Information Modelling and Knowledge Bases XXXIV
M. Tropmann-Frick et al. (Eds.)

doi:10.3233/FAIA220494

Scalp the Foreign Exchange Market with Deep Reinforcement Learning

Marina TROPMANN-FRICK [a,1] and Phuc TRAN [b]
[a] *University of Applied Sciences Hamburg, Berliner Tor 7, 20099 Hamburg, Germany.*
marina.tropmann-frick@haw-hamburg.de
[b] *thienphuc.tran@haw-hamburg.de*

Abstract. This paper presents a reinforcement learning approach for foreign exchange trading. Inspired by technical analysis methods, this approach makes use of technical indicators by encoding them into Gramian Angular Fields and searches for patterns that indicate price movements using convolutional neural networks (CNN). In addition to the policy that determines the action to take, an extra regression head is utilized to determine the size of market orders. This paper also experimentally shows that maximizing the return of individual trade or cumulative reward in a finite time window results to better performance.

Keywords. Foreign Exchange Trading and Gramian Angular Field and Deep Reinforcement Learning.

1. Introduction

The development of reinforcement learning is accelerating, Deep Reinforcement Learning (DRL) algorithms have been widely used in diverse fields such as robotics, autonomous driving, healthcare and finance. Various DRL approaches for algorithmic trading have been proposed over the last two decades; the problem, however, still can not be considered as solved. Movements in financial markets are the result of human decisions and interactions; thus, they are affected by political, natural events and a variety of unknown factors. This high grade of uncertainty has made the markets extremely challenging to predict. Because understanding human decisions and interactions is essential for trading in financial markets, not only Time-Series Prediction but also Natural Language Processing and Sentiment Analysis are strongly involved in this field, which require a lot of effort and resources.

Apart from the Stock Exchange Market, there is the Foreign Exchange (or Forex) Market, which is the largest financial market in the world and is open 24-hours a day, 5 days a week. The high level of liquidity in the Forex market makes it easier to buy or sell currencies, orders are executed nearly immediately without large price difference. This characteristics is also advantageous for RL agents since delayed execution suffers from market fluctuations, and thus, the actual market trend at execution time could be different

[1] Corresponding Author: Marina Tropmann-Frick, University of Applied Sciences Hamburg, Berliner Tor 7, 20099 Hamburg, Germany; E-mail: marina.tropmann-frick@haw-hamburg.de.

from the trend at order placement time. Although the market dynamics is in general unpredictable, the human behavior is considered invariant, or in other words predictable. Psychological and emotional patterns can be found in the price movements, especially when no rare event is occurring.

RL studies for algorithmic trading mainly learn to trade in the daily time frame and aim to gain large profit in each order. This work, inspired by technical analysis and scalping strategies, proposes an RL approach that uses candlestick charts and technical indicators to learn price movement patterns, aims to make profits from minor changes in price by trading in a short-term manner.

2. Related work

A Double DQN [12] with feed-forward fully-connected layers was used in [9]. Gradient clipping is utilized to prevent the gradient exploding problem. Another study proprosed a policy gradient based method [5] that uses a single recurrent neural network (RNN) and the tanh activation function to approximate the policy; [7] also uses RNN but with Deep Q-Learning approach.

In order to enrich the data and to prevent the model from overfitting, extra minute-candle data were used to train the agent that trades on daily-candle data in [6]. Although the data have the same structure, movement patterns in minute-candle data are different from patterns in daily-candle data. Therefore, only few instruments can be traded using this method; they must be selected by a mechanism based on skewness and kurtosis.

One can consider the classification/regression head of the deep neural network as the policy and the rest as an encoder that learns the representation of the market state. A stack of denoising autoencoders were used in [8], followed by an LSTM layer. In [11], wavelet transforms were added as an extra pre-processing step. Instead of using RNN to process time series data, the present work uses CNN to process time series data, since recurrent operations are much more computational expensive in comparison to convolutional operations.

