Expert Systems With Applications 173 (2021) 114632

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An application of deep reinforcement learning to algorithmic trading

Thibaut Théate *, Damien Ernst

Montefiore Institute, University of Liège, Allée de la découverte 10, 4000 Liège, Belgium

ARTICLE INFO

Keywords:
Artificial intelligence
Deep reinforcement learning
Algorithmic trading
Trading policy

ABSTRACT

This scientific research paper presents an innovative approach based on deep reinforcement learning (DRL) to solve the algorithmic trading problem of determining the optimal trading position at any point in time during a trading activity in the stock market. It proposes a novel DRL trading policy so as to maximise the resulting Sharpe ratio performance indicator on a broad range of stock markets. Denominated the Trading Deep Q-Network algorithm (TDQN), this new DRL approach is inspired from the popular DQN algorithm and significantly adapted to the specific algorithmic trading problem at hand. The training of the resulting reinforcement learning (RL) agent is entirely based on the generation of artificial trajectories from a limited set of stock market historical data. In order to objectively assess the performance of trading strategies, the research paper also proposes a novel, more rigorous performance assessment methodology. Following this new performance assessment approach, promising results are reported for the TDQN algorithm.

1. Introduction

For the past few years, the interest in *artificial intelligence* (AI) has grown at a very fast pace, with numerous research papers published every year. A key element for this growing interest is related to the impressive successes of *deep learning* (DL) techniques which are based on *deep neural networks* (DNN) - mathematical models directly inspired by the human brain structure. These specific techniques are nowadays the state of the art in many applications such as speech recognition, image classification or natural language processing. In parallel to DL, another field of research has recently gained much more attention from the research community: *deep reinforcement learning* (DRL). This family of techniques is concerned with the learning process of an intelligent agent (i) interacting in a sequential manner with an unknown environment (ii) aiming to maximise its cumulative rewards and (iii) using DL techniques to generalise the information acquired from the interaction with the environment. The many recent successes of DRL techniques highlight their ability to solve complex sequential decision-making problems.

Nowadays, an emerging industry which is growing extremely fast is the *financial technology* industry, generally referred to by the abbreviation *FinTech*. The objective of FinTech is pretty simple: to extensively take advantage of technology in order to innovate and improve activities in finance. In the coming years, the FinTech industry is expected to revolutionise the way many decision-making problems related to the financial sector are addressed, including the problems related to trading, investment, risk management, portfolio management, fraud detection and financial advising, to cite a few. Such complex decision-making problems are extremely complex to solve as they generally have a sequential nature and are highly stochastic, with an environment partially observable and potentially adversarial. In particular, *algorithmic trading*, which is a key sector of the FinTech industry, presents particularly interesting challenges. Also called quantitative trading, algorithmic trading is the methodology to trade using computers and a specific set of mathematical rules.

The core objective of this research paper is to answer the following question: how to design a novel trading policy (algorithm) based on AI techniques that could compete with the popular algorithmic trading strategies widely adopted in practice? To answer this question, this scientific article presents and analyses a novel DRL solution to tackle the algorithmic trading problem of determining the optimal trading position (long or short) at any point in time during a trading activity in the stock market. The algorithmic solution presented in this research paper is inspired by the popular Deep Q-Network (DQN) algorithm, which has been adapted to the particular sequential decision-making problem at hand. The research question to be answered is all the more relevant as the trading environment presents very different characteristics from those which have already been successfully solved by DRL approaches, mainly significant stochasticity and extremely poor observability.

The main contributions of this research paper are threefold:

* Corresponding author.
E-mail addresses: thibaut.theate@uliege.be (T. Théate), dernst@uliege.be (D. Ernst).

https://doi.org/10.1016/j.eswa.2021.114632
Received 7 April 2020; Received in revised form 13 January 2021; Accepted 17 January 2021
Available online 28 January 2021

- The algorithmic trading decision-making problem considered is rigorously formalised and framed into a reinforcement learning (RL) problem.
- A novel DRL algorithm, denominated *Trading Deep Q-Network* (TDQN), is presented to solve this RL problem. This new algorithm is inspired by the popular DQN algorithm and significantly adapted to the particular decision-making problem at hand, which presents very different characteristics compared to the task of playing Atari games for which the DQN algorithm was originally designed.
- A new performance assessment methodology is proposed to fairly evaluate the performance of the TDQN algorithm as well as other trading strategies operated in the stock market.

The scientific research paper is structured as follows. First of all, a brief review of the scientific literature around the algorithmic trading field and its main AI-based contributions is presented in Section 2. Afterwards, Section 3 introduces and rigorously formalises the particular algorithmic trading problem considered. Additionally, this section makes the link with the reinforcement learning (RL) approach. Then, Section 4 covers the complete design of the TDQN trading strategy based on DRL concepts. Subsequently, Section 5 proposes a novel methodology to objectively assess the performance of trading strategies. Section 6 is concerned with the presentation and discussion of the results achieved by the TDQN trading strategy. To end this research paper, Section 7 discusses interesting leads as future work and draws meaningful conclusions.

2. Literature review

To begin this brief literature review, two facts have to be emphasised. Firstly, it is important to be aware that many sound scientific works in the field of algorithmic trading are not publicly available. As explained in Li (2017), due to the huge amount of money at stake, private FinTech firms are very unlikely to make their latest research results public. Secondly, it should be acknowledged that making a fair comparison between trading strategies is a challenging task, due to the lack of a common, well-established framework to properly evaluate their performance. Instead, the authors generally define their own framework with their evident bias. Another major problem is related to the trading costs which are variously defined or even omitted.

First of all, most of the works in algorithmic trading are techniques developed by mathematicians, economists and traders who do not exploit AI. Typical examples of classical trading strategies are the *trend following* and *mean reversion* strategies, which are covered in detail in Chan (2009), Chan (2013) and Narang (2009). Then, the majority of works applying machine learning (ML) techniques in the algorithmic trading field focus on forecasting. If the financial market evolution is known in advance with a reasonable level of confidence, the optimal trading decisions can easily be computed. Following this approach, DL techniques have already been investigated with good results, see e.g. Arévalo, Niño, Hernández, and Sandoval (2016) introducing a trading strategy based on a DNN, and especially Bao, Yue, and Rao (2017) using wavelet transforms, stacked autoencoders and long short-term memory (LSTM). One can also mention Paiva, Cardoso, Hanaoka, and Duarte (2019) proposing a decision-making model based on the combination of a support vector machine method for forecasting stock returns and on the mean–variance method for portfolio selection. Alternatively, several authors have already investigated RL techniques to solve this algorithmic trading problem. For instance, Moody and Saffell (2001) introduced a recurrent RL algorithm for discovering new investment policies without the need to build forecasting models, and Dempster and Leemans (2006) used adaptive RL to trade in foreign exchange markets. More recently, a few works investigated DRL techniques in a scientifically sound way to solve this particular algorithmic trading problem. For instance, one can first mention Deng, Bao, Kong, Ren, and Dai (2017) which introduced the fuzzy recurrent deep neural network structure to obtain a technical-indicator-free trading system taking advantage of fuzzy learning to reduce the time series uncertainty. One can also mention Carapuço, Neves, and Horta (2018) which studied the application of the deep Q-learning algorithm for trading in foreign exchange markets. Another interesting research work is Jeong and Kim (2019) which proposed to determine the action strategies together with the number of shares to trade by combining RL and a specific DNN, and using transfer learning to address problems of insufficient financial data. One can also mention Almahdi and Yang (2019) which introduced a new constrained portfolio trading system by combining particle swarm and recurrent RL. Another work is Lei, Zhang, Li, Yang, and Shen (2020) which presented a time-driven feature-aware jointly DRL model leveraging both DL and RL to improve the financial signal representation learning and action decision making in algorithmic trading. One can cite as well Park, Sim, and Choi (2020) which proposed an approach for financial portfolio trading using deep Q-learning together with a discrete combinatorial action space. Finally, there exist a few interesting works studying the application of DRL techniques to algorithmic trading in specific markets, such as in the field of energy, see e.g. the article Boukas et al. (2020).

To finish with this short literature review, a sensitive problem in the scientific literature is the tendency to prioritise the communication of good results or findings, sometimes at the cost of a proper scientific approach with objective criticism. Going even further, Ioannidis (2005) even states that most published research findings in certain sensitive fields are probably false. Such concern appears to be all the more relevant in the field of financial sciences, especially when the subject directly relates to trading activities. Indeed, Bailey, Borwein, de Prado, and Zhu (2014) claims that many scientific publications in finance suffer from a lack of a proper scientific approach, instead getting closer to pseudo-mathematics and financial charlatanism than rigorous sciences. Aware of these concerning tendencies, the present research paper intends to deliver an unbiased scientific evaluation of the novel DRL algorithm proposed.

3. Algorithmic trading problem formalisation

In this section, the sequential decision-making algorithmic trading problem studied in this research paper is presented in detail. Moreover, a rigorous formalisation of this particular problem is performed. Additionally, the link with the RL formalism is highlighted.

3.1. Algorithmic trading

Algorithmic trading, also called quantitative trading, is a subfield of finance, which can be viewed as the approach of automatically making trading decisions based on a set of mathematical rules computed by a machine. This commonly accepted definition is adopted in this research paper, although other definitions exist in the literature. Indeed, several authors differentiate the trading decisions (quantitative trading) from the actual trading execution (algorithmic trading). For the sake of generality, algorithmic trading and quantitative trading are considered synonyms in this research paper, defining the entire automated trading process. Algorithmic trading has already proven to be very beneficial to markets, the main benefit being the significant improvement in liquidity, as discussed in Hendershott, Jones, and Menkveld (2011). For more information about this specific field, please refer to Treleaven, Galas, and Lalchand (2013) and Nuti, Mirghaemi, Treleaven, and Yingsaeree (2011).

There are many different markets suitable to apply algorithmic trading strategies. Stocks and shares can be traded in the stock markets, FOREX trading is concerned with foreign currencies, or a trader could invest in commodity futures, to only cite a few. The recent rise of cryptocurrencies, such as the Bitcoin, offers new interesting possibilities as well. Ideally, the DRL algorithms developed in this research paper should be applicable to multiple markets. However, the focus will be set

on stock markets for now, with an extension to various other markets planned in the future.

In fact, a trading activity can be viewed as the management of a portfolio, which is a set of assets including diverse stocks, bonds, commodities, currencies, etc. In the scope of this research paper, the portfolio considered consists of one single stock together with the agent cash. The portfolio value v_t is then composed of the trading agent cash value v_t^c and the share value v_t^s, which continuously evolves over time t. Buying and selling operations are simply cash and share exchanges. The trading agent interacts with the stock market through an order book, which contains the entire set of buying orders (*bids*) and selling orders (*asks*). An example of a simple order book is depicted in Table 1. An order represents the willingness of a market participant to trade and is composed of a price p, a quantity q and a side s (bid or ask). For a trade to occur, a match between bid and ask orders is required, an event which can only happen if $p_{max}^{bid} \geqslant p_{min}^{ask}$, with p_{max}^{bid} (p_{min}^{ask}) being the maximum (minimum) price of a bid (ask) order. Then, a trading agent faces a very difficult task in order to generate profit: what, when, how, at which price and which quantity to trade. This is the algorithmic trading complex sequential decision-making problem studied in this scientific research paper.

