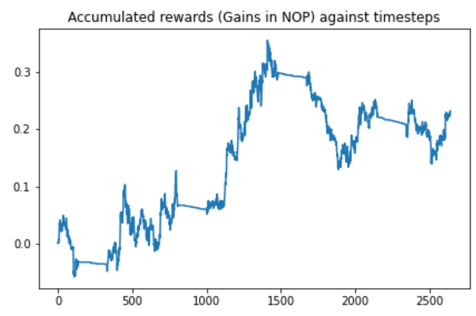
### 4.2 Test Environment Performance

**January 2017**



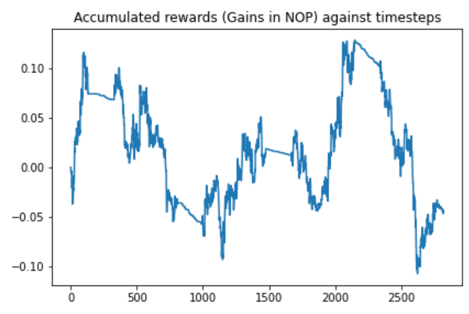
| | |
|---|---|
| Returns: $585.55 (0.586%) | Sortino Ratio: 1.119 |

**February 2017**



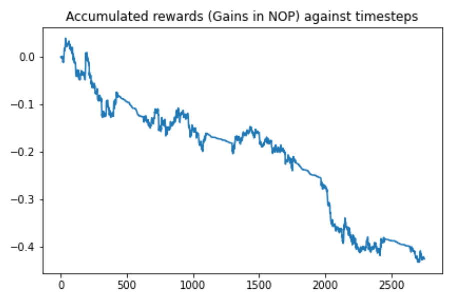
| | |
|---|---|
| Returns: $2318 (2.318%) | Sortino Ratio: 4.093 |

**March 2017**



| | |
|---|---|
| Returns: $-433.25 (-0.437%) | Sortino Ratio: -0.832 |

**April 2017**



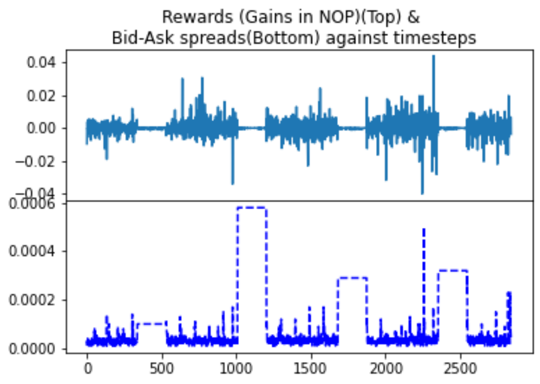
| | |
|---|---|
| Returns: $-4246.33 (-1.994%) | Sortino Ratio: -10.790 |

### 4.1 Discussion

Only January, February and to some extent, March, had some convergence in the training period, and only January and February saw positive returns when evaluated on the test period.

The April 2017 training did not result in policy convergence, and it was not surprising that the evaluation performed significantly worse compared to the other months.

It could be possible that there is some annual seasonality involved and the dynamics of the environment in one month remains slightly similar in the same month of the next year.



Rewards (zero rewards implies a consolidation to USD) vs EURUSD spreads in January 2018 test period.

It is also interesting to note that the agent has learned not to diversify the portfolio, convert all balances back to USD and halt trading when the spread (i.e., transaction cost) is high

Two main challenges that we have faced in this research, aside from the experiment being computationally expensive, was the non-stationarity of the environment and the difficulty in reaching convergence. Future works could focus on experimenting different algorithms and techniques to strive for policy convergence and also generalising performance across similar environments with meta-learning techniques. [6]

## 5 CONCLUSION

We used a A2C algorithm to train on 4 environments surrounding the trading periods of January, February, March and April 2017 and tuned the hyperparameters using bayesian optimization. We then evaluated each environment in the same month of the following year and yielded mixed results. We have learned that training across several months was difficult to reach convergence, but it was possible when we trained on each individual months. The results also suggested that in order to achieve desired results, the agent needs to perform significantly well in the training period prior to evaluating it on the test period.

## 6. ACKNOWLEDGMENT

*We would like to acknowledge the funding support from Nanyang Technological University – URECA Undergraduate Research Programme for this research project. We would also like to acknowledge the open source contributions to the Stable Baselines framework and FinRL project that has helped with our implementation of the experiment.*

## REFERENCES

[1] H. Allen and M. Taylor, "Charts, Noise and Fundamentals in the London Foreign Exchange Market," *Economic Journal,* 1990.

[2] L. Menkhoff and M. P. Taylor, "The Obstinate Passion of Foreign Exchange Professionals:," *Centre for Economic Policy Research ,* 2006.

[3] J.P. Morgan, "2020 e-Trading Survey," New York, 2020.

[4] Z. Hu, Y. Zhao and M. Khushi, "A Survey of Forex and Stock Price Prediction Using," *MDPI Appl. Syst. Innov.,* 2021.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv cs.LG,* 2013.

[6] X. Gao, "Deep reinforcement learning for time series: playing idealized trading games," *Georgia Institute of Technology,* 2018.

[7] J. Moody and M. Saffell, "Reinforcement Learning for Trading Systems and Portfolios," *Oregon Graduate Institue,* 1998.

[8] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," *arXiv cs.LG,* 2016.

[9] J. X. Wang, Z. Kurth-Nelson, D. Kumaran and D. Tirumala, "Prefrontal cortex as a meta-reinforcement learning system," *Nature Neuroscience,* 2018.

[10] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python".

[11] A. H. Z. W. I. A. J. S. Yutian Chen, "Bayesian Optimization in AlphaGo," *arXiv,* 2018.

[12] C. Marney, "Are price updates a good proxy," *FX Trader Magazine,* 2011.

[13] M. McCloskey and N. J. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem," *Psychology of LEarning and Motivation,* 1989.

[14] J. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Leibo, R. Munos, C. Blundell, D. K. and M. Botvinick, "LEARNING TO REINFORCEMENT LEARN," *arXiv cs.LG,* 2017.

[15] BIS, "Triennial Central Bank Survey of Foreign Exchange and Over-the-counter (OTC) Derivatives Markets in 2019," *BIS Quarterly Review,* 2019.