The reward function in the most studies is mainly the raw profit/loss, i.e. the change in equity after each time step:

$$R_t = v_{t+1} - v_t$$

Another popular reward function is the percentage return:

$$R_t = (v_{t+1} - v_t)/v_t$$

The most popular objective is to maximize the infinite horizon discounted return. Because the infinite horizon discounted return is extremely difficult to approximate in algorithmic trading due to the high grade of uncertainty. In addition, infinite horizon discounted return also takes long temporal dependency in account, which is not desired under the perspective of scalping strategies. This work hypothesizes that aiming to approximate and to maximize short-term return can achieve better result.

Using continuous action space to choose action and trading amount simultaneously is practical; however, the money management strategy will be built-in in the learned policy and extremely difficult to replace. Thus, this work attempts to construct a separate money management module.

3. Method

Each minute can have thousands of ticks; hence, training an agent with historical tick data consumes a vast amount of time. This work aims to construct an RL agent that uses Open-High-Low-Close (OHLC) data and makes a decision every 15 minutes in order to reduce training time and the required computational capacity. However, this approach should also work with tick data (or in real-time), since tick data can be resampled into OHLC format. An update in tick data will change the last candle in the candlestick chart, and the agent can make a new decision after each tick. The RL agent can have only one open position at a time; i.e. the current position must be closed first in order to open a new one.

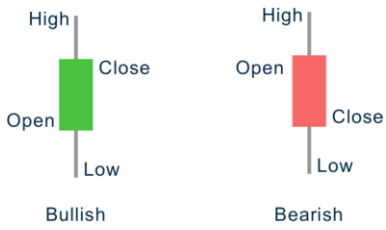

Figure 1. An illustration of candlesticks

3.1. Time series input

OHLC data in 15-minute and 1-hour time frames were used to generate inputs. Each time frame has 32 latest candlesticks. The 15-minute candles capture short-term price actions, while the 1-hour ones help to identify the market trend.

The price movement patterns do not depend the actual price; they rather depend on the relative positions of the past prices. Thus, instead of using raw prices, the log return function is used to transform them into relative movements.

$$LogReturn_t = log(p_{t+1}/p_t)$$

Apart from the price movements, the following technical indicators were utilized in order to predict market trends better:

- Relative Strength Index (RSI).
- Bollinger Bands and %B.
- Money Flow Index (MFI).
- Moving Average Convergence Divergence (MACD).

Technical traders often use MACD by looking at its relative position to the signal line, which is a fast exponential moving average of the MACD line. Therefore, the difference of the two lines is taken instead of both lines.

3.2. Gramian Angular Field (GAF)

The time series input data need to be transformed into a kind of image, so that the 2D convolutional layers of the CNN can process them. Gramian Angular Field is a Gram Matrix; hence, it can preserve temporal dependency since time increase from left to right and from top to bottom. The main idea of GAF is to converts the values into polar coordinates, then uses the sine and cosine functions to compute the final values. Therefore, the input data X need to be scaled to the interval $[-1,1]$ using the following equation:

$$\tilde{x}_i = clip\left(\frac{2x_i - max(X_{train}) - min(X_{train})}{max(X_{train}) - min(X_{train})}, -1, 1\right)$$

Due to the fact that the scaled test data can still lie outside the interval $[-1,1]$, the clip function is applied on the scaled data. This work assumes that the training data must contain enough diverse movement patterns, ranging from slow, small to sudden and large movements.

The variant Gramian Angular Difference Field (GADF) was used in this work, which is defined as follows:

$$\phi_i = arccos(\tilde{x}_i)$$

$$GADF = \begin{bmatrix} sin(\phi_1 - \phi_1) & \dots & sin(\phi_1 - \phi_n) \\ \vdots & \ddots & \vdots \\ sin(\phi_n - \phi_1) & \dots & sin(\phi_n - \phi_n) \end{bmatrix}$$

Each time series of length n is transformed into a GADF of size n x n. Each technical indicator produces a time series; therefore, m indicators result to an output of shape m x n x n, which the CNN can treat as an image of size n x n with m channels.