3.2. Timeline discretisation

Since trading decisions can be issued at any time, the trading activity is a continuous process. In order to study the algorithmic trading problem described in this research paper, a discretisation operation of the continuous timeline is performed. The trading timeline is discretized into a high number of discrete trading time steps t of constant duration Δt. In this research paper, for the sake of clarity, the increment (decrement) operations $t+1$ ($t-1$) are used to model the discrete transition from time step t to time step $t+\Delta t$ ($t-\Delta t$).

The duration Δt is closely linked to the trading frequency targeted by the trading agent (very high trading frequency, intraday, daily, monthly, etc.). Such discretisation operation inevitably imposes a constraint with respect to this trading frequency. Indeed, because the duration Δt between two time steps cannot be chosen as small as possible due to technical constraints, the maximum trading frequency achievable, equal to $1/\Delta t$, is limited. In the scope of this research paper, this constraint is met as the trading frequency targeted is daily, meaning that the trading agent makes a new decision once every day.

3.3. Trading strategy

The algorithmic trading approach is rule based, meaning that the trading decisions are made according to a set of rules: a *trading strategy*. In technical terms, a trading strategy can be viewed as a programmed policy $\pi(a_t|i_t)$, either deterministic or stochastic, which outputs a trading action a_t according to the information available to the trading agent i_t at time step t. Additionally, a key characteristic of a trading strategy is its sequential aspect, as illustrated in Fig. 1. An agent executing its trading strategy sequentially applies the following steps:

1. Update of the available market information i_t.
2. Execution of the policy $\pi(a_t|i_t)$ to get action a_t.

Table 1
Example of a simple order book.

Side s	Quantity q	Price p
Ask	3000	107
Ask	1500	106
Ask	500	105
Bid	1000	95
Bid	2000	94
Bid	4000	93

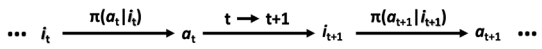


Fig. 1. Illustration of a trading strategy execution.

3. Application of the designated trading action a_t.
4. Next time step $t \rightarrow t+1$, loop back to step 1.

In the following subsection, the algorithmic trading sequential decision-making problem, which shares similarities with other problems successfully tackled by the RL community, is casted as an RL problem.

3.4. Reinforcement learning problem formalisation

As illustrated in Fig. 2, reinforcement learning is concerned with the sequential interaction of an agent with its environment. At each time step t, the RL agent firstly observes the RL environment of internal state s_t, and retrieves an observation o_t. It then executes the action a_t resulting from its RL policy $\pi(a_t|h_t)$ where h_t is the RL agent history and receives a reward r_t as a consequence of its action. In this RL context, the agent history can be expressed as $h_t = \{(o_\tau, a_\tau, r_\tau)|\tau = 0, 1, \ldots, t\}$.

Reinforcement learning techniques are concerned with the design of policies π maximising an optimality criterion, which directly depends on the immediate rewards r_t observed over a certain time horizon. The most popular optimality criterion is the expected discounted sum of rewards over an infinite time horizon. Mathematically, the resulting optimal policy π^* is expressed as the following:

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}[R|\pi] \tag{1}$$

$$R = \sum_{t=0}^{\infty} \gamma^t r_t \tag{2}$$

The parameter γ is the discount factor ($\gamma \in [0, 1]$). It determines the importance of future rewards. For instance, if $\gamma = 0$, the RL agent is said to be myopic as it only considers the current reward and totally discards the future rewards. When the discount factor increases, the RL agent tends to become more long-term oriented. In the extreme case where $\gamma = 1$, the RL agent considers each reward equally. This key parameter should be tuned according to the desired behaviour.

3.4.1. RL observations

In the scope of this algorithmic trading problem, the RL environment is the entire complex trading world gravitating around the RL agent. In fact, this trading environment can be viewed as an abstraction including the trading mechanisms together with every single piece of information capable of having an effect on the trading activity of the agent. A major challenge of the algorithmic trading problem is the extremely poor observability of this environment. Indeed, a significant amount of information is simply hidden to the trading agent, ranging from some companies' confidential information to the other market participants'

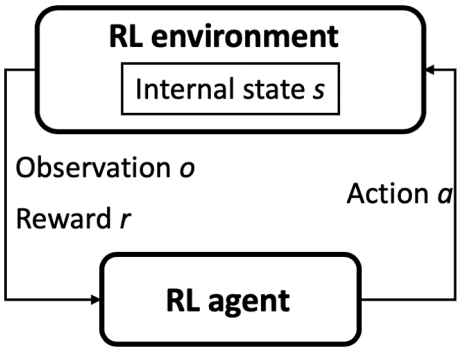


Fig. 2. Reinforcement learning core building blocks.

strategies. In fact, the information available to the RL agent is extremely limited compared to the complexity of the environment. Moreover, this information can take various forms, both quantitative and qualitative. Correctly processing such information and re-expressing it using relevant quantitative figures while minimising the subjective bias is capital. Finally, there are significant time correlation complexities to deal with. Therefore, the information retrieved by the RL agent at each time step should be considered sequentially as a series of information rather than individually.

At each trading time step t, the RL agent observes the stock market whose internal state is $s_t \in \mathscr{S}$. The limited information collected by the agent on this complex trading environment is denoted by $o_t \in \mathscr{O}$. Ideally, this observation space $\mathscr{O}$ should encompass all the information capable of influencing the market prices. Because of the sequential aspect of the algorithmic trading problem, an observation o_t has to be considered as a sequence of both the information gathered during the previous τ time steps (history) and the newly available information at time step t. In this research paper, the RL agent observations can be mathematically expressed as the following:

$$o_t = \{S(t'),\ D(t'),\ T(t'),\ I(t'),\ M(t'),\ N(t'),\ E(t')\}_{t'=t-\tau}^{t} \tag{3}$$

where:

- $S(t)$ represents the state information of the RL agent at time step t (current trading position, number of shares owned by the agent, available cash).
- $D(t)$ is the information gathered by the agent at time step t concerning the OHLCV (Open-High-Low-Close-Volume) data characterising the stock market. More precisely, $D(t)$ can be expressed as follows:

$$D(t) = \{p_t^O,\ p_t^H,\ p_t^L,\ p_t^C,\ V_t\} \tag{4}$$

  where:
  - p_t^O is the stock market price at the opening of the time period $[t-\Delta t,\ t[$.
  - p_t^H is the highest stock market price over the time period $[t-\Delta t,\ t[$.
  - p_t^L is the lowest stock market price over the time period $[t-\Delta t,\ t[$.
  - p_t^C is the stock market price at the closing of the time period $[t-\Delta t,\ t[$.
  - V_t is the total volume of shares exchanged over the time period $[t-\Delta t,\ t[$.
- $T(t)$ is the agent information regarding the trading time step t (date, weekday, time).
- $I(t)$ is the agent information regarding multiple technical indicators about the stock market targeted at time step t. There exist many technical indicators providing extra insights about diverse financial phenomena, such as moving average convergence divergence (MACD), relative strength index (RSI) or average directional index (ADX), to only cite a few.
- $M(t)$ gathers the macroeconomic information at the disposal of the agent at time step t. There are many interesting macroeconomic indicators which could potentially be useful to forecast markets' evolution, such as the interest rate or the exchange rate.
- $N(t)$ represents the news information gathered by the agent at time step t. These news data can be extracted from various sources such as social media (Twitter, Facebook, LinkedIn), the newspapers, specific journals, etc. Complex sentiment analysis models could then be built to extract meaningful quantitative figures (quantity, sentiment polarity and subjectivity, etc.) from the news. The benefits of such information has already been demonstrated by several authors, see e.g. Leinweber and Sisk (2011), Bollen, Mao, and jun Zeng (2011) and Nuij, Milea, Hogenboom, Frasincar, and Kaymak (2014).
- $E(t)$ is any extra useful information at the disposal of the trading agent at time step t, such as other market participants trading strategies, companies' confidential information, similar stock market behaviours, rumours, experts' advice, etc.

Observation space reduction:

In the scope of this research paper, it is assumed that the only information considered by the RL agent is the classical OHLCV data $D(t)$ together with the state information $S(t)$. Especially, the reduced observation space $\mathscr{O}$ encompasses the current trading position together with a series of the previous $\tau+1$ daily open–high-low-close prices and daily traded volume. With such an assumption, the reduced RL observation o_t can be expressed as the following:

$$o_t = \left\{\{p_{t'}^O,\ p_{t'}^H,\ p_{t'}^L,\ p_{t'}^C,\ V_{t'}\}_{t'=t-\tau}^{t},\ P_t\right\} \tag{5}$$

with P_t being the trading position of the RL agent at time step t (either *long* or *short*, as explained in the next subsection of this research paper).

3.4.2. RL actions

At each time step t, the RL agent executes a trading action $a_t \in \mathscr{A}$ resulting from its policy $\pi(a_t|h_t)$. In fact, the trading agent has to answer several questions: whether, how and how much to trade? Such decisions can be modelled by the quantity of shares bought by the trading agent at time step t, represented by $Q_t \in \mathbb{Z}$. Therefore, the RL actions can be expressed as the following:

$$a_t = Q_t \tag{6}$$

Three cases can occur depending on the value of Q_t:

- $\underline{Q_t > 0}$: The RL agent *buys* shares on the stock market, by posting new *bid* orders on the order book.
- $\underline{Q_t < 0}$: The RL agent *sells* shares on the stock market, by posting new *ask* orders on the order book.
- $\underline{Q_t = 0}$: The RL agent *holds*, meaning that it does not buy nor sell any shares on the stock market.

Actually, the real actions occurring in the scope of a trading activity are the orders posted on the order book. The RL agent is assumed to communicate with an external module responsible for the synthesis of these true actions according to the value of Q_t: the *trading execution system*. Despite being out of the scope of this paper, it should be mentioned that multiple execution strategies can be considered depending on the general trading purpose.

The trading actions have an impact on the two components of the portfolio value, namely the cash and share values. Assuming that the trading actions occur close to the market closure at price $p_t \simeq p_t^C$, the updates of these components are governed by the following equations:

$$v_{t+1}^c = v_t^c - Q_t\ p_t \tag{7}$$

$$v_{t+1}^s = \underbrace{(n_t + Q_t)}_{n_{t+1}}\ p_{t+1} \tag{8}$$

with $n_t \in \mathbb{Z}$ being the number of shares owned by the trading agent at time step t. In the scope of this research paper, negative values are allowed for this quantity. Despite being surprising at first glance, a negative number of shares simply corresponds to shares borrowed and sold, with the obligation to repay the lender in shares in the future. Such a mechanism is particularly interesting as it introduces new possibilities for the trading agent.

