Expert Systems With Applications 202 (2022) 117259

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Outperforming algorithmic trading reinforcement learning systems: A supervised approach to the cryptocurrency market

Leonardo Kanashiro Felizardo [a,*], Francisco Caio Lima Paiva [a], Catharine de Vita Graves [a], Elia Yathie Matsumoto [b], Anna Helena Reali Costa [a], Emilio Del-Moral-Hernandez [a], Paolo Brandimarte [c]

[a] Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil
[b] Escola de Economia de São Paulo, Fundação Getulio Vargas, São Paulo, Brazil
[c] Department of Mathematical Sciences, Politecnico di Torino, Turin, Italy

ARTICLE INFO

Keywords:
Deep neural network
Reinforcement learning
Stock trading
Time series classification
Cryptocurrencies

ABSTRACT

The interdisciplinary relationship between machine learning and financial markets has long been a theme of great interest among both research communities. Recently, reinforcement learning and deep learning methods gained prominence in the active asset trading task, aiming to achieve outstanding performances compared with classical benchmarks, such as the Buy and Hold strategy. This paper explores both the supervised learning and reinforcement learning approaches applied to active asset trading, drawing attention to the benefits of both approaches. This work extends the comparison between the supervised approach and reinforcement learning by using state-of-the-art strategies with both techniques. We propose adopting the ResNet architecture, one of the best deep learning approaches for time series classification, into the ResNet-LSTM actor (RSLSTM-A). We compare RSLSTM-A against classical and recent reinforcement learning techniques, such as recurrent reinforcement learning, deep Q-network, and advantage actor–critic. We simulated a currency exchange market environment with the price time series of the Bitcoin, Litecoin, Ethereum, Monero, Nxt, and Dash cryptocurrencies to run our tests. We show that our approach achieves better overall performance, confirming that supervised learning can outperform reinforcement learning for trading. We also present a graphic representation of the features extracted from the ResNet neural network to identify which type of characteristics each residual block generates.

1. Introduction

Individual traders and hedge funds try to profit and reduce risk as their primary goals in financial markets. For this, they use various strategies to exploit the patterns and anomalies in the market. An active trader continuously observes the market to trade at the best time, improving performance metrics related to profit and risk. It is a usual assumption that there is no point in trying to extract profit from the financial markets since the agents are in an efficient market (Fama, 1965), meaning that all information is equally available to every agent precluding market exploitation. Furthermore, considering the efficient market hypothesis, any trading strategy that attempts to exploit patterns in time series of asset prices would fail. However, behavioral finance (Kapoor & Prosad, 2017) suggests that market participants may not be wholly rational or fully informed. Consequently, decision-makers may have behavioral biases, generating time series anomalies in asset prices. Analyzing only the time series of prices, technical analysts try to employ several methods to profit from the observation of the dynamics of the asset prices (Brock et al., 1992). It is possible to automate these methods with an algorithmic trading system that can trade without human intervention. The algorithmic trading system captures the market information to optimize the trading decisions automatically.

Lately, algorithmic trading systems have incorporated new techniques that employ recent machine learning developments. Among the wide range of machine learning approaches, some address the sequential decision problem using dynamic programming, and other techniques. Over the past few years, many approaches, such as reinforcement learning (RL), have become quite popular to solve the active single asset trading problem (Aboussalah & Lee, 2020; Almahdi & Yang,

* Corresponding author.
E-mail addresses: leonardo.felizardo@usp.br (L.K. Felizardo), francisco.paiva@usp.br (F.C. Lima Paiva), catharine.graves@usp.br (C. de Vita Graves), elia.matsumoto@fgv.br (E.Y. Matsumoto), anna.reali@usp.br (A.H.R. Costa), emilio.delmoral@usp.br (E. Del-Moral-Hernandez), paolo.brandimarte@polito.it (P. Brandimarte).

https://doi.org/10.1016/j.eswa.2022.117259
Received 31 October 2021; Received in revised form 12 April 2022; Accepted 13 April 2022
Available online 19 April 2022

List of Acronyms and Variables

s	State
a	Action
a^*	Best action given by the future return(s)
A	Action space
t	Time step
p	Price
ρ	Return
ρ^a	Agent return
ρ_f	Risk-free daily rate
μ	Position size
o	Observation
O_h	List of observed states
r	Reward
π	Policy
$\mathcal{F}$	Residual map
$\mathcal{H}$	Original underlying map
x	Input variable
M	Past window size (in hours)
M^f	Future window size
w	Model parameters or neural networks weights
w_f	Model parameters after training
Y	Labels
H_a	Executed actions stored
R_h	Received rewards stored
L	Episode length
$prob$	Estimated probability of best action
η	controls the influence of the asset price return on the Sharpe ratio
DS_t	Differential Sharpe ratio
$ReLU$	Rectified Linear Unit
$B\&H$	Buy and Hold
RL	Reinforcement learning
MDP	Markov decision process
$LSTM$	Long short-term memory
BTS	Bootstrapped Thompson sampling
$A2C$	Advantage actor–critic
$A3C$	Asynchronous advantage actor–critic
DQN	Deep Q-network
RRL	Recurrent reinforcement learning
$DDPG$	Deep deterministic policy gradient
$RSLSTM-A$	Estimated ResNet-LSTM actor
SR	Sharpe ratio
AR	Annualized returns
ACR	Area under the cumulative agent asset price return curve
API	Application programming interface
D	Number of trading days
BTC	Bitcoin
ETH	Ethereum
LTC	Litecoin
$DASH$	Dash
NXT	Nxt
XMR	Monero

2019; Deng et al., 2017; Feuerriegel & Prendinger, 2016; Li et al., 2019). Conventional RL methods assume that the autonomous agent's external environment is affected by the agent's actions and, therefore, entails a complex set of tools to deal with this interaction. However, in the active trading problem, the financial market is not affected by the trades of individual investors or smaller hedge funds. Hence, the market state does not transition based on a single-agent interaction. Considering this market characteristic in the problem formulation, we can focus on a different set of solutions that can directly learn which action yields more immediate or cumulative reward given a past price time series, such as supervised learning. Deng et al. (2017) previously compared RL approaches against supervised learning approaches achieving better results with RL under transaction costs. Nevertheless, since the publication of this comparison (Deng et al., 2017), there have been advances in time series classification that facilitate the elaboration of classifiers that are better at dealing with noisy time series. RL is vulnerable in noisy environments such as in financial environments, generating misled trading rules that are a result of small perturbations in the observations (Deng et al., 2017; Henderson et al., 2018; Wang et al., 2020). Furthermore, RL may also present signs of overfitting in the training set when applied to the trading task as mentioned by Dempster and Romahi (2002). In this sense, supervised learning provides a robust approach that does not require any additional procedure to account for the influence of the action on the states, simplifies the model, and allows easier control of convergence and overfitting issues, presenting a promising direction for research efforts.

This paper proposes the actor ResNet-LSTM (RSLSTM-A) as novel architecture to address the asset trading task via a supervised learning approach. RSLSTM-A uses a ResNet architecture given its outstanding performance against other deep learning models for time series classification tasks (Ismail Fawaz et al., 2019). Moreover, RSLSTM-A efficiency comes from the ResNet capacity to extract features from time series, capturing nonlinear behaviors of the financial market. By adopting state-of-the-art supervised learning for time series classification, which can better capture features in the time series, we can now revisit the comparison made by Deng et al. (2017) and evaluate the supervised learning approach against an RL agent. Our approach focuses on extracting features from the time series of prices and giving us the best action instead of learning complex temporal relations between state and actions that suffer from convergence under very noisy time series. We hypothesize that treating the trading active asset decision problem as a classification problem and using modern deep learning architectures may be advantageous to generate profitable trading decisions.

We simulate a currency exchange environment using real-world cryptocurrency data to verify our proposed method's performance. These markets are not likely to be efficient, and active trading may work in this condition. We show that our method outperformed other methods in most currency assets and outperformed the Buy and Hold (B&H) strategy in all tests. Also, we adapted and included the bootstrapped Thompson sampling (BTS) (Eckles & Kaptein, 2014) in the comparison, which is an RL technique that does not consider the action influence on the environment as another performance baseline for our model. Using BTS and RSLSTM-A, we present some evidence that the gain in the decision process is not related to complex RL techniques, which may not be required to solve this type of decision problem. Finally, we provide a visualization of the feature maps generated by our model to assist with the interpretability and feature extraction of the artificial neural networks. We summarize the contributions of this article in the following topics:

- We **proposed the RSLSTM-A**, a customized supervised learning approach for the trading task, capable of surpassing other approaches and the B&H strategy. Our architecture adopted the ResNet model of the latest generation time series classifier to identify nonlinear behaviors replacing linear classifiers usually employed in similar approaches. Besides, we modified the ResNet architecture for a one-dimensional time series by adding recurrence for better performance.

- We present evidence that the **selected supervised learning technique provides stable and competitive performance** compared to the most recent RL approaches in the literature. To overcome problems related to transaction costs, we define a window for trading frequency and use cumulative future rewards as labels.
- We provide some **insightful interpretations** of the ResNet learning process for a time series sequential decision process through visual representations of the feature extraction by depicting a comparison for the original time series and the channel outputs. The visualization of the channel's output may provide an insight into the importance of volatility quantification and asset price return historical distribution as a feature for trading decisions.

We divided this article into the following sections. Section 2 provides a brief overview of supervised learning and RL applied to the active trading and portfolio optimization problem. Section 3 defines the active trading problem. Section 4 presents our proposed solution to the problem. The 5th section shows the parameters and results of the experiments and the methods applied to compare the performance of the algorithms. Finally, Section 6 concludes the article and suggests paths for future research.

