An application of deep reinforcement learning to algorithmic trading

Anas Essounaini
Université de Paris VII - $\mathcal{M}2\mathcal{MO}$
Paris, France
anas.essounaini@centrale.centralelille.fr

Taha-Abderrahman El Ouahabi
ENSAE Paris - $\mathcal{FGR}$
Palaiseau, France
taha.abderrahman.el.ouahabi@ensae.fr

Key Words Quantitative finance, Algorithmic trading, Deep Reinforcement Learning

1 INTRODUCTION

Deep Reinforcement Learning (DRL) is a subfield of Machine Learning that uses deep learning to generalize information fitted from its interactions with the environment. Recently, DRL has gained significant traction due to its state-of-the-art performance in many applications such as NLP, speech recognition.. The paper at hands explores the possibility of generating such performance in an Algorithmic Trading environment.

Using DRL under an algorithmic trading setting poses some significant challenges:

(1) Algorithmic trading is characterized by a significant level of noise, poor observability, and significant stochasticity.
(2) The agent should at each timestep determine whether to short or long the stock, and also the proportion of the portfolio that will be invested in the risky asset.

The paper's main objective is to overcome these challenges. For that, a new trading policy (TDQN) -based on the Deep Q-Learning algorithm- is formulated. Finally, to practically measure the success of the policy, we compare it against some investment strategies:

- **Passive Strategies**: Buy & Hold, Sell & Hold
- **Active Strategies**: Based on technical indicators, Relative Strength Indicator (RSI), Moving Average Convergance Divergence (MACD), Bollinger Band® (BBANDS), Moving Average (SMA)

2 ALGORITHMIC TRADING: A REINFORCEMENT LEARNING SETUP

2.1 A brief overview of algorithmic trading

Algorithmic trading can be defined as the process of making trading decisions in an automatic manner. It has been proven highly beneficial to financial markets, especially in terms of liquidity improvement.

In this study, we present empirical results applied to the *stock markets*. However, such approach can be easily used for other classes of assets such as FOREX, commodity futures, cryptocurrencies ... etc

In this setup, algorithmic trading problem is equivalent to managing a single stock portfolio. Hence, the value of the trading portfolio can be decomposed as follows:

$$\underbrace{v_t}_{\text{value of the trading portfolio}} = \underbrace{v_t^c}_{\text{cash component}} + \underbrace{v_t^s}_{\text{stock component}} \quad (1)$$

The trading agent interacts sequentially with the stock market through an order book containing a set of *bids* and *asks*. In our setup, we will consider a discretized trading activity on a *daily* basis with *only one decision per day*.

In real life, the trader makes at time t an investment decision a_t given information to which he has access i_t. In other terms, trading activity can be quantified as a policy $\pi(a_t|i_t)$ (probability distribution).

To be more concise, here is the trading activity procedure in a nutshell:

(1) Update available market information i_t
(2) Execute trading policy $\pi(a_t|i_t)$ to get action a_t
(3) Perform a_t
(4) $t \longrightarrow t+1$, and loop back to step 1

2.2 Reinforcement learning paradigm

2.2.1 *General framework.*

Reinforcement learning can be described as learning:

- by *"trial and error"*
- to behave in an unknown, shochastic environement by maximizing some real-valued *reward* signal

A learning agent sequentially interacts with its environment by performing actions. Each action provides an instantaneous reward and leads to an evolution of the agent's state. More precisely, the agent's goal is to find a : proper strategy of choosing an action in a given state, called *policy* that maximizes the cumulative reward, *value function* in each state.

The Reinforcement Learning modelling paradigm is based on *Markov Decision Processes*defined as follows.

