Deep Reinforcement Learning for Active High Frequency Trading

Antonio Briola*, Jeremy Turiel*, Riccardo Marcaccioli†, Alvaro Cauderan§, Tomaso Aste*‡
*Department of Computer Science, University College London
London, UK
{a.briola, jeremy.turiel.18, t.aste}@ucl.ac.uk
†Chair of Econophysics and Complex Systems, Ecole Polytechnique
Paris, FR
{riccardo.marcaccioli}@ladhyx.polytechnique.fr
‡Systemic Risk Center, London School of Economics
London, UK
{acauderan}@ethz.ch
§Department of Computer Science, ETH
Zurich, CH

***Abstract*—We introduce the first end-to-end Deep Reinforcement Learning (DRL) based framework for active high frequency trading in the stock market. We train DRL agents to trade one unit of Intel Corporation stock by employing the Proximal Policy Optimization algorithm. The training is performed on three contiguous months of high frequency Limit Order Book data, of which the last month constitutes the validation data. In order to maximise the signal to noise ratio in the training data, we compose the latter by only selecting training samples with largest price changes. The test is then carried out on the following month of data. Hyperparameters are tuned using the Sequential Model Based Optimization technique. We consider three different state characterizations, which differ in their LOB-based meta-features. Analysing the agents' performances on test data, we argue that the agents are able to create a dynamic representation of the underlying environment. They identify occasional regularities present in the data and exploit them to create long-term profitable trading strategies. Indeed, agents learn trading strategies able to produce stable positive returns in spite of the highly stochastic and non-stationary environment. The source code is available at https://github.com/FinancialComputingUCL/DRL_for_Active_High_Frequency_Trading.**

***Index Terms*—Artificial Intelligence, Deep Reinforcement Learning, High Frequency Trading, Market Microstructure**

I. Introduction

Can machines learn how to trade automatically? What data do they need to achieve this task? If they successfully do so, can we use them to learn something about the price formation mechanism? In this paper, we try to answer such questions by means of a Deep Reinforcement Learning (DRL) approach.

During the last ten years, DRL algorithms have been applied to a variety of scopes and contexts ranging from robotics [1], [2] to healthcare [3]. For a complete overview of the main application domains we refer the interested reader to the work by Li [4]. Financial markets are known to be stochastic environments with very low signal to noise ratios, dominated by markedly non-stationary dynamics and characterised by strong feedback loops and non-linear effects [5]. This is true especially for data at highest level of granularity [5], [6], i.e. at what is called the microstructure level. The Limit Order Book (LOB) is the venue where market participants express their intentions to buy or sell a specific amount (volume) of a security at a specific price level by submitting orders at that given price. The interest in applying DRL algorithms to such a context is undeniably strong. First of all, all the algorithms which exploit the power of ANNs are well known to be data hungry models and, thanks to the spreading of electronic trading, everything that happens in the LOB of an assets can be recorded and stored with up to nanosecond resolution. Secondly, several different problems, potentially solvable using Reinforcement Learning, can be defined on such data.

There are several research questions, with immediate practical interest that must be investigated. Can an agent learn how to assign a given amount of capital among a set of assets? If we need to buy or sell a large amount of assets, can we train an agent to optimally do so? Can we train an agent to directly learn trading strategies at different frequencies and for different numbers of assets?

In the present work, we choose to tackle the latter problem. We show how it is possible to train DRL agents on LOB data to buy and sell single units of a given asset and achieve long term profits. We choose to trade a single unit because our goal is not to maximize the agent's virtual profits, rather to present the first end-to-end framework to successfully deploy DRL algorithms for active High Frequency Trading. This is discussed further in Section IV-B. Specifically, we apply the Proximal Policy Optimization (PPO) algorithm [7] to train our agents. We adopt a continual learning paradigm and we tune the hyperparameters of the model through the Sequential

TA and JT acknowledge the EC Horizon 2020 FIN-Tech project for partial support and useful opportunities for discussion. JT acknowledges support from EPSRC (EP/L015129/1). TA acknowledges support from ESRC (ES/K002309/1), EPSRC (EP/P031730/1) and EC (H2020-ICT-2018-2 825215).

Model Based Optimisation (SMBO) [8] technique. Motivated by the work of Briola *et al.* [9], we use the simplest policy object offered by Stable Baselines [10] that implements actor-critic through the usage of a Multilayer Perceptron. Finally, in order to regularize the dynamics of the profit and loss (P&L) function, we construct the training set by developing a sampling strategy which selects periods of high price activity that are often characterised by higher signal to noise ratios. This provides a more informative training set for the agent.

II. Background

A. Limit Order Book

Nowadays, to facilitate trading, most of the major to minor financial market exchanges employ an electronic trading mechanism called Limit Order Book (LOB). The formation of the price of an asset in a LOB is a self-organising process driven by the submissions and cancellations of orders. Orders can be thought of as visible declarations of a market participant's intention to buy or sell a specified quantity of an asset at a specified price. Orders are put in a queue until they are either cancelled by their owner or executed against an order of opposite direction. The execution of a pair of orders implies that the owners of such orders trade the agreed quantity of the asset at the agreed price. Order execution follows a *FIFO* (first in, first out) mechanism. An order x is defined by four attributes: the sign or direction ϵ_x indicates if the owner wants to buy or sell the given asset, the price p_x at which the order is submitted, the number of shares (or volume) V_x that the owner of the order wants to trade and the time τ_x of submission. It is therefore defined by a tuple $x = (\epsilon_x, p_x, V_x, \tau_x)$. Whenever a trader submits a buy (respectively, sell) order, an automated trade-matching algorithm checks whether it is possible to execute x against one or more existing orders of the opposite sign with a price smaller or equal to (respectively, greater or equal to) p_x. Conventionally, sell orders have a negative signature $\epsilon = -1$ while buy orders have a positive sign $\epsilon = 1$. If any partial matching of the whole volume V_x is possible, it occurs immediately. Instead, any portion of V_x that is not immediately matched becomes active at the price p_x, and it remains so until either it is matched to an incoming sell (respectively, buy) order or it is cancelled by the owner.