2. Related work

Technical analysis is the term used for trading techniques that try to discover hidden relations in asset price changes. Identifying trading rules has long been an attractive problem that dates from the late 1800s with techniques attributed to Charles Dow (see Hamilton (1922) for reference), whereas the efficient market hypothesis (Fama, 1965) supports the concept that financial markets follow a random walk. It is thus impossible to predict the asset prices' future directions. Interestingly, different studies (Brock et al., 1992; Chopra et al., 1992; Fama & French, 1988) contested this efficient market hypothesis, with the main argument that noticeable market inefficiencies can occur. For instance, Brock et al. (1992) explores some popular trading rules based on moving average and trading range break, obtaining returns not compatible with the random walk assumption. Additionally, Bessembinder and Chan (1995) reinforce the ideas of Brock et al. (1992) by employing a set of popular trading rules (e.g., trading range break and fixed and variable-length moving average) that point to opposite conclusions to the efficient market hypothesis.

We can use approaches other than the popular mathematical tools such as moving averages and others to find trading rules. Machine learning is usually employed to forecast the asset price, which is essential to establish trading rules, but it could also be employed to find the trading rules directly. Trippi and DeSieno (1992) proposed a neural network system that could act assuming long or short positions based on the asset market price outperforming the random trading. In works such as Allen and Karjalainen (1999), another type of machine learning approach extracts the trading rules without exogenously specifying them as in Brock et al. (1992). However, using the genetic algorithm to make trading policies did not outperform the B&H strategy after transaction cost using out-of-sample data with daily frequency. As a next step, Kuo et al. (2001) proposes a genetic algorithm-based fuzzy neural network. Kuo et al. (2001) proposed method employs a genetic algorithm for initial weights selection and fuzzy logic to capture qualitative features of the stock market for the buy and sell decisions. Nowadays, different papers have used supervised learning with artificial neural networks to trade in the financial market. Nakano et al. (2018) use fixed trading rules based on the prediction made by an artificial neural network. Sang and Di Pierro (2019) also explore the supervised learning approach by using the long short-term memory (LSTM) architecture to select the technical analysis strategy given a market state.

In parallel, other authors developed reinforcement learning methods to solve the active asset trading problem. Neuneier (1995) made the first attempt to apply RL (Q-learning technique) to the portfolio optimization problem, optimally allocating assets at each step of the time. Later on, Moody and Wu (1997) used direct recurring reinforcements to approximate an optimal asset allocation policy. These two methods applied by these authors can be classified as critic (Neuneier, 1995, 1998) and actor (Moody, Saffell et al., 1998; Moody & Wu, 1997; Moody, Wu et al., 1998), respectively. The critic method uses the values of each state to approximate an optimal policy indirectly. On the other hand, actors use only the objective function to approach the optimal policy directly. As we observe, subsequent works improved the techniques by merging new approaches and methods with the concepts presented by the previous works. Maringer and Ramtohul (2010) further developed the recurrent reinforcement learning (RRL) (Moody & Wu, 1997) approach with a regime-switching mechanism to better adapt to the multiple regimes that a price time series would assume. Shen et al. (2015) uses the multi-armed bandit approach balancing the portfolio using a linear policy approximation without feature extraction. Recently, many other approaches to the actor (Aboussalah & Lee, 2020; Almahdi & Yang, 2019; Wang et al., 2019) and critic (Jeong & Kim, 2019; Park et al., 2020; Zarkias et al., 2019) frameworks were published. Lately, authors have explored the union of actor and critic frameworks in actor–critic algorithms. Just a few papers, such as the ones by Kang et al. (2018), Li et al. (2019), Lima Paiva et al. (2021), Ponomarev et al. (2019), Ye et al. (2020) and Yu et al. (2019), used the actor–critic approach. The predominant methods used in the actor–critic works approaches are asynchronous advantage actor–critic (A3C) and deep deterministic policy gradient (DDPG).

Which approach is the best for active asset trading is an open question. Pendharkar and Cusatis (2018) classified the problem of active trading in portfolio optimization as a multi-armed bandit problem since the common assumption is that trader transactions do not impact market prices. If we assume this premise, it is possible to establish a simple rule to estimate the next return, making the problem a supervised learning one. Reinforcement learning becomes relevant in the trading problem by introducing transaction costs. However, just a few articles (Almahdi & Yang, 2017; Deng et al., 2017) extensively compared the reinforcement and supervised approaches with different transaction costs. It is difficult to affirm that the reinforcement learning algorithms perform better even without transaction costs. Therefore, we compare the recent reinforcement learning approaches and the state-of-the-art deep learning architecture for time series classification and forecasting.

In the following sections, we present the problem definition to clarify the assumptions adopted in our solution.

3. Problem definition

Our work addresses the active single asset trading problem. To profit, traders actively trade an individual asset over time to take advantage of any market anomaly. They can observe information about the market such as financial reports, news, asset price time series, and financial indexes to decide the best timing to trade an asset. Using information about the market, the trader creates a belief over the next asset price and decides to buy the asset or borrow an asset and sell it. These two actions may increase or decrease the trader's wealth depending on the actual price movement. When a trader buys an asset, the trader then owns that asset, assuming a long position. Contrarily to the long position, expecting that the asset price will decrease, traders borrow an asset and sell it, meaning they are now in debt, assuming a short position. To assume a short position, traders need to borrow the asset from a financial institution, such as a brokerage firm and sell at the asset price moment. At some point, traders will have to return the borrowed asset. In turn, the traders have a short position over an asset and the price decreases. They can buy that same asset for a lower price and return it to the financial institution they borrowed from, obtaining profit. The difference between the sale and purchase prices

is the trader's profit or loss. To summarize, traders observe the market information at each time step, decide which position (long or short) to assume, and observe the price changes affecting their wealth.

We assume that traders can make optimal trading decisions using only the history of past asset prices. This assumption is commonly made by technical analysts, especially for high-frequency trading, whereby the traders have to make decisions every hour or minute (Kirkpatrick II & Dahlquist, 2010). Instead of directly using the past asset price time series, we calculate and use a series of past asset price changes $\rho_{t-1}, \ldots, \rho_1$ at each time step t:

$$\rho_t = p_t - p_{t-1}, \tag{1}$$

where p_t is the price at the time step t of a past time series of prices $p_{t-1}, \ldots, p_1, p_0$. There is a high correlation between p_t and p_{t-1}, which can generate a multicollinearity problem that reduces the statistical significance of the independent variable (Allen, 1997). Using the asset price changes instead of the asset price gives the time series desirable statistical properties (Campbell et al., 1997), such as stationarity. The stationarity property makes the time series less prone to the multicollinearity problem.

Ideally, the agent can use the entire asset price return history to observe the market state. However, we assume that a window with a fixed size of the past price time series can express the market state without losing relevant information. The asset price return time series compose the state space in a predefined window of M hours. In this sense, the trader will observe only the M past asset price changes $\rho_{t-1}, \ldots, \rho_{t-M}$ to determine the following action. The fixed window size assumes that recent prices can provide sufficient information about future asset prices for high-frequency trading. Furthermore, defining a constant window size M is a common approach in other studies (Aboussalah & Lee, 2020; Almahdi & Yang, 2019; Deng et al., 2017).

Now that we have defined the market information adopted, we must delimit which actions the trader can select and how these actions affect the trader's wealth. We adopt the premise that the trader can only assume a long or a short position at each time step with a position size equal to μ. As a simplification of the problem, we limit the size of μ to a constant value for each time step, avoiding problems related to the asset price changes magnitude (Moody, Wu et al., 1998). By fixing the size of μ, we also have stationary asset price changes with the statistical benefits cited before (Campbell et al., 1997). The fixation of μ is essential only during the model training, but we can relax this assumption during a real-world operation. Depending on the trader's position (long or short), the trader will receive a positive or a negative return. The traders who own the asset will have a positive asset price return as the price increases. Conversely, traders that borrowed the asset and sold it for an initial price will have a negative return if the price increases, as they will need to repurchase the asset to return it. Therefore, we can model long and short positions as changes in the subsequent asset price return signal. If the trader assumed a long position, the trader's asset price return would have the same sign as the asset price return, $\rho_t^a = \rho_t$, but if the trader assumed a short position, the sign would be the opposite, $\rho_t^a = -\rho_t$. Since we can change a part of the μ available assets to a long or a short position, we can interpret the trader actions at each time step as a spectrum in the range $[-1, 1]$. In Fig. 1, we summarize our model of the interaction between the trader and the market.

Another important aspect to consider is the market response to the trader's position (long or short). The standard assumption is that trades do not significantly affect market supply and demand. This premise is valid for a small trade when considering the market trade volumes (number and size of trades). Most works in the literature usually assume this premise (Aboussalah & Lee, 2020; Deng et al., 2017; Moody, Saffell et al., 1998).

We can summarize our interpretation of the problem as a decision problem in which the trader has to decide what the best action to perform would be, given the observations made regarding the market.

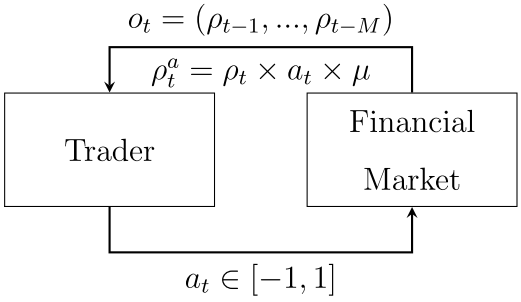

Fig. 1. Model of the trader interaction with the financial market.

The trader, here called agent, observes the past time series of asset price changes – the states – and decides to take long or short positions, interpreted as an agent's action a_t. The market supply and demand will change the asset prices, and then it is possible to determine the agent's asset price return ρ_t^a for the chosen action. This process repeats every time step. Our goal is to automatize this process as efficiently as possible to maximize the total agent return.

