Pertanika J. Sci. & Technol. 30 (4): 2683 - 2705 (2022)

PERTANIKA JOURNALS

SCIENCE & TECHNOLOGY

Journal homepage: http://www.pertanika.upm.edu.my/

Automated Cryptocurrency Trading Bot Implementing DRL

Aisha Peng*, Sau Loong Ang and Chia Yean Lim

Department of Computing, UOW Malaysia KDU Penang University College, 10400 George Town, Pulau Pinang, Malaysia

ABSTRACT

A year ago, one thousand USD invested in Bitcoin (BTC) alone would have appreciated to three thousand five hundred USD. Deep reinforcement learning (DRL) recent outstanding performance has opened up the possibilities to predict price fluctuations in changing markets and determine effective trading points, making a significant contribution to the finance sector. Several DRL methods have been tested in the trading domain. However, this research proposes implementing the proximal policy optimisation (PPO) algorithm, which has not been integrated into an automated trading system (ATS). Furthermore, behavioural biases in human decision-making often cloud one's judgement to perform emotionally. ATS may alleviate these problems by identifying and using the best potential strategy for maximising profit over time. Motivated by the factors mentioned, this research aims to develop a stable, accurate, and robust automated trading system that implements a deep neural network and reinforcement learning to predict price movements to maximise investment returns by performing optimal trading points. Experiments and evaluations illustrated that this research model has outperformed the baseline buy and hold method and exceeded models of other similar works.

Keywords: Automated trading system, deep neural network, reinforcement learning

INTRODUCTION

Cryptocurrencies and cryptocurrency trading have significantly succeeded as an emerging business and research topic, with a noticeable uptick in interest and activity. For example, Fang et al. (2022) survey found

ARTICLE INFO

Article history:
Received: 28 October 2021
Accepted: 05 April 2022
Published: 21 September 2022

DOI: https://doi.org/10.47836/pjst.30.4.22

E-mail addresses:
aishaaishapeng@gmail.com (Aisha Peng)
sauloong.ang@kdupg.edu.my (Sau Loong Ang)
cylim@usm.my (Chia Yean Lim)
* Corresponding author

Current affiliation:
Chia Yean Lim
School of Computer Sciences, Universiti Sains Malaysia, 11800, Pulau Pinang, Malaysia

ISSN: 0128-7680
e-ISSN: 2231-8526

that there are emergent trading technologies, including machine learning technology and other emergent trading methods, portfolio and cryptocurrency assets research, and market condition research since 2018.

There are a few distinct advantages of trading cryptocurrency. For instance, its drastic fluctuations, its 24/7 availability thanks to its decentralised market, low transaction costs using peer-to-peer transactions, and programmable "smart" capabilities bring benefits to holders.

However, trading activities can be controversial as some people relate them to "legal" gambling. However, a good trader looks for consistency, whereas a gambler looks for quick profits. Thus, this research aims to investigate a suitable architecture to integrate deep reinforcement learning (DRL) into a trading bot.

The motive of this trading bot is to train a model to learn from carefully selected technical indicators to learn to trade for profits consistently and accurately. This research aims to expose more people to cryptocurrency applications and provide an option for beginner investors who have no prior knowledge of trading but would like to earn passive income. Additionally, this research hopes to affiliate with people interested in fintech and would like to explore the potential of DRL in trading.

The reason for undertaking this research includes the lack of a baseline. Despite much work and research published on applying DRL to video gameplay and robotics, there is relatively little work on applying DRL to financial trading (Li, 2017). No standard baseline or publicly acknowledged architecture exists to create the proposed model (Huang, 2018). Additionally, this research aims to fill the absence of commercially available products. The current commercially available product commonly uses a trading strategy based on technical indicators to control the trading points, with no application of artificial intelligence (AI).

In addition, simple trading systems may be sufficient for capturing recurrent patterns. However, they are incompetent for the dynamically random cryptocurrency market as they assume a linear relationship with no feedback from the environment. At the same time, the cryptocurrency market has many exception values and is largely affected by external factors, including political and geographical news. Chaos theory—a conceptual framework of nonlinear dynamic systems resolves the unpredictability with distinctive patterns (Cartwright, 1991). Neurons in deep learning architecture can exhibit chaotic dynamics; hence this research aims to utilise deep learning techniques to reconcile the unpredictability of the chaotic trading problems.

This research also aspires to resolve the human hindrance of being always available. Humans have limitations; they cannot focus on the market all the time and always think rationally. Although there are automated functions such as grid trading, these functionalities are simply fixed positions with no flexibility and AI integrated. Not to mention, humans tend to be their own greatest enemy when it comes to financial trading due to emotions. For example, humans fear missing trading opportunities during market euphoria and panic selling in a market downtrend.

LITERATURE REVIEW

Automated Trading System

Automated trading systems (ATS) take past market prices as inputs and generate a trading signal suggesting the optimum. In other words, it is a decision-making system where the dependability and performance of a system are subjective to the efficiency of big data analysis and modelling (Huang et al., 2019).

Many attempts have been made to develop a consistently profitable system in different fields ranging from fundamental analysis and econometric modelling of financial markets to machine learning (Dempster & Romahi, 2002; Moody et al., 1999). Nonetheless, few attempts were successful, and most could not be used to trade actual markets due to associated practical disadvantages such as large draw-downs in profits and excessive switching behaviour resulting in very high transaction costs (Dempster & Leemans, 2006).

Generally, the main advantages of ATS include (1) emotional trading elimination, (2) greater discipline in following strategy rules and more consistent behaviour, (3) virtually guaranteed participation in every important trend, and (4) minimised losses. Regardless, the known cons of ATS are (1) trend-following systems rely on major trends to be profitable, (2) non-trending markets are non-profitable, and (3) unable to recognise that market is not trending (Tucnik, 2010).

Reinforcement Learning

The RL framework includes an agent that optimises its behaviour by interacting with its environment. After taking action in some state, the agent receives a scalar reward from the environment, indicating the quality of that action. The agent's main goal is to find a policy that maximises the total accumulated reward. By iteratively interacting with the environment, the agent uses past experiences to decide on future actions to take in or around a certain state (Grondman et al., 2012).

Proximal Policy Optimisation Algorithm

PPO algorithm is a policy optimisation method where the agent directly learns the policy function that maps state to action. The policy optimisation approach maintains a parameterised action-selection policy. It updates the policy parameters by moving them toward an estimate of the gradient of a performance measure (Van Otterlo & Wiering, 2012). The most commonly used gradient estimator is obtained by differentiating the policy gradient objective function in Equation 1:

$$L^{PG}(\theta) = \hat{\mathrm{E}}_t[\log \pi_\theta (a_t|s_t)\hat{\mathrm{A}}_t] \quad [1]$$

where θ represents the parameter values (weights), $\pi_\theta\,(a_t|s_t)$ is the policy, and $\hat{\mathrm{A}}_t$ is an estimator of the advantage function at timestep t.

However, the advantage estimate $\hat{\mathrm{A}}_t$ uses a value function to estimate how good it is for the agent to be in a certain state (Schulman et al., 2015). Since this is only an estimate of a neural network, the estimation will be noisy. Thus, multiple optimisation steps based on the policy gradient loss often lead to destructively large policy updates (Schulman et al., 2017). Furthermore, it will push the policy network into a region of parameter space where the subsequent data will be collected under a very poor policy, causing it never to recover again.