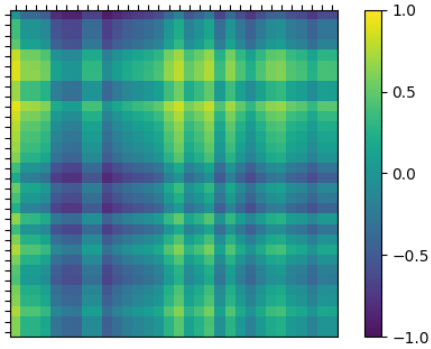

Figure 2. An GADF representation of 32 RSI values

Piecewise Aggregation Approximation (PAA) [13] is often used in combination with GAF in order to smooth the data and to identify trends easier. This work, however, discards PAA because it may remove information that is relevant to match price movement patterns. The trends can still be recognized in a larger time frame (1-hour) without having to apply PAA on the data.

3.3. Proximal Policy Optimization (PPO)

A small batch can contains only one or a few patterns, which may lead to poor policy or model collapse after updating the weights if the step size is large enough. Proximal Policy Optimization (PPO) [2], based on the idea of Trust Region Policy Optimization (TRPO) [3], makes sure the policy is not stepping to far away from the starting point in each update. Instead of using an extra Kullback-Leibler divergence term in the loss function, the Clipped Surrogated Objective function [2] is used:

$$r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

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

Generalized Advantage Estimation (GAE) [4] is used to approximate the advantage function, which is the exponentially-decayed sum of TD residuals:

$$\delta_t^V = R_t + \gamma V(s_{t+1}) - V(s_t)$$

$$\hat{A}_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}^V$$

Apart from GAE, this work hypothesizes that maximizing the cumulative reward in short term can result to better performance. In particular, the ultimate objective is maximizing the return of individual trades, which will act as a guidance for the model to find the optimal policy; namely, profit from a trade does not compensate any loss from previous trades. Hence, the model should always learn to cut loss and take profit at the right time. In order to implement this idea, an additional flag that indicates the end of a trade is stored in the experience memory. The returns are computed as usual but with discount factor $\gamma = 1$ and the return will be reset to zero after seeing the flag.

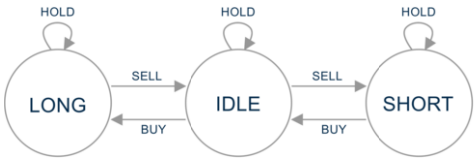

Figure 3. Illustration of the state transitions

3.3.1. Actions

There are 3 discrete actions: *BUY, SELL* and *HOLD*. For instance, a long position can be opened by taking the action *BUY* if the there is currently no open position; a long position can be closed by taking the action *SELL* (see Figure 3). Because there can be only one open position at a time, there are some invalid actions; e.g. the agent cannot take the action *BUY* if a long position is currently open. Invalid actions are handled by using

a valid action mask $m_t \in \{0,1\}^3$ that is provided by the environment. A possible method is applying the mask to the probabilities directly by using element-wise multiplication, followed by redistributing the probabilities for the remaining actions. Another method is applying the mask to the activation of the actions by setting the activation to a large negative number, e.g. $-1e+8$, where the value in the mask is 0. The softmax function will return zero probability for the masked out actions.

3.3.2. Reward

To support the idea mentioned above, the reward function is the change in profit/loss of the opening position (hence, also the change in equity) plus the change in balance, which only changes after closing the position. This can be formulated as the following equation, where E_t is the equity and B_t is the balance at time step t:

$$R_t = (E_t - E_{t-1}) + (B_t - B_{t-1})$$

3.3.3. Loss function

The loss function combines the following components: the Clipped Surrogated Objective L^{CLIP}, the squared-error term of the value function L^{VF} (weighted by c_1) and a small entropy term H (weighted by c_2) to encourage exploration:

$$H(\pi(\cdot|s_t)) = -\sum_{a \in A} \pi(a|s_t) log \pi(a|s_t)$$

$$L(\theta) = -L^{CLIP}(\theta) + c_1 L^{VF}(\theta) - c_2 H(\pi_\theta(\cdot|s_t))$$