Two important constraints are assumed concerning the quantity of traded shares Q_t. Firstly, contrarily to the share value v_t^s which can be both positive or negative, the cash value v_t^c has to remain positive for every trading time steps t. This constraint imposes an upper bound on the number of shares that the trading agent is capable of purchasing, this volume of shares being easily derived from Eq. (7). Secondly, there exists a risk associated with the impossibility to repay the share lender if the agent suffers significant losses. To prevent such a situation from happening, the cash value v_t^c is constrained to be sufficiently large when a negative number of shares is owned, in order to be able to get back to a

neutral position ($n_t = 0$). A maximum relative change in prices, expressed in % and denoted $\epsilon \in \mathbb{R}^+$, is assumed by the RL agent prior to the trading activity. This parameter corresponds to the maximum market daily evolution supposed by the agent over the entire trading horizon, so that the trading agent should always be capable of paying back the share lender as long as the market variation remains below this value. Therefore, the constraints acting upon the RL actions at time step t can be mathematically expressed as follows:

$$v_{t+1}^c \geqslant 0 \tag{9}$$

$$v_{t+1}^c \geqslant -n_{t+1}\ p_t\ (1+\epsilon) \tag{10}$$

with the following condition assumed to be satisfied:

$$\left|\frac{p_{t+1}-p_t}{p_t}\right| \leqslant \epsilon \tag{11}$$

Trading costs consideration:

Actually, the modelling represented by Eq. (7) is inaccurate and will inevitably lead to unrealistic results. Indeed, whenever simulating trading activities, the trading costs should not be neglected. Such omission is generally misleading as a trading strategy, highly profitable in simulations, may be likely to generate large losses in real trading situations due to these trading costs, especially when the trading frequency is high. The trading costs can be subdivided into two categories. On the one hand, there are explicit costs which are induced by transaction costs and taxes. On the other hand, there are implicit costs, called slippage costs, which are composed of three main elements and are associated to some of the dynamics of the trading environment. The different slippage costs are detailed hereafter:

- **Spread costs:** These costs are related to the difference between the minimum ask price p_{min}^{ask} and the maximum bid price p_{max}^{bid}, called the *spread*. Because the complete state of the order book is generally too complex to efficiently process or even not available, the trading decisions are mostly based on the middle price $p^{mid} = (p_{max}^{bid} + p_{min}^{ask})/2$. However, a buying (selling) trade issued at p^{mid} inevitably occurs at a price $p \geqslant p_{min}^{ask}$ ($p \leqslant p_{max}^{bid}$). Such costs are all the more significant that the stock market liquidity is low compared to the volume of shares traded.
- **Market impact costs:** These costs are induced by the impact of the trader's actions on the market. Each trade (both buying and selling orders) is potentially capable of influencing the price. This phenomenon is all the more important that the stock market liquidity is low with respect to the volume of shares traded.
- **Timing costs:** These costs are related to the time required for a trade to physically happen once the trading decision is made, knowing that the market price is continuously evolving. The first cause is the inevitable latency which delays the posting of the orders on the market order book. The second cause is the intentional delays generated by the trading execution system. For instance, a large trade could be split into multiple smaller trades spread over time in order to limit the market impact costs.

An accurate modelling of the trading costs is required to realistically reproduce the dynamics of the real trading environment. While explicit costs are relatively easy to take into account, the valid modelling of slippage costs is a truly complex task. In this research paper, the integration of both costs into the RL environment is performed through a heuristic. When a trade is executed, a certain amount of capital equivalent to a percentage C of the amount of money invested is lost. This parameter was realistically chosen equal to 0.1% in the forthcoming simulations.

Practically, these trading costs are withdrawn from the trading agent cash. Following the heuristic previously introduced, Eq. (7) can be re-expressed with a corrective term modelling the trading costs:

$$v_{t+1}^c = v_t^c - Q_t\ p_t - \underbrace{C\ |Q_t|\ p_t}_{\text{Trading costs}} \tag{12}$$

Moreover, the trading costs have to be properly considered in the constraint expressed in Eq. (10). Indeed, the cash value v_t^c should be sufficiently large to get back to a neutral position ($n_t = 0$) when the maximum market variation ϵ occurs, the trading costs being included. Consequently, Eq. (10) is re-expressed as follows:

$$v_{t+1}^c \geqslant -n_{t+1}\ p_t\ (1+\epsilon)(1+C) \tag{13}$$

Eventually, the RL action space $\mathscr{A}$ can be defined as the discrete set of acceptable values for the quantity of traded shares Q_t. Derived in detail in Appendix A, the RL action space $\mathscr{A}$ is mathematically expressed as the following:

$$\mathscr{A} = \left\{ Q_t \in \mathbb{Z} \cap \left[\underline{Q_t},\ \overline{Q_t} \right] \right\} \tag{14}$$

where:

- $\overline{Q_t} = \frac{v_t^c}{p_t\ (1+C)}$
- $\underline{Q_t} = \begin{cases} \dfrac{\Delta_t}{p_t\ \epsilon(1+C)} & \text{if } \Delta_t \geqslant 0 \\ \dfrac{\Delta_t}{p_t\ (2C+\epsilon(1+C))} & \text{if } \Delta_t < 0 \end{cases}$

  with $\Delta_t = -v_t^c - n_t\ p_t\ (1+\epsilon)(1+C)$.

Action space reduction:

In the scope of this scientific research paper, the action space $\mathscr{A}$ is reduced in order to lower the complexity of the algorithmic trading problem. The reduced action space is composed of only two RL actions which can be mathematically expressed as the following:

$$a_t = Q_t \in \left\{ Q_t^{Long},\ Q_t^{Short} \right\} \tag{15}$$

The first RL action Q_t^{Long} maximises the number of shares owned by the trading agent, by converting as much cash value v_t^c as possible into share value v_t^s. It can be mathematically expressed as follows:

$$Q_t^{Long} = \begin{cases} \left\lfloor \dfrac{v_t^c}{p_t\ (1+C)} \right\rfloor & \text{if } a_{t-1} \neq Q_{t-1}^{Long}, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

The action Q_t^{Long} is always valid as it is obviously included into the original action space $\mathscr{A}$ defined by Eq. (14). As a result of this action, the trading agent owns a number of shares $N_t^{Long} = n_t + Q_t^{Long}$. On the contrary, the second RL action, designated by Q_t^{Short}, converts share value v_t^s into cash value v_t^c, such that the RL agent owns a number of shares equal to $-N_t^{Long}$. This operation can be mathematically expressed as the following:

$$\widehat{Q_t^{Short}} = \begin{cases} -2n_t - \left\lfloor \dfrac{v_t^c}{p_t\ (1+C)} \right\rfloor & \text{if } a_{t-1} \neq Q_{t-1}^{Short}, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

However, the action $\widehat{Q_t^{Short}}$ may violate the lower bound $\underline{Q_t}$ of the action space $\mathscr{A}$ when the price significantly increases over time. Eventually, the second RL action Q_t^{Short} is expressed as follows:

$$Q_t^{Short} = \max\left\{ \widehat{Q_t^{Short}},\ \underline{Q_t} \right\} \tag{18}$$

To conclude this subsection, it should be mentioned that the two reduced RL actions are actually related to the next trading position of the agent, designated as P_{t+1}. Indeed, the first action Q_t^{Long} induces a *long*

trading position because the number of owned shares is positive. On the contrary, the second action Q_t^{Short} always results in a number of shares which is negative, which is generally referred to as a *short* trading position in finance.

3.4.3. RL rewards

For this algorithmic trading problem, a natural choice for the RL rewards is the strategy daily returns. Intuitively, it makes sense to favour positive returns which are an evidence of a profitable strategy. Moreover, such quantity has the advantage of being independent of the number of shares n_t currently owned by the agent. This choice is also motivated by the fact that it allows to avoid a sparse reward setup, which is more complex to deal with. The RL rewards can be mathematically expressed as the following:

$$r_t = \frac{v_{t+1} - v_t}{v_t} \tag{19}$$

3.5. Objective

Objectively assessing the performance of a trading strategy is a tricky task, due to the numerous quantitative and qualitative factors to consider. Indeed, a well-performing trading strategy is not simply expected to generate profit, but also to efficiently mitigate the risk associated with the trading activity. The balance between these two goals varies depending on the trading agent profile and its willingness to take extra risks. Although intuitively convenient, maximising the profit generated by a trading strategy is a necessary but not sufficient objective. Instead, the core objective of a trading strategy is the maximisation of the *Sharpe ratio*, a performance indicator widely used in the fields of finance and algorithmic trading. It is particularly well suited for the performance assessment task as it considers both the generated profit and the risk associated with the trading activity. Mathematically, the Sharpe ratio S_r is expressed as the following:

$$S_r = \frac{\mathbb{E}\left[R_s - R_f\right]}{\sigma_r} = \frac{\mathbb{E}\left[R_s - R_f\right]}{\sqrt{\text{var}\left[R_s - R_f\right]}} \simeq \frac{\mathbb{E}[R_s]}{\sqrt{\text{var}[R_s]}} \tag{20}$$

where:

- R_s is the trading strategy return over a certain time period, modelling its profitability.
- R_f is the risk-free return, the expected return from a totally safe investment (negligible).
- σ_r is the standard deviation of the trading strategy excess return $R_s - R_f$, modelling its riskiness.

In order to compute the Sharpe ratio S_r in practice, the daily returns achieved by the trading strategy are firstly computed using the formula $\rho_t = (v_t - v_{t-1})/v_{t-1}$. Then, the ratio between the returns mean and standard deviation is evaluated. Finally, the annualised Sharpe ratio is obtained by multiplying this value by the square root of the number of trading days in a year (252).

Moreover, a well-performing trading strategy should ideally be capable of achieving acceptable performance on diverse markets presenting very different patterns. For instance, the trading strategy should properly handle both bull and bear markets (respectively strong increasing and decreasing price trends), with different levels of volatility. Therefore, the research paper's core objective is the development of a novel trading strategy based on DRL techniques to maximise the average Sharpe ratio computed on the entire set of existing stock markets.

Despite the fact that the ultimate objective is the maximisation of the Sharpe ratio, the DRL algorithm adopted in this scientific paper actually maximises the expected discounted sum of rewards (daily returns) over an infinite time horizon. This optimisation criterion, which does not exactly corresponds to maximising profits but is very close to that, can in fact be seen as a relaxation of the Sharpe ratio criterion. A future interesting research direction would be to narrow the gap between these two objectives.