Therefore, the team of OpenAI designed the PPO algorithm based on the trust region policy optimisation (TRPO) to resolve the mentioned issue (Schulman et al., 2017). The core idea is to include a constraint directly into the optimisation objective so the updated policy does not move too far away from the old policy, sticking close to a region where everything works fine. Hence, the objective function for PPO is in Equation 2.

$$L^{CLIP}(\theta) \;=\; \hat{\mathrm{E}}_t[\min(r_t(\theta)\hat{\mathrm{A}}_t, clip(r_t(\theta), 1 \;-\; \varepsilon, 1 \;+\; \varepsilon)\hat{\mathrm{A}}_t)] \qquad [2]$$

$(r_t(\theta)$ is simply the probability ratio between the updated policy network outputs and the old policy network outputs:

$$r_t(\theta) \;=\; \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta\ old}(a_t|s_t)} \qquad [3]$$

In Equation 3, when the estimated advantage is positive, the L^{CLIP} function clips the r value to limit the effect of gradient update. Similarly, when r goes near 0. Similarly, the update will be limited to prevent reducing the action probability to 0. Since the estimated advantage is noisy, destroying a policy based on a single estimate is unwise.

To summarise, PPO is a family of policy optimisation methods that use multiple epochs of stochastic gradient ascent to perform each policy update. As a result, PPO provides stability and reliability and is simple to implement, requiring only a few lines of code change to a vanilla policy gradient implementation (Schulman et al., 2017).

Deep Neural Network

In 1957, Frank Rosenblatt created the perceptron. It is the prototype of the now-known neural network (NN) (Rosenblatt, 1958). NN is an ML technique inspired by the human brain. It consists of many simple, connected processors called neurons, each producing a sequence of real-valued activations. Simply put, NN is nothing more than a certain type of nonlinear function (Anthony et al., 1999).

A deep neural network (DNN) is a type of NN modelled as a multilayer perceptron (MLP). DNN is trained to learn representations from data sets without manual feature extractors (Shrestha & Mahmood, 2019). It is used in RL to approximate the value function (Li, 2017). It allows RL to be applied in larger problems as the state action pair grows, unlike non-deep RL, which requires storing the values via a tabular function.

Convolutional Neural Network

Convolutional Neural Network (CNN) is used in this research to leverage its ability to extract useful knowledge and learn the internal representation of the cryptocurrency market data (Livieris et al., 2020).

CNN is mainly composed of two parts: the convolution layer and the pooling layer. Each convolution layer contains a plurality of convolution kernels. The convolution kernel (filter) can be considered a tiny window that ''slides'' all over the input matrix, applying convolution operation on each subregion (patch) that this specified window ''meets'' across the input matrix.

After the convolution operation of the convolution layer, the extracted feature dimensions are very high. Therefore, a pooling layer is added after the convolution layer to reduce the feature dimension (Lu et al., 2020). A pooling layer is a subsampling technique to extract certain values from the convolved features to produce a lower dimension matrix. As a result, the pooling layer produces new matrices, which can be considered summarised versions of the convolved features produced by the convolutional layer. The pooling operation can help the system to be more robust since small changes in the input will not change the pooled output values (Livieris et al., 2020).

Long Short-Term Memory

Long short-term memory (LSTM) is a network model designed to solve the longstanding problems of gradient explosion and gradient disappearance in recurrent neural networks (RNN) (Hochreiter & Schmidhuber, 1997).

The LSTM networks are composed of an input layer, one or more hidden layers, and an output layer. Each LSTM memory cell has three main gates (input, output, and forget) to maintain and adjust its cell state. The purpose of the forget gates is to provide a way for the memory cells to reset themselves, which is proved to be important for tasks that require the network to "forget" previous inputs (Graves, 2012). The input gate specifies which information is added to the memory (cell state). In contrast, the output gate specifies which information from memory (cell state) will be used as output information (Wu et al., 2018).

As CNNs are not usually adapted to manage complex and long temporal dependencies, the LSTM network is used in this research to cope with temporal correlations. However, simultaneously, CNN exploits features provided in the training set, which LSTM cannot do.

Therefore, a time-series model which exploits the benefits of both deep learning techniques could improve the prediction performance (Livieris et al., 2020).

Related Works

Several past works have implemented DRL into an ATS. All the results have proven that DRL techniques can outperform baseline methods in the stock market or cryptocurrency market. However, none of the mentioned works implemented a CNN-LSTM in their model. In contrast, this research leverages CNN's high sensitivity to undergo feature extraction from the historical data and technical indicators of the crypto market and LSTM time delay characteristics to build a robust and consistent trading bot.

Sattarov et al. (2020) developed a cryptocurrency trading points recommending system using DRL. Their experimental results on the BTC historical prices are positive, with a 74.6% profit within a month (19 March 2019 to 21 April 2019). However, the trend in that month is generally trending upwards. At the same time, this research will be evaluated in a complete business cycle, where a specific downtrend period is included to ensure fairness and truly evaluate the robustness and stability of the model.

Li et al. (2019) implemented the deep Q-network (DQN) and the asynchronous advantage actor-critic (A3C) as their DRL algorithm to autonomously make trading decisions and gain profits in the dynamic financial markets. They also used an LSTM model as a part of the function approximator. As expected, the A3C outperformed the DQN algorithm as it is too complex to learn the Q function with a value-based algorithm. On the other hand, the policy-based algorithm can learn a good policy since it directly operates in the policy space. Therefore, this research implemented the policy-based PPO algorithm.

Xiong et al. (2018) also implemented a Deep Deterministic Policy Gradient (DDPG) to find the best trading strategy in the dynamic stock market. The results show the DDPG algorithm to achieve a higher return than the traditional min-variance portfolio allocation method and the Dow Jones Industrial Average (DJIA), with an annualised return of 22.24%. The model also achieved a much higher Sharpe ratio indicating it is more robust than the others in balancing risk and return. Nonetheless, there is no mention of data transformation. Cryptocurrencies as an activity in a payment system are influenced by seasonal effects (Haferkorn & Diaz, 2014). Thus, this research implemented differencing techniques to remove seasonality from the historical prices dataset. Furthermore, technical indicators are introduced into the dataset to capture different types of information about the market. It is proven that combining information from technical indicators and macroeconomic variables produces superior forecasts to better track the market's substantial countercyclical fluctuations (Neely et al., 2014).

Lucarelli and Borrotti (2019) made a comparison of Double Deep Q-Network (D-DQN), Dueling Double Deep Q-Network (DD-DQN) and DQN in implementing a BTC

ATS. Their proposed system has an intact stop-loss and take-profit strategy applied upon a certain threshold. Besides, each model is built with two different settings: profit reward function and Sharpe ratio reward function. Their Sharpe D-DQN model shows the best result with an average return of 5.81% and a maximum return of 26.14% from 1 December 2014 to 27 June 2018. Nevertheless, the fact that the model is trained in the 1-minute interval may not be practical for real-time trading as the markets are too volatile, and the model may not be able to consider time delays. Besides, the function implemented was the standard mean squared error (MSE) loss function, which, as mentioned, might cause large inaccurate updates to the model due to the noisy value function, which is a serious issue in RL as updates are carried out iteratively. Unlike the PPO algorithm, which clips the estimated loss, the model is updated in a trusted, safe region.