3.4. Model architecture

3.4.1. Architecture overview

The CNN extracts features from the GAFs and flattens them into a vector, which is then concatenated with 4 additional information: the current position state (-1 for buying, 1 for selling and 0 otherwise), profit/loss of the opening position, the current hour and the elapsed hours since the position opened divided by 24. The result of the concatenation forms a state of the environment. The actor and the critic are just shallow networks that consist of a hidden layer and an output layer. Figure 4 illustrates the overview of the architecture.

3.4.2. CNN architecture

A CNN with residual connections [14] is used to construct the feature extractor (see Figure 5) . Unlike image classification tasks where only layer-wide activation is important since spatial translation is irrelevant, the cell position in this approach contains temporal meaning and has significant contribution to understanding the patterns. Thus, global pooling layer is not used and downsampling techniques should only be used with consideration since it also reduces temporal information.

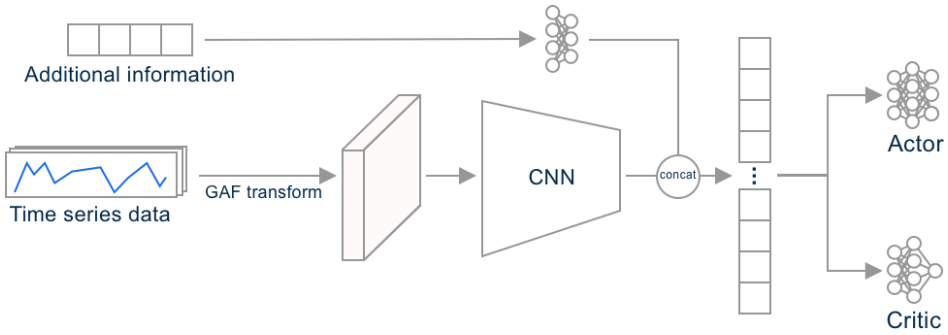

Figure 4. Overview of the architecture

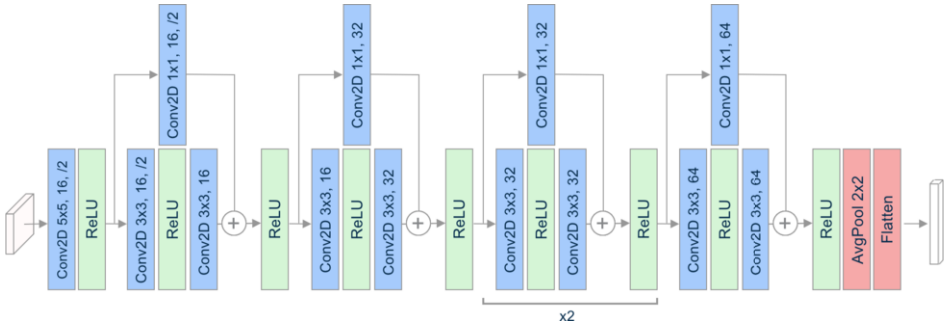

Figure 5. Illustration of the CNN architecture

3.5. Money Management with an Extra Actor

In order to assign the size of market orders dynamically, an extra regression head is used that acts as a second actor. The goal of this second actor is to estimate the chance of winning positive return in the current state given the actual policy. The output is a single value that lies in the interval $[0,1]$. Given a range of possible sizes for market orders $[S_{min}, S_{max}]$, the actual size can be determined by the following equation:

$$S_i = S_{min} + \hat{y}_i(S_{max} - S_{min})$$

The second actor estimate the chance of winning based on the current policy; thus, gradients from this actor should not flow back to the convolutional network during back-propagation. However, because the policy is constantly changing during the training phase, training both the actors simultaneously can easily makes the model suffer from model collapse, or even worse, hit NaN loss during training. It is better to train this actor separately in a self-supervised manner, where the labels are generated on-the-fly by the actual returns of the agent. The binary cross entropy loss function is used in this work, which can be formulated as follows:

$$L^{MM} = -\frac{1}{N}\sum_{i=1}^{N}(\mathbb{1}_{G_i>0} log(\hat{y}_i) + \mathbb{1}_{G_i\leq 0}(1-\hat{y}_i))$$