4. Deep reinforcement learning algorithm design

In this section, a novel DRL algorithm is designed to solve the algorithmic trading problem previously introduced. The resulting trading strategy, denominated the Trading Deep Q-Network algorithm (TDQN), is inspired from the successful DQN algorithm presented in Mnih et al. (2013) and is significantly adapted to the specific decision-making problem at hand. Concerning the training of the RL agent, artificial trajectories are generated from a limited set of stock market historical data.

4.1. Deep Q-Network algorithm

The Deep Q-Network algorithm, generally referred to as DQN, is a DRL algorithm capable of successfully learning control policies from high-dimensional sensory inputs. It is in a way the successor of the popular Q-learning algorithm introduced in Watkins and Dayan (1992). This DRL algorithm is said to be *model-free*, meaning that a complete model of the environment is not required and that trajectories are sufficient. Belonging to the *Q-learning* family of algorithms, it is based on the learning of an approximation of the state-action value function, which is represented by a DNN. In such context, learning the Q-function amounts to learning the parameters θ of this DNN. Finally, the DQN algorithm is said to be *off-policy* as it exploits in batch mode previous experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ collected at any point during training.

For the sake of brevity, the DQN algorithm is illustrated in Fig. 3, but is not extensively presented in this paper. Besides the original publications (Mnih et al., 2013; Mnih et al., 2015), there exists a great scientific literature around this algorithm, see for instance van Hasselt, Guez, and Silver (2015), Wang, de Freitas, and Lanctot (2015), Schaul, Quan, Antonoglou, and Silver (2016), Bellemare, Dabney, and Munos (2017), Fortunato et al. (2018) and Hessel et al. (2017). Concerning DL techniques, interesting resources are LeCun, Bengio, and Hinton (2015), Goodfellow, Bengio, and Courville (2015) and Goodfellow, Bengio, and Courville (2016). For more information about RL, the reader can refer to the following textbooks and surveys: Sutton and Barto (2018), Szepesvari (2010), Busoniu, Babuska, De Schutter, and Ernst (2010), Arulkumaran, Deisenroth, Brundage, and Bharath (2017) and Shao, Tang, Zhu, Li, and Zhao (2019).

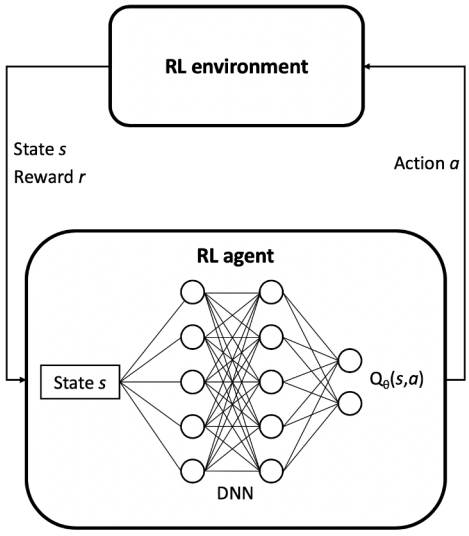


Fig. 3. Illustration of the DQN algorithm.

4.2. Artificial trajectories generation

In the scope of the algorithmic trading problem, a complete model of the environment $\mathscr{E}$ is not available. The training of the TDQN algorithm is entirely based on the generation of artificial trajectories from a limited set of stock market historical daily OHLCV data. A trajectory τ is defined as a sequence of observations $o_t \in \mathscr{O}$, actions $a_t \in \mathscr{A}$ and rewards r_t from an RL agent for a certain number T of trading time steps t:

$$\tau = (\{o_0, a_0, r_0\}, \{o_1, a_1, r_1\}, \ldots, \{o_T, a_T, r_T\})$$

Initially, although the environment $\mathscr{E}$ is unknown, one disposes of a single real trajectory, corresponding to the historical behaviour of the stock market, i.e. the particular case of the RL agent being inactive. This original trajectory is composed of the historical prices and volumes together with long actions executed by the RL agent with no money at its disposal, to represent the fact that no shares are actually traded. For this algorithmic trading problem, new fictive trajectories are then artificially generated from this unique true trajectory to simulate interactions with the environment $\mathscr{E}$. The historical stock market behaviour is simply considered unaffected by the new actions performed by the trading agent. The artificial trajectories generated are simply composed of the sequence of historical real observations associated with various sequences of trading actions from the RL agent. For such practice to be scientifically acceptable and lead to realistic simulations, the trading agent should not be able to influence the stock market behaviour. This assumption generally holds when the number of shares traded by the trading agent is low with respect to the liquidity of the stock market.

In addition to the generation of artificial trajectories just described, a trick is employed to slightly improve the exploration of the RL agent. It relies on the fact that the reduced action space $\mathscr{A}$ is composed of only two actions: long (Q_t^{Long}) and short (Q_t^{Short}). At each trading time step t, the chosen action a_t is executed on the trading environment $\mathscr{E}$ and the opposite action a_t^- is executed on a copy of this environment $\mathscr{E}^-$. Although this trick does not completely solve the challenging exploration/exploitation trade-off, it enables the RL agent to continuously explore at a small extra computational cost.

4.3. Diverse modifications and improvements

The DQN algorithm was chosen as starting point for the novel DRL trading strategy developed, but was significantly adapted to the specific algorithmic trading decision-making problem at hand. The diverse modifications and improvements, which are mainly based on the numerous simulations performed, are summarised hereafter:

- **Deep neural network architecture:** The first difference with respect to the classical DQN algorithm is the architecture of the DNN approximating the action-value function $Q(s,a)$. Due to the different nature of the input (time-series instead of raw images), the convolutional neural network (CNN) has been replaced by a classical feedforward DNN with some leaky rectified linear unit (Leaky ReLU) activation functions.
- **Double DQN:** The DQN algorithm suffers from substantial overestimations, this overoptimism harming the algorithm performance. In order to reduce the impact of this undesired phenomenon, the article van Hasselt et al. (2015) presents the double DQN algorithm which is based on the decomposition of the target max operation into both action selection and action evaluation.
- **ADAM optimiser:** The classical DQN algorithm implements the RMSProp optimiser. However, the ADAM optimiser, introduced in Kingma and Ba (2015), experimentally proves to improve both the training stability and the convergence speed of the DRL algorithm.
- **Huber loss:** While the classical DQN algorithm implements a mean squared error (MSE) loss, the Huber loss experimentally improves the stability of the training phase. Such observation is explained by the fact that the MSE loss significantly penalises large errors, which is generally desired but has a negative side-effect for the DQN algorithm because the DNN is supposed to predict values that depend on its own input. This DNN should not radically change in a single training update because this would also lead to a significant change in the target, which could actually result in a larger error. Ideally, the update of the DNN should be performed in a slower and more stable manner. On the other hand, the mean absolute error (MAE) has the drawback of not being differentiable at 0. A good trade-off between these two losses is the Huber loss H (see Fig. 4):

$$H(x) = \begin{cases} \frac{1}{2}x^2 & \text{if } |x| \leqslant 1, \\ |x| - \frac{1}{2} & \text{otherwise.} \end{cases} \tag{21}$$

- **Gradient clipping:** The gradient clipping technique is implemented in the TDQN algorithm to solve the gradient exploding problem which induces significant instabilities during the training of the DNN.
- **Xavier initialisation:** While the classical DQN algorithm simply initialises the DNN weights randomly, the Xavier initialisation is implemented to improve the algorithm convergence. The idea is to set the initial weights so that the gradients variance remains constant across the DNN layers.
- **Batch normalisation layers:** This DL technique, introduced by Ioffe and Szegedy (2015), consists in normalising the input layer by adjusting and scaling the activation functions. It brings many benefits including a faster and more robust training phase as well as an improved generalisation.
- **Regularisation techniques:** Because a strong tendency to overfit was observed during the first experiments with the DRL trading strategy, three regularisation techniques are implemented: *Dropout*, *L2 regularisation* and *Early Stopping*.
- **Preprocessing and normalisation:** The training loop of the TDQN algorithm is preceded by both a preprocessing and a normalisation operation of the RL observations o_t. Firstly, because the high-frequency noise present in the trading data was experimentally observed to lower the algorithm generalisation, a low-pass filtering operation is executed. However, such a preprocessing operation has a cost as it modifies or even destroys some potentially useful trading patterns and introduces a non-negligible lag. Secondly, the resulting data are transformed in order to convey more meaningful information about market movements. Typically, the daily evolution of

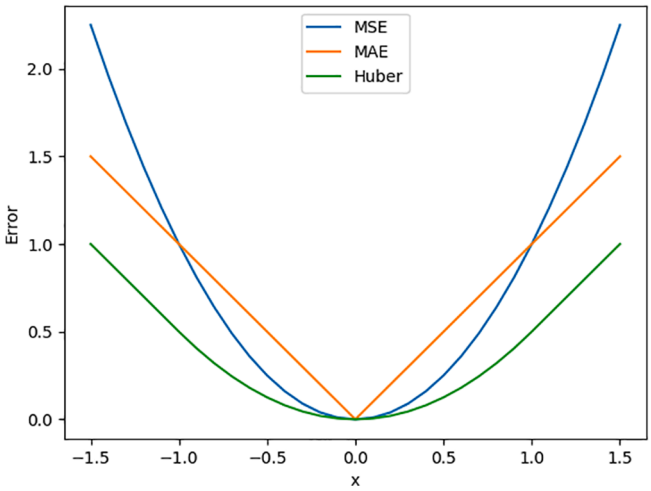


Fig. 4. Comparison of the MSE, MAE and Huber losses.

prices is considered rather than the raw prices. Thirdly, the remaining data are normalised.

- **Data augmentation techniques:** A key challenge of this algorithmic trading problem is the limited amount of available data, which are in addition generally of poor quality. As a counter to this major problem, several data augmentation techniques are implemented: signal shifting, signal filtering and artificial noise addition. The application of such data augmentation techniques will artificially generate new trading data which are slightly different but which result in the same financial phenomena.

Finally, the algorithm underneath the TDQN trading strategy is depicted in detail in Algorithm 1.