As of the writing of this journal, there has yet to be the implementation of deep reinforcement learning, especially with the PPO algorithm for trading the cryptocurrency market. Besides, using two different deep learning models is also an upcoming noteworthy area of research. It has not been thoroughly configured to be applied in predicting trading signals in the cryptocurrency markets. This research aims to revolutionise the implementation of deep reinforcement learning on fintech domains, as well as showcase the outstanding trading performance of deep reinforcement learning in trading problems.

METHODS

Research Methodology

There are four main stages conducted in this research. The first stage is the data acquisition from Binance exchange REST API v3. The second stage is the data transformation applied to the dataset before training. Technical indicators and differences, as well as normalisation techniques, are applied to transform the input data. The third stage is the model's training, where RL and DNN architecture is used to update the policy network. The fourth stage is where evaluation analysis is conducted on the trained model to evaluate the accuracy and robustness of the model.

Dataset Interpretation

The dataset of historical k-line is obtained from Binance exchange REST API. The obtained dataset is separated into market cycles, as seen in Figure 1, for training and testing purposes and to allow the model to identify the market cycle phase in different timesteps. It is important to split the training and testing dataset. Just because a model performs well on its training data does not mean it will perform well on data it has never seen, and what matters in this study is that the model can predict in real-time unseen data. (Chollet, 2017). The dataset is first split on Bitcoin's halving date (11 May 2020). Cycles 1 and 2 will be

used to train the model, whereas cycle 3 is the testing dataset. The imbalanced ratio (70/30) ensures the model has enough training dataset to learn for an effective mapping of inputs to outputs. As there is no limitation for setting a splitting ratio, 70/30 is a common dataset splitting proportion. Nonetheless, the optimal splitting ratio varies on the characteristics of different datasets (Vrigazova, 2021).

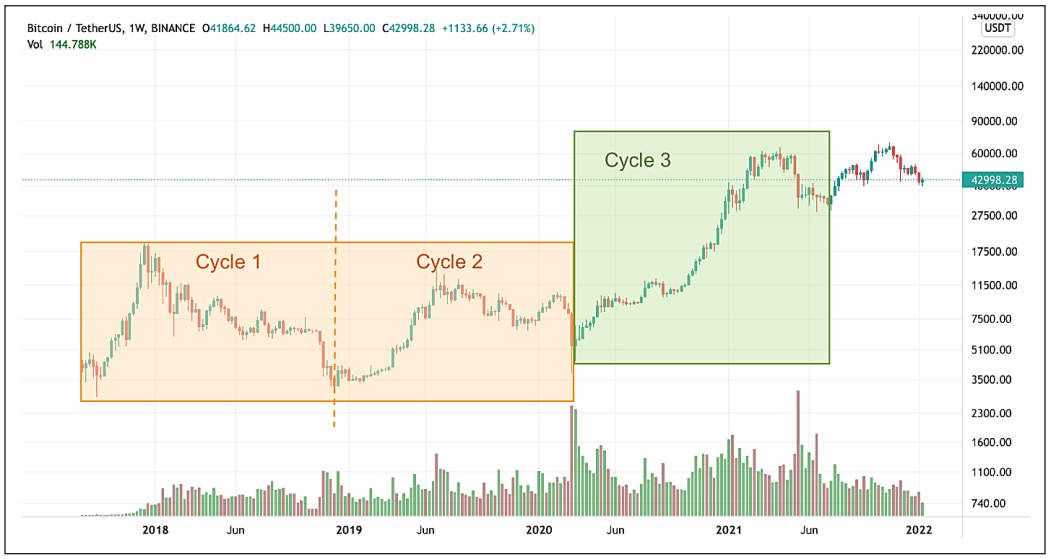

Figure 1. Splitting training and testing dataset

Technical Indicators

Technical indicators are included in the dataset to capture various market information and reduce the noise of the volatile cryptocurrency markets. The chosen technical indicators are true average range (volatility indicator), relative strength index (momentum indicator), and Chaikin money flow indicator (volume indicator). It shall be observed that different categories of technical indicators are used. It improves the model by minimising highly correlated dataset features, as high feature correlation increases redundancy in representations, which is undesirable (Zhang et al., 2018). Furthermore, correlated features add noise and redundancy to the model, making knowledge discovery during the training phase more difficult (Kotsiantis et al., 2006).

These indicators are also chosen because their values are in a limited range. For example, the relative strength index ranges from 0 to 100, and the Chaikin money flow indicator has a zero-centred value. If other indicators were to be used, such as Bollinger Bands, the values are highly dependent on seasonal variations. As PPO uses experience replay to train the model at the end of each episode, having time-dependent values will introduce bias to certain states; as the model goes into implementation, it might not be able to perform well as it does not recognise the values anymore as time changes. For example,

it would be hard for the model to predict a downtrend if it was learned on uptrend data while training.

Data Pre-Processing

In practice, it is nearly always advantageous to apply pre-processing transformations to the input data before it is presented to a network (Bishop, 1995). As stated, time-series data is not stationary, as the model also uses the historical close prices to act as a trend indicator; the data needs to be processed.

The first stage in any analysis should be to see if there is any indication of a trend or seasonal impacts, and if so, remove them. Therefore, the data fed to the stationary model are a realisation of a stationary process (Cowpertwait & Metcalfe, 2009). Figures 2 and 3 show the difference before and after processing the closing data prices. Before normalising the data, a technique known as differencing is applied. Differencing can help stabilise the mean of the time series by removing changes in the level of a time series and eliminating or reducing trend and seasonality (Hyndman & Athanasopoulos, 2018).