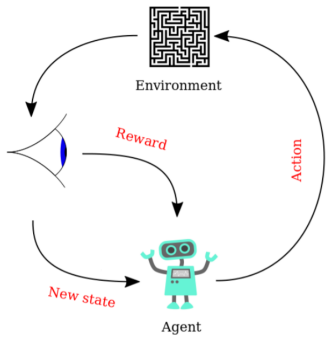
Definition 2.1 (Markov Decision Process - MDP). *A MDP is parameterized by a tuple* $(\mathcal{S}, \mathcal{A}, R, P)$ *where*

- $\mathcal{S}$ *is the state space*
- $\mathcal{A}$ *is the action space*
- $R = (v(s,a))_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ *where* $v(s,a) \in \Delta(\mathbf{R})$ *is the reward distribution for the state-action pair* (s,a)
- $P = (p(\cdot|s,a))_{(s,a)\in\mathcal{S}\times\mathcal{A}}$ *where* $p(\cdot|s,a) \in \Delta(\mathcal{S})$ *is the transition kernel associated to the state-action pair* (s,a)

In a MDP, the sequence of sucessive states / actions / rewards :

$$s_0, a_0, r_0, s_1, a_1, r_1, \ldots, s_{t1}, a_{t-1}, r_{t-1}, s_t$$

Figure 1: Reinforcement Learning paradigm



satisfies some extension of the Markov property :

$$\mathbf{P}(s_t = s, r_{t-1} = r | s_0, a_0, r_0, s_1, a_1, r_1, \ldots, a_{t-1}, r_{t-1}, s_{t-1}) = \mathbf{P}(s_t = s, r_{t-1} = r | s_{t-1}, a_{t-1}) \quad (2)$$

In an informal manner, the objective of RL is to train an agent to act according to a *good policy* in a potentially unknown MDP. So how can we quantify in a more formal way a "good policy" ?

Definition 2.2 (Policies). *A (Markovian) policy is a sequence* $\pi = (\pi_t)_{t \in \mathbf{N}}$ *of mappings :*

$$\pi_t : \mathcal{S} \mapsto \Delta(\mathcal{A})$$

where $\Delta(\mathcal{A})$ *is the set of probability distributions over the action space.*

A good policy is a policy that yield a large value in each state, which is always some notion of *cumulative reward*, called value function. In our case, we consider infinite time horizon framework for such functional.

Definition 2.3 (Infinite time horizon with a discount parameter). *Given a discount factor* $\gamma \in (0, 1)$*, we define :*

- *Value function :*

$$V^{\pi}(s) = \mathbf{E}^{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s] \quad (3)$$

- *Q function :*

$$Q^{\pi}(s, a) = \mathbf{E}^{\pi}[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a] \quad (4)$$

Definition 2.4 (Optimal policy). *An optimal policy satisfies :*

$$\pi^* \in \underset{\pi}{\operatorname{argmax}} V^{\pi} \quad (5)$$

And in this case,

$$V^* = V^{\pi^*}$$

is the optimal value function

Remark. $Q^* = Q^{\pi^*}$ is the optimal Q function

2.2.2 *Algorithmic trading setup.*

Observation space $\mathcal{S}$. The reinforcement learning environment is the complex trading world gravitating around the agent. The main challenge is the observability of its states, a considerable amount of information is withheld and hidden from the trading agent such as companies' confidential information and other agents' strategies. At each time step t, we consider an available public information stream i_t and the current market state s_t. This state will be resumed to o_t defined as :

$$o_t = \{S_u, D_u, T_u, I_u, M_u, N_u, E_u\}_{t-\tau \leq u \leq t} \quad (6)$$

Where τ is a time window and :

- S_u: state information of the RL agent at time step u (current trading position, number of shares owned by the agent, available cash)
- D_u: information gathered by the agent at time step u concerning the OHLCV (Open-High-Low-Close-Volume) data characterising the stock market
- T_u: information regarding the trading time step u
- I_u: information regarding multiple technical indicators about the stock market targeted at time step u
- M_u: macroeconomic information at the disposal of the agent at time step u
- N_u: news information gathered by the agent at time step u
- E_u: any extra useful information at the disposal of the trading agent at time step u

In our case, we will reduce o_t to :

$$o_t = \{\{ \overbrace{p_u^O}^{\text{opening price}}, \overbrace{p_u^H}^{\text{highest price}}, \overbrace{p_u^L}^{\text{lowest price}}, \underbrace{p_u^C}_{\text{closing price}}, \underbrace{V_u}_{\text{volume}} \}_{t-\tau \leq u \leq t}, \underbrace{P_t}_{\text{position}} \} \quad (7)$$

Action space $\mathcal{A}$. At each step t, the trading agent executes an action $a_t \in \mathcal{A}$ resulting from policy $\pi(a_t | o_t)$. The trader (agent) needs to answer to the following questions: How, whether and how much ?