4. Solving the problem

Usually, a typical approach employed in the literature to solve this trading problem is to model with the mathematical framework of the Markov decision process (MDP) and solve it with reinforcement learning (RL) techniques. RL is concerned with autonomous agents learning successful actuation policies by experimenting in their environment in a trial-and-error approach. The agent observes the state of the environment and then chooses an action to perform. This action changes the state of the environment, and the agent receives a reward, which is a numerical value that can be positive or negative (a penalization). The goal of the RL agent is to learn an actuation policy that chooses actions that maximize the reward accumulated over time by the agent. RL agents receive indirect, delayed, and occasional rewards. The reward function indicates the desirability of the resulting situation or state.

However, MDP assumes that the observed state encompasses all the information necessary to choose the best action. In the active single asset trading problem, we cannot fully observe the market state, and we only observe some information about the market. Furthermore, MDP assumes a transition probability to the next state that solely depends on the agent's action and the last observed state (Feinberg and Shwartz, 2002). As we have discussed before, as a common assumption, the trader's actions do not influence the market, meaning that the trader's action will not influence the following states. In addition to these characteristics, it is worth mentioning that one of the main challenges of RL is the trade-off between exploration and exploitation. An agent wants to maximize its accumulated reward for each action it takes by exploiting the experience stored in memory (represented in a tabular way or using an approximation function). However, this decreasing exploration of new actions improves knowledge about getting the highest reward possible. Thus, the agent must balance the trade-off between exploration and exploitation to aim at behaving optimally, maximizing the expected accumulated reward.

In the problem we are addressing, optimal actions can be defined *a posteriori* by looking at past asset prices, i.e., by looking at the sequence of past prices, we can decide which the best possible action would have been at each moment. Thus, we can label the series of past prices with the actions that would have been the best possible at each moment and adopt a supervised learning approach to train the agent to perform optimal actions (instead of an RL approach). In addition, the supervised learning approach does not suffer from problems such as the dilemma between exploration and exploitation.

This article proposes, develops, and analyzes a supervised learning model based on time series classification to decide which action to

take in each observed state. As in the many usual financial trading approaches, we use a supervised learning algorithm to classify each state into different groups: the group in which the agent can benefit from a long position and the group in which the agent can benefit from a short position. The correct action is the action that gives our autonomous agent the maximum agent asset price return.

The input to our model is composed of the past price changes window of size M, $(\rho_{t-1}, \cdot, \rho_{t-M})$, and the output is the estimated best action. To generate the labels for the classification task, we consider which action would provide the maximum financial return when reaching the subsequent time step. In this sense, the most profitable immediate action is the one that determines the label used for the supervised learning algorithm training phase. Moreover, we apply this same procedure to train, validate, and test our approach.

Our contributions are two-fold. First, we show that using a supervised approach like ours can lead to similar or even better results for the active trading problem than the RL approach. Second, we present and modify the ResNet architecture as our time series classifier to solve the decision problems. We depict features extracted by ResNet in a graphical representation of the outputs of the convolutional layers and discuss how this may help our approach outperform the experiments' RL counterparts. Since we approach the problem using a supervised method, ResNet architecture perfectly fits as it is one of the best time series classifiers (Ismail Fawaz et al., 2019). We name our approach ResNet LSTM actor, or RSLSTM-A.

4.1. ResNet classifier: RSLSTM-A

We use the ResNet (He et al., 2016) architecture to select among discrete actions, $\{short, long\}$ with the encoding of $\{-1, 1\}$ using the context (state) expressed by the past asset price return window. ResNet can fit nonlinear functions and does not rely on any assumption about the time series underlying function. As stated by Ismail Fawaz et al. (2019), ResNet had the best overall results in time series univariate and multivariate classification problems, among many other deep learning approaches.

The ResNet architecture was first proposed for image classification problems, addressing the problem of degradation of the training accuracy caused by the vanishing or exploding gradient (He et al., 2016), a typical problem with deep learning architectures. ResNet fits the residual mapping, $\mathcal{F}(x)$, using a block of stacked convolutional layer, which proved to have some advantages over the typical approach of directly learning the original underlying mapping, $\mathcal{H}(x)$. The underlying mapping, $\mathcal{H}(x)$, is the unknown function that, in our case, gives the best actions for each state that encodes input x, which is the market state. Then, the residual mapping is defined as $\mathcal{F}(x) = \mathcal{H}(x) - x$. The ResNet architecture assumes that, by learning the difference between the input $\mathcal{H}(x)$ and the output x of the block of stacked layers, we can reduce the variation of learned features. The direct use of the input x with the output is called a shortcut connection. Fig. 2 depicts the combined features of residual approximation and shortcut connection. We use the Rectified Linear Unit (ReLU) activation function in each convolutional block. $\text{ReLU}(x) = max(0, x)$ is the most popular activation function because it learns faster than sigmoid functions.

To solve the decision problem of trading, we employ the ResNet to approximate policy function $\pi(o_t)$ by indirectly approximating the underlying function $\mathcal{H}(x)$. The reinforcement learning agent has predefined actions $(a \in A)$ that we define according to the task. In our work, we define that, depending on the algorithm, the agent can use a discrete set of actions $a = \{-1, 1\}$ or a continuous set of actions $a = [-1, 1]$. A policy function (π) provides the action $a \in [-1, 1]$ given an observation $o_t = (\rho_{t-1}, \ldots, \rho_{t-M})$. Consider the equivalence between input and observation $x_t \equiv o_t$ while underlying function $\mathcal{H}(x_t)$ is equivalent to policy function $(\mathcal{H}(x_t) \equiv \pi(o_t))$, so that $\mathcal{H}(x_t) = \pi(\rho_{t-1}, \ldots, \rho_{t-M})$. Then, the convolutional layers of the ResNet model approximate the underlying function $\mathcal{H}(x)$ by the residual function $\mathcal{F}(x)$. We try to maximize the next agent's reward (or the agent's return, expressed by ρ^a_{t+1}) by reducing the loss between action a provided by the ResNet approximated policy π and the best action a^* (the one that maximizes the immediate return) given by the signal of the following asset price return ρ_{t+1}.

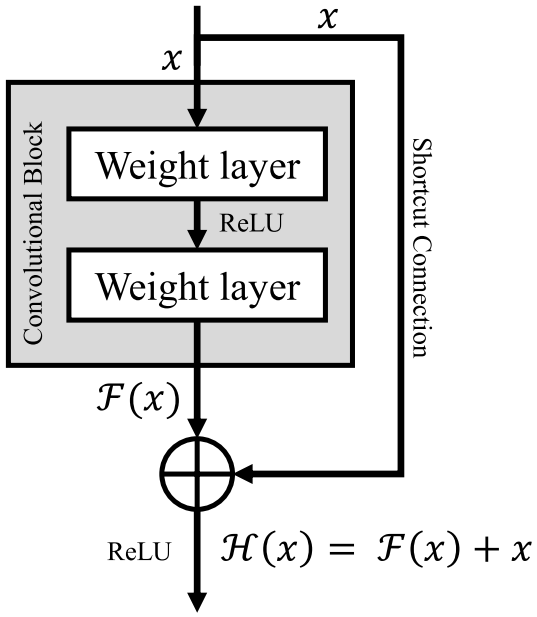

Fig. 2. Graphical representation of the approximation of underlying function $(\mathcal{H}(x))$ using the residual approximation $(\mathcal{F}(x))$ and a shortcut connection of x.
Source: Adapted from He et al. (2016).

A supervised learning algorithm uses labeled data to update the model parameters (i.e., artificial neural network internal weights). The label, as described before, is the best action given a state represented by $\{\rho_{t-1}, \ldots, \rho_{t-M}\}$. We update the artificial neural network weights using a loss function calculated as the difference between the model's output and the actual labels (i.e., the best action). For the backpropagation training algorithm, we applied the Adam optimizer with a learning rate of 0.01 that adapts over time. We use a batch of size 64 to correct our model parameters training the algorithm through 150 epochs. In the output layer, we have a softmax layer that generates a vector of probabilities of success (success meaning the positive agent asset price return), each probability associated with an action. Later, we describe the training procedure with Algorithm 1.

Then, once our model learns the underlying function $\mathcal{H}(x)$, we reach the second phase: online execution. In the online execution phase, the model receives the observations $o_t = (\rho_{t-1}, \ldots, \rho_{t-M})$ at instant t, and decides which action a_t to select; at instant $t + 1$, observes the $o_{t+1} = (\rho_t, \rho_{t-1}, \ldots, \rho_{t+1-M})$ and decides action a_{t+1}, and so on. We describe the online execution procedure with Algorithm 2. Ultimately, the online execution phase validates if the trained model generalizes well in unseen observations. In essence, online execution helps verify if the mappings from observations to actions learned during training are still profitable during testing.

Moreover, under the presence of transaction costs, we test different trading frequencies training the agent based on the cumulative rewards of the future window of size M^f and not only on the next ρ^a. Reducing the trading frequency and basing the best action on the cumulative future reward of a predefined window may reduce transaction costs, increasing the overall performance. We fixed the trading to happen every M^f time step. The future window and the trading frequency do not have to match, but we make them equal to simplify.

Fig. 3 also depicts the RSLSTM-A architecture with our modifications over the original implementation of Ismail Fawaz et al. (2019).