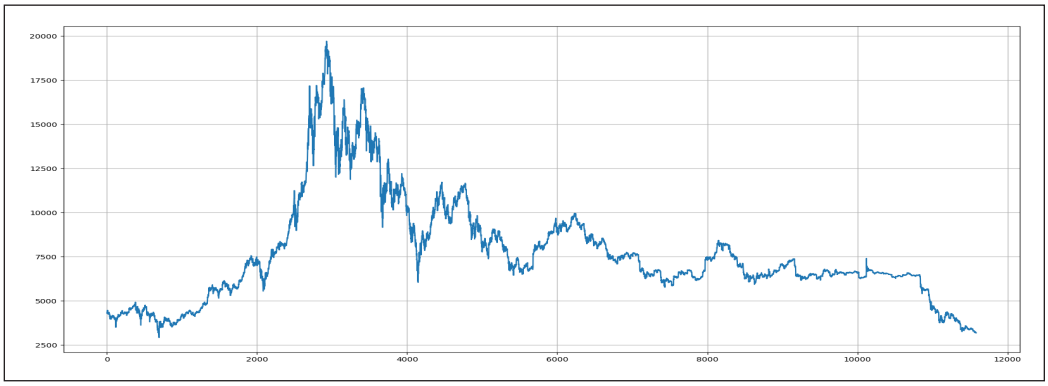

Figure 2. Data before differencing

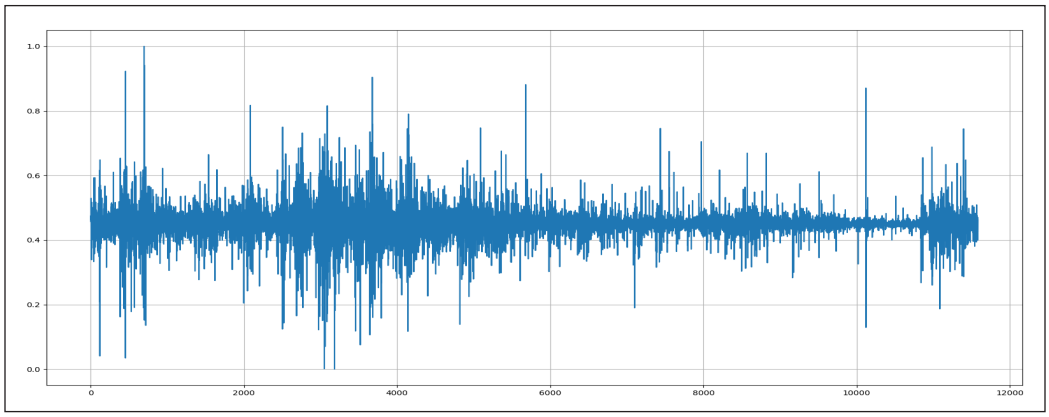

Figure 3. Data after differencing

After applying differencing to the closing data prices, all the feature data, including the technical indicators, are normalised. Research has found that input data normalisation prior to training is crucial to obtaining good results and significantly fastening the calculations (Sola & Sevilla, 1997).

Reinforcement Learning

PPO is chosen as the RL algorithm due to a few factors. Firstly, it must be a model-free algorithm. It is due to the volatility of the crypto market, where the prices are extremely dynamic, and short-term decline can happen unexpectedly. Implementing the model-free approach allows the agent in this research to rapidly adapt when the environment changes its way of reacting to the agent's actions.

Furthermore, PPO uses the actor-critic framework. Typically, RL methods fall under two categories: policy-based (actor) or value-based (critic). Actor-critic methods combine the best of both worlds. The critic uses an approximation architecture to learn a value function, which is then used to update the actor's policy parameters in the direction of performance improvement. As a result, it reduces the high variance of the policy networks, resulting in a more robust model. Additionally, actor-critic methods promise variance reduction, thus resulting in faster convergence (Konda & Tsitsiklis, 1999).

As the trading environment is a continuous state-action space, the actor can produce continuous actions without optimisation procedures on a value function because the critic is there to evaluate the current actor's policy by approximating and updating the value function using samples, which also speeds up the learning process (Grondman et al., 2012). As a result, the updated policy analyses how good the action is and how much better it could be. Hence, the stability provided by the characteristics of the actor-critic framework makes it a great model for the financial trading environment (Yang, 2020).

Keeping in mind that this research aims to build a stable and robust training bot, PPO is a good fit to adapt to a trading strategy that incurs the most profit in the long term instead of an unsustainable strategy that only works for a rapid gain. Generally, PPO is a simple to implement yet reliable RL algorithm; thus, it is a competent algorithm to start for an automated trading domain (Schulman et al., 2017). Additionally, the RL trading environment simulation improves the agent's generalisation and robustness to the environment's changes (Ganesh et al., 2019). A good, simulated environment provides a high degree of reduction to simulate multiple policies for evaluation and improvement, including various cases such as the different phases of market cycles (Zhang, 2021).

Given the cryptocurrency market domain, the RL environment is custom-built to adapt to the cryptocurrency market domain. This RL environment trains the agent through positive reinforcement by interacting with the environment. Essentially, the agent conducts "trial and error" to maximise its reward (Kolm & Ritter, 2019). The agent will be interacting

with the environment using state space, action space and reward functions that are specifically designed. The state-space specifies the observations the agent receives from the environment. The action space specifies the possible interactions between the agent and the environment. Finally, the reward function incentivises to drive the agent to learn a better action (Liu et al., 2020).

Table 1
Example of input states to model at each step. Each state has 100 hours of market information containing the closing price and three different technical indicators

Timestamp	Closing price	Relative strength index indicator	Normalised average true range indicator	On-balance volume indicator
1502942400	4261.32	49.48	1.42	-0.14
1502946000	4291.37	51.89	1.46	-0.07
1502949600	4309.37	56.01	1.48	0.01
…	…	…	…	…
1503302400	4036.3	48.80	1.33	-0.02

As shown in Table 1, with each step, the states are a lookback window of 100 hours consisting of the market information, including the technical indicator. The technical indicators are derived from the unprocessed dataset's open, high, low, close, and volume (OHLCV). In other words, the model will predict the action based on the 100 hours of information generated by the environment at each step.

Table 2
Possible agent actions as predicted by the model

Actions	Effect
Buy	Buy crypto assets at current price with all the current net worth. Deduct net worth and add holding crypto amount.
Hold	Do nothing.
Sell	Sell all crypto assets. Deduct holding crypto amount and add net worth.

As depicted in Table 2, the possible actions by the agent are buying, holding, or selling. In the RL environment, there is a "net worth" variable that keeps track of the agent's current net worth and another "crypto holding" variable that keeps track of the agent's current holding amount of crypto assets. To simulate a real trading environment, the agent will perform a "buy" action each time the model predicts a buying signal. As a result, the environment will deduct the "net worth" variable and add the amount of crypto bought to the "crypto holding" variable. The amount bought is calculated by Equation 4.

$$Amount\ bought = \frac{Current\ net\ worth}{Current\ crypto\ closing\ price} \quad [4]$$

If the predicted action is held, the agent will not do anything. Whereas, if the predicted action is sold, the environment will now deduct all the current amount of the "crypto holding" variable and adds the equivalent amount of money to the "net worth" variable, which can be calculated by Equation 5.

$$Amount\ sold = Current\ crypto\ amount\ held \times Current\ crypto\ closing\ price \quad [5]$$

After taking action, the reward for the predicted action is computed using the formula in Equation 6.

$$Reward = Current\ net\ worth - Previous\ net\ worth \quad [6]$$

For instance, if the agent's selling action results in a smaller net worth value than its previous net worth, the reward will be negative, incurring a penalty to the model. On the contrary, if the action taken results in a greater net worth value than its previous net worth, the reward will be a positive value. As the model training aims to increase the agent's net worth, this reward system will drive the model to learn by attempting to maximise the reward and avoid negative rewards.

Figure 4 depicts the model's flowchart of the training and predicting process. As the actor-critic framework is implemented, the flowchart shows how the critic model is