Algorithm 1. TDQN algorithm

Initialise the experience replay memory M of capacity C.
Initialise the main DNN weights θ (Xavier initialisation).
Initialise the target DNN weights $\theta^{-} = \theta$.
for episode $= 1$ **to** N **do**
  Acquire the initial observation o_1 from the environment $\mathscr{E}$ and preprocess it.
  for t $= 1$ **to** T **do**
    With probability ϵ, select a random action a_t from $\mathscr{A}$.
    Otherwise, select $a_t = \text{argmax}_{a \in \mathscr{A}} Q(o_t, a; \theta)$.
    Copy the environment $\mathscr{E}^{-} = \mathscr{E}$.
    Interact with the environment $\mathscr{E}$ (action a_t) and get the new observation o_{t+1} and reward r_t.
    Perform the same operation on $\mathscr{E}^{-}$ with the opposite action a_t^{-}, getting o_{t+1}^{-} and r_t^{-}.
    Preprocess both new observations o_{t+1} and o_{t+1}^{-}.
    Store both experiences $e_t = (o_t, a_t, r_t, o_{t+1})$ and $e_t^{-} = (o_t, a_t^{-}, r_t^{-}, o_{t+1}^{-})$ in M.
    if t % T$' = 0$ **then**
      Randomly sample from M a minibatch of N_e experiences $e_i = (o_i, a_i, r_i, o_{i+1})$.
      Set $y_i = \begin{cases} r_i & \text{if the state } s_{i+1} \text{ is terminal,} \\ r_i + \gamma\, Q(o_{i+1}, \text{argmax}_{a \in \mathscr{A}} Q(o_{i+1}, a; \theta); \theta^{-}) & \text{otherwise.} \end{cases}$
      Compute and clip the gradients based on the Huber loss $H(y_i,\ Q(o_i, a_i; \theta))$.
      Optimise the main DNN parameters θ based on these clipped gradients.
      Update the target DNN parameters $\theta^{-} = \theta$ every N^{-} steps.
    end if
    Anneal the ϵ-Greedy exploration parameter ϵ.
  end for
end for

5. Performance assessment

An accurate performance evaluation approach is capital in order to produce meaningful results. As previously hinted, this procedure is all the more critical because there has been a real lack of a proper performance assessment methodology in the algorithmic trading field. In this section, a novel, more reliable methodology is presented to objectively assess the performance of algorithmic trading strategies, including the TDQN algorithm.

5.1. Testbench

In the literature, the performance of a trading strategy is generally assessed on a single instrument (stock market or others) for a certain period of time. Nevertheless, the analysis resulting from such a basic approach should not be entirely trusted, as the trading data could have been specifically selected so that a trading strategy looks profitable, even though it is not the case in general. To eliminate such bias, the performance should ideally be assessed on multiple instruments presenting diverse patterns. Aiming to produce trustful conclusions, this research paper proposes a testbench composed of 30 stocks presenting diverse characteristics (sectors, regions, volatility, liquidity, etc.). The testbench is depicted in Table 2. To avoid any confusion, the official reference for each stock (ticker) is specified in parentheses. To avoid any ambiguities concerning the training and evaluation protocols, it should be mentioned that a new trading strategy is trained for each stock included in the testbench. Nevertheless, for the sake of generality, all the algorithm hyperparameters remain unchanged over the entire testbench.

Regarding the trading horizon, the eight years preceding the publication year of the research paper are selected to be representative of the current market conditions. Such a short-time period could be criticised because it may be too limited to be representative of the entire set of financial phenomena. For instance, the financial crisis of 2008 is rejected, even though it could be interesting to assess the robustness of trading strategies with respect to such an extraordinary event. However, this choice was motivated by the fact that a shorter trading horizon is less likely to contain significant market regime shifts which would seriously harm the training stability of the trading strategies. Finally, the trading horizon of eight years is divided into both training and test sets as follows:

- **Training set:** 01/01/2012 → 31/12/2017.
- **Test set:** 01/01/2018 → 31/12/2019.

A validation set is also considered as a subset of the training set for the tuning of the numerous TDQN algorithm hyperparameters. Note that the RL policy DNN parameters θ are fixed during the execution of the trading strategy on the entire test set, meaning that the new experiences acquired are not valued for extra training. Nevertheless, such practice constitutes an interesting future research direction.

To end this subsection, it should be noted that the proposed testbench could be improved thanks to even more diversification. The obvious addition would be to include more stocks with different financial situations and properties. Another interesting addition would be to consider different training/testing time periods while excluding the

Table 2
Performance assessment testbench.

Sector	Region		
	American	*European*	*Asian*
Trading index	Dow Jones (DIA) S&P 500 (SPY) NASDAQ (QQQ)	FTSE 100 (EZU)	Nikkei 225 (EWJ)
Technology	Apple (AAPL) Google (GOOGL) Amazon (AMZN) Facebook (FB) Microsoft (MSFT) Twitter (TWTR)	Nokia (NOK) Philips (PHIA.AS) Siemens (SIE.DE)	Sony (6758.T) Baidu (BIDU) Tencent (0700.HK) Alibaba (BABA)
Financial services	JPMorgan Chase (JPM)	HSBC (HSBC)	CCB (0939.HK)
Energy	ExxonMobil (XOM)	Shell (RDSA.AS)	PetroChina (PTR)
Automotive	Tesla (TSLA)	Volkswagen (VOW3.DE)	Toyota (7203.T)
Food	Coca Cola (KO)	AB InBev (ABI.BR)	Kirin (2503.T)

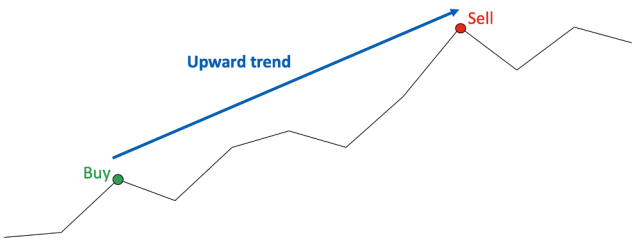


Fig. 5. Illustration of a typical trend following trading strategy.

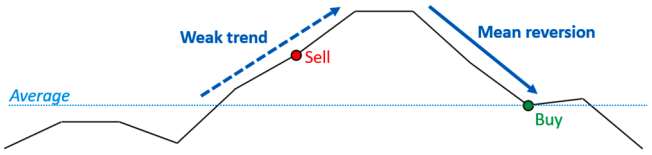


Fig. 6. Illustration of a typical mean reversion trading strategy.

significant market regime shifts. Nevertheless, this last idea was discarded in this scientific article due to the important time already required to produce results for the proposed testbench. Indeed, generating the results presented in Section 6, especially Table 6, takes approximately a full month of processing with GPU hardware acceleration.

5.2. Benchmark trading strategies

In order to properly assess the strengths and weaknesses of the TDQN algorithm, some benchmark algorithmic trading strategies were selected for comparison purposes. Only the classical trading strategies commonly used in practice were considered, excluding for instance strategies based on DL techniques or other advanced approaches. Despite the fact that the TDQN algorithm is an active trading strategy, both passive and active strategies are taken into consideration. For the sake of fairness, the strategies share the same input and output spaces presented in Section 3.4.2 ($\mathscr{O}$ and $\mathscr{A}$). The following list summarises the benchmark strategies selected:

- Buy and hold (B&H).
- Sell and hold (S&H).
- Trend following with moving averages (TF).
- Mean reversion with moving averages (MR).

For the sake of brevity, a detailed description of each strategy is not provided in this research paper. The reader can refer to Chan, 2009; Chan, 2013 or Narang, 2009 for more information. The first two benchmark trading strategies (B&H and S&H) are said to be passive, as there are no changes in trading position over the trading horizon. On the contrary, the other two benchmark strategies (TF and MR) are active trading strategies, issuing multiple changes in trading positions over the trading horizon. On the one hand, a trend following strategy is concerned with the identification and the follow-up of significant market trends, as depicted in Fig. 5. On the other hand, a mean reversion strategy, illustrated in Fig. 6, is based on the tendency of a stock market to get back to its previous average price in the absence of clear trends. By design, a trend following strategy generally makes a profit when a mean reversion strategy does not, the opposite being true as well. This is due to the fact that these two families of trading strategies adopt opposite positions: a mean reversion strategy always denies and goes against the trends while a trend following strategy follows the movements.

5.3. Quantitative performance assessment

The quantitative performance assessment consists in defining one performance indicator or more to numerically quantify the performance of an algorithmic trading strategy. Because the core objective of a trading strategy is to be profitable, its performance should be linked to the amount of money earned. However, such reasoning omits to consider the risk associated with the trading activity which should be efficiently mitigated. Generally, a trading strategy achieving a small but stable profit is preferred to a trading strategy achieving a huge profit in a very unstable way after suffering from multiple losses. It eventually depends on the investor profile and the willingness to take extra risks to potentially earn more.

Multiple performance indicators were selected to accurately assess the performance of a trading strategy. As previously introduced in Section 3.5, the most important one is certainly the Sharpe ratio. This performance indicator, widely used in the field of algorithmic trading, is particularly informative as it combines both profitability and risk. Besides the Sharpe ratio, this research paper considers multiple other performance indicators to provide extra insights. Table 3 presents the entire set of performance indicators employed to quantify the performance of a trading strategy.

Complementarily to the computation of these numerous performance indicators, it is interesting to graphically represent the trading strategy behaviour. Plotting both the stock market price p_t and portfolio value v_t evolutions together with the trading actions a_t issued by the trading strategy seems appropriate to accurately analyse the trading policy. Moreover, such visualisation could also provide extra insights about the performance, the strengths and weaknesses of the strategy analysed.

6. Results and discussion

In this section, the TDQN trading strategy is evaluated following the performance assessment methodology previously described. Firstly, a detailed analysis is performed for both a case that give good results and a case for which the results were mitigated. This highlights the strengths, weaknesses and limitations of the TDQN algorithm. Secondly, the performance achieved by the DRL trading strategy on the entire testbench is summarised and analysed. Finally, some additional discussions about the discount factor parameter, the trading costs influence and the main challenges faced by the TDQN algorithm are provided. The experimental code supporting the results presented is publicly available at the

Table 3
Quantitative performance assessment indicators.

Performance indicator	Description
Sharpe ratio	Return of the trading activity compared to its riskiness.
Profit & loss	Money gained or lost at the end of the trading activity.
Annualised return	Annualised return generated during the trading activity.
Annualised volatility	Modelling of the risk associated with the trading activity.
Profitability ratio	Percentage of winning trades made during the trading activity.
Profit and loss ratio	Ratio between the trading activity trades average profit and loss.
Sortino ratio	Similar to the Sharpe ratio with the negative risk penalised only.
Maximum drawdown	Largest loss from a peak to a trough during the trading activity.
Maximum drawdown duration	Time duration of the trading activity maximum drawdown.

Table 4
Performance assessment for the Apple stock.

Performance indicator	B&H	S&H	TF	MR	TDQN
Sharpe ratio	1.239	−1.593	1.178	−0.609	1.484
Profit & loss [$]	79823	−80023	68738	−34630	100288
Annualised return [%]	28.86	−100.00	25.97	−19.09	32.81
Annualised volatility [%]	26.62	44.39	24.86	28.33	25.69
Profitability ratio [%]	100	0.00	42.31	56.67	52.17
Profit and loss ratio	∞	0.00	3.182	0.492	2.958
Sortino ratio	1.558	−2.203	1.802	−0.812	1.841
Max drawdown [%]	38.51	82.48	14.89	51.12	17.31
Max drawdown duration [days]	62	250	20	204	25

following link: https://github.com/ThibautTheate/An-Application-of-Deep-Reinforcement-Learning-to-Algorithmic-Trading.