Actions will be modelled with $Q_t \in \mathbf{Z}$, quantity of shares to buy, *i.e* :

$$a_t = Q_t \quad (8)$$

We consider the following cases:

- $Q_t > 0$: agent buys by posting new bid orders on the order book
- $Q_t < 0$: agent sells by posting new ask orders on the order book
- $Q_t = 0$: agent holds

Key assumptions

(1) Trading actions occur at market closure with execution price $p_t \approx p_t^c$.
(2) We need to account for the trading costs for a realistic and accurate setup. They are due to:

- costs and taxes
- slippage costs (spread costs, market impact and timing costs)

If the first type is easy to model, the second one poses a true challenge for modelling. For the sake of simplicity, we make the hypothesis that costs are proportional to asset's price.

Portfolio update

$$v_{t+1}^c = v_t^c - Q_t p_t - \overbrace{C}^{\text{cost proportionality constant}} |Q_t| p_t \tag{9}$$

$$v_{t+1}^s = \underbrace{(n_t + Q_t)}_{n_{t+1}} p_{t+1} \tag{10}$$

where $n_t \in \mathcal{Z}$ is the number of shares owned at time step t. Note that negative values are allowed, this corresponds to shares borrowed and sold with obligation to repay the lender in the future.

Constraints on the portfolio

- We need the cash value v_t^{cash} to remain positive:

$$v_t^c \geq 0 \tag{11}$$

- We need to adress the risk associated with the impossibility to repay the share lender if agent suffers significant losses. Consequently, v_t^c must be sufficiently large when a negative number of shares is owned. The agent must be able to get back to a neutral position. In a formal manner,

$$v_{t+1}^c \geq -n_{t+1} p_t (1+\epsilon)(1+C) \tag{12}$$

where ϵ quantifies maximum relative change in prices assumed by RL agent prior to trading activity with following condition condition assumed to be fulfilled:

$$\left|\frac{p_{t+1} - p_t}{p_t}\right| \leq \epsilon$$

Action space

(11) and (12) yield upper and lower bounds and we write :

$$\mathcal{A} = \{Q_t \in \mathbf{Z} \cap [\underline{Q_t}, \overline{Q_t}]\} \tag{13}$$

Where:

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

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

Action space reduction We reduce $\mathcal{A}$ to:

$$a_t = Q_t \in \mathcal{A} = \{Q_t^{long}, Q_t^{short}\} \tag{14}$$

for feasibility purposes.

- Long action:

Q_t^{long} induces a long trading position with $n_t \geq 0$. It converts maximum of $v_t^c \to v_t^s$. And we write:

$$Q_t^{long} = \begin{cases} \frac{v_t}{p_t(1+C)} & , a_{t-1} \neq Q_{t-1}^{long} \\ 0 & , otherwise \end{cases} \tag{15}$$

- Short action:

Q_t^{short} converts $v_t^s \to v_t^c$ such that RL agent owns a number of shares equal to $-N_t^{long}$. And we write:

$$\hat{Q}_t^{short} = \begin{cases} -2n_t - \frac{v_t}{p_t(1+C)} & , a_{t-1} \neq Q_{t-1}^{short} \\ 0 & , otherwise \end{cases} \tag{16}$$

However, Q_t^{short} needs to respect lower bound of $\mathcal{A}$. Hence,

$$Q_t^{short} = max(\hat{Q}_t^{short}, \underline{Q_t}) \tag{17}$$

Agent's reward. We set the RL agent's reward to be the daily return of the portfolio :

$$r_t = \frac{v_{t+1} - v_t}{v_t} \tag{18}$$

There are several reasons to justify this choice:

(1) This reward is independent from the number of held shares
(2) $r_t \geq 0 \implies$ profitable strategy
(3) (18) enables us to avoid sparsity that is very complex to adress in a Reinforcement Learning setup

3 DEEP REINFORCEMENT LEARNING: FRAMEWORK AND ALGORITHM

3.1 Dynamic Programming

Definition 3.1. *(Dynamic Programming operator)*
The optimal Bellman operator is defined as follows:

(1) *Value function :*

$$T^*: \mathbf{R}^{|\mathcal{S}|} \longrightarrow \mathbf{R}^{|\mathcal{S}|}$$
$$V \longmapsto T^*(V)$$

Where $T^*(V)(s) = \max_a \{r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V(s')\}$

(2) *Q function :*

$$T^*: \mathbf{R}^{|\mathcal{S}|} \times \mathbf{R}^{|\mathcal{A}|} \longrightarrow \mathbf{R}^{|\mathcal{S}|} \times \mathbf{R}^{|\mathcal{A}|}$$
$$Q \longmapsto T^*(Q)$$

Where $T^*(Q)(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) \max_b Q(s', b)$

Theorem 3.2. *Q^* and V^* are fixed points for their respective dynamic programming operators.*

Remark. In a finite horizon setup, this theorem is replaced with a backward linear system.

3.2 Temporal difference methods

TD (temporal difference) learning is a key pillar in reinforcement learning. The key idea is to find the solution of the fixed point problem for the optimal Value function and Q function given their noisy evaluation with **Stochastic Approximation : Robbins-Monro algorithm**. It usually refers to TD, Q-learning and SARSA algorithms.

The TD update rule for :

(1) Value function :

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)] \tag{19}$$

(2) Q-function :

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)] \tag{20}$$

Where α is the learning rate.

Bootstrapping estimates state or action value function based on subsequent estimates. Such methods are usually faster to learn, and enable learning to be online and continual.

We will present only Q-learning as its extension is the algorithm used in this project.

Q-learning

Algorithm parameters: step size $\alpha \in (0, 1]$, small *epsilon* > 0 among other learning and environment parameters
- Initialize $Q(s, a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}$, arbitrarily except that $Q(\text{terminal}, \cdot) = 0$

foreach *episode* **do**
- Initialize S
- **foreach** *step of episode* **do**
  - Choose A from S using policy derived from Q (e.g., ε-greedy)
  - Take action A, observe R, S'
  - $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$
  - $S \leftarrow S'$

end foreach

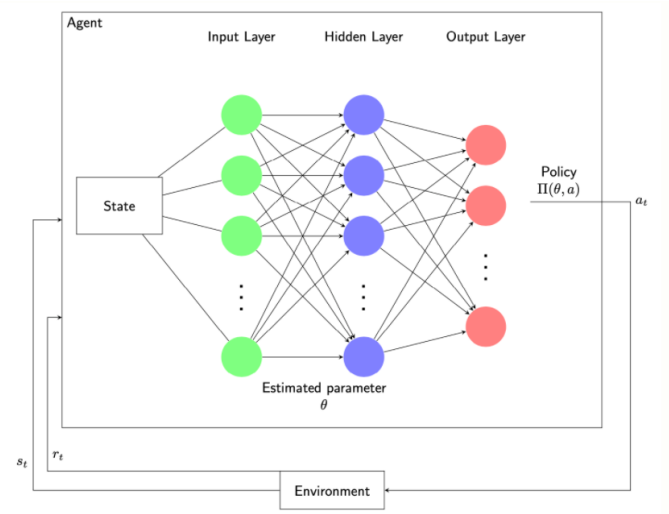
end foreach

3.3 Deep Q Learning

Q-learning is a classical algorithm for learning state and action value functions. Once we have an optimal value function, we may derive an optimal policy. However, it does not scale up well for large state/action spaces. A simple way to generalise over state/action spaces to learn faster is to use an approximation paradigm for the Q-function. For the deep Q-learning algorithm, we utilize neural nets considered as *universal approximation functions*. Such methods can have state of the art performance on complex tasks still with very little theoretical explanation.