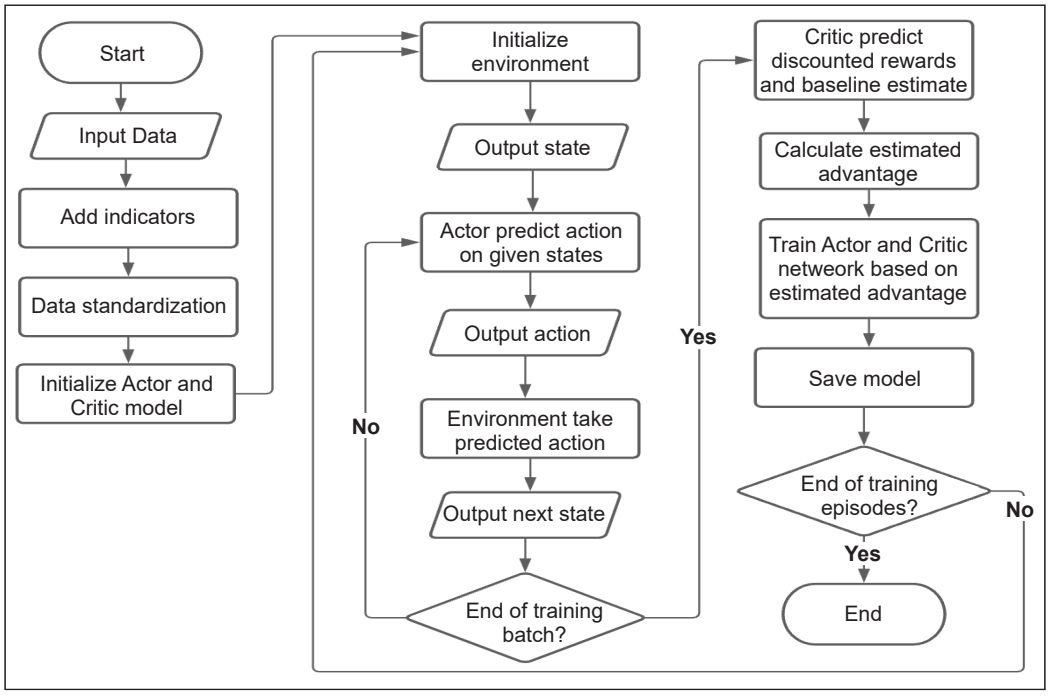

Figure 4. Flowchart of training and predicting process

responsible for predicting the discounted rewards, which will be used to calculate the estimated advantage. The estimated advantage is then used to train the actor and critic models.

Figure 5 depicts the pseudocode for how RL model training is conducted. The environment is reset for each training batch, and the trajectories of the end of each training batch are collected. The collected trajectories are used to compute the estimated advantage and update the policy by training the actor and critic models.

Figure 6 illustrates how RL is integrated with deep learning architecture. The RL environment passes the states and generates the rewards for the deep learning model, and the deep learning model will return predicted action onto the RL environment. In addition, the model loss will be clipped by the PPO loss function to restrict the weight updates during model training.

```
for i = 1, 2, ..., training episodes do
        Environment reset
        for j = 1, 2, ..., training batch size do
                Collect trajectories by running old policy in environment
        end for
        Compute estimated advantage
        Update policy
end for
```

Figure 5. Pseudocode for model training

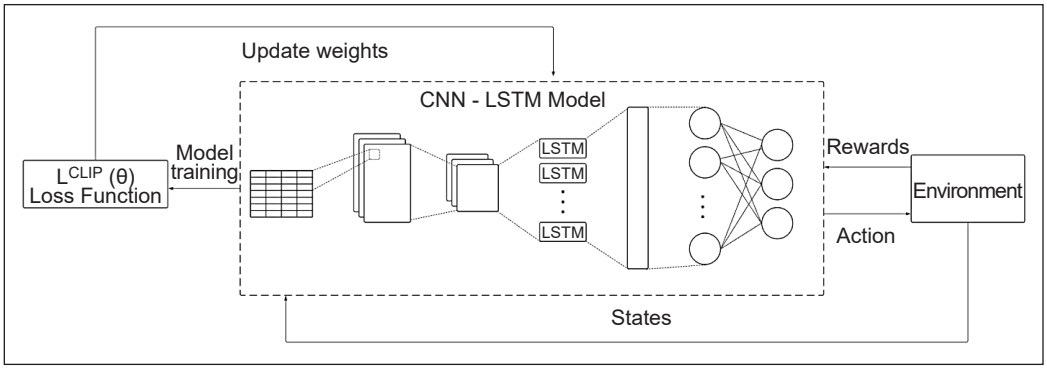

Figure 6. Integration of reinforcement learning and deep learning

Deep Learning Architecture

The CNN-LSTM architecture in this research is designed as shown in Figure 7. As CNNs are not usually adapted to manage complex and long temporal dependencies, the LSTM network is used in this research to cope with temporal correlations. Furthermore, CNN is paying attention to the most obvious features in the line of sight, so it is used in this model to perform feature engineering. Along with its convolutional and pooling layers, it

can filter the input data and extract useful information as an input, enhancing the model's performance (Livieris et al., 2020). Aside from feature engineering, CNN is also superior in its ability to deliberate the relationship between the extracted features and hence perform classification (Liao et al., 2017). Hence, CNN is popular in applications such as image analysis and financial time-series prediction.

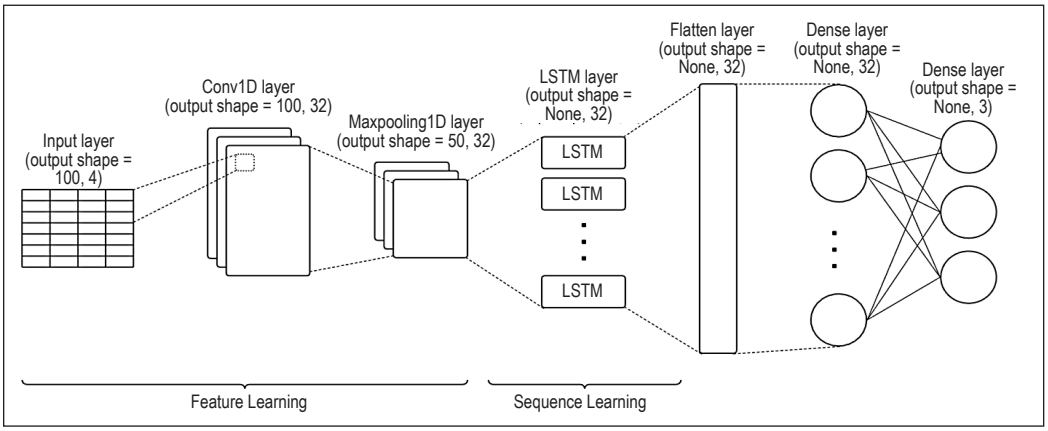

Figure 7. Actor deep learning model architecture

However, CNN is a highly sensitive model as even small translations or rescaling of the input data can drastically change the network's prediction (Azulay & Weiss, 2018). However, it might introduce large variance into the function approximator, causing major updates to the policy network due to the noisy value function. By integrating LSTM with CNN, LSTM can tone down the high sensitivity of CNN by introducing a delay into the model. In addition, LSTM allows the model to be more stable by considering past market information with its long memory by storing the information in its cell state as compared to shorter memory RNN. It allows the model to avoid pitfalls such as a sudden market spike.

The research uses transfer learning to overcome the problem of hard-to-train financial data due to high volatility and insufficient data (Jeong & Kim, 2019). However, this learning approach is not implemented in this research as it may hurt the learning performance in the target domain, a situation often referred to as negative transfer (Pan & Yang, 2009). Furthermore, it prevents the pre-trained model from introducing additional bias into the model, causing the learning process to side-track, especially when BTC holds unique market characteristics due to extreme price movements. Not to mention that investors' attention to the crypto market drastically changed over time (Gronwald, 2014).

The main parameter involved in this proposed model is its weights, which will be optimised through iterative updates during experience replay in the training process. Furthermore, the proposed model integrates with RL to perform optimal weight updating based on the stable PPO loss function instead of the conventional noisy estimated value function.