6.1. Good results – Apple stock

The first detailed analysis concerns the execution of the TDQN trading strategy on the Apple stock, resulting in promising results. Similar to many DRL algorithms, the TDQN algorithm is subject to a non-negligible variance. Multiple training experiments with the exact same initial conditions will inevitably lead to slightly different trading strategies of varying performance. As a consequence, both a typical run of the TDQN algorithm and its expected performance are presented hereafter.

Typical run: Firstly, Table 4 presents the performance achieved by each trading strategy considered, the initial amount of money being equal to $100,000. The TDQN algorithm achieves good results from both an earnings and a risk mitigation point of view, clearly outperforming all the benchmark active and passive trading strategies. Secondly, Fig. 7 plots both the stock market price p_t and RL agent portfolio value v_t evolutions, together with the actions a_t outputted by the TDQN algorithm. It can be observed that the DRL trading strategy is capable of accurately detecting and benefiting from major trends, while being more hesitant during market behavioural shifts when the volatility increases. It can also be seen that the trading agent generally lags slightly behind the market trends, meaning that the TDQN algorithm learned to be more reactive than proactive for this particular stock. This behaviour is expected with such a limited observation space $\mathcal{O}$ not including the reasons for the future market directions (new product announcement, financial report, macroeconomics, etc.). However, this does not mean that the policies learned are purely reactive. Indeed, it was observed that the RL agent may decide to adapt its trading position before a trend inversion by noticing a sudden increase in volatility, therefore anticipating and being proactive.

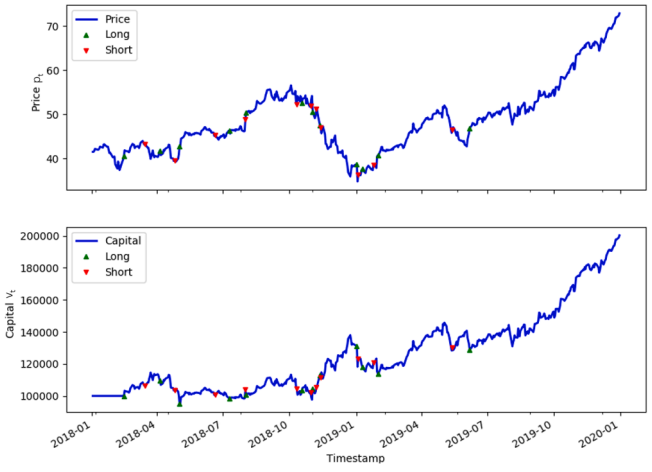


Fig. 7. TDQN algorithm execution for the Apple stock (test set).

Expected performance: In order to estimate the expected performance as well as the variance of the TDQN algorithm, the same RL trading agent is trained multiple times. Fig. 8 plots the averaged (over 50 iterations) performance of the TDQN algorithm for both the training and test sets with respect to the number of training episodes. This expected performance is comparable to the performance achieved during the typical run of the algorithm. It can also be noticed that the overfitting tendency of the RL agent seems to be properly handled for this specific market. Please note that the test set performance being temporarily superior to the training set performance is not a mistake. It simply indicates an easier to trade and more profitable market for the test set

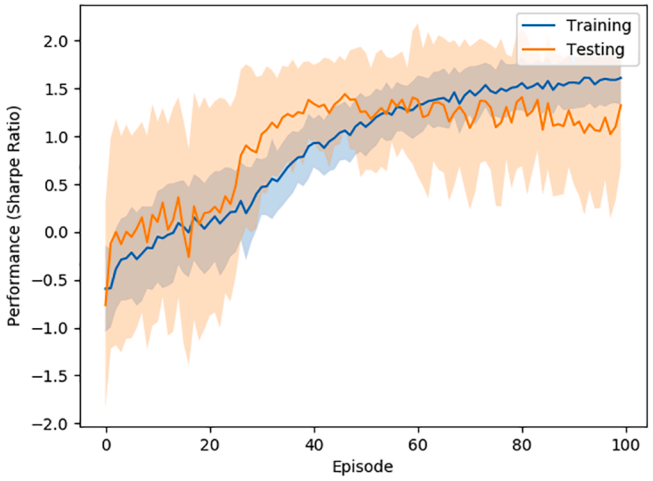


Fig. 8. TDQN algorithm expected performance for the Apple stock.

trading period for the Apple stock. This example perfectly illustrates a major difficulty of the algorithmic trading problem: the training and test sets do not share the same distributions. Indeed, the distribution of the daily returns is continuously changing, which complicates both the training of the DRL trading strategy and its performance evaluation.

6.2. Mitigated results – Tesla stock

The same detailed analysis is performed on the Tesla stock, which presents very different characteristics compared to the Apple stock, such as a pronounced volatility. In contrast to the promising performance achieved on the previous stock, this case was specifically selected to highlight the limitations of the TDQN algorithm.

Typical run: Similar to the previous analysis, Table 5 presents the performance achieved by every trading strategies considered, the initial amount of money being equal to $100,000. The mitigated results achieved by the benchmark active strategies suggest that the Tesla stock is quite difficult to trade, which is partly due to its significant volatility. Even though the TDQN algorithm achieves a positive Sharpe ratio,

Table 5
Performance assessment for the Tesla stock.

Performance indicator	B&H	S&H	TF	MR	TDQN
Sharpe ratio	0.508	−0.154	−0.987	0.358	0.261
Profit & loss [$]	29847	−29847	−73301	8600	98
Annualised return [%]	24.11	−7.38	−100.00	19.02	12.80
Annualised volatility [%]	53.14	46.11	52.70	58.05	52.09
Profitability ratio [%]	100	0.00	34.38	67.65	38.18
Profit and loss ratio	∞	0.00	0.534	0.496	1.621
Sortino ratio	0.741	−0.205	−1.229	0.539	0.359
Max drawdown [%]	52.83	54.09	79.91	65.31	58.95
Max drawdown duration [days]	205	144	229	159	331

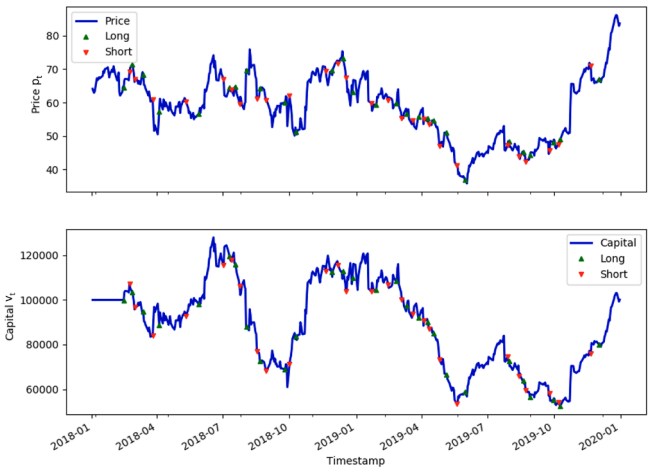


Fig. 9. TDQN algorithm execution for the Tesla stock (test set).

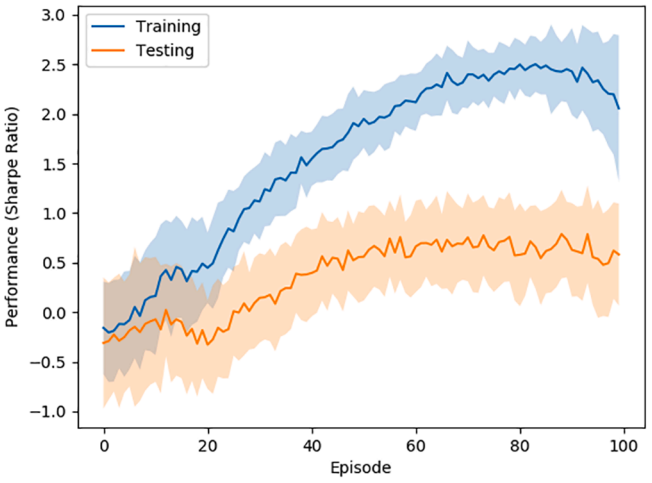


Fig. 10. TDQN algorithm expected performance for the Tesla stock.

almost no profit is generated. Moreover, the risk level associated with this trading activity cannot really be considered acceptable. For instance, the maximum drawdown duration is particularly large, which would result in a stressful situation for the operator responsible for the trading strategy. Fig. 9, which plots both the stock market price p_t and RL agent portfolio value v_t evolutions together with the actions a_t outputted by the TDQN algorithm, confirms this observation. Moreover, it can be clearly observed that the pronounced volatility of the Tesla stock induces a higher trading frequency (changes in trading positions, which correspond to the situation where $a_t \neq a_{t-1}$) despite the non-negligible trading costs, which increases even more the riskiness of the DRL trading strategy.

Expected performance: Fig. 10 plots the expected performance of the TDQN algorithm for both the training and test sets as a function of the number of training episodes (over 50 iterations). It can be directly noticed that this expected performance is significantly better than the performance achieved by the typical run previously analysed, which can therefore be considered as not really representative of the average behaviour. This highlights a key limitation of the TDQN algorithm: the substantial variance which may result in selecting poorly performing policies compared to the expected performance. The significantly higher performance achieved on the training set also suggests that the DRL algorithm is subject to overfitting in this specific case, despite the multiple regularisation techniques implemented. This overfitting phenomenon can be partially explained by the observation space $\mathscr{O}$ which is too limited to efficiently apprehend the Tesla stock. Even though this overfitting phenomenon does not seem to be too harmful in this particular case, it may lead to poor performance for other stocks.

Table 6
Performance assessment for the entire testbench.