In a nutchell, the deep Q-learning algorithm consists of the following steps:

Figure 2: Deep Reinforcement Learning



Deep Q-learning

Algorithm parameters: step size $\alpha \in (0, 1]$, discount factor $\gamma \in (0, 1)$ among other learning and environment parameters
- Use two neural networks:
  (1) A slow-learning target network (θ^-)
  (2) A fast learning Q-network (θ)
- Use experience replay (fill in a replay buffer D with transitions generated by $\pi = f(Q_{\theta^-}(s, a))$
- Shuffle samples in the replay buffer and minimize:

$$J(\theta) = \sum_{(s,a,r,s') \in D} [r + \gamma max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)]^2$$

$$\theta \leftarrow \theta + \alpha(r + \gamma max_{a'} Q_{\theta^-}(s', a') - Q_\theta(s, a)) \nabla_\theta Q_\theta(s, a)$$

- Every N training steps $\theta^- \leftarrow \theta$

Some improvments. Dueling, regularization, exploration with decay, processing state/action pair ...etc

4 EMPIRICAL RESULTS: DISCUSSION

4.1 The Learning Process

Here, we discuss the empirical setup for TDQN learning. Our main objective is to understand the performance of the DRL under different situations (bearish, bullish, volatile market). For that, the initial idea was to create a universe of stocks that would take into account a large fraction of market heterogeneity (top 50 US companies by Market Cap for example), the TDQN would then be backtested for each stock of the universe, and its performance compared to the other strategies, allowing us to get an unbiased view of the performance of the TDQN under different environments. We were not able to implement this approach due to the important training

time of the TDQN agent. Because of this limitation, we have chosen a single stock that exhibits numerous features throughout the study period.

Figure 3: MSFT Stock Price between 2012-1-26 and 2018-12-25

To train the model, we have chosen Microsoft stock (NASDAQ: MSFT) for dates between 2012-1-26 and 2018-1-25. During this period, the stock price was characterized by an important positive trend. This trend made us curious as to the performance of the agent for stocks that exhibits a positive trend, and also stocks exhibiting no trend and negative trend [1].

4.2 Non DRL Trading Strategies

Here, our aim is to briefly explain the trading strategies that are used. In the original paper, authors include 2 passive trading strategies and 2 active strategies. In this project, we extend the list of active strategies based on technical analysis. These strategies are slightly more sophisticated which would help us better understand the performance of the agent from different perspectives.

Passive strategies:

Buy & Hold: this strategy consists of entering the Long position at the start date and exiting it at the final date.

Sell & Hold: the inverse of the Buy & Hold strategy. It consists of shorting the stock at the start date and exiting it at the final date.

Active strategies:[2]

Simple Moving Average :[3] The simple moving average (SMA) indicator is used mainly to identify trends, it is one of the most commonly used indicators across all financial markets. In this report, the SMA indicator is the difference between shorter-term MA a longer-term MA[3], if the indicator changes signs form negative to positive, it's a long signal, and a short signal if there was a change in the opposite direction.

Relative Strength Index: RSI uses the momentum concept [4] to evaluate overbought and oversold stocks. Here, a long signal is generated when RSI of the given stock crosses down 30. A short signal is generated when 70 is crossed.

MACD:[3] Moving Average Convergence Divergence (MACD) is a trend-following momentum indicator that provides an indication of the stock strength, direction and trend. Similar to the SMA cross over strategy, if the indicator changes signs form negative to positive, it's a long signal, and a short signal if there was a change in the opposite direction.

Bollinger Bands®: is a volatility indicator that is composed of three lines: A simple moving average (middle band) and an upper and lower band. The long signal is generated when the price crosses down the lower band, and the short signal when the upper band is crossed.

For all of the Technical Indicators, we used the default parameters.

Back-Testing Set up for Active strategies

For each of the Technical Indicators, we generate the trading signal accordingly. Next, to keep the comparison between the performance of the DRL agent and the Technical Indicators as fair as possible, the starting date for calculating technical indicators precedes the actual start of the backtesting. This step would allow for technical indicators to be fully computed and for signals to correctly reflect the investment strategy. The algorithm is idle until the first Long signal where the initial capital is invested in the stock. This Long position is closed when the next Short Signal is triggered. Meanwhile, the portfolio stays inactive until the next long signal when the cycle is restarted.