A schematic representation of a LOB's dynamics is provided in Figure 1. The collection of all active sell limit orders constitutes the ask side of the LOB while the collection of all active buy limit orders forms the bid side. The price difference $\sigma_\tau = p^{best}_{a,\tau} - p^{best}_{b,\tau}$ between the sell and buy orders at their best available price, respectively, (i.e. the sell/buy with lower/greater price) defines the bid-ask spread, while the average between the best bid/ask prices of a security is called the mid-price $m_\tau = (p^{best}_{a,\tau} + p^{best}_{b,\tau})/2$.

Orders that are executed at the best bid/ask price are called market orders (MOs). MOs gain time priority (since their order is executed immediately), but at the cost of paying every unit of the given asset an extra amount at least equal to the spread σ_τ. This is the reason why the spread is also said to be the cost of a market order. Orders not entirely executed upon arrival

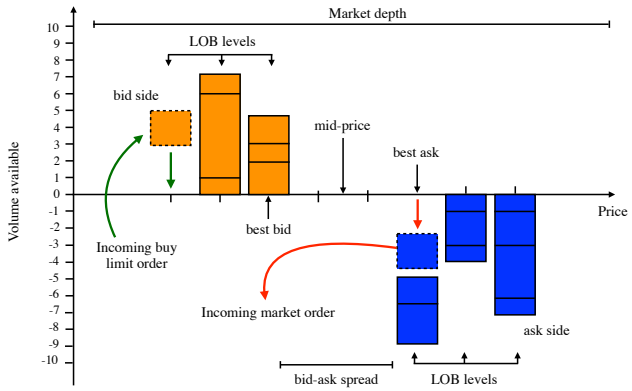


Fig. 1. Graphical illustration of a Limit Order Book, its components, related quantities and dynamics.

(which therefore sit in full or in part in the LOB until matched or cancelled) are called limit orders. Limit orders sitting at the best bid or ask price are matched with incoming MOs using mechanisms which may differ between markets. The most common priority mechanism (which also rules the dynamics of the LOB considered in this work) is called price-time priority: buy/sell limit orders sitting at the same price are executed by selecting first the limit order with the earliest submission time τ_x. We can therefore define the LOB as the collection of unmatched limit orders for a given asset on a given platform at time τ. The time parameter τ is naturally a continuous variable which can be studied on its own, e.g. by characterising the distribution of waiting times between two or more types of order or by studying possible seasonalities of one or more LOB observables. However, when someone wants to characterize the effect of a given event on the LOB (like the average price movement after a market order), it is easier to normalize the time and work in tick time. In this system/frame of reference time evolves as a discrete variable which is updated each time an event is recorded on the LOB. For ease of exposition, we terminate here our description of the LOB. However, it can be already noticed how such venue is at the same time a very intriguing yet hard testing ground for RL algorithms. The rules are simple but able to produce complex patterns, the dynamics of the environment are the result of mediated interactions where agents try to cover their real intentions as much as possible and where different agents act according to their own specific strategies and objectives. We refer the interested reader to the book by Bouchaud *et al.* [11] for a detailed introduction on the topic and a deep characterization of the dynamics of the various quantities which can be defined on a LOB.

B. Proximal Policy Optimization Algorithm

Reinforcement Learning (RL) can be described as the problem faced by a learner which attempts to achieve a specific goal while minimizing related costs, through the exploitation of trial-and-error interactions with a dynamic environment. During this learning exercise, the agent experiences different states (i.e. points in the phase space covered by the envi-

ronment's trajectories) and potentially performs a variety of actions. The effect associated with the specific state-action pair is then evaluated by means of a scoring function and a numerical reward determining the goodness of the chosen action sequence. In this context, the agent needs to find an optimal balance between the exploitation of already experienced state-action pairs and the exploration of unseen ones. Finding this equilibrium is far from trivial. The intrinsic stochasticity involved in solving this maximization problem, makes RL different from other learning paradigms since it does not try to approximate any specific underlying function or maximize a well defined and deterministic score.

This corresponds to finding the optimal mapping (i.e. the policy π) between a set of states $\mathcal{S}$ and a probability distribution over a set of actions $\mathcal{A}$ able to maximize a discounted, cumulative reward over time $T \in [0, \infty)$. Approaches to solving RL problems can be grouped into different classes. Our focus will be on policy search-based methods, but we refer the interested reader to the book by Richard S. Sutton and Andrew G. Barto [12] for a complete review of the subject.

While effective for a vast range of problems in their basic configuration, RL algorithms have proven to be impacted by the curse of dimensionality [13]. The higher the number of features used to describe the agent's state, the harder is for the agent to learn the optimal strategy. Indeed, it will need to search for it in a space whose size scales combinatorially with the number of features. In order to scale classical RL algorithms to high-dimensional problems, Deep Reinforcement Learning (DRL) was introduced. The usage of Artificial Neural Networks (ANNs) to automatically produce lower-dimensional feature representations from a high dimensional input can allow to overcome the curse of dimensionality. Yet, DRL algorithms can require a large amount of data to properly generalise and the training task is notoriously hard [4].

Among other issues, high sensitivity to hyperparameter tuning and initialization can affect the DRL agent's ability to generalize. For example, a wrong learning rate could push the policy network into a region of the parameter space where it is going to collect the next batch of data under a very poor policy, thereby causing it never to recover. In order to overcome this issue, the Proximal Policy Optimization (PPO) algorithm [7] has been introduced. PPO is known to be scalable, data efficient, robust [14] and one of the most suitable for applications to highly stochastic domains. As such, it is a valid candidate for the applicative domain considered here.