Stock	Sharpe Ratio				
	B&H	*S&H*	*TF*	*MR*	*TDQN*
Dow Jones (DIA)	0.684	−0.636	−0.325	−0.214	0.684
S&P 500 (SPY)	0.834	−0.833	−0.309	−0.376	0.834
NASDAQ 100 (QQQ)	0.845	−0.806	0.264	0.060	0.845
FTSE 100 (EZU)	0.088	0.026	−0.404	−0.030	0.103
Nikkei 225 (EWJ)	0.128	−0.025	−1.649	0.418	0.019
Google (GOOGL)	0.570	−0.370	0.125	0.555	0.227
Apple (AAPL)	1.239	−1.593	1.178	−0.609	1.424
Facebook (FB)	0.371	−0.078	0.248	−0.168	0.151
Amazon (AMZN)	0.559	−0.187	0.161	−1.193	0.419
Microsoft (MSFT)	1.364	−1.390	−0.041	−0.416	0.987
Twitter (TWTR)	0.189	0.314	−0.271	−0.422	0.238
Nokia (NOK)	−0.408	0.565	1.088	1.314	−0.094
Philips (PHIA.AS)	1.062	−0.672	−0.167	−0.599	0.675
Siemens (SIE.DE)	0.399	−0.265	0.525	0.526	0.426
Baidu (BIDU)	−0.699	0.866	−1.209	0.167	0.080
Alibaba (BABA)	0.357	−0.139	−0.068	0.293	0.021
Tencent (0700.HK)	−0.013	0.309	0.179	−0.466	−0.198
Sony (6758.T)	0.794	−0.655	−0.352	0.415	0.424
JPMorgan Chase (JPM)	0.713	−0.743	−1.325	−0.004	0.722
HSBC (HSBC)	−0.518	0.725	−1.061	0.447	0.011
CCB (0939.HK)	0.026	0.165	−1.163	−0.388	0.202
ExxonMobil (XOM)	0.055	0.132	−0.386	−0.673	0.098
Shell (RDSA.AS)	0.488	−0.238	−0.043	0.742	0.425
PetroChina (PTR)	−0.376	0.514	−0.821	−0.238	0.156
Tesla (TSLA)	0.508	−0.154	−0.987	0.358	0.621
Volkswagen (VOW3.DE)	0.384	−0.208	−0.361	0.601	0.216
Toyota (7203.T)	0.352	−0.242	−1.108	−0.378	0.304
Coca Cola (KO)	1.031	−0.871	−0.236	−0.394	1.068
AB InBev (ABI.BR)	−0.058	0.275	0.036	−1.313	0.187
Kirin (2503.T)	0.106	0.156	−1.441	0.313	0.852
Average	0.369	−0.202	−0.331	−0.056	0.404

6.3. *Global results – Testbench*

As previously suggested in this research paper, the TDQN algorithm is evaluated on the testbench introduced in Section 5.1, in order to draw more robust and trustful conclusions. Table 6 presents the expected Sharpe ratio achieved by both the TDQN and benchmark trading strategies on the entire set of stocks included in this testbench.

Regarding the performance achieved by the benchmark trading strategies, it is important to differentiate the passive strategies (B&H and S&H) from the active ones (TF and MR). Indeed, this second family of trading strategies has more potential at the cost of an extra non-negligible risk: continuous speculation. Because the stock markets were mostly *bullish* (price p_t mainly increasing over time) with some instabilities during the test set trading period, it is not surprising to see the buy and hold strategy outperforming the other benchmark trading strategies. In fact, neither the trend following nor the mean reversion strategy managed to generate satisfying results on average on this testbench. It clearly indicates that there is a major difficulty to actively trade in such market conditions. This poorer performance can also be explained by the fact that such strategies are generally well suited to exploit specific financial patterns, but they lack versatility and thus often fail to achieve good average performance on a large set of stocks presenting diverse characteristics. Moreover, such strategies are generally more impacted by the trading costs due their higher trading frequency (for relatively short moving averages durations, as it is the case in this research paper).

Concerning the innovative trading strategy, the TDQN algorithm achieves promising results on the testbench, outperforming the benchmark active trading strategies on average. Nevertheless, the DRL trading

strategy only barely surpasses the buy and hold strategy on these particular bullish markets which are so favourable to this simple passive strategy. Interestingly, it should be noted that the performance of the TDQN algorithm is identical or very close to the performance of the passive trading strategies (B&H and S&H) for multiple stocks. This is explained by the fact that the DRL strategy efficiently learns to tend toward a passive trading strategy when the uncertainty associated to active trading increases. It should also be emphasized that the TDQN algorithm is neither a trend following nor a mean reversion trading strategy as both financial patterns can be efficiently handled in practice. Thus, the main advantage of the DRL trading strategy is certainly its versatility and its ability to efficiently handle various markets presenting diverse characteristics.

6.4. Discount factor discussion

As previously explained in Section 3.4, the discount factor γ is concerned with the importance of future rewards. In the scope of this algorithmic trading problem, the proper tuning of this parameter is not trivial due to the significant uncertainty of the future. On the one hand, the desired trading policy should be long-term oriented ($\gamma \rightarrow 1$), in order to avoid a too high trading frequency and being exposed to considerable trading costs. On the other hand, it would be unwise to place too much importance on a stock market future which is particularly uncertain ($\gamma \rightarrow 0$). Therefore, a trade-off intuitively exists for the discount factor parameter.

This reasoning is validated by the multiple experiments performed to tune the parameter γ. Indeed, it was observed that there is an optimal value for the discount factor, which is neither too small nor too large. Additionally, these experiments highlighted the hidden link between the discount factor and the trading frequency, due to the trading costs. From the point of view of the RL agent, these costs represent an obstacle to overcome for a change in trading position to occur, due to the immediate reduced (and often negative) reward received. It models the fact that the trading agent should be sufficiently confident about the future in order to overcome the extra risk associated with the trading costs. The discount factor determining the importance assigned to the future, a small value for the parameter γ will inevitably reduce the tendency of the RL agent to change its trading position, which decreases the trading frequency of the TDQN algorithm.

6.5. Trading costs discussion

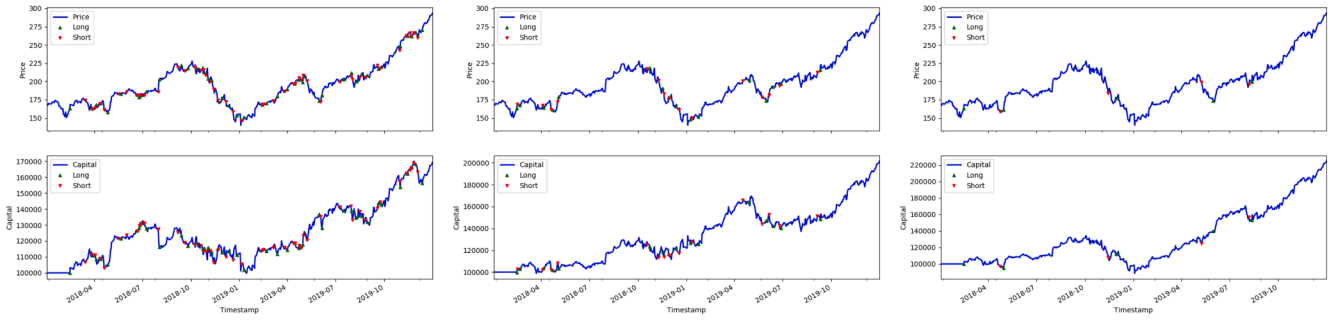
The analysis of the trading costs influence on a trading strategy behaviour and performance is capital, due to the fact that such costs represent an extra risk to mitigate. A major motivation for studying DRL solutions rather than pure prediction techniques that could also be based on DL architectures is related to the trading costs. As previously explained in Section 3, the RL formalism enables the consideration of these additional costs directly into the decision-making process. The optimal policy is learned according to the trading costs value. On the contrary, a purely predictive approach would only output predictions about the future market direction or prices without any indications regarding an appropriate trading strategy taking into account the trading costs. Although this last approach offers more flexibility and could certainly lead to well-performing trading strategies, it is less efficient by design.

In order to illustrate the ability of the TDQN algorithm to automatically and efficiently adapt to different trading costs, Fig. 11 presents the behaviour of the DRL trading strategy for three different costs values, all other parameters remaining unchanged. It can clearly be observed that the TDQN algorithm effectively reduces its trading frequency when the trading costs increase, as expected. When these costs become too high, the DRL algorithm simply stops actively trading and adopts a passive approach (buy and hold or sell and hold strategies).

6.6. Core challenges

Nowadays, the main DRL solutions successfully applied to real-life problems concern specific environments with particular properties such as games (see e.g. the famous AlphaGo algorithm developed by Google Deepmind Silver et al., 2016). In this research paper, an entirely different environment characterised by a significant complexity and a considerable uncertainty is studied with the algorithmic trading problem. Obviously, multiple challenges were faced during the research around the TDQN algorithm, the major ones being summarised hereafter.

Firstly, the extremely poor observability of the trading environment is a characteristic that significantly limits the performance of the TDQN algorithm. Indeed, the amount of information at the disposal of the RL agent is really not sufficient to accurately explain the financial phenomena occurring during training, which is necessary to efficiently learn to trade. Secondly, although the distribution of the daily returns is continuously changing, the past is required to be representative enough of the future for the TDQN algorithm to achieve good results. This makes the DRL trading strategy particularly sensitive to significant market regime shifts. Thirdly, the TDQN algorithm overfitting tendency has to be properly handled in order to obtain a reliable trading strategy. As suggested in Zhang, Vinyals, Munos, and Bengio (2018), more rigorous evaluation protocols are required in RL due to the strong tendency of common DRL techniques to overfit. More research on this particular topic is required for DRL techniques to fit a broader range of real-life applications. Lastly, the substantial variance of DRL algorithms such as DQN makes it rather difficult to successfully apply these algorithms to certain problems, especially when the training and test sets differ considerably. This is a key limitation of the TDQN algorithm which was previously highlighted for the Tesla stock.



(a) Trading costs: 0% (b) Trading costs: 0.1% (c) Trading costs: 0.2%

Fig. 11. Impact of the trading costs on the TDQN algorithm, for the Apple stock.

7. Conclusion

This scientific research paper presents the Trading Deep Q-Network algorithm (TDQN), a deep reinforcement learning (DRL) solution to the algorithmic trading problem of determining the optimal trading position at any point in time during a trading activity in stock markets. Following a rigorous performance assessment, this innovative trading strategy achieves promising results, surpassing on average the benchmark trading strategies. Moreover, the TDQN algorithm demonstrates multiple benefits compared to more classical approaches, such as an appreciable versatility and a remarkable robustness to diverse trading costs. Additionally, such data-driven approach presents the major advantage of suppressing the complex task of defining explicit rules suited to the particular financial markets considered.

Nevertheless, the performance of the TDQN algorithm could still be improved, from both a generalisation and a reproducibility point of view, to cite a few. Several research directions are suggested to upgrade the DRL solution, such as the use of LSTM layers into the deep neural network which should help to better process the financial time-series data, see e.g. Hausknecht and Stone (2015). Another example is the consideration of the numerous improvements implemented in the Rainbow algorithm, which are detailed in Sutton and Barto (2018), van Hasselt et al. (2015), Wang et al. (2015), Schaul et al. (2016), Bellemare et al. (2017), Fortunato et al. (2018) and Hessel et al. (2017). Another interesting research direction is the comparison of the TDQN algorithm with Policy Optimisation DRL algorithms such as the Proximal Policy Optimisation (PPO – Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) algorithm.

The last major research direction suggested concerns the formalisation of the algorithmic trading problem into a reinforcement learning one. Firstly, the observation space $\mathscr{O}$ should be extended to enhance the observability of the trading environment. Similarly, some constraints about the action space $\mathscr{A}$ could be relaxed in order to enable new trading possibilities. Secondly, advanced RL reward engineering should be performed to narrow the gap between the RL objective and the Sharpe ratio maximisation objective. Finally, an interesting and promising research direction is the consideration of distributions instead of expected values in the TDQN algorithm in order to encompass the notion of risk and to better handle uncertainty.