4. Experiments & Results

Two years data of the currency pair EUR/USD[2] were used to train and test the agents:

Training data: Jan. 2017 $\rightarrow$ Dec. 2017

Test data: Jan. 2018 $\rightarrow$ Dec. 2018

All the experiments have the following hyper-parameters:

- Learning rate: 0.0003
- Batch size: 128
- Replay memory size: 1024
- Clipping range ε: 0.2
- Value function coefficient $c1$: 0.5
- Entropy coefficient $c2$: 0.001

The discount factor γ is 0.99 and GAE-λ is 0.95 when using GAE, whereas γ is 1.0 when maximizing finite horizon undiscounted return.

Figure 6. Training data and test data

The optimizer used in the experiments is Adam [15]. The weights were initialized by the Xavier Uniform method [16], which inits the weights from a uniform distribution $\mathscr{U}(-a,a)$:

$$a = gain \times \sqrt{\frac{6}{fan\_in + fan\_out}}$$

Because the models in this work use only ReLU as activation function, *gain* is always $\sqrt{2}$. In addition, a dropout layer is added after each convolutional block, the dropout rate is 0.2. L2 regularization term is also added to the loss function in order to keep the weights small; $\lambda = 0.0001$ is used in the experiments. This can be formulated as follows, where L_0 denotes the unregularized loss.

[2]Provided by FXCM. Online available at https://github.com/fxcm/MarketData

$$L = L_0 + \frac{\lambda}{2n} \sum_w w^2$$

This papers assumes that the agent trades in micro-lots, i.e. 1 lot is equal to 1,000 units of the base currency instead of 100,000 units when trading in standard-lots. Every agent starts to trade with a capital of € 100.

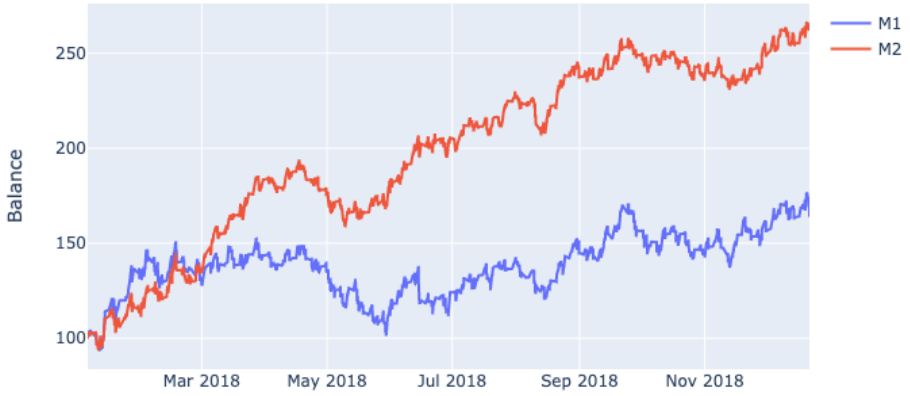

Figure 7. Comparison between M1 and M2

The first experiment compared the two approaches: maximizing infinite horizon discounted return (denoted as M1) and maximizing undiscounted return of individual trades (denoted as M2). Both agents were trained with the same initialization, hyper-parameters. The size of orders is constant in this experiment: 1 lot. The results in Figure 7 shows that maximizing undiscounted return of individual trades indeed helped to achieve better performance. Although the market trends in 2018 are very different from the trends in 2017 (see Figure 6), both agents managed to achieve positive return at the end of the year. However, the performance of M2 is significantly better than that of M1. Moreover, max drawdown of M1 is 33.5%, while M2 has only **18.1%**.