The PPO algorithm is part of a family of policy gradients methods for DRL, which are mainly on-policy methods, i.e. they search for the next action based on the present state. In order to do this, policy gradient methods adopt the Policy Loss Function $L^{PG}(Q)$

$$L^{PG}(Q) = E_t[\log \pi_Q(a_t|s_t)]\hat{A}_t \ , \tag{1}$$

where $E_t[\log \pi_Q(a_t|s_t)]$ represents the expected log probability of taking action a in state s and the estimated Advantage function $\hat{A}_t$, provides an estimate of the value of the current action compared to the average over all other possible actions at the present state s.

PPO methods ensure the stability of the policy update process by directly controlling the size of the gradient ascent step implied by the Policy Loss Function. In order to do this, the objective function is written as per Equation 2

$$L^{CPI}(Q) = E_t\left[\frac{\pi_Q(a_t|s_t)}{\pi_{Q_{old}}(a_t|s_t)}\hat{A}_t\right] = E_t[r_t(Q)\hat{A}_t] \ , \tag{2}$$

where the term $\log \pi_Q(a_t|s_t)$ is replaced by the ratio between the probability of the action under the current policy and the probability of the action under the previous policy $r_t(Q) = \frac{\pi_Q(a_t|s_t)}{\pi_{Q_{old}}(a_t|s_t)}$. This implies that for $r_t(Q) > 1$ the action is more probable in the current policy than in the previous one, while if $0 < r_t(Q) < 1$ the action was more probable in the previous policy than in the current one. We can then notice that when $r_t(Q)$ deviates from 1 an excessively large policy update would be performed. In order to avoid this, the objective function is modified as per Equation 3 to penalize $r_t(Q)$ values that lie outside of $[1-\epsilon, 1+\epsilon]$. We take $\epsilon = 0.2$ here as in the original paper [7].

$$\begin{aligned} L^{CLIP}(Q) = E_t\Big[min\Big[\Big(r_t(Q)\hat{A}_t\Big), \\ clip\Big(r_t(Q), [1-\epsilon, 1+\epsilon]\Big)\hat{A}_t\Big]\Big] \ . \end{aligned} \tag{3}$$

The Clipped Surrogate Objective function in Equation 3 takes the minimum between the original $L^{CPI}(Q)$ and the clipped $clip\Big(r_t(Q), [1-\epsilon, 1+\epsilon]\Big)\hat{A}_t$, which removes the incentive for $r_t(Q)$ outside the $[1-\epsilon, 1+\epsilon]$ range. The minimum constitutes a lower bound (i.e. pessimistic bound) on the unclipped objective.

III. Related Work

As mentioned in Section I, the growth of electronic trading has sparked the interest in DL-based applications to the context of LOB data. The literature on forecasting LOB quantities using supervised learning models is growing rapidly. The work by Kearns and Nevmyvaka [15] presents an overview of AI-based applications to market microstructure data and tasks, including return prediction based on previous LOB states. The first attempt to produce an extensive analysis of DL-based methods for stock price prediction based upon the LOB was recently presented in [16]. Starting from a horse-racing type comparison between classical Machine Learning approaches, such as Support Vector Machines, and more structured DL ones, Tsantekidis et al. then considers the possibility to apply CNNs to detect anomalous events in the financial markets, and take profitable positions. The work by Sirignano and Cont [17] provides a more theoretical approach. It contains a comparison between multiple DL architectures for return forecasting based on order book data, and it discusses the ability of these models to capture and generalise to universal price formation mechanisms throughout assets. More recently,

Zhang *et al.* produced two works aimed at forecasting mid-price returns. The first work [18] directly forecasts returns at the microstructure level by employing Bayesian Networks, while the second [19] develops a DL architecture aimed at the simultaneous modelling of the return quantiles of both buy and sell positions. The latter introduces the present state-of-the-art modeling architecture combining CNNs and LSTMs to delve deeper into the complex structure of the LOB. Lastly, a recent review of all the aforementioned methods applied to the context of return quantile forecasting can be found in [9], which highlights how modelling the LOB through its temporal and spatial dimensions (with the CNN-LSTM model introduced in [19]) provides a good approximation, but not a necessary nor optimal one. These findings justify the use of an MLP in the present work as the underlying policy network for the DRL agent.

RL applications, involving financial market data, date back to the early 2000s. These typically focused on low frequency single asset (or index) trading [20], [21], portfolio allocation [22]–[24], up to early forms of intra-day trading [25]. For a complete review of the subject, we refer the interested reader to [26]. The recent spike of interest in DRL and its various applications has naturally brought researchers to apply such algorithms to financial market data or simulations [27], [28]. Concerning trading applications, researchers mostly looked at active trading on single assets with simulated or real data in the low frequency (typically daily) domain [29]–[36]. Other works focused on a variety of tasks such as multi-asset allocation [37]–[39] or portfolio management in general [40]. The body of literature involving DRL applications to the high-frequency domain is surprisingly scarce. Besides recent works aiming to apply DRL algorithms to market making on simulated high frequency data [41], [42], no literature on DRL applications to real high-frequency LOB data is present. In the present work we try to fill this gap and propose the first end-to-end framework to train and deploy a Deep Reinforcement Learning agent on real high frequency market data for the task of active high frequency trading.