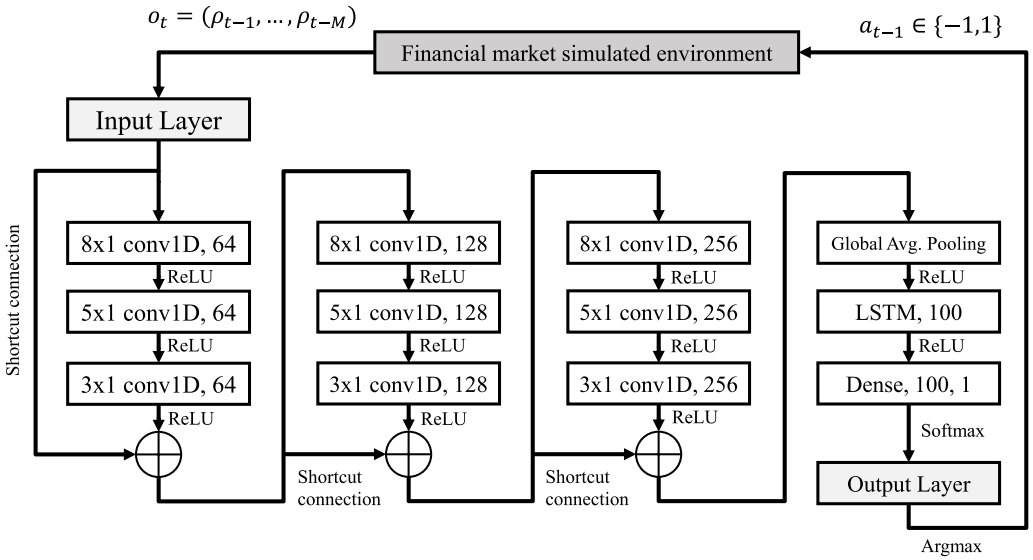

Fig. 3. RSLSTM-A architecture and financial market online execution (evaluation) using actions (a_{t-1}) and receiving informational state observed (o_t) of the environment.

We maintained the one-dimensional convolutions as in the original implementation for time series classification. After each convolution, we have a batch normalization layer, the ReLU activation function, and a max-pooling layer. We have three sets of convolutions, batch normalization, and max pooling at each block. We increase the number of channels at each block to increase the number of extracted features of the time series (64, 128, and 256). We added an LSTM layer, removing the average global pooling layer used in the original implementation so that the model can account for the representation of the previous actions and observations. We chose to adopt LSTM as previous research (Choi et al., 2018; He et al., 2019) has shown evidence that it can retain extended temporal sequence information and thus improve performance. This improvement relates to the fact that even though we defined a fixed size of M past asset price changes for observation, the LSTM constructs maintain the memory of previous information that is not bound to just the last M features.

In Fig. 3, we represent our model input and output in the decision process problem context.

As we previously described, we train our proposed method in a data set and measure our system performance in an online execution on an out-of-sample (test) data set. The following sections describe our proposed method for algorithm training and testing (execution).

4.2. RSLSTM-A model training

Here we detail the training and testing procedures with the pseudo-code for RSLSTM-A divided into training in the Algorithm 1 and testing in the Algorithm 2. RSLSTM-A generates the probability vector employed to decide the best action in the training algorithm at each time step. Based on the agent's outputs, we update the weights of the neural network by calculating the loss with binary cross-entropy (He et al., 2016) between the best action probability vector and the RSLSTM-A probability vector. Then, the *backpropagation* procedure updates the model's internal weights, ultimately mapping which actions are best suited for each situation (i.e., defining the actuation policy).

We divide the data set into a training, validation, and test sets. The training set is employed to update the model parameters. After each episode, model validation occurs in the validation set and stores the updated parameters (weights of the neural network) with the best performance so far. The function *eval_on_validation_set* evaluates the total loss attained in the validation set by the trained RSLSTM-A. Finally, we use the best parameters obtained through RSLSTM-A

Algorithm 1: Training RSLSTM-A algorithm

input : episode list of observed states O_h
output: Trained parameters: w_f
initialization: model parameters: w_0, $max\_episodes$, set of actions: $A = \{-1, 1\}$, storage of actions: $H_a = []$, storage of best actions: $Y = []$, get best action function: f;
$validation\_loss_0 \longleftarrow eval\_on\_validation\_set(w_0)$;
foreach *episode in range(0,max_episodes)* **do**
  foreach o_t *of the list of observed states* O_h **do**
    $prob_t \longleftarrow model(o_t)$;
    $a_t \longleftarrow A[argmax(prob_t)]$;
    $a^* \longleftarrow f(o_{t+1}, o_t)$;
    $Y \longleftarrow concatenate(Y, a^*)$;
    $H_a \longleftarrow concatenate(H_a, a_t)$;
  $w_t \longleftarrow backpropagate(w_{t-1}, loss(H_a, Y))$;
  $validation\_loss_t \longleftarrow eval\_on\_validation\_set(w_t)$;
  if $validation\_loss_t < validation\_loss_{t-1}$ **then**
    $w_f \longleftarrow w_t$

interactions with the environment, in the training phase, for evaluation in the test data set. In the evaluation and training phase, we define the *environment* class that simulates the market. The *environment* class sets the next observation and immediate reward.

4.3. RSLSTM-A model online execution

After training the RSLSTM-A, we measure the performance of our proposed method in the out-of-sample data set, in which we did not fit our model. This approach is standard in supervised learning methods, and it is part of the evaluation process to verify the generalization capabilities, ensuring that we did not overfit our model to the in-sample (training) data. We present the pseudo-code that describes our model interaction with the simulated market environment. In this phase, we measure the accumulated profit obtained by our model. Then, using the model actions and considering the asset price changes, we can calculate the profits and the other financial metrics used to evaluate our model. We further describe our metrics in the experiment description section.

Algorithm 2: Online execution and Evaluation of the RSLSTM-A algorithm

input : Model parameters w_f
output: Evaluation metrics
initialization: storage of rewards: R_h=[], episode length: L;
$o_0 \longleftarrow environment()$;
foreach *step t in range(0,L)* **do**
  $a_t \longleftarrow model(w_f, o_t)$;
  $\langle o_{t+1}, r_t \rangle \longleftarrow environment(a_t)$;
  $R_h \longleftarrow concatenate(R_h, r_t)$;
  $o_t \longleftarrow o_{t+1}$
$\langle SR, AR, ACR \rangle \longleftarrow calculate\_evaluation\_metrics(R_h)$

Table 1
RSLSTM-A hyperparameters list.

Hyperparameter	Value
Loss function	Binary cross entropy
Batch size	64
Number of epochs	150
Optimizer	Adam
Learning rate	0.01
Minimum learning rate	0.001

5. Experimental analysis

We executed the experiments comparing the mainstream RL algorithms, the bootstrapped Thompson sampling, and the RSLSTM-A algorithm for the active trading problem. Machine learning community is continually developing other RL techniques, but here we focus on the algorithms already tested in the trading problem. Recent applications of RL solutions to the trading problem (i.e., portfolio management and single asset trading) (Aboussalah & Lee, 2020; Almahdi & Yang, 2019; Li et al., 2019; Park et al., 2020; Ponomarev et al., 2019; Zarkias et al., 2019) have predominantly employed the recurrent reinforcement learning (RRL) (Moody & Wu, 1997), the deep Q-network (DQN) (Mnih et al., 2013), and the asynchronous advantage actor–critic (A3C) (Baird, 1993; Mnih et al., 2016) (here we apply non-asynchronous version, A2C) algorithms. We used a core i7-7700HQ 2.80 GHz, a GTX 1060 (6 GB), and 16 GB of RAM. All codes are in Python language, and the main libraries used are Keras, Tensorflow, Pytorch, and Scikit-learn. All the codes are available in the Github repository (https://github.com/leokan92/Contextual-bandit-Resnet-trading)

5.1. Hyperparameters

In addition, to choose the hyperparameters of all the methods and RSLSTM-A, we made an empirical evaluation over various configurations that we noticed to most influence the overall results. Finally, we employ the hyperparameters here presented. For instance, we reduced the learning rate from 0.01, changing every five training epochs if there were no improvements in the loss until it reached 0.001. Thus, the model started with a higher learning rate for faster updates of the neural network weights. Then, a subsequent reduction to the learning rate avoiding fluctuation around a local or global minimum. In our verification, the batch size did not cause relevant performance differences. Therefore, we selected a batch size of 64 maintaining consistency with previous work (Ismail Fawaz et al., 2019). Noting that, after around 50 epochs, the validation and training loss do not change, so we limited the training epochs to 150. In Table 1, we present the hyperparameters adopted for the RSLSTM-A model that complement the architecture shown in Fig. 3.

5.2. Action and state experimental configuration

We employed discrete action spaces to increase performance and evaluate value-based approaches, such as DQN. For policy gradient-based methods such as RRL, we employ continuous actions spaces. The agent acts to change its position affecting the next agent asset price return (r_t^a):

$$r_t^a = \mu \sum_{t=1}^{T} \left\{ \rho_t^f + a_{t-1}(\rho_t - \rho_t^f) - \sigma |a_t - a_{t-1}| \right\}, \tag{2}$$

where σ is the transaction cost, ρ_t is the asset price return at the time step t, μ the position size (number of an asset in the portfolio), and ρ^f is the risk-free asset (such as T-Bills). For better comparison, we propose to use only the series of asset price changes. We assume that the observations over the asset price return time series represent the market state, although we know the actual market state is more complex than the price changes time series. To capture daily seasonality we employed in the experiments the past window size M of 50 h.

5.3. Reward function experimental configuration

Notwithstanding that the RL rewards are easy to define in finance, we can select various reward functions in a market simulated environment. Nevertheless, the agent's immediate asset price return ρ^a is the most commonly adopted reward r (Lima Paiva et al., 2021; Moody & Wu, 1997). Unlike a supervised approach, the RL framework can maximize the expected accumulated rewards (Deng et al., 2017; Li et al., 2019). One technique to balance risk and asset price returns is to consider these both metrics in the reward function. The most employed risk-adjusted reward function is the Sharpe ratio (SR).

The RRL algorithm (Moody & Wu, 1997) adopts the differential Sharpe ratio as a reward function. The differential Sharpe ratio (DS_t) is a marginal utility that computes the reward r_t at time t. The differential Sharpe ratio is exponentially weighted moving averages and the price changes' standard deviation calculated over the past states. The differential Sharpe ratio considers the expansion to first order in the decay rate of η used in the approximation:

$$SR_t|_{\eta>0} \approx SR_t|_{\eta=0} + \eta \frac{\partial SR_t}{\partial \eta} + O(\eta^2), \tag{3}$$

being $DS_t = \frac{\partial SR_t}{\partial \eta}$ the partial derivative of the Sharpe ratio, the differential Sharpe ratio. Note η is the term that controls the influence of the asset price return on the Sharpe ratio (SR).