Evaluation Metric

The evaluation metric is important for discriminating and obtaining the optimal model architecture (Hossin & Sulaiman, 2015). The performance of the proposed model will be evaluated against the baseline buy-and-hold method. The single-period rate of return of the proposed method and the buy and hold method will be compared to determine the profitability of the trained model. Additionally, the model's accuracy will be determined to assess the correctness of the model's trading decision. Finally, the model will be tested on unseen data to determine the system's performance and ensure fairness against the buy-and-hold method.

RESULTS AND DISCUSSIONS

Training Process

The training process will take in the pre-processed training dataset containing two market cycles; as mentioned, the model will interact with input states and react by acting on the predicted action. Therefore, a successful training process for this experiment will decrease model loss and increase net worth. Normally, the model loss will be high initially as the actions predicted are inconsistent with the target output but should decrease in time as the model learns to predict actions based on input states. As a result, the model is expected to yield an increase in net worth from generating accurate trading signals.

The model loss, net worth, and order per episode are recorded to visualise the training process better. The training process took approximately 100 hours to complete 4000 training episodes. Each episode is made up of 1000-time steps. The training parameters for model training in each episode are five epochs with a batch size of 32. In other words, the model will go over each episode five times, where the model will update its weight parameters every 32 batches. As observed in Figure 8, the model can learn as the loss per replay decreases.

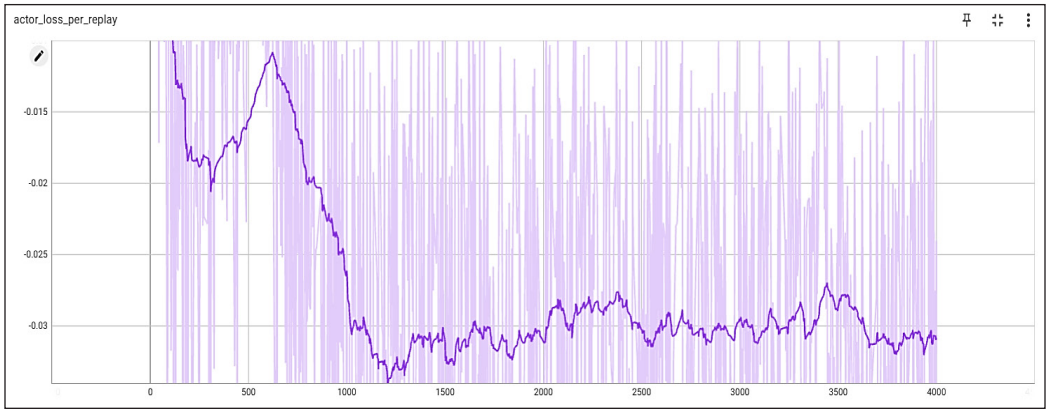

Figure 8. Actor loss per replay

As a result, the model has proven its ability to perform profitable trades as its net worth increases as more training episodes continue, as shown in Figure 9. Notably, the model also learned to perform more trading orders per episode, as illustrated in Figure 10, which is believed to reflect how it can achieve more profits.

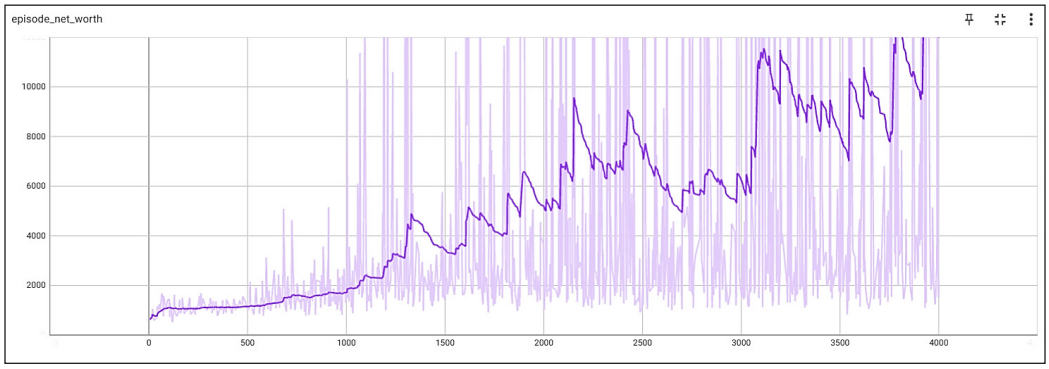

Figure 9. Net worth per episode

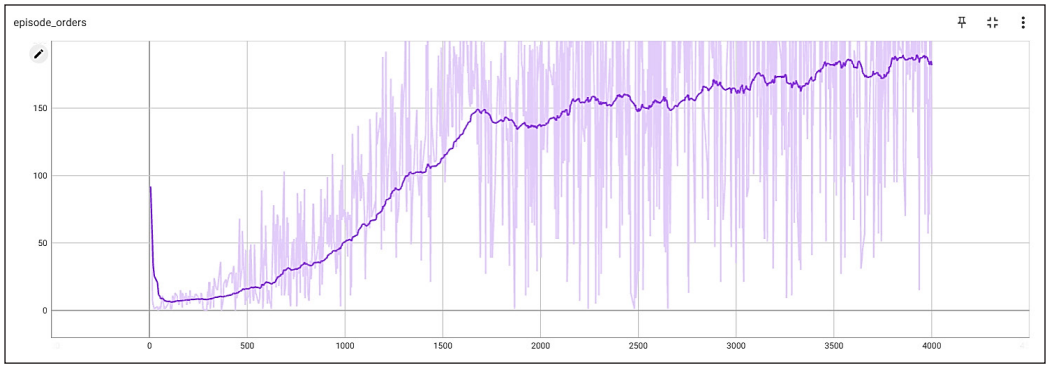

Figure 10. Orders made per episode

Evaluation and Findings

Model Accuracy. The trained model is tested with an unseen dataset with 436 days, ranging from 10 May 2020 to 20 July 2021 for all the cryptocurrencies, except for DOT, as it was only launched on 18 August 2020. Table 3 shows the model's performance, including the number of trades, number of profitable trades, accuracy, total return rate given the time, and annualised return. The model has proved its capability to make profitable trades.

The model's accuracy is evaluated by the number of profitable trades made over the total number of trades made. A trade is considered profitable if the current selling price of the trade is higher than the previous buying price. As the model is initially trained using BTC historical dataset, the accuracy for BTC is the highest among the 10 cryptocurrencies. Nonetheless, the model can perform well with all the cryptocurrencies with high accuracy. Although, this may be due to the nature of cryptocurrencies generally trending upwards.