IV. Methods

A. Data

High quality LOB data from the LOBSTER dataset [43] represent the cornerstone of the research presented in the current work. This is a widely adopted dataset in the market microstructure literature [11], [44]–[46] which provides a highly detailed, event-by-event description of all micro-scale market activities for each stock listed on the NASDAQ exchange. The dataset lists every market order arrival, limit order arrival and cancellation that occurs on the NASDAQ platform between 09:30am – 04:00pm on each trading day. Trading does not occur on weekends or public holidays, so these days are excluded from all the analyses performed in the present work. A minimum tick size of $\theta = 0.01\$$ is present for all stocks in the NASDAQ exchange. The flow of orders, together with their size and side, is recorded in the event files. LOBSTER also provides the derived LOB aggregation, which forms the input data for the present work. For each active trading day, the evolution of the LOB up to 10 levels is provided. It is worth noting that, each order's dollar price is multiplied by 10000. Experiments described in the next sections are performed using the reconstructed LOB of Intel Corporation's stock (INTC), which is known to be one of the most traded large tick stocks on the NASDAQ exchange [11]. While high market activity may aid DRL training (the agents will explore a wider region of the phase space) or not (lower signal to noise ratio as the price process is more efficient), the fact that we have chosen to consider a large tick stock is definitively beneficial. Indeed, prices of large tick stocks, i.e. stocks with wide price levels, are known to be less stochastic and more dependent on the underlying available LOB information than small tick stocks [11]. Further, as each tick is of meaningful size, the LOB is dense which allows to pass volumes alone to the DRL agent without loss of information.

The whole training dataset consists of 60 files (one per trading day between 04-02-2019 and 30-04-2019). The validation dataset consists of 22 (one per trading day between 01-05-2019 and 31-05-2019). The test dataset consists of 20 files (one per trading day between 03-06-2019 and 28-06-2019). All the experiments presented in the current work are conducted on snapshots of the LOB with a depth (number of tick size-separated limit order levels per side of the Order Book) of 10. Due to the widely different dynamics and higher volatility during the market open and market close periods, these are excluded from both the training and test set for each day (first and last 2×10^5 ticks). This is done in order to facilitate the agent's learning under *normal* market conditions.

B. Models

In the present work, we consider three different training and testing scenarios $c_{i\in[201,202,203]}$, which only differ in the way the DRL agents' state $\mathcal{S}$ is characterized:

- $\mathcal{S}_{c_{201}}$: The state of the agent at time τ is defined by the LOB's volumes for the first ten levels on both the bid and ask sides. In addition, the agent is provided the LOB states for the last 9 ticks and the agent's current position (long, short, neutral).
- $\mathcal{S}_{c_{202}}$: The state of the agent at time τ is defined as per $\mathcal{S}_{c_{201}}$, with the addition of the mark to market value for the agent's current open position (zero if not in a position, by definition). The mark to market value is defined as the profit that the agent would obtain if it decides to close its position at the current time. This allows the agent to evaluate an action with knowledge of how its trade has been performing and what closing the position might imply in terms of reward and, hence, return. Moreover, an improved return profile with respect to $\mathcal{S}_{c_{201}}$ would suggest that the optimal exit of a trade is not independent of its return at the present time.
- $\mathcal{S}_{c_{203}}$: The state of the agent at time τ is defined as per $\mathcal{S}_{c_{202}}$, with the addition of the current bid-ask spread. This allows the agent to evaluate the trading costs before

opening/closing a position. These impact the profitability of the potential trade, and hence the ability to estimate the potential net return of a trade.

For all the different state definitions, at each tick time τ, the agent must choose to perform a specific action $a \in \mathcal{A}$. The set $\mathcal{A}$ of all actions available to the agent is $\mathcal{A} = \{a_i\}_{i=0}^{3} = \{$*sell*, *stay*, *buy*, *daily_stop_loss*$\}$. The outcome of each action depends on the agent's position $\mathcal{P} = \{\mathfrak{p}_i\}_{i \in N,L,S} = \{$*neutral*, *long*, *short*$\}$. A position-action pair $(\mathfrak{p}_i, a_i)$ fully defines how the agent will interact with the LOB environment. We hence list the possible pairs and their respective outcomes.

- $(\mathfrak{p}_N, a_0)$: The agent decides to sell (i.e. a_0), even if it does not currently own the stock (i.e. N). As such, it decides to enter a short position $\mathfrak{p} : N \to S$ for one unit of the underlying stock. Entering a short position is the decision to sell the stock at the current best bid price to buy it back at a future time for the market price available at that time. This action is based on the expectation that the price will fall in the future.
- $(\mathfrak{p}_N, a_2)$: The agent decides to buy (i.e. a_2), while not owning the stock at present (i.e. N). As such, it decides to enter a long position $\mathfrak{p} : N \to L$ for one unit of the underlying stock. Entering a long position is the decision to buy the stock at the current best ask price to sell it at a future time for the market price available at that time. This action will probably be based on the expectation that the price will rise in the future.
- $(\cdot, a_1)$: Independently on its state, if the agent decides to sit A_1, it will not perform any action.
- $(\mathfrak{p}_S, a_0)$: The agent decides to sell (i.e. a_0), even if it has already a short position open (i.e. S). The position is kept open and no additional position is taken on (no change). This sparks from the essential approach of this work, where the agent can only take unit-stock positions.
- $(\mathfrak{p}_L, a_2)$: The agent decides to buy (i.e. a_2), even if it has already a long position open (i.e. L). The position is kept open and no additional position is taken on (no change). This sparks from the essential approach of this work, where the agent can only take unit-stock positions.
- $(\mathfrak{p}_L, a_0)$: The agent decides to sell (i.e. a_0) while in a long position (i.e. L). The position is closed and restored to null $\mathfrak{p} : L \to N$. The profit associated with the trade is hence calculated and given to the agent as feedback for its action.
- $(\mathfrak{p}_S, a_2)$: The agent decides to buy (i.e. a_2) while in a short position (i.e. S). The position is closed and restored to null $\mathfrak{p} : S \to N$. The profit associated with the trade is hence calculated and given to the agent as feedback for its action.
- $(\cdot, a_3)$: The agent checks the current cumulative profit for the day R_{day}; if $R_{day} < 0$ the current position is closed (daily stop-loss action a_3) and further trading for the day is disabled to avoid excessive losses, otherwise a sit action (i.e. a_1) is performed.