The Sharpe ratio is calculated by:

$$SR_t = \frac{\text{Average } (\rho_t, \ldots, \rho_0)}{\text{Standard Deviation } (\rho_t, \ldots, \rho_0)}. \tag{4}$$

5.4. Market data

We evaluated the models using cryptocurrency due to the data availability when analyzing the hourly frequency and the previously cited market inefficiency. In cryptocurrencies markets, trades can occur over the full day (24 h). We selected the following six assets to perform experimental tests: Bitcoin (BTC), Ethereum (ETH), Litecoin (LTC), Dash (DASH), Nxt (NXT), and Monero (XMR) cryptocurrencies. Additionally, our market simulation adopts an intraday hourly frequency. The lack of standardized benchmark high-frequency public data sets of cryptocurrencies and many other assets has long been a problem in trading research (Khadjeh Nassirtoussi et al., 2014; Pineau et al., 2021). Fortunately, it is possible to gather the data required to reproduce the results by directly downloading it using the parameters here provided from any number of web portals or API. However, this procedure may require some effort to find a reliable source that can provide the data, which is the usual approach in financial studies and is the procedure we took. All the data was extracted from the Poloniex broker API. Each

data set of each asset contain approximately 29 600 data points starting from September 01, 2017, and finishing on November 20, 2020

Our data set is divided into training, validation, and test sets in the proportions of 0.8/0.1/0.1, respectively. We opted to take the traditional approach employed by other articles in RL research applied to trading and used a validation set to calibrate the RL and supervised learning models hyperparameters, and further selecting the calibrations that yielded the best performance. To build the training, validation, and test data set for the RSLSTM-A, we consider for each input the past window of price changes, $(\rho_{t-1}, \dots, \rho_{t-M})$. We set the target as minus one if the next return, ρ_t, is negative and one of the following returns is positive. In the case of transactions with costs, we use the label -1 for the negative sum of M^f future price changes and $+1$ for the positive sum of M^f future price changes. In the case of the RL methods, the environment simulator used the price time series to produce, for each time step of the decision process, the observation, $o_t = \rho_{t-1}, \dots, \rho_{t-M}$, and the reward r_t that depends on the reward function employed.

5.5. Performance metrics

First, we define two of the most used performance metrics: Annualized returns (AR) and Sharpe ratio (SR). The annualized asset price return is calculated by:

$$AR = (E(\rho_t, \dots, \rho_0) + 1)^D - 1, \tag{5}$$

where $E(\rho_t, \dots, \rho_0)$ is the expected daily asset price return rate of the tested method. We calculate the daily asset price return powered by the number of trading days (D).

Another way to calculate the Sharpe ratio, differently from Eq. (4), is to consider the risk-free ratio to first calculate the expected annualized mean excess return, $E((\rho_t, \dots, \rho_0) - \rho^f)$, and divide it by the standard deviation, $std(\rho_t, \dots, \rho_0)$, of the daily return:

$$SR = \frac{E((\rho_t, \dots, \rho_0) - \rho^f)}{std(\rho_t, \dots, \rho_0)}, \tag{6}$$

where ρ_f is the risk-free daily rate that we define as 0.01%, which is a common adopted value in the asset trading works. Using this SR different calculation, we avoid any assumption related to the risk-free ratio helping in the comparability of the results. We also include a different metric to better compare the algorithms performance, the total area under the cumulative asset price return curve that is calculated by adding the daily returns as expressed by:

$$ACR = \sum_{i=0}^{T} cumulative\ sum(\rho_i)_i. \tag{7}$$

We attempted to provide an average asset price return over time with the area under the cumulative asset price return curve. One way to calculate the area under the curve is to execute a sum of the discrete values. The cumulative sum returns the T accumulated values of the T ρ values. The area under the cumulative asset price return curve (ACR) gives traders a more accurate perception of the profit they would make if they decided to liquidate the assets and stop trading at any point in time.

5.6. Experimental results

Table 2 compiles the sum of the returns obtained for each algorithm and each asset for the case of absence of transaction costs. Zero transaction costs is not very realistic but is a conservative starting point to verify any potential to outperform the B&H benchmark. We included the transaction cost in a second test and reduced the algorithm trading frequency, accounting for the cumulative reward (agent asset return) in a future window.

Note that RSLSTM-A had an overall better performance when considering all the metrics (ACR, SR, and AR) for four assets (BTC, DASH, LTC, and NXT). Moreover, RSLSTM-A achieved the second-best performance regarding the ACR metric for the remaining assets (ETH and XMR). Recall from the previous Section 5.5 that the ACR metric shows that for an arbitrary stop point, on average, the RSLSTM-A would give a higher agent asset price return independently of the stopping point. Thus, from the results of the experiments, we can infer that for an investor that desires to disinvest at any given moment, RSLSTM-A would be the best technique to employ. SR considers the risk as a factor that shows the trading system's capacity to be stable under high volatility situations. The results show that RSLSTM-A is also the best algorithm to avoid risk while maximizing profit, being adequate for conservative risk investors. Finally, AR is the annualized agent asset price return which is the mainstream metric for comparison, on which RSLSTM-A had the best performance.

Interestingly, the bootstrapped Thompson sampling (BTS) baseline displayed a competitive performance, achieving better results than the mainstream RL methods for several evaluated assets. The BTS performance could also improve by using better approximation functions, and thus it is a promising technique for tackling the exploration and exploitation dilemma using artificial neural networks for online learning. We observe in the results depicted in Fig. 4 that our proposed method, RSLSTM-A (dotted green line), can be a better approach to choose the best action in the trading environment. Reflecting the table results, Fig. 4 shows that RSLSTM-A profit equals or exceeds the B&H benchmark and outperforms the other RL algorithms in most cases. Although we achieved higher annualized returns and overall cumulative returns, the stability of the returns was not maintained. Considering the SR metric, the RSLSTM-A was better in four out of six assets, being the best model to balance risk and agent asset price return.

When considering SR as a success metric, the advantage of RSLSTM-A over other techniques is smaller, but it remains the technique with the highest SR for most assets. Note that RSLSTM-A is better than the B&H benchmark for all assets. The B&H benchmark has proved to be an excellent strategy considering that B&H has often outperformed RL techniques. Also, the B&H strategy cannot be affected by the transaction cost and is supported by the efficient market hypothesis. However, in the case of assets such as Bitcoin and Ethereum, depicted in Fig. 4, the great majority of the algorithms outperformed the B&H benchmark. Interestingly, some of the algorithm behavior follows the B&H strategy, represented by the continuous red line, which could be considered a conservative behavior since it at least exhibits similar performance to the benchmark. We believe that one of the main reasons for the better results of the RSLSTM-A is the excellent performance of ResNet in time series classification tasks. This better performance of RSLSTM-A raises the question regarding the necessity of the RL method to solve the active single asset trading problem. In a problem whereby the agent actions do not affect the state, the focus should be on the time series classification and minimizing transaction costs. From the work of Ismail Fawaz et al. (2019), we can already observe that ResNet outperforms the MLP in the classification task. Therefore, we would expect our method to outperform the others considering only the classification capability.

Another remarkable aspect of the RSLSTM-A is the smoothness in the loss reduction observed during training and validation, which facilitates overfitting control, likely favoring its generalization capacity and better performance. RL methods working with very noisy data are unstable during the training phase (Henderson et al., 2018; Lima Paiva et al., 2021). Consequently, it is harder to control overfitting, while supervised learning can use many techniques to have a smooth convergence with overfitting control using validation sets, giving our method a generalization advantage. Finally, regarding the LSTM, as previously pointed out, it plays an essential role in the proposed architecture by improving the time series classification performance when it may also depend on previous behaviors. Here, the LSTM possibly helped ResNet deal with the inconsistency between different observed periods of the time series. LSTM, as also observed by Choi et al. (2018),

Table 2
Consolidated results for the accumulated agent asset price returns (ACR), Sharpe ratio (SR), and annualized agent asset price return (AR) for all the models employed and assets without transaction costs. The best results are the ones in bold.