Table 3
Model performance on the testing dataset

	ADA	BCH	BNB	BTC	DOT
Trades	850	652	1100	1184	313
Profitable trades	763	569	1012	1145	263
Accuracy	89.76%	87.27%	92.00%	96.71%	84.03%
Total return rate	49835618642.99%	15199675.56%	38527152540.10%	169543702.22%	647835.54%
Annualised return	2246218700.00%	2397972.87%	1807081748.09%	18416004.14%	1570920.32%

	ETH	LINK	LTC	XLM	XRP
Trades	824	632	592	895	1152
Profitable trades	741	536	514	804	1096
Accuracy	89.93%	84.81%	86.82%	89.83%	95.14%
Total return rate	421040489.53%	410081019.47%	26941402.38%	4071603763.66%	1494947924270.84%
Annualised return	39728250.52%	2397972.87%	1807081748.09%	18416004.14%	1570920.32%

Table 4
Baseline buy-and-hold method performance on the testing dataset

	ADA	BCH	BNB	BTC	DOT
Total return rate	2280.73%	78.02%	1723.91%	258.64%	253.65%
Annualised return	64632.43%	3777.43%	51117.58%	10395.64%	41160.71%

	ETH	LINK	LTC	XLM	XRP
Total return rate	886.64%	286.06%	170.68%	248.98%	182.74%
Annualised return	29269.75%	11316.05%	7323.75%	10067.89%	7758.00%

The performance of the baseline buys and hold method is shown in Table 4. As compared to the baseline buy and hold method on BTCUSDT, a trader would have acquired a total return rate of 258.64%. Therefore, the model has significantly outshone the baseline method and outperformed previous researchers' models. This evaluation result also substantiates humans' significant limitations of not always being able to trade and missing out on trading opportunities without an ATS.

Table 5 shows the historical annualised return of BTC from 2017 to 2021. Despite having an exceptionally high return rate, the trained model can outmatch the performance of the baseline buy and hold method of all historical annualised returns of BTC with an annualised return of 10395.64% on the testing dataset.

Table 5
Historical annualised return of BTC with baseline buy and hold method

Year	Annualised Return
2017	1331%
2018	-73%
2019	95%
2020	301%
2021	58%

Model Performance on Market Downtrend

To evaluate the performance of the model in a market downtrend, the model is tested on the date ranging from 9 May 2021 to 19 July 2021 as seen in Figure 11. The evaluation results are depicted in Table 6. Despite the falling prices, the model can trade with outstanding results. The accuracy for trading BTC is once again the highest, with 88.79%. It showcased the model's adaptability in different market conditions.

Figure 11. Downtrend in 2021 market cycle

Table 6
Model performance on a market downtrend

	ADA	BCH	BNB	BTC	DOT
Trades	83	66	86	107	41
Profitable trades	58	50	57	95	20
Accuracy	69.88%	75.76%	66.28%	88.79%	48.78%
Total return rate	147.14%	329.53%	324.93%	316.64%	7.30%
Annualised return	13937.88%	287727.96%	271293.66%	243593.93%	46.95%

	ETH	LINK	LTC	XLM	XRP
Trades	90	64	76	77	67
Profitable trades	76	44	44	60	34
Accuracy	84.44%	68.75%	57.89%	77.92%	50.75%
Total return rate	410.91%	266.72%	67.73%	142.45%	-37.47%
Annualised return	742776.57%	121225.78%	1588.02%	12543.58%	-92.31%

However, the results of trading DOTUSDT and XRPUSDT were not as good as others. It may be due to the model's inability to adapt to the different market characteristics compared to BTCUSDT. As a result, its low accuracy causes it to predict an incorrect action, getting caught in a market down spike. Table 7 shows the total return rate of the baseline buy and hold method; all the crypto pairings result in negative results. Comparably, implementing the baseline buy and hold method for XRPUSDT would have lost -63.04%, while the model was able to cut loss to -37.47%. In conclusion, the proposed model is appropriately evaluated and accepted as effective.

Table 7
Baseline buy-and-hold method performance on a market downtrend

	ADA	BCH	BNB	BTC	DOT
Total return rate	-26.87%	-68.91%	-53..17%	-46.00%	-68.78%

	ETH	LINK	LTC	XLM	XRP
Total return rate	-50.85%	-68.13%	-65.86%	-62.93%	-63.04%

CONCLUSION AND RECOMMENDATIONS

This research proposes a new automated cryptocurrency trading system integrated with DRL. This research has perfectly demonstrated its capability to resolve the stated problem statement – overcoming human hindrance and bridging the gap between human and automated trading. The goal of presenting the concept and use of cryptocurrencies to all audiences and utilising the power of machine learning to earn profits has been accomplished. With that said, the experimental analysis of the model manifested exceptional results, even

surpassing similar works by other researchers, thus again fortifying the implementation of DRL in the finance sector.

Undeniably, there are underlying limitations in this research. Different training parameters have not been explored since the time constraint arises from hardware constraints. In the future, this work hopes to explore different architecture to induce a comparison, thus reaching a consensus on the best-performing architecture. Furthermore, the input data features can be further improved, and different technical indicators can be tested. Correlation analysis such as the Pearson correlation coefficient can be further conducted to drop correlated features to achieve optimal noise reduction (Benesty et al., 2009). To effectively address each cryptocurrency's market behaviour, models should also be trained on each cryptocurrency's historical data to attain even greater accuracy and adaptability.

ACKNOWLEDGEMENT

This research is supported through the KDU Penang University College Internal Research Grant.

REFERENCES

Anthony, M., Bartlett, P. L., & Bartlett, P. L. (1999). *Neural network learning: Theoretical foundations*. Cambridge University Press.

Azulay, A., & Weiss, Y. (2018). Why do deep convolutional networks generalize so poorly to small image transformations? *Journal of Machine Learning, 20*(184), 1-25. https://doi.org/10.48550/arXiv.1805.12177

Benesty, J., Chen, J., Huang, Y., & Cohen, I. (2009). On the importance of the Pearson correlation coefficient in noise reduction. *IEEE Transactions on Audio, Speech, and Language Processing, 16*(4), 757-765. https://doi.org/10.1109/tasl.2008.919072

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford University Press.

Cartwright, T. J. (1991). Planning and chaos theory. *Journal of the American Planning Association, 57*(1), 44-56. https://doi.org/10.1080/01944369108975471

Chollet, F. (2017). *Deep learning with Python*. Simon and Schuster.

Cowpertwait, P. S. P., & Metcalfe, A. V. (2009) *Time series data*. In *Introductory time series with R* (pp. 1-25). Springer. https://doi.org/10.1007/978-0-387-88698-5_1

Dempster, M. A. H., & Romahi, Y. S. (2002). Intraday FX trading: An evolutionary reinforcement learning approach. In H. Yin, N. Allinson, R. Freeman, J. Keane & S. Hubbard (Eds.), *Intelligent Data Engineering and Automated Learning - IDEAL 2002* (pp. 347-358). Springer. https://doi.org/10.1007/3-540-45675-9_52

Dempster, M. A., & Leemans, V. (2006). An automated FX trading system using adaptive reinforcement learning. *Expert Systems with Applications, 30*(3), 543-552. https://doi.org/10.1016/j.eswa.2005.10.012

Fang, F., Ventre, C., Basios, M., Kong, H., Kanthan, L., Li, L., Martinez-Regoband, D., & Wu, F. (2022). Cryptocurrency trading: A comprehensive survey. *Financial Innovation, 8*(13). https://doi.org/10.1186/s40854-021-00321-6

Ganesh, S., Vadori, N., Xu, M., Zheng, H., Reddy, P., & Veloso, M. (2019). *Reinforcement learning for market making in a multi-agent dealer market*. arXiv Preprint. https://doi.org/10.48550/arXiv.1911.05892

Graves, A. (2012). Long short-term memory. In *Supervised sequence labelling with recurrent neural networks* (pp. 37-45). Springer. https://doi.org/10.1007/978-3-642-24797-2_4

Grondman, I., Busoniu, L., Lopes, G. A., & Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 42*(6), 1291-1307. https://doi.org/10.1109/tsmcc.2012.2218595