As described above, for some position-action pairs a non-zero profit (or loss) is experienced by the agent. The profit associated with closing at time τ a long/short position opened at time $\tau - t$ is intuitively defined as $R_{l/s} = p^{best}_{a/b,\tau} - p^{best}_{b/a,\tau-t}$, i.e. the net dollar return of the trade (price difference minus cost of trade). Incorporating the spread crossing dynamics (trading at bid/ask rather than mid price) directly incorporates transaction costs and makes the experiment more realistic.

Further, the rationale for introducing the *daily_stop_loss* action is that the agent may decide to skip an entire environment (day) if, given the current policy, it is too hard to implement a profitable trading strategy.

The reward function $r_{e,E,d}$ (experienced by the agent at the end of the e-th tick of the E-th epoch of the d-th trading day) is a function of the action-state pair which coincides with the profit $R_{l/c}$ for all actions except the stop-loss.

At this point a final observation is due. The careful reader will have noticed that the agent we define is able to trade and hold in inventory only one unit of the underlying asset at each point in time. Even though this may seem like an unrealistic assumption which affects the applicability of our method, we argue just the opposite.

Every action that an agent performs on a LOB will alter the LOB itself and influence the future actions and reactions of other market participants. Every time we buy (sell) one or more units of an asset, we are consuming liquidity at the ask (bid). This simple action creates a surplus of demand which is going to push the price up (down) proportionally to the square root of the size of the order [11]. By trading only one unit we are sure that there is enough liquidity (as the minimum is one unit) for the agent to execute at the given best bid/ask price without the need to consume the liquidity at higher levels in the LOB (which would worsen the execution price and impact returns). Further, larger trades are usually broken down into smaller units which are executed separately. A variable order size would imply the need for a more complex execution strategy and the requirement to account for self-impact on the price of execution as the orders are placed by the agent. Hence, our choice of single unit trading supports the simplicity and applicability of our framework and its results to live scenarios.

C. Training-Test Pipeline

All experiments are organised into three separate stages.

1) The training phase is performed using a continual learning approach. For each one of the 82 available trading days, 5 LOB snapshots of 10^4 consecutive ticks each are selected. The snapshots include the largest absolute differences between mid-prices in the day. Among them, only 25 snapshots are sampled and transformed into vectorised DRL environments [10]. This sampling procedure allows to choose data containing high signal-to-noise ratios and price changes which are more likely to be actionable. Each environment is then run over 30 epochs. The underlying agent's policy network is represented by a simple Multilayer Perceptron which consists of two hidden layers of 64 neurons each, shared between the action network and the value network. The

last hidden layer is then mapped directly to the two final single-valued network outputs (action and value).

2) A Bayesian Optimisation (BO) [47]–[50] of specific model hyperparameters (i.e. the learning rate and the entropy coefficient) is performed on the validation set whose environments (i.e. LOB snapshots) are sampled analogously to the training ones. A Gaussian Process (GP) regressor with a Squared Exponential kernel [51] is used to fit the unknown function which maps hyperparameter values to out of sample rewards, Based on the cumulative profit registered over all the validation trading days, the next realization to be evaluated is iteratively chosen exploiting the Expected Improvement function [52], [53]. The fitted function then yields the optimal hyperparameter values for training.
3) The policy learned by the agent is independently tested on all days of the test set during the last stage. The cumulative daily profit R_{day} is updated each time a position is closed and the corresponding reward is recorded in order to define a P&L trajectory. Obtained results are hence used to compare the effect of the different state characterizations on the agent's out of sample performance.

V. Results and Discussion

We report out of sample (test) results for ensembles of DRL models defined according to the state definitions in Section IV-B. To compare performances we consider the following perspectives on the returns generated by the agent: cumulative P&L across test days, daily P&L mean and standard deviation across different iterations and return distribution for the agents' trades across test days and iterations.

Figures 2, 3, 4 present results for the three analyses described above and the three state characterizations from Section IV-B.

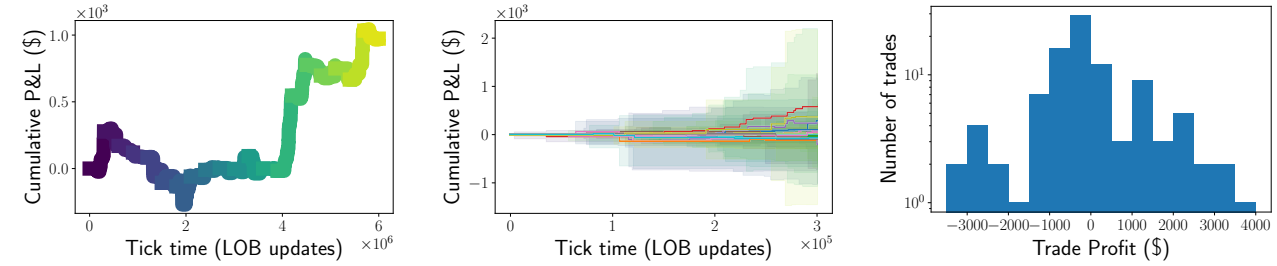
We start by noticing that the agents are able to produce profitable strategies (net of trading costs) on average across their ensembles for each state characterization. From Figures 2(a), 3(a), 4(a) we observe a significant improvement both in magnitude of the cumulative profit as well as in the more persistent upward trend in time between $\mathcal{S}_{c_{201}}$ and $\mathcal{S}_{c_{202}}$. $\mathcal{S}_{c_{203}}$ shows good performance when compared to $\mathcal{S}_{c_{201}}$, while underperforming $\mathcal{S}_{c_{202}}$. This suggests that the mark to market added to the $\mathcal{S}_{c_{202}}$ state definition is highly informative of the agent's expected reward and strongly improves convergence and performance. On the other hand the spread, which is included in $\mathcal{S}_{c_{203}}$, does not improve performance, likely due to the fact that this is mostly constant in large tick stocks. The outperformance of $\mathcal{S}_{c_{203}}$ when compared to $\mathcal{S}_{c_{201}}$ is likely due to the state definition including the mark to market. Future work should investigate a larger sample of environment definitions, as this is beyond the scope of this more foundational and elementary work.