4.3 Comparison

4.3.1 Observing TDQN decisions. We backtest the 7 trading strategies using Microsoft stock (NASDAQ: MSFT) for dates between 2018-1-26 and 2018-12-26. We selected this particular time period specifically for its interesting features. During the chosen period, MSFT stock price exhibited multiple "regimes" : a region of important volatility with no observable trend, a region of an increasing trend with moderate volatility and towards the end of the period a negative trend. These numerous features would allow us to better understand the performance of the agent and gauge its robustness and generalizability.

Figure 4: MSFT Stock Price between 2018-1-26 and 2018-12-26 showcasing the three regimes

Figure 6 shows that the agent was initially inactive [5] during more than 40 days, but eventually made an important number of trades throughout the relevant time period. The agent's performance exhibited more profitable returns and less volatility compared to the basic Buy & Hold strategy. This observation points to a promising potential for (TDQN) which surpassed all passive strategies. The agent also outperformed other active trading strategies, including

[1] These observations will be dealt with in the next section

[2] Check Appendix for more detail regarding the construction of these Technical Indicators

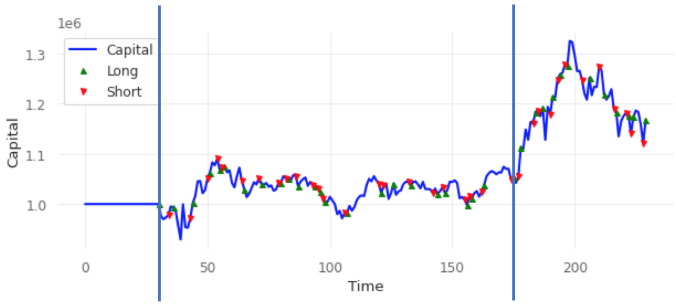
[3] shorter-term (resp. longer-term) means fewer (more) days for calculating the Moving Average

[4] the speed at which the price of a stock is changing

[5] possibly due to the important volatility

RSI and MACD. Table 1 showcases the performance for used trading strategies, Table 2 lists the metrics we used.

Figure 5: Agent performance with an initial amount of 1M$



When closely investigating the performance of the agent, one observes that the initial positive trend was not taken advantage of. At the beginning, the capital was roughly constant and oscillating around the initial amount. The fact that the agent was trained in an environment characterized by an important positive trend reflects the possibility that the agent was unable to recognize the profitability of a positive trend.

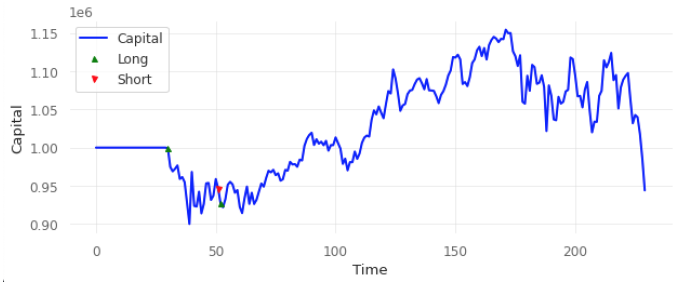
The agent's performance gained significant momentum when the regime changed to a negative trend meaning that the agent was able to perform relatively well in a bearish environment.

From a theoretical point of view, one would expect that the characteristics of the environment in which the agent is operating would greatly impact its performance. This property is also observed empirically before as the previous comments point to an important stylized fact: If we were to subdivide the performance of the agent into periods, the optimal number of periods would be 3. Furthermore, these periods greatly coincide with the regimes observed previously with the stock time series as visualized in Figure 6. Although further backtesting [6] is required to gain more insight and certainty, we can see that the training environment greatly impacts the performance of the agent. Practically, to mitigate the risk that would be generated from such variability we propose to:

(1) Extend the training time series to include maximum heterogeneity (this would also limit overfitting issues)
(2) In case 1. was unfeasible, one can use some changepoint detection framework/models to detect regime shifts[7] (and thus improve the performance)