Gronwald, M. (2014). *The economics of bitcoins - Market characteristics and price jumps*. (Working Paper No. 5121). https://www.cesifo.org/DocDL/cesifo1_wp5121.pdf

Haferkorn, M., & Diaz, J. M. Q. (2014). Seasonality and interconnectivity within cryptocurrencies - An analysis on the basis of bitcoin, litecoin and namecoin. In A. Lugmayr (Ed). *International Workshop on Enterprise Applications and Services in the Finance Industry* (pp. 106-120). Springer. https://doi.org/10.1007/978-3-319-28151-3_8

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hossin, M., & Sulaiman, M. N. (2015). A review on evaluation metrics for data classification evaluations. *International Journal of Data Mining & Knowledge Management Process, 5*(2). https://doi.org/10.5121/ijdkp.2015.5201

Huang, B., Huan, Y., Xu, L. D., Zheng, L., & Zou, Z. (2019). Automated trading systems statistical and machine learning methods and hardware implementation: A survey. *Enterprise Information Systems, 13*(1), 132-144. https://doi.org/10.1080/17517575.2018.1493145

Huang, C. Y. (2018). *Financial trading as a game: A deep reinforcement learning approach*. arXiv Preprint. https://doi.org/10.48550/arXiv.1807.02787

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts.

Jeong, G., & Kim, H. Y. (2019). Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications, 117*, 125-138. https://doi.org/10.1016/j.eswa.2018.09.036

Kolm, P. N., & Ritter, G. (2019). Modern perspectives on reinforcement learning in finance. *Journal of Machine Learning in Finance, 1*(1).

Konda, V. R., & Tsitsiklis, J. N. (1999). Actor-critic algorithms. In S. Solla, T. Leen & K. Müller (Eds.), *NIPS'99: Proceedings of the 12th International Conference on Neural Information Processing Systems* (pp. 1008-1014). MIT Press.

Kotsiantis, S. B., Kanellopoulos, D., & Pintelas, P. E. (2006). Data preprocessing for supervised leaning. *International Journal of Computer and Information Engineering, 1*(12), 4104-4109. https://doi.org/10.5281/zenodo.1082415

Li, Y. (2017). *Deep reinforcement learning: An overview*. arXiv Preprint. https://doi.org/10.48550/arXiv.1701.07274

Liao, S., Wang, J., Yu, R., Sato, K., & Cheng, Z. (2017). CNN for situations understanding based on sentiment analysis of twitter data. *Procedia Computer Science, 111*, 376-381. https://doi.org/10.1016/j.procs.2017.06.037

Liu, X. Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., & Wang, C. (2020). *FinRL: A deep reinforcement learning library for automated stock trading in quantitative finance.* arXiv Preprint. https://doi.org/10.48550/arXiv.2011.09607

Livieris, I. E., Pintelas, E., & Pintelas, P. (2020). A CNN-LSTM model for gold price time-series forecasting. *Neural Computing and Applications, 32*, 17351-17360. https://doi.org/10.1007/s00521-020-04867-x

Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020). A CNN-LSTM-based model to forecast stock prices. *Artificial Intelligence for Smart System Simulation, 2020*, Article 6622927 https://doi.org/10.1155/2020/6622927

Lucarelli, G., & Borrotti, M. (2019). A deep reinforcement learning approach for automated cryptocurrency trading. In J. MacIntyre, I. Maglogiannis, L. Iliadis & E. Pimenidis (Eds.), *Artificial Intelligence Applications and Innovations* (pp. 247-258). Springer. https://doi.org/10.1007/978-3-030-19823-7_20

Moody, J., Saffell, M., Andrew, W. L., Abu-Mostafa, Y. S., LeBaraon, B., & Weigend, A. S. (1999). Minimizing downside risk via stochastic dynamic programming. *Computational Finance*, 403-415.

Neely, C. J., Rapach, D. E., Tu, J., & Zhou, G. (2014). Forecasting the equity risk premium: The role of technical indicators. *Management Science, 60*(7), 1772-1791. https://doi.org/10.1287/mnsc.2013.1838

Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345-1359. https://doi.org/10.1109/tkde.2009.191

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65*(6), 386-408. https://doi.org/10.1037/h0042519

Sattarov, O., Muminov, A., Lee, C. W., Kang, H. K., Oh, R., Ahn, J., Oh, H. J., & Jeon, H. S. (2020). Recommending cryptocurrency trading points with deep reinforcement learning approach. *Applied Sciences, 10*(4), Article 1506. https://doi.org/10.3390/app10041506

Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2015). *High-dimensional continuous control using generalized advantage estimation.* arXiv Preprint. https://doi.org/10.48550/arXiv.1506.02438

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). *Proximal policy optimization algorithms*. arXiv Preprint. https://doi.org/10.48550/arXiv.1707.06347

Shrestha, A., & Mahmood, A. (2019). Review of deep learning algorithms and architectures. *IEEE Access, 7*, 53040-53065. https://doi.org/10.1109/access.2019.2912200

Sola, J., & Sevilla, J. (1997). Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE Transactions on Nuclear Science, 44*(3), 1464-1468. https://doi.org/10.1109/23.589532

Tucnik, P. (2010). Optimization of automated trading system's interaction with market environment. In P. Forbrig & H. Günther (Eds.), *Perspectives in Business Informatics Research* (pp. 55-61). Springer. https://doi.org/10.1007/978-3-642-16101-8_5

Van Otterlo, M., & Wiering, M. (2012). Reinforcement learning and Markov decision processes. In M. Wiering & M. Van Otterlo (Eds.), *Reinforcement Learning. Adaptation, Learning, and Optimization* (pp. 3-42). Springer. https://doi.org/10.1007/978-3-642-27645-3_1

Vrigazova, B. (2021). The proportion for splitting data into training and test set for the bootstrap in classification problems. *Business Systems Research Journal, 12*(1) 228-242. https://doi.org/10.2478/bsrj-2021-0015

Wu, C. H., Lu, C. C., Ma, Y. F., & Lu, R. S. (2018). A new forecasting framework for bitcoin price with LSTM. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)* (pp. 168-175). IEEE Publishing. https://doi.org/10.1109/icdmw.2018.00032

Xiong, Z., Liu, X. Y., Zhong, S., Yang, H., & Walid, A. (2018). *Practical deep reinforcement learning approach for stock trading*. arXiv Preprint. https://doi.org/10.48550/arXiv.1811.07522

Yang, H., Liu, X. Y., Zhong, S., & Walid, A. (2020, October 15-16). Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the First ACM International Conference on AI in Finance* (pp. 1-8). ACM Publishing. https://doi.org/10.1145/3383455.3422540

Zhang, W., Yang, Z., Shen, J., Liu, M., Huang, Y., Zhang, X., Tang, R., & Li, Z. (2021). Learning to build high-fidelity and robust environment models. In N. Oliver, F. Pérez-Cruz, S. Kramer, J. Read & J. A. Lozano (Eds.), *Machine Learning and Knowledge Discovery in Databases* (pp. 104-121). Springer. https://doi:10.1007/978-3-030-86486-6_7

Zhang, Z., Zhang, Y., & Li, Z. (2018). *Removing the feature correlation effect of multiplicative noise*. arXiv Preprint. https://doi.org/10.48550/arXiv.1809.07023