The plots in Figures 2(b), 3(b), 4(b) show cumulative daily mean profits with confidence intervals of one standard deviation across ensemble elements per tick. We observe a positive skew of the means and of the confidence intervals as well as a trend in time, with specific days being more volatile than others. These plots allow to see the agents' performances across different environments (i.e. market conditions) and clearly show that profits are not characterised by daily seasonalities or concentrated in very few days alone. Indeed we notice throughout that most of the days in the test set are characterised by positive returns and that positive returns are larger than negative ones, on average.

We observe that $\mathcal{S}_{c_{202}}$ reaches P&L levels an order of magnitude above the other state characterizations. This result suggests that more trading opportunities are exploited by the agent. This can be understood as the agent having information about its current unrealised profit which allows it to trade in and out of positions more smoothly. Indeed, we observe smoother curves in the daily return plots and a number of trades two orders of magnitude higher than $\mathcal{S}_{c_{201}}$ (and one order of magnitude higher than $\mathcal{S}_{c_{203}}$).

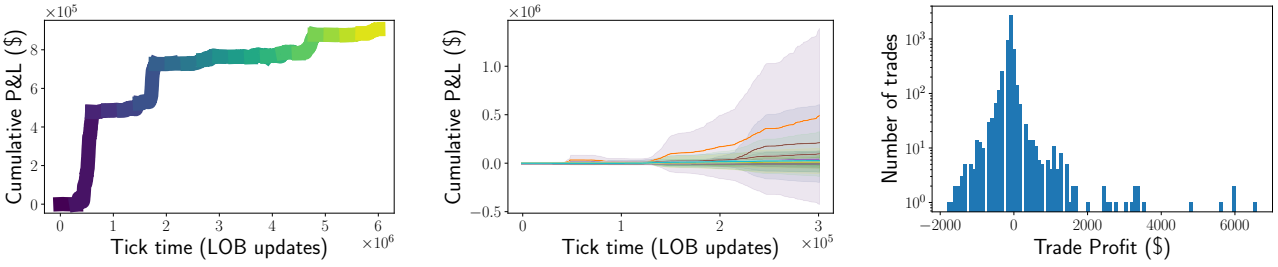
When looking at the return distributions for each trade performed by the agent across days and ensemble realisations (histograms in Figures 2(c), 3(c), 4(c)) we immediately notice the different levels of trading frequencies across state characterizations. As discussed above, information about the current unrealised profit allows the agent to manage its trading activity more accurately and frequently. Yet, the realised profit distributions are broadly similar in terms of the shape of the body and the presence of tails with positive skewness and a peak around zero. The cutoff on the negative distribution tail may be attributed to the day stop loss action available to the agent. As expected, the nearly symmetric distribution body observable in Figures 3(c), 4(c) suggests a limited price predictability. Indeed, many trades are abandoned early with equally likely positive and negative outcomes. The likely difference arises for larger returns for which the authors suggest the following possible explanations: the spread (cost of trading) has less impact, larger price moves arise from inefficiencies which are exploitable and that, in the absence of meaningful price predictability, good position management allows to obtain positive P&L. The latter is likely applicable to $\mathcal{S}_{c_{202}}$, $\mathcal{S}_{c_{203}}$, while rare inefficiencies might be the source of profit for $\mathcal{S}_{c_{201}}$. Yet, this directly shows how markets are inefficient (not in the mere sense of price predictability, but rather of being exploitable for profit) at least for short periods of time in the microstructural domain. This finding is likely due to the strategy being based on innovative DRL techniques which are not (yet) widely implemented in this domain and hence have not exploited and removed these inefficiencies.

When looking at the body of the distributions and their tails, there is no strong evidence to justify the widely different return profiles between different trading state characterizations. What seems to really drive the different return profiles, is the number of trades, as reflected in the histogram plots. Indeed, agents with knowledge of their mark to market have exact information about their potential reward and only need to evaluate whether the potential for further upside is worth keeping the position open. This allows to exploit more trading opportunities which



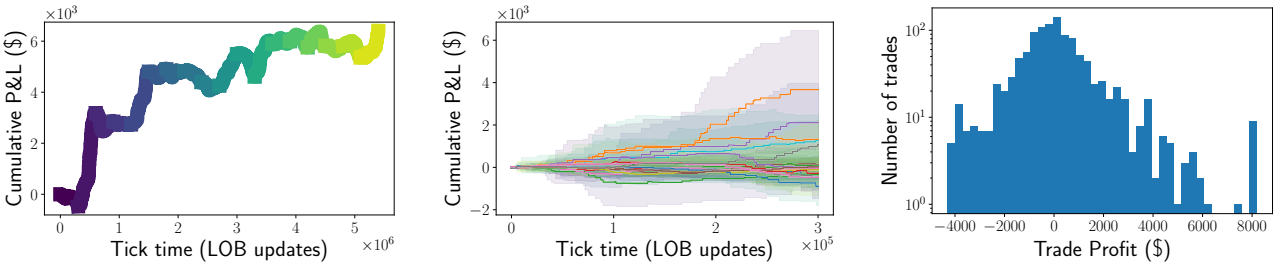
(a) (b) (c)

Fig. 2. Cumulative mean return across the 30 ensemble elements (a), daily mean cumulative return and standard deviation (b) and trade return distribution (c) for the full test (20 days) and agent state definition $\mathcal{S}_{c_{201}}$. Notice that the values of P&L directly depend on the fact that, in LOBSTER data, each order's dollar price is multiplied by 10000 (see Section IV-A).