4.3.2 Effects of transaction costs. Another important question that we seek to answer is the effects of transaction costs. Transaction costs is an useful tool for to get a better idea concerning the flexibility of the agent. In an environment with transaction costs, each transaction limits the performance of the agent, requiring ever more precise trades and a lower number of transactions. Practically, the agent reflects this observation. Compared to the case where transactions costs are null, TDQN has effectively limited its number of transactions and generated 3 investment signals, compared to 93 for the null transaction costs. However, the drop in the number of transactions came at cost, as the performance was mediocre compared to a simple Buy & Hold strategy. The fact that the agent was able to limit its transactions to a minimum reflects its flexibility, however, one would expect more trades than three but with greater precision [8]

Figure 6: Agent performance with an initial amount of 1M$ under transaction cost of 0.1%



We can also observe the difference in terms of the agent's behaviour with or without transaction costs during the learning process. The evolution of score as function of episodes for the null transaction costs case was roughly increasing, whereas it changed monotonicity abruptly for the case of non null transaction costs. This means that the agent was able to understand at some point that the current strategy (consisting of carrying many trades) was not an optimal strategy and consequently changed abruptly its strategy. For more details check Figures 7 and 8.

5 CONCLUSION

In this project, we were able to demonstrate a promising potential for the Trading Deep Q-Network (TDQN) algorithm. Following a portfolio backtesting approach, the TDQN's surpassed some classical strategies (B&H and Technical Indicators) both in terms of performance, flexibility and versatility. Furthermore, the TDQN is more than an investment signal, it takes into account the important detail of the exact proportion of the portfolio invested in the risky asset. This detail is neglected in the classical Technical Analysis approach, and is considered one of its main issues.

We were also able to prove the important dependence of the agent on the underlying dynamics of the environment both during training and back-testing. We have also suggested some possible solutions to this phenomena.

Finally, the measurement of the TDQN performance could be improved. As discussed earlier, assessing the performance over one stock cannot replace the information gained from a large scale backtest. The large scale Backtest would test the TDQN under multiple Market Situations producing an unbiased point of view of its pitfalls and under which situations it excels.

REFERENCES

[1] Théate, T., Ernst, D. (2021). - An application of deep reinforcement learning to algorithmic trading *(Expert Systems with Applications, 173, 114632).*

[6] Back Testing the agent in an environment exhibiting an important heterogeneity
[7] This includes online Bayesian change point detection, Hidden Markov Chains…
[8] Here, precision means the winning trades

[2] Kaufmann,E. (INRIA Lille) , Valko,M. (INRIA Lille & Deepmind) - Lecture notes on reinforcement learning *(Ecole Centrale de Lille).*
[3] Michel Fliess (LIX, INRIA Saclay - Ile de France), Cédric Join (INRIA Saclay - Ile de France, CRAN) - A mathematical proof of the existence of trends in financial time series *(arXiv:0901.1945).*

Appendices

A TDQN

Figure 7: Score as function of episodes for the null transaction costs case

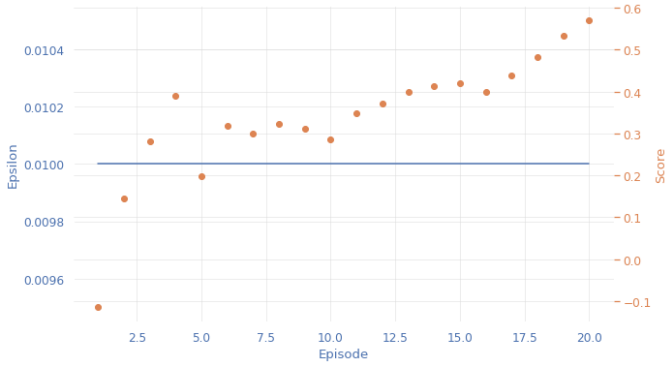
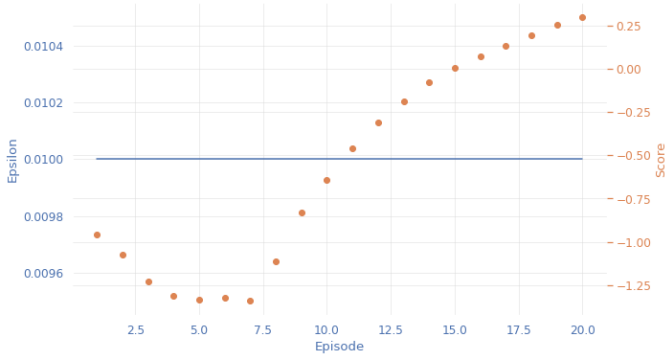