	Asset	B&H	RRL	DQN	A2C	BTS	RSLSTM-A
ACR	BTC	−5.31e+06	−4.90e+06	5.74e+06	−4.07e+05	5.93e+06	**1.14e+07**
	DASH	−4.09e+04	−9.60e+05	−5.11e+03	−3.20e+05	2.95e+05	**5.10e+05**
	ETH	−4.44e+04	2.25e+05	**9.22e+04**	2.24e+05	−4.37e+05	4.63e+04
	LTC	−5.04e+03	−5.85e+04	−7.14e+04	−5.31e+04	8.26e+03	**1.77e+05**
	NXT	4.70e+03	3.52e+03	−5.35e+03	−4.45e+03	−6.16e+03	**1.07e+04**
	XMR	−2.57e+04	−1.89e+05	**2.07e+05**	2.13e+04	−5.19e+04	−5.92e+02
SR	BTC	−2.49e−06	2.32e−05	1.54e−04	1.09e−04	1.23e−04	**1.46e−04**
	DASH	4.55e−04	−3.04e−03	8.84e−04	2.09e−04	1.22e−03	**1.26e−03**
	ETH	7.15e−04	1.09e−03	8.61e−04	**1.33e−03**	−4.26e−04	7.90e−04
	LTC	2.66e−03	−2.43e−03	−1.03e−03	1.06e−03	5.47e−03	**7.86e−03**
	NXT	2.35e−01	2.30e−01	−2.53e−01	−2.25e−01	−2.54e−01	**3.00e−01**
	XMR	1.06e−03	−3.68e−04	**3.99e−03**	3.29e−03	1.46e−03	1.65e−03
AR	BTC	3.51e−02	5.47e−02	1.51e−01	9.98e−02	1.44e−01	**1.65e−01**
	DASH	7.35e−02	−1.64e−01	1.08e−01	5.08e−02	1.17e−01	**1.41e−01**
	ETH	9.15e−02	1.17e−01	9.88e−02	**1.25e−01**	1.42e−02	9.73e−02
	LTC	8.60e−02	−2.06e−03	1.96e−02	5.19e−02	1.17e−01	**1.88e−01**
	NXT	3.91e−01	3.63e−01	−2.42e+00	−2.37e+00	−2.38e+00	**5.07e−01**
	XMR	6.97e−02	2.68e−02	**1.63e−01**	1.14e−01	7.30e−02	8.81e−02

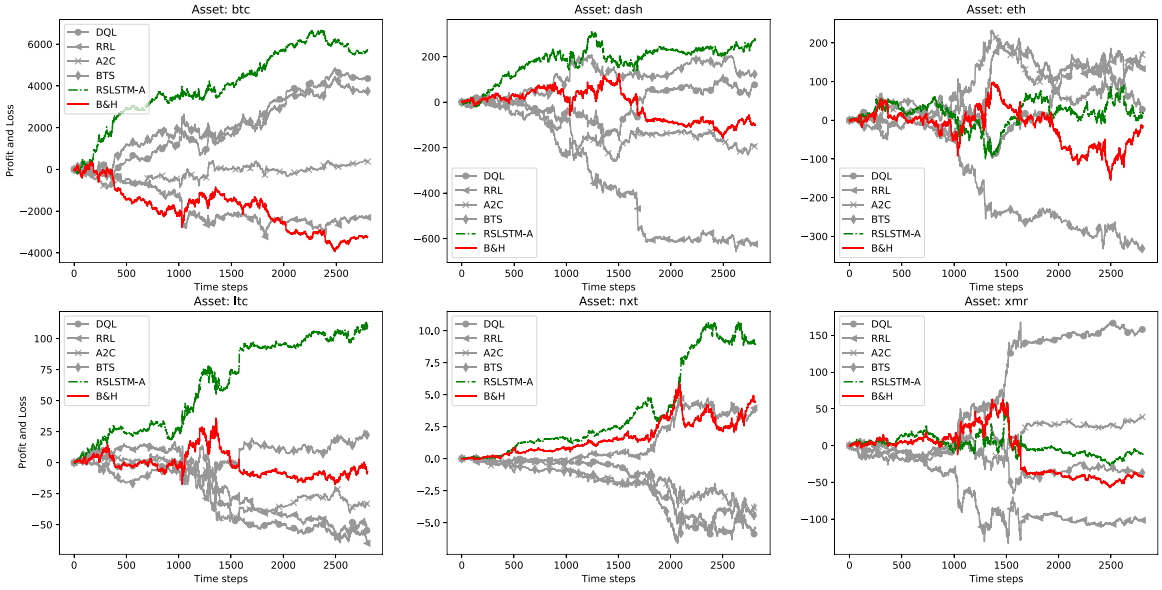

Fig. 4. Cumulative agent asset price return for the test set of all assets (BTC, DASH, ETH, LTC, NXT, XMR) for transaction cost equals to zero.

holds the properties to learn data dynamics better and is an excellent combination for the ResNet.

Convolutional neural networks can be considered a pre-processing step to extract features that help the agent choose the best action. The extracted features are used by the LSTM network that retains the memory of previous actions and states to extract new features. In the end, a linear layer generates the probability of success for each action given a state. Our analysis shows some of the convolution channels outputs in the RSLSTM-A architecture – the first and last convolutional layers – by looking at the extracted features from the original time series. Fig. 5 depicts the transformation that the first convolutional layer made to the features. We extracted two samples of the features generated by the convolutional layers. In the last convolutional layer, depicted in Fig. 5, we observe a more accentuated difference of behavior when compared to the original input. The features extracted by the last layers seem to be a trend approximation or even a volatility estimation. From the sample in Fig. 5, we can see that convolutional neural networks generate forecasting in some channels. In the extracted sample from the 128 channels, we can also observe that channels 7 and 16 appear to show an intensified representation of the price changes trend. Notably, more than one channel generates similar behavior. Also, considering the improved performance observed in the experiments, we might assume that these convolutional mechanics can enhance the most relevant signals as features for decision-making. Here, we presented just one possible interpretation of the phenomenon through a graphical analysis of channels' outputs. Nonetheless, this analysis of outputs is a promising method to understand the network's learning process and give insights to interpret the features extracted from convolutional neural networks.

5.7. The effect of transaction costs

One main line of argument to use RL techniques is that we may have to account for future rewards, not only the immediate rewards, under the presence of transaction costs. We modified the training to consider the cumulative future value of the price changes as the right action to deal with the possible excessive transaction costs. Also, we limit the frequency of trades to reduce the impact of transaction costs. We adopted a future window of size $M^f = 80$ to calculate the future rewards used in the training of our RSLSTM-A model. Using this simple approach, RSLSTM-A was also able to outperform the other models in most of the assets and still surpass the B&H benchmark in most of the

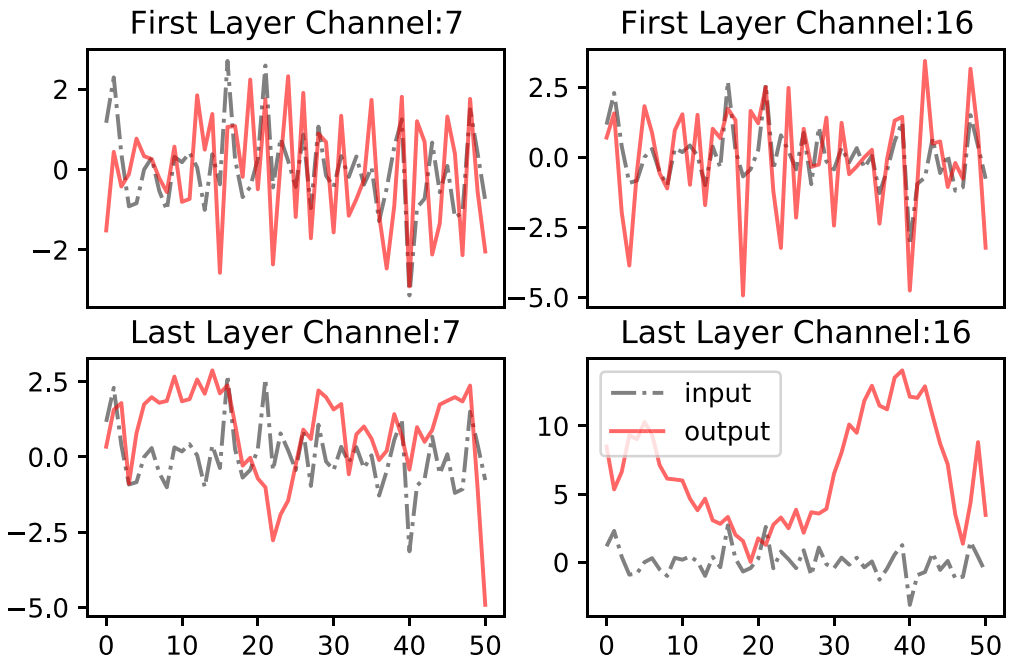

Fig. 5. Two samples of the first and the last convolutional layer outputs. The x-axis ranging from zero to 50 is the input size of the network (state dimension, or the look-back window time series), and the y-axis displays the agent asset price return values for each time step.

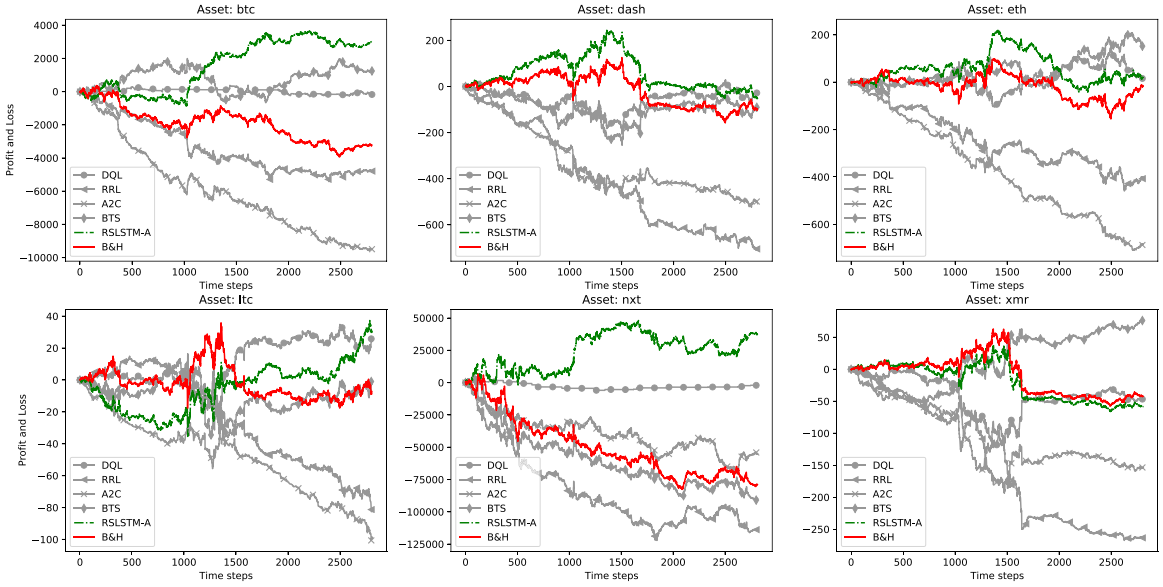

Fig. 6. Cumulative agent asset price return for the test set of all assets (BTC, DASH, ETH, LTC, NXT, XMR) for transaction cost equals to 0.001.

cases. Fig. 6 and Table 3 allow observing that, for some assets, the RSLSTM-A followed the asset price but stayed above it. In cases such as the NXT asset, RSLSTM-A could outperform the other techniques and the B&H by far.