CRediT authorship contribution statement

Thibaut Théate: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization, Funding acquisition.
Damien Ernst: Supervision, Validation, Resources, Project administration.

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.

Acknowledgements

Thibaut Théate is a Research Fellow of the F.R.S.-FNRS, of which he acknowledges the financial support.

Appendix A. Derivation of action space $\mathscr{A}$

Theorem 1. *The RL action space $\mathscr{A}$ admits an upper bound $\overline{Q_t}$ such that:*

$$\overline{Q_t} = \frac{v_t^c}{p_t\ (1+C)}$$

Proof. The upper bound of the RL action space $\mathscr{A}$ is derived from the fact that the cash value v_t^c has to remain positive over the entire trading horizon (Eq. (9)). Making the hypothesis that $v_t^c \geqslant 0$, the number of shares Q_t traded by the RL agent at time step t has to be set such that $v_{t+1}^c \geqslant 0$ as well. Introducing this condition into Eq. (12) expressing the update of the cash value, the following expression is obtained:

$$v_t^c - Q_t\ p_t - C\ |Q_t|\ p_t \geqslant 0$$

Two cases arise depending on the value of Q_t:

Case of $Q_t < 0$: The previous expression becomes

$v_t^c - Q_t\ p_t + C\ Q_t\ p_t \geqslant 0$

$\Leftrightarrow Q_t \leqslant \frac{v_t^c}{p_t\ (1-C)}$.

The expression on the right side of the inequality is always positive due to the hypothesis that $v_t^c \geqslant 0$. Because Q_t is negative in this case, the condition is always satisfied.

Case of $Q_t \geqslant 0$: The previous expression becomes

$v_t^c - Q_t\ p_t - C\ Q_t\ p_t \geqslant 0$

$\Leftrightarrow Q_t \leqslant \frac{v_t^c}{p_t\ (1+C)}$.

This condition represents the upper bound (positive) of the RL action space $\mathscr{A}$. □

Theorem 2. *The RL action space $\mathscr{A}$ admits a lower bound $\underline{Q_t}$ such that:*

$$\underline{Q_t} = \begin{cases} \dfrac{\Delta_t}{p_t\ \epsilon(1+C)} & \text{if } \Delta_t \geqslant 0 \\ \dfrac{\Delta_t}{p_t\ (2C + \epsilon(1+C))} & \text{if } \Delta_t < 0 \end{cases}$$

with $\Delta_t = -v_t^c - n_t\ p_t\ (1+\epsilon)(1+C)$.

Proof. The lower bound of the RL action space $\mathscr{A}$ is derived from the fact that the cash value v_t^c has to be sufficient to get back to a neutral position

($n_t = 0$) over the entire trading horizon (Eq. 13). Making the hypothesis that this condition is satisfied at time step t, the number of shares Q_t traded by the RL agent should be such that this condition remains true at the next time step $t+1$. Introducing this constraint into Eq. 12, the following inequality is obtained:

$$v_t^c - Q_t\ p_t - C\ |Q_t|\ p_t \geqslant -\left(n_t + Q_t\right)\ p_t\ \left(1+C\right)\left(1+\epsilon\right)$$

Two cases arise depending on the value of Q_t:

Case of $Q_t \geqslant 0$: The previous expression becomes

$$v_t^c - Q_t\ p_t - C\ Q_t\ p_t \geqslant -(n_t + Q_t)\ p_t\ (1+C)(1+\epsilon)$$
$$\Leftrightarrow v_t^c \geqslant -n_t\ p_t\ (1+C)(1+\epsilon) - Q_t\ p_t\ \epsilon\ (1+C)$$
$$\Leftrightarrow Q_t \geqslant \frac{-v_t^c - n_t\ p_t\ (1+C)(1+\epsilon)}{p_t\ \epsilon\ (1+C)}$$

The expression on the right side of the inequality represents the first lower bound for the RL action space $\mathscr{A}$.

Case of $Q_t < 0$: The previous expression becomes

$$v_t^c - Q_t\ p_t + C\ Q_t\ p_t \geqslant -(n_t + Q_t)\ p_t\ (1+C)(1+\epsilon)$$
$$\Leftrightarrow v_t^c \geqslant -n_t\ p_t\ (1+C)(1+\epsilon) - Q_t\ p_t\ (2C + \epsilon + \epsilon C)$$
$$\Leftrightarrow Q_t \geqslant \frac{-v_t^c - n_t\ p_t\ (1+C)(1+\epsilon)}{p_t\ (2C+\epsilon(1+C))}$$

The expression on the right side of the inequality represents the second lower bound for the RL action space $\mathscr{A}$.

Both lower bounds previously derived have the same numerator, which is denoted Δ_t from now on. This quantity represents the difference between the maximum assumed cost to get back to a neutral position at the next time step $t+1$ and the current cash value of the agent v_t^c. The expression tests whether the agent can pay its debt in the worst assumed case or not at the next time step, if nothing is done at the current time step ($Q_t = 0$). Two cases arise depending on the sign of the quantity Δ_t:

Case of $\Delta_t < 0$: The trading agent has no problem paying its debt in the situation previously described. This is always true when the agent owns a positive number of shares ($n_t \geqslant 0$). This is also always true when the agent owns a negative number of shares ($n_t < 0$) and when the price decreases ($p_t < p_{t-1}$) due to the hypothesis that Eq. (13) was verified for time step t. In this case, the most constraining lower bound of the two is the following:

$$\underline{Q_t} = \frac{\Delta_t}{p_t\ (2C + \epsilon(1+C))}$$

Case of $\Delta_t \geqslant 0$: The trading agent may have problem paying its debt in the situation previously described. Following a similar reasoning than for the previous case, the most constraining lower bound of the two is the following:

$$\underline{Q_t} = \frac{\Delta_t}{p_t\ \epsilon(1+C)} \qquad \square$$

References

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

Arévalo, A., Niño, J., Hernández, G., & Sandoval, J. (2016). High-frequency trading strategy based on deep neural networks. In *ICIC*.

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). A brief survey of deep reinforcement learning. CoRR, abs/1708.05866.

Bailey, D. H., Borwein, J. M., de Prado, M. L., & Zhu, Q. J. (2014). Pseudo-mathematics and financial charlatanism: the effects of backtest overfitting on out-of-sample performance. *Notice of the American Mathematical Society*, 458–471.

Bao, W. N., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one, 12*.

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A distributional perspective on reinforcement learning. CoRR, abs/1707.06887.

Bollen, J., Mao, H., & jun Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computer Science, 2*, 1–8.

Boukas, I., Ernst, D., Théate, T., Bolland, A., Huynen, A., Buchwald, M., Wynants, C., & Cornélusse, B. (2020). A deep reinforcement learning framework for continuous intraday market bidding. ArXiv, abs/2004.05940.

Busoniu, L., Babuska, R., De Schutter, B., & Ernst, D. (2010). *Reinforcement learning and dynamic programming using function approximators*. CRC Press.

Carapuço, J., Neves, R. F., & Horta, N. (2018). Reinforcement learning applied to Forex trading. *Applied Soft Computing, 73*, 783–794.

Chan, E. P. (2009). *Quantitative trading: how to build your own algorithmic trading business*. Wiley.

Chan, E. P. (2013). *Algorithmic trading: winning strategies and their rationale*. Wiley.

Dempster, M. A. H., & Leemans, V. (2006). An automated FX trading system using adaptive reinforcement learning. *Expert Systems With Applications, 30*, 543–552.

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*, 653–664.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., & Legg, S. (2018). Noisy networks for exploration. CoRR, abs/1706.10295.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Goodfellow, I. J., Bengio, Y., & Courville, A. C. (2015). Deep learning. *Nature, 521*, 436–444.

Hausknecht, M. J., & Stone, P. (2015). Deep recurrent Q-Learning for partially observable MDPs. CoRR, abs/1507.06527.

Hendershott, T., Jones, C. M., & Menkveld, A. J. (2011). Does algorithmic trading improve liquidity? *Journal of Finance, 66*, 1–33.

Hessel, M., Modayil, J., van Hasselt, H. P., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M. G., & Silver, D. (2017). Rainbow: Combining improvements in deep reinforcement learning. CoRR, abs/1710.02298.

Ioannidis, J. P. A. (2005). Why most published research findings are false. *PLoS Med, 2*, 124.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR, abs/1502.03167.

Jeong, G., & Kim, H. (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.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. CoRR, abs/1412.6980.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*.

Lei, K., Zhang, B., Li, Y., Yang, M., & Shen, Y. (2020). Time-driven Feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading. *Expert Systems With Applications, 140*.

Leinweber, D., & Sisk, J. (2011). Event-driven trading and the "New News". *The Journal of Portfolio Management, 38*, 110–124.

Li, Y. (2017). Deep reinforcement learning: An overview. CoRR, abs/1701.07274.

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.

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

Moody, J. E., & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks, 12*(4), 875–889.

Narang, R. K. (2009). *Inside the black box*. Wiley.

Nuij, W., Milea, V., Hogenboom, F., Frasincar, F., & Kaymak, U. (2014). An automated framework for incorporating news into stock trading strategies. *IEEE Transactions on Knowledge and Data Engineering, 26*, 823–835.

Nuti, G., Mirghaemi, M., Treleaven, P. C., & Yingsaeree, C. (2011). Algorithmic trading. *Computer, 44*, 61–69.

Paiva, F. D., Cardoso, R. T. N., Hanaoka, G. P., & Duarte, W. M. (2019). Decision-making for financial trading: A fusion approach of machine learning and portfolio selection. *Expert Systems With Applications, 115*, 635–655.

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.

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. CoRR, abs/1511.05952.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. CoRR, abs/1707.06347.

Shao, K., Tang, Z., Zhu, Y., Li, N., & Zhao, D. (2019). A survey of deep reinforcement learning in video games. ArXiv, abs/1912.10944.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*, 484–489.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). The MIT Press.

Szepesvari, C. (2010). *Algorithms for reinforcement learning*. Morgan and Claypool Publishers.

Treleaven, P. C., Galas, M., & Lalchand, V. (2013). Algorithmic trading review. *Communications of the ACM, 56*, 76–85.

van Hasselt, H. P., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double Q-Learning. CoRR, abs/1509.06461.

Wang, Z., de Freitas, N., & Lanctot, M. (2015). Dueling network architectures for deep reinforcement learning. CoRR, abs/1511.06581.

Watkins, C. J. C. H., & Dayan, P. (1992). Technical note: Q-Learning. *Machine Learning, 8*, 279–292.

Zhang, C., Vinyals, O., Munos, R., & Bengio, S. (2018). A study on overfitting in deep reinforcement learning. CoRR, abs/1804.06893.