Received March 18, 2021, accepted April 12, 2021, date of publication April 21, 2021, date of current version April 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3074782

Market Making With Signals Through Deep Reinforcement Learning

BRUNO GAŠPEROV AND ZVONKO KOSTANJČAR, (Member, IEEE)
Laboratory for Financial and Risk Analytics, Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia

Corresponding author: Bruno Gašperov (bruno.gasperov@fer.hr)

This work was supported in part by the Croatian Science Foundation under Project 5241, and in part by the European Regional Development Fund under Grant KK.01.1.1.01.0009 (DATACROSS).

ABSTRACT Deep reinforcement learning has recently been successfully applied to a plethora of diverse and difficult sequential decision-making tasks, ranging from the Atari games to robotic motion control. Among the foremost such tasks in quantitative finance is the problem of optimal market making. Market making is the process of simultaneously quoting limit orders on both sides of the limit order book of a security with the goal of repeatedly capturing the quoted spread while minimizing the inventory risk. Most of the existing analytical approaches to market making tend to be predicated on a set of strong, naïve assumptions, whereas current machine learning-based approaches either resort to crudely discretized quotes or fail to incorporate additional predictive signals. In this paper, we present a novel framework for market making with signals based on model-free deep reinforcement learning, addressing these shortcomings. A new state space formulation incorporating outputs from standalone signal generating units, as well as a novel action space and reward function formulation, are introduced. The framework is underpinned by both ideas from adversarial reinforcement learning and neuroevolution. Experimental results on historical data demonstrate the superior reward-to-risk performance of the proposed framework over several standard market making benchmarks. More specifically, the resulting reinforcement learning agent achieves between 20-30% higher terminal wealth than the benchmarks while being exposed to only around 60% of their inventory risks. Finally, an insight into its policy is provided for the sake of interpretability.

INDEX TERMS Deep reinforcement learning, genetic algorithms, high-frequency trading, machine learning, market making, stochastic control.

I. INTRODUCTION

In the field of high-frequency trading (HFT), market making (MM) is particularly salient as it is estimated to constitute more than 65% of trading volume and 80% of limit order traffic in some markets [1]. MM is the process of simultaneously quoting limit orders on both sides of the limit order book (LOB) of a given security with the goal of repeatedly capturing the quoted spread (i.e., the difference between the quotes) while minimizing the inventory risk. The inventory risk arises from uncertainties in the value of the asset held in the market maker's inventory, which is in general non-zero as it depends on *when* and *whether* the orders on the two sides of the LOB get executed. Market makers, in addition to maximizing their risk-adjusted returns, serve a critical role in the market by providing liquidity and immediacy in transactions, thereby playing a vital part in the security price formation process, for which they are commonly incentivized via rebates. This role is particularly important during periods of either heightened volatility or inadequate order flow. It should therefore not come as surprising that market makers are often colloquially referred to as "market catalysts".

The associate editor coordinating the review of this manuscript and approving it for publication was Yongming Li.

A. RELATED WORK

Most commonly, MM is formalized as a problem in stochastic inventory control, and especially so since the seminal work by Avellaneda and Stoikov [2], which was in turn strongly inspired by the earlier work by Ho and Stoll [3]. Under such a formalization, the goal is to continuously select optimal prices at which the market maker should set its bid and ask quotes (limit orders) such as to maximize the expected terminal utility, which should account for both the profit and loss (PnL) and risk. Analytically, this is usually done by using the associated Hamilton-Jacobi-Bellman equations to derive closed-form approximations to the optimal bid and ask quotes, i.e., to the optimal MM controls.

In the original Avellaneda-Stoikov (AS) model [2], a driftless diffusion process for the mid-price evolution is assumed as well as the terminal time T when all positions need to be closed. The authors provide approximations to the optimal quotes, given by the Taylor expansions around T. On a related note, Guéant *et al.* [4] consider a variant of the AS model with inventory limits and derive asymptotic approximations to the optimal quotes in the limit $T \to \infty$, whereas in [5] the model is extended by considering both non-martingale mid-price dynamics and inventory limits. More sophisticated analytical MM approaches, all more or less in the spirit of the original AS model, include additional features such as: stochastic market spreads [6], trading via market orders [6], interdependence in mid-price moves and limit order execution probabilities via mutually exciting Hawkes processes [7], short-term alpha trend dynamics [7], model uncertainty [8], latency [33], and LOB signals [9].

Even though analytical approaches differ in the choice of the utility function, modeling of the underlying processes, and other assumptions, they share in common being predicated on mathematical models underpinned by strong (naïve) assumptions about market behavior and employing many parameters that require often laborious estimation from historical data. For example, it is often assumed that the intensities at which the limit orders get executed decay either exponentially or linearly with the distance from the mid-price, or that the mid-price innovations are normal i.i.d. Despite being conducive to mathematical tractability, such assumptions render the model less realistic. To sidestep these shortcomings, it seems promising to consider approaches such as reinforcement learning (RL), which are both data-driven, i.e., learn directly from the data, and suitable for solving the stochastic control problem of optimal MM. RL methods are particularly favorable since they are capable of solving problems in a completely model-free fashion and *tabula rasa*, i.e. without any explicit modeling of the underlying processes or any prior knowledge. Moreover, since the optimal MM controls (policies) might be non-linear and highly complex, deep reinforcement learning (DRL) [35], the combination of RL and deep learning in which deep neural networks (DNNs) are used to represent RL policies and value functions, is particularly suitable. We emphasize that, recently, DRL methods achieved outstanding success in a variety of diverse, complex sequential decision-making tasks, ranging from the Atari games [10] to robotic manipulation [11] and resource management in network slicing [34]. Research on (D)RL for MM, despite still being relatively scarce, has also recently been swiftly growing. Some other applications of DRL in finance include portfolio optimization [19], risk management and hedging [36], and adaptive rolling window selection [37].