(a) (b) (c)

Fig. 3. Cumulative mean return across the 30 ensemble elements (a), daily mean cumulative return and standard deviation (b) and trade return distribution (c) for the full test (20 days) and agent state definition $\mathcal{S}_{c_{202}}$. Notice that the values of P&L directly depend on the fact that, in LOBSTER data, each order's dollar price is multiplied by 10000 (see Section IV-A).



(a) (b) (c)

Fig. 4. Cumulative mean return across the 30 ensemble elements (a), daily mean cumulative return and standard deviation (b) and trade return distribution (c) for the full test (20 days) and agent state definition $\mathcal{S}_{c_{203}}$. Notice that the values of P&L directly depend on the fact that, in LOBSTER data, each order's dollar price is multiplied by 10000 (see Section IV-A).

lead to an increase in P&L with no apparent effect on the quality and profitability of individual trades. Indeed this suggests a characteristic trade and return horizon throughout strategies, perhaps characteristic of the asset and its price dynamics.

VI. Limitations

The method presented in the current research work presents some limitations. One of them is its sample inefficiency, a common issue in RL. The fact that the agent receives a reward only when closing a position means that feedback is only as frequent as the agent's trade. In our case, positions are held for tens, hundreds, or even thousands of time steps, hindering convergence speed and leading to a sparse reward function. Furthermore, considering the task's challenging nature and the reward's sparseness, we observe that, sporadically, the agent needs to properly initialize (i.e. it converges to sub-optimal policy such as not trading at all). This finding is expected and results from the PPO algorithm's exploration policy and the negative expected return the agent experiences at the beginning of the exploration stage. A few solutions could alleviate these issues, such as temporal aggregation of the input time series to reduce the sparseness of the reward or the usage of different RL algorithms with a better-suited exploration policy, but we leave this for future work.

VII. Conclusions

In the present work we showed how to to successfully train and deploy DRL models in the context of high frequency trading. We investigate how three different state definitions affect the out-of-sample performance of the agents and we find that the knowledge of the mark-to-market return of their current position is highly beneficial both for the global P&L function and for the single trade reward. However, it should be noticed that, independently on the state definition, the agents are always able to "beat the market"and achieve a net positive profit in the whole training sample considered and in most of the single trading days it is composed of.

Even though we have considered an agent who buys at the best ask and sells at the best bid (and therefore needs to pay the spread to close a position), the exercise here performed is only a first attempt to realistically employ DRL algorithm in a markedly non-stationary environment as the Limit Order Book and financial markets. First of all, we considered an agent who is able to trade only one single unit of the underlying asset. As a result, we safely considered zero the impact of its own trades on the market. However, it is well known that in a realistic setting, where multiple units of the underlying asset can be bought and sold, the impact of our own trades influences in a severe way the final profit of a strategy. Nevertheless, the very own fact that, even in a simplistic setting, a DRL agent can potentially perform well in a highly stochastic and non-stationary environment is worth noticing per se. Further, the ability to produce out of sample profitable strategies shows that temporary market inefficiencies exists and can be exploit to produce long lasting profits and therefore adds a proof against the long-lasting belief that markets are intrinsically efficient.

References

[1] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[2] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[3] Y. Ling, S. A. Hasan, V. Datla, A. Qadir, K. Lee, J. Liu, and O. Farri, "Diagnostic inferencing via improving clinical concept extraction with deep reinforcement learning: A preliminary study," in *Machine Learning for Healthcare Conference*, 2017, pp. 271–285.

[4] Y. Li, "Deep reinforcement learning: An overview," *arXiv preprint arXiv:1701.07274*, 2017.

[5] C. Comerton-Forde and T. J. Putniņš, "Dark trading and price discovery," *Journal of Financial Economics*, vol. 118, no. 1, pp. 70–92, 2015.

[6] M. O'Hara, "High frequency market microstructure," *Journal of Financial Economics*, vol. 116, no. 2, pp. 257–270, 2015.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[8] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*. Springer, 2011, pp. 507–523.

[9] A. Briola, J. Turiel, and T. Aste, "Deep learning modeling of limit order book: a comparative perspective," *arXiv preprint arXiv:2007.07319*, 2020.

[10] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[11] J.-P. Bouchaud, J. Bonart, J. Donier, and M. Gould, *Trades, Quotes and Prices: Financial Markets Under the Microscope*. Cambridge University Press, 2018.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[13] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[14] P. Abbeel and J. Schulman, "Deep reinforcement learning through policy optimization," *Tutorial at Neural Information Processing Systems*, 2016.

[15] M. Kearns and Y. Nevmyvaka, "Machine learning for market microstructure and high frequency trading," *High Frequency Trading: New Realities for Traders, Markets, and Regulators*, 2013.

[16] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Forecasting stock prices from the limit order book using convolutional neural networks," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, vol. 1. IEEE, 2017, pp. 7–12.

[17] J. Sirignano and R. Cont, "Universal features of price formation in financial markets: perspectives from deep learning," *Quantitative Finance*, vol. 19, no. 9, pp. 1449–1459, 2019.

[18] Z. Zhang, S. Zohren, and S. Roberts, "Bdlob: Bayesian deep convolutional neural networks for limit order books," *arXiv preprint arXiv:1811.10041*, 2018.

[19] ——, "Extending deep learning models for limit order books to quantile regression," *arXiv preprint arXiv:1906.04404*, 2019.