In the second experiment, the same agent was evaluated 3 times with different settings: fixed order sizes (1 and 2 lots; denoted as F1 and F2, respectively), and dynamic order size with the money management module (denoted as MM), $S_{min} = 1$ and $S_{max} = 2$.

The results of the second experiment is shown in Figure 8. The agent MM achieved slightly better performance than F2. The max drawdown of F2 is 26.68%, whereas the max-drawdown of MM is **23.29%**. The results confirmed that the money management module can indeed estimate the chance of winning with and therefore, helps to reduce loss and maximize profit.

5. Conclusion

This work experimentally showed that reinforcement learning can be applied for Automated Forex Trading with scalping strategies, which only use technical indicators to predict short-term market trends. Although scalping strategies only catch small price movements, they can still manage to earn a large amount of profit since the agent trades in relatively high frequency.

Figure 8. Comparison between fixed order sizes and dynamic order size

Gramian Angular Fields help to process time series data with convolutional neural networks, which reduce computation time since RNN is slower than CNN by design. The model architecture for CNNs that handle GAF input, however, is different from that for CNN in computer vision tasks, since the cell position can contain relevant temporal meaning. Downsampling techniques can be used but with caution.

The first experiment confirmed the hypothesis that maximizing cumulative reward of individual trades can help to achieve better performance. The main reason could be the elimination of the chance to compensate losses from previous trades, which relies heavily on upcoming price movements at that time. This behavior reduces the generalization capability of the model. Thus, the elimination acts as a guidance for the model to know where to focus on.

An additional money management module can help to reduce loss and maximize profit, since it can estimate the chance of winning in the current state. This, however, needs to be trained separately in a self-supervised manner because the chance of winning strongly depends on the current policy. Training this module with the main model simultaneously is extremely unstable. Further methods for money management strategy could be an interesting topic for future work.

References

[1] Wang, Zhiguang, and Tim Oates. "Encoding time series as images for visual inspection and classification using tiled convolutional neural networks." Workshops at the twenty-ninth AAAI conference on artificial intelligence. Vol. 1. 2015.

[2] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[3] Schulman, John, et al. "Trust region policy optimization." International conference on machine learning. PMLR, 2015.

[4] Schulman, John, et al. "High-dimensional continuous control using generalized advantage estimation." arXiv preprint arXiv:1506.02438 (2015).

[5] Deng, Yue, et al. "Deep direct reinforcement learning for financial signal representation and trading." IEEE transactions on neural networks and learning systems 28.3 (2016): 653-664.

[6] Yuan, Yuyu, Wen Wen, and Jincui Yang. "Using Data Augmentation Based Reinforcement Learning for Daily Stock Trading." Electronics 9.9 (2020): 1384.
[7] Zengeler, Nico, and Uwe Handmann. "Contracts for Difference: A Reinforcement Learning Approach." Journal of Risk and Financial Management 13.4 (2020): 78.
[8] Li, Yang, Wanshan Zheng, and Zibin Zheng. "Deep robust reinforcement learning for practical algorithmic trading." IEEE Access 7 (2019): 108014-108022.
[9] Théate, Thibaut, and Damien Ernst. "An application of deep reinforcement learning to algorithmic trading." Expert Systems with Applications 173 (2021): 114632.
[10] Park, Hyungjun, Min Kyu Sim, and Dong Gu Choi. "An intelligent financial portfolio trading strategy using deep Q-learning." Expert Systems with Applications 158 (2020): 113573.
[11] Bao, Wei, Jun Yue, and Yulei Rao. "A deep learning framework for financial time series using stacked autoencoders and long-short term memory." PloS one 12.7 (2017): e0180944.
[12] Van Hasselt, Hado, Arthur Guez, and David Silver. "Deep reinforcement learning with double q-learning." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 30. No. 1. 2016.
[13] Keogh, Eamonn, et al. "Dimensionality reduction for fast similarity search in large time series databases." Knowledge and information Systems 3.3 (2001): 263-286.
[14] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[15] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
[16] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2010.