In the presence of transaction costs, surprisingly, the DQN was a competitive algorithm against the RSLSTM-A, as observed in Table 3. It is essential to highlight that without the presence of transaction costs, DQN had a worse performance. Considering the SR metric, DQN surpass RSLSTM-A, but that metric was over-estimated because DQN tends to choose a neutral position in most cases. We also depict in Fig. 7 how the RSLSTM-A performance changes as we change the future window (M^f), and the frequency of trades. As observed in Fig. 7, the best performances are in the range from 40 to 100. This result can strongly indicate the importance of the restriction of trading frequency and the consideration of cumulative future price changes for the trading decision.

6. Conclusions and insights for future works

This paper explored the supervised learning approach to solve the decision problem of the financial asset trading task. We compared our proposed method with RL algorithms that achieved state-of-the-art in trading. Consequently, this comparison shows pieces of evidence that supervised learning can match and surpass the RL method performance.

Table 3
Consolidated results for the accumulated agent asset returns (ACR), Sharpe ratio (SR), and annualized agent asset price return (AR) for all the models employed and assets with transaction costs equals to 0.001. The best results are the ones in bold.

	Asset	B&H	RRL	DQN	A2C	BSTS	RSLSTM-A
ACR	btc	−5.31e+06	−9.40e+06	6.89e+04	−1.66e+07	2.18e+06	**4.44e+06**
	dash	−4.09e+04	−1.14e+06	−1.59e+05	−9.19e+05	−2.68e+05	**1.71e+05**
	*e*th	−4.44e+04	−6.68e+05	3.14e+04	−1.04e+06	1.47e+05	**1.64e+05**
	ltc	−5.04e+03	−6.77e+04	**2.82e+04**	−1.35e+05	−2.91e+04	−1.39e+04
	nxt	−1.41e+08	−2.35e+08	−8.31e+06	−1.11e+08	−1.65e+08	**6.77e+07**
	xmr	−2.57e+04	−4.10e+05	−1.15e+05	−2.96e+05	**4.02e+04**	−5.41e+04
SR	btc	−2.49e−06	−5.72e−05	**2.21e−04**	−2.09e−04	8.92e−05	1.14e−04
	dash	4.55e−04	−7.23e−03	**6.56e−04**	−1.80e−03	4.96e−04	5.89e−04
	*e*th	7.15e−04	−7.13e−04	7.77e−04	−5.25e−03	**1.09e−03**	7.90e−04
	ltc	2.66e−03	−5.20e−03	4.43e−03	−1.25e−02	3.08e−03	**4.79e−03**
	nxt	−2.16e−05	−1.16e−05	**3.69e−05**	−6.16e−06	−9.60e−06	6.31e−06
	xmr	1.06e−03	−5.19e−03	**9.61e−04**	−2.32e−03	3.29e−03	7.23e−04
AR	btc	3.51e−02	−4.48e−03	9.15e−02	−2.20e+00	1.12e−01	**1.36e−01**
	dash	7.35e−02	−3.73e−01	**8.88e−02**	−6.79e−02	7.48e−02	8.44e−02
	*e*th	9.15e−02	−1.03e−02	9.63e−02	−2.06e−01	1.20e−01	**9.73e−02**
	ltc	8.60e−02	−4.61e−02	1.19e−01	−1.02e−01	9.12e−02	**1.26e−01**
	nxt	−2.67e−01	−2.22e+00	9.04e−02	−5.11e−02	−2.20e+00	**1.48e−01**
	xmr	**6.97e−02**	−2.20e+00	6.64e−02	−2.38e−02	1.31e−01	5.92e−02

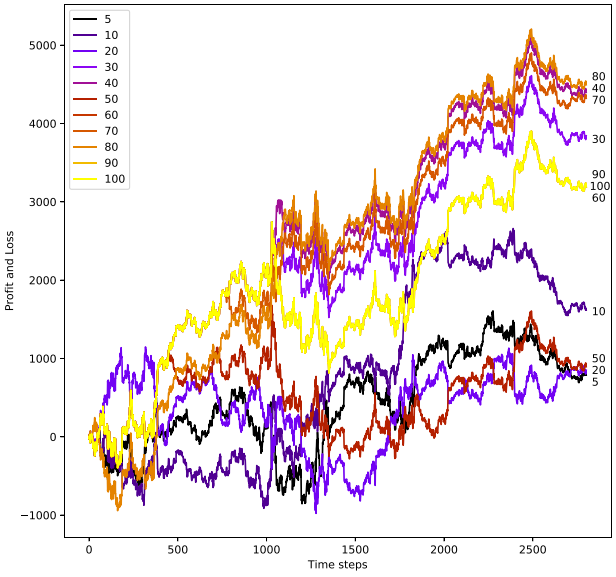

Fig. 7. Testing the ResNet performance for the BTC asset trading considering different future windows sizes from 5 to 100 for cumulative price changes and trading frequency.

We chose cryptocurrency time series to perform the tests and show the experimental results based on real financial data. We proposed the ResNet LSTM actor (RSLSTM-A), a supervised learning approach to solve the MDP for trading financial assets. RSLSTM-A obtained the best performance among the techniques employed with and without transaction costs, outperforming B&H in both scenarios. State-of-the-art time series classifiers seem to be an interesting method for trading. Also, with our proposed method, we present a graphical interpretation of the extracted characteristics using a convolutional neural network, the features of which relate to local volatility and the price changes distribution, being important to solve the financial asset trading MDP.

We list some future research paths to be explored and improved in future studies. First, to statistically differentiate the results of the algorithms, it is necessary to use the Friedman test and the Nemenyi critical differences as suggested by Demšar (2006). To take advantage of the Demšar (2006) recommendations, we would need many more data sets to evaluate the algorithms. With more computational resources, we could apply the models to more data sets to generate a distribution of the desired performance metric that would improve our confidence in the results. Another approach would be to compare the performance metrics distribution for each pair of models using a k-fold process with the scroll window to maintain time consistency. The k-fold method requires sacrificing part of the data set used for each training, possibly decreasing the algorithm's potential. These two proposed extensions suggest that using data augmentation can be very useful. Future work may use transfer learning to initialize parameters to overcome this k-fold method problem. Researchers can also use a multiple initialization approach to sample metric values, providing elements to perform bias and variance analysis to extend this work. Due to the high computational cost required, we did not use multiple initialization techniques in this study. However, we present some evidence that can be analyzed with more computational resources, following our proposed methods and using our available code.

CRediT authorship contribution statement

Leonardo Kanashiro Felizardo: Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Project administration. **Francisco Caio Lima Paiva:** Writing – original draft, Writing – review & editing. **Catharine de Vita Graves:** Conceptualization, Methodology, Formal analysis, Investigation. **Elia Yathie Matsumoto:** Writing – review. **Anna Helena Reali Costa:** Validation, Writing – review & editing, Supervision, Funding acquisition. **Emilio Del-Moral-Hernandez:** Validation, Writing – review & editing, Supervision, Funding acquisition. **Paolo Brandimarte:** Validation, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES - Coordination for the Improvement of Higher Education Personnel, Finance Code 001, grant 88882.333380/2019-01), Brazil, and *Conselho Nacional de Desenvolvimento Científico e Tecnológico* (CNPq - Brazilian National Council for Scientific and Technological Development, grants 100085/2020-9 and

310085/2020-9), and by *Itaú Unibanco S.A.* through the *Programa de Bolsas Itaú* (PBI) of the *Centro de Ciência de Dados* (C^2D, EP-USP).

Any views and opinions expressed in this article are those of the authors and do not necessarily reflect the official policy or position of the funding companies.

References

Aboussalah, A. M., & Lee, C.-G. (2020). Continuous control with stacked deep dynamic recurrent reinforcement learning for portfolio optimization. *Expert Systems with Applications*, *140*, Article 112891. http://dx.doi.org/10.1016/j.eswa.2019.112891.

Allen, M. P. (1997). The problem of multicollinearity. In *Understanding regression analysis* (pp. 176–180). Boston, MA: Springer US, http://dx.doi.org/10.1007/978-0-585-25657-3_37.

Allen, F., & Karjalainen, R. (1999). Using genetic algorithms to find technical trading rules. *Journal of Financial Economics*, *51*(2), 245–271. http://dx.doi.org/10.1016/S0304-405X(98)00052-X.

Almahdi, S., & Yang, S. Y. (2017). An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications*, *87*, 267–279. http://dx.doi.org/10.1016/j.eswa.2017.06.023.

Almahdi, S., & Yang, S. Y. (2019). A constrained portfolio trading system using particle swarm algorithm and recurrent reinforcement learning. *Expert Systems with Applications*, *130*, 145–156.

Baird, L. (1993). *Advantage updating: Technical Report WL-TR-93-1146*.

Bessembinder, H., & Chan, K. (1995). The profitability of technical trading rules in the Asian stock markets. *Pacific-Basin Finance Journal*, *3*(2), 257–284. http://dx.doi.org/10.1016/0927-538X(95)00002-3.

Brock, W., Lakonishok, J., & LeBaron, B. (1992). Simple technical trading rules and the stochastic properties of stock returns. *The Journal of Finance*, *47*(5), 1731–1764, http://www.jstor.org/stable/2328994.

Campbell, J. Y., Lo, A. W., & MacKinlay, A. (1997). *The econometrics of financial markets*. Princeton University Press, http://www.jstor.org/stable/j.ctt7skm5.