Figure 8: Score as function of episodes for the non-null transaction costs case



B TECHNICAL INDICATORS

RSI is one of the most used formulas in technical analysis. It uses the momentum concept [9] to evaluate overbought and oversold stocks. Its formula is given by U and D where:

$$\begin{aligned} U &= (\text{ close }_{t} - \text{ close }_{t\text{-}1}).\mathbf{1}_{\text{ close }_{t} > \text{ close }_{t\text{-}1}} \\ D &= (\text{ close }_{t\text{-}1} - \text{ close }_{t}).\mathbf{1}_{\text{ close }_{t} < \text{ close }_{t\text{-}1}} \end{aligned} \tag{21}$$

[9] the speed at which the price of a stock is changing

We then define the quantity:

$$RS = \frac{\text{SMMA}(U, n)}{\text{SMMA}(D, n)} \tag{22}$$

Where SMMA (X, n) is a smoothed moving average of X and with a time window of n[10] days.

The RSI is then given by:

$$RSI = 100 - \frac{100}{1 + RS} \tag{23}$$

Moving Average Convergence Divergence (MACD) is a trend-following momentum that provides an indication of the stock strength, direction and trend.

The trading signal is the difference between the MACD line and the signal line. The MACD line is calculated by subtracting a slow-period Exponential Moving Average (EMA) from the fast-period exponential moving average and the signal line is an exponential moving average of the MACD line values. The MACD is defined as:

$$MACD = EMA(Close, p)EMA(Close, q) \tag{24}$$

And the MACD histogram:

$$MACD_{hist} = MACD - EMA(MACD, s) \tag{25}$$

p, q, s are respectively the fast line, slow line and signal line parameters[11].

Bollinger Band®: Bollinger Band® is a volatility indicator that is composed of three lines: A simple moving average (middle band) and an upper (BOLUP) and lower (BOLLO) band. Its formula is given by:

$$\text{BOLUP} = \text{MA}(\text{Close}, n) + m * \sigma[\text{Close}, n]$$

$$\text{BOLLO} = \text{MA}(\text{Close}, n) - m * \sigma[\text{Close}, n]$$

[10] In this report, n takes the value 14
[11] Traditionally they take the value (12, 26, 9)

	Trading Strategy						
Metric	B&H	S&H	RSI	BBANDS	MACD	SMA	TDQN
Sharpe Ratio (Higher is better)	0.21	0.07	NaN	1.3	0.17	0.81	0.79
Sortino Ratio (Higher is better)	0.29	0.11	NaN	2.01	0.23	1.17	1.2
Max Drawdown	-0.18	-0.27	0	-0.09	-0.12	-0.12	-0.15
Profit Ratio	0.71	1.39	NaN	0.03	0.07	0.55	0.57
Annualized Retruns (Higher is better)	6%	%	0%	27%	3%	22%	20%
Annualized Volatility (Lower is better)	28%	%	0%	21%	15%	27%	26%

Table 1: Performance assessment for all strategies

Metric	Utility
Sharpe Ratio	Average return of the trading activity compared to its volatility
Sortino Ratio	Same as Sharpe ratio. Taking into account only downside volatility
Max Drawdown	Maximum observed loss from a peak to a valley of a portfolio
Profit Ratio	Percentage of winning trades made during the trading activity
Annualized Retruns	Annualised return during trading activity
Annualized Volatility	Annualised volatility during trading activity

Table 2: Metrics used for measuring performance