[20] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE transactions on neural Networks*, vol. 12, no. 4, pp. 875–889, 2001.

[21] J. E. Moody and M. Saffell, "Reinforcement learning for trading," in *Advances in Neural Information Processing Systems*, 1999, pp. 917–923.

[22] J. Moody, L. Wu, Y. Liao, and M. Saffell, "Performance functions and reinforcement learning for trading systems and portfolios," *Journal of Forecasting*, vol. 17, no. 5-6, pp. 441–470, 1998.

[23] R. Neuneier, "Optimal asset allocation using adaptive dynamic programming," in *Advances in Neural Information Processing Systems*, 1996, pp. 952–958.

[24] O. Mihatsch and R. Neuneier, "Risk-sensitive reinforcement learning," *Machine learning*, vol. 49, no. 2-3, pp. 267–290, 2002.

[25] J. Moody, M. Saffell, W. L. Andrew, Y. S. Abu-Mostafa, B. LeBaraon, and A. S. Weigend, "Minimizing downside risk via stochastic dynamic programming," *Computational Finance*, pp. 403–415, 1999.

[26] T. L. Meng and M. Khushi, "Reinforcement learning in financial markets," *Data*, vol. 4, no. 3, p. 110, 2019.

[27] D. Byrd, M. Hybinette, and T. H. Balch, "Abides: Towards high-fidelity market simulation for ai research," *arXiv preprint arXiv:1904.12066*, 2019.

[28] M. Karpe, J. Fang, Z. Ma, and C. Wang, "Multi-agent reinforcement learning in a realistic limit order book market simulation," *arXiv preprint arXiv:2006.05574*, 2020.

[29] F. Bertoluzzo and M. Corazza, "Testing different reinforcement learning configurations for financial trading: Introduction and applications," *Procedia Economics and Finance*, vol. 3, pp. 68–77, 2012.

[30] L. Chen and Q. Gao, "Application of deep reinforcement learning on automated stock trading," in *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*. IEEE, 2019, pp. 29–33.

[31] Q.-V. Dang, "Reinforcement learning in stock trading," in *International Conference on Computer Science, Applied Mathematics and Applications*. Springer, 2019, pp. 311–322.

[32] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 3, pp. 653–664, 2016.

[33] G. Jeong and H. Y. Kim, "Improving financial trading decisions using deep q-learning: Predicting the number of shares, action strategies, and transfer learning," *Expert Systems with Applications*, vol. 117, pp. 125–138, 2019.

[34] Y. Kim, W. Ahn, K. J. Oh, and D. Enke, "An intelligent hybrid trading system for discovering trading rules for the futures market using rough

sets and genetic algorithms," *Applied Soft Computing*, vol. 55, pp. 127–140, 2017.
[35] Z. Zhang, S. Zohren, and S. Roberts, "Deep reinforcement learning for trading," *The Journal of Financial Data Science*, vol. 2, no. 2, pp. 25–40, 2020.
[36] T. Théate and D. Ernst, "An application of deep reinforcement learning to algorithmic trading," *arXiv preprint arXiv:2004.06627*, 2020.
[37] H. Yang, X.-Y. Liu, S. Zhong, and A. Walid, "Deep reinforcement learning for automated stock trading: An ensemble strategy," *Available at SSRN*, 2020.
[38] Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning," in *2017 Intelligent Systems Conference (IntelliSys)*. IEEE, 2017, pp. 905–913.
[39] Z. Zhang, S. Zohren, and S. Roberts, "Deep learning for portfolio optimisation," *arXiv preprint arXiv:2005.13665*, 2020.
[40] Y.-J. Hu and S.-J. Lin, "Deep reinforcement learning for optimizing finance portfolio management," in *2019 Amity International Conference on Artificial Intelligence (AICAI)*. IEEE, 2019, pp. 14–20.
[41] Y.-S. Lim and D. Gorse, "Reinforcement learning for high-frequency market making." in *ESANN*, 2018.
[42] Y. Wang, "Electronic market making on large tick assets," Ph.D. dissertation, The Chinese University of Hong Kong (Hong Kong), 2019.
[43] R. Huang and T. Polak, "Lobster: Limit order book reconstruction system," *Available at SSRN 1977207*, 2011.
[44] T. H. Balch, M. Mahfouz, J. Lockhart, M. Hybinette, and D. Byrd, "How to evaluate trading strategies: Single agent market replay or multiple agent interactive simulation?" *arXiv preprint arXiv:1906.12010*, 2019.
[45] M. Bibinger, C. Neely, and L. Winkelmann, "Estimation of the discontinuous leverage effect: Evidence from the nasdaq order book," *Journal of Econometrics*, vol. 209, no. 2, pp. 158–184, 2019.
[46] M. Bibinger, M. Jirak, M. Reiss *et al.*, "Volatility estimation under one-sided errors with applications to limit order books," *The Annals of Applied Probability*, vol. 26, no. 5, pp. 2754–2790, 2016.
[47] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.
[48] P. I. Frazier, "Bayesian optimization," in *Recent Advances in Optimization and Modeling of Contemporary Problems*. INFORMS, 2018, pp. 255–278.
[49] F. Archetti and A. Candelieri, *Bayesian Optimization and Data Science*. Springer, 2019.
[50] A. Candelieri and F. Archetti, "Global optimization in machine learning: the design of a predictive analytics application," *Soft Computing*, vol. 23, no. 9, pp. 2969–2977, 2019.
[51] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer School on Machine Learning*. Springer, 2003, pp. 63–71.
[52] J. Močkus, "On bayesian methods for seeking the extremum," in *Optimization techniques IFIP technical conference*. Springer, 1975, pp. 400–404.
[53] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *Journal of Global optimization*, vol. 13, no. 4, pp. 455–492, 1998.