Choi, H., Ryu, S., & Kim, H. (2018). Short-term load forecasting based on ResNet and LSTM. In *2018 IEEE international conference on communications, control, and computing technologies for smart grids (SmartGridComm)* (pp. 1–6). http://dx.doi.org/10.1109/SmartGridComm.2018.8587554.

Chopra, N., Lakonishok, J., & Ritter, J. R. (1992). Measuring abnormal performance: Do stocks overreact? *Journal of Financial Economics*, *31*(2), 235–268. http://dx.doi.org/10.1016/0304-405X(92)90005-I.

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). Berlin, Heidelberg: Springer Berlin Heidelberg.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*.

Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2017). Deep direct reinforcement learning for financial signal representation and trading. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(3), 653–664. http://dx.doi.org/10.1109/TNNLS.2016.2522401.

Eckles, D., & Kaptein, M. (2014). Thompson sampling with the online bootstrap.

Fama, E. F. (1965). Random walks in stock market prices. *Financial Analysts Journal*, http://dx.doi.org/10.2469/faj.v21.n5.55.

Fama, E. F., & French, K. R. (1988). Permanent and temporary components of stock prices. *Journal of Political Economy*, *96*(2), 246–273.

Feinberg, E. A., & Shwartz, A. (Eds.), (2002). *Handbook of Markov decision processes*. Springer US, http://dx.doi.org/10.1007/978-1-4615-0805-2.

Feuerriegel, S., & Prendinger, H. (2016). News-based trading strategies. *Decision Support Systems*, *90*, 65–74. http://dx.doi.org/10.1016/j.dss.2016.06.020.

Hamilton, W. P. (1922). *The stock market barometer: study of its forecast value based on Charles H. Dow's theory of the price movement. with an analysis of the market and its history since 1897*. Harper & Brothers.

He, R., Liu, Y., Wang, K., Zhao, N., Yuan, Y., Li, Q., & Zhang, H. (2019). Automatic cardiac arrhythmia classification using combination of deep residual network and bidirectional LSTM. *IEEE Access*, *7*, 102119–102135. http://dx.doi.org/10.1109/ACCESS.2019.2931500.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE computer society conference on computer vision and pattern recognition*. http://dx.doi.org/10.1109/CVPR.2016.90.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. In *32nd AAAI conf. on artificial intelligence (AAAI-18)*. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16669.

Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, *33*(4), 917–963. http://dx.doi.org/10.1007/s10618-019-00619-1.

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. http://dx.doi.org/10.1016/j.eswa.2018.09.036.

Kang, Q., Zhou, H., & Kang, Y. (2018). An asynchronous advantage actor-critic reinforcement learning method for stock selection and portfolio management. In *Proceedings of the 2nd international conference on big data research - ICBDR 2018* (pp. 141–145). New York, New York, USA: ACM Press, http://dx.doi.org/10.1145/3291801.3291831.

Kapoor, S., & Prosad, J. M. (2017). Behavioural finance: A review. *Procedia Computer Science*, *122*, 50–54. http://dx.doi.org/10.1016/j.procs.2017.11.340, 5th International Conference on Information Technology and Quantitative Management, ITQM 2017.

Khadjeh Nassirtoussi, A., Aghabozorgi, S., Ying Wah, T., & Ngo, D. C. L. (2014). Text mining for market prediction: A systematic review. *Expert Systems with Applications*, *41*(16), 7653–7670. http://dx.doi.org/10.1016/j.eswa.2014.06.009.

Kirkpatrick II, C. D., & Dahlquist, J. A. (2010). *Technical analysis: the complete resource for financial market technicians*. FT Press.

Kuo, R., Chen, C., & Hwang, Y. (2001). An intelligent stock trading decision support system through integration of genetic algorithm based fuzzy neural network and artificial neural network. *Fuzzy Sets and Systems*, *118*(1), 21–45. http://dx.doi.org/10.1016/S0165-0114(98)00399-6.

Li, Y., Zheng, W., & Zheng, Z. (2019). Deep robust reinforcement learning for practical algorithmic trading. *IEEE Access*, *7*, 108014–108022. http://dx.doi.org/10.1109/ACCESS.2019.2932789.

Lima Paiva, F. C., Felizardo, L. K., Bianchi, R. A. d. C. B., & Costa, A. H. R. (2021). Intelligent trading systems: A sentiment-aware reinforcement learning approach. In *Proceedings of the Second ACM International Conference on AI in Finance* (pp. 1–9). New York, NY, USA: ACM, http://dx.doi.org/10.1145/3490354.3494445, arXiv:2112.02095.

Maringer, D., & Ramtohul, T. (2010). Threshold recurrent reinforcement learning model for automated trading. In *Applications of evolutionary computation* (pp. 212—-221). Berlin, Heidelberg: Springer Berlin Heidelberg, http://dx.doi.org/10.1007/978-3-642-12242-2_22.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. CoRR, abs/1602.01783, arXiv:1602.01783.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. A. (2013). Playing atari with deep reinforcement learning. CoRR, abs/1312.5602, arXiv:1312.5602.

Moody, J., Saffell, M., Liao, Y., & Wu, L. (1998). Reinforcement learning for trading systems and portfolios: Immediate vs future rewards. In A.-P. N. Refenes, A. N. Burgess, & J. E. Moody (Eds.), *Decision technologies for computational finance: proceedings of the fifth international conference computational finance* (pp. 129–140). Boston, MA: Springer US, http://dx.doi.org/10.1007/978-1-4615-5625-1_10.

Moody, J., & Wu, L. (1997). Optimization of trading systems and portfolios. In *Proceedings of the IEEE/IAFE 1997 computational intelligence for financial engineering (CIFEr)* (pp. 300–307). IEEE, http://dx.doi.org/10.1109/CIFER.1997.618952.

Moody, J., Wu, L., Liao, Y., & Saffell, M. (1998). Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, *17*(5–6), 441–470. http://dx.doi.org/10.1002/(SICI)1099-131X(1998090)17:5/6<441::AID-FOR707>3.0.CO;2-#.

Nakano, M., Takahashi, A., & Takahashi, S. (2018). Bitcoin technical trading with artificial neural network. *Physica A: Statistical Mechanics and its Applications*, *510*, 587–609. http://dx.doi.org/10.1016/j.physa.2018.07.017.

Neuneier, R. (1995). Optimal asset allocation using adaptive dynamic programming. In *2: vol. 32, Advances in neural information processing systems 8* (pp. 952–958). http://papers.nips.cc/paper/1121-optimal-asset-allocation-using-adaptive-dynamic-programming.pdf.

Neuneier, R. (1998). Enhancing Q-learning for optimal asset allocation. In *Advances in neural information processing systems*.

Park, H., Sim, M. K., & Choi, D. G. (2020). An intelligent financial portfolio trading strategy using deep Q-learning. *Expert Systems with Applications*, *158*, Article 113573. http://dx.doi.org/10.1016/j.eswa.2020.113573.

Pendharkar, P. C., & Cusatis, P. (2018). Trading financial indices with reinforcement learning agents. *Expert Systems with Applications*, *103*, 1–13. http://dx.doi.org/10.1016/j.eswa.2018.02.032.

Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché Buc, F., Fox, E., & Larochelle, H. (2021). Improving reproducibility in machine learning research: a report from the neurips 2019 reproducibility program. *Journal of Machine Learning Research*, *22*.

Ponomarev, E. S., Oseledets, I. V., & Cichocki, A. S. (2019). Using reinforcement learning in the algorithmic trading problem. *Journal of Communications Technology and Electronics*, *64*(12), 1450–1457. http://dx.doi.org/10.1134/S1064226919120131.

Sang, C., & Di Pierro, M. (2019). Improving trading technical analysis with TensorFlow long short-term memory (LSTM) neural network. *The Journal of Finance and Data Science*, *5*(1), 1–11. http://dx.doi.org/10.1016/j.jfds.2018.10.003.

Shen, W., Wang, J., Jiang, Y.-G., & Zha, H. (2015). Portfolio choices with orthogonal bandit learning. In *IJCAI'15, Proceedings of the 24th international conference on artificial intelligence* (pp. 974–980). AAAI Press.

Trippi, R. R., & DeSieno, D. (1992). Trading equity index futures with a neural network. *The Journal of Portfolio Management*, *19*(1), 27–33. http://dx.doi.org/10.3905/jpm.1992.409432.

Wang, J., Liu, Y., & Li, B. (2020). Reinforcement learning with perturbed rewards. In *Proceedings of the AAAI conference on artificial intelligence, vol. 34* (04), (pp. 6202–6209). http://dx.doi.org/10.1609/aaai.v34i04.6086.

Wang, J., Zhang, Y., Tang, K., Wu, J., & Xiong, Z. (2019). AlphaStock. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining - KDD '19* (pp. 1900–1908). New York, New York, USA: ACM Press, http://dx.doi.org/10.1145/3292500.3330647.

Ye, Y., Pei, H., Wang, B., Chen, P.-Y., Zhu, Y., Xiao, J., & Li, B. (2020). Reinforcement-learning based portfolio management with augmented asset movement prediction states. In *Proceedings of the AAAI conference on artificial intelligence, vol. 34* (01), (pp. 1112–1119). http://dx.doi.org/10.1609/aaai.v34i01.5462.

Yu, P., Lee, J. S., Kulyatin, I., Shi, Z., & Dasgupta, S. (2019). Model-based deep reinforcement learning for dynamic portfolio optimization. http://arxiv.org/abs/1901.08740.

Zarkias, K. S., Passalis, N., Tsantekidis, A., & Tefas, A. (2019). Deep reinforcement learning for financial trading using price trailing. In *ICASSP 2019 - 2019 IEEE international conference on acoustics, speech and signal processing (ICASSP), vol. 117* (pp. 3067–3071). IEEE, http://dx.doi.org/10.1109/ICASSP.2019.8683161.