Current non-deep RL approaches to optimal MM are mainly based on temporal-difference RL, for example via simple discrete Q-Learning [12] or SARSA with a linear combination of tile codings used as a value function approximator [13]. Among DRL approaches, Sadighian [14] proposes an MM framework based on advanced policy gradient-based algorithms (A2C and PPO) and an observation space consisting of LOB data and order flow arrival statistics. Kumar [15] introduces a realistic LOB simulator which is then used to train MM agents based on Deep Recurrent Q-Networks (DRQNs). Additional notable research in DRL for MM includes multi-asset MM over a large universe of bonds [16] as well as MM for a multi-agent dealer market [17].

However, most such approaches either resort to crudely discretized action spaces with only a dozen of possible actions, i.e., quote pairs, or fail to incorporate additional predictive signals (besides the time and inventory) into the state space. Moreover, they tend to disregard the issue of interpretability of the learned MM controls, including their relationship with the existing analytical closed-form approximations. Hence, in this paper, we focus on precisely these aspects.

B. OUR WORK AND CONTRIBUTIONS

The main goal of this paper is the development of a novel comprehensive framework for MM with signals. To this end, we use a combination of two standalone supervised learning-based signal generating units (SGUs) and a DRL unit for MM that exploits the generated signals. By linking signal generation capabilities with RL-based MM control, we leverage the advantages of both supervised and reinforcement learning, primarily (i) the existence of labels, and (ii) consideration of the intrinsic sequentiality of the problem. The choice of the SGUs is based on the following idea: when setting quotes for the next period, the market maker effectively has two degrees of freedom, corresponding to how wide (the *width*) and how asymmetrically (the *skew*) w.r.t. the mid-price it sets the quotes. Clearly, the market maker should set the width (skew) in accordance with the price range (trend) forecasts for the next period. If it could do this with perfect accuracy, it would be able to both capture the maximum possible spread in each period and avoid accumulating any inventory. We therefore include exactly these predictions as signals into the state space. The main contributions are the following.

First, we propose a novel state space, action space, and reward function formulation. Specifically, we introduce a tick-based action space that is both continuous (unlike is the case with most current DRL approaches) and takes into account the tick-based nature of MM on stocks (unlike most analytical approaches).

Second, encouraged by the recent results [18] in the genetic algorithms community, we use neuroevolution for training DNNs representing DRL agents. To the authors' knowledge, this is the first study to investigate the viability of using neuroevolution for training DNN agents for MM. We abandon gradient-based DRL in favor of a neuroevolution-based approach in which DNNs directly map states into actions, instead of probability distributions over possible actions.

(For a somewhat similar approach in portfolio optimization that also uses direct mapping to actions see [19].) This eschews the noisy gradient problem and ameliorates learning instabilities, like catastrophic forgetting, which often plague state-of-the-art gradient-based DRL methods. Additionally, a particular choice of DNN initialization, orthogonal for weights and constant for biases, that improves the diversity of the zeroth-generation agents, is proposed.

Third, inspired by ideas from adversarial reinforcement learning [20], we use perturbations by an opposing agent — the adversary — to render the MM agent more robust to model uncertainty and consequently improve generalization. The novel state and action space design for the adversary, which represents the market itself, ensures that it learns *when* and *how* to act in order to best hinder the MM agent. We point out that, in our approach, the adversary directly perturbs the MM agent's actions, whereas in the existing research on ARL for MM [21] the simulation model parameters are altered instead.

Fourth and final, we focus on the interpretability of the learned MM controls. Recently, the topics of interpretability and transparency in deep learning, including DRL, have received immense interest. As a recent example from the HFT literature, Leal *et al.* [22] tackle interpretability of the "black box" controls learned by a DNN controller for optimal execution. In a somewhat similar vein, we shed some light on the resulting DRL agent by providing explanations of the learned MM controls.

II. PRELIMINARIES

A. (DEEP) REINFORCEMENT LEARNING

RL is a large class of machine learning algorithms for efficiently solving (usually discrete-time) Markov decision processes (MDPs) based on learning by interacting with the environment via trial-and-error. A discrete-time MDP is given by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ a set of actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ a transition probability function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ a reward function, and $\gamma \in [0, 1]$ a discount factor.

At each time t, the decision-maker, also known as *the agent*, observes the state of the environment S_t and selects an action A_t based on the current state. The selected action generally influences the next state of the environment S_{t+1}, as well as the reward R_{t+1}. The procedure is then iterated and a (possibly infinite) sequence of states, actions, and rewards $(S_0, A_0, R_1, \ldots, S_t, A_t, R_{t+1}, \ldots)$ — the *trajectory* — is obtained. Since the next state S_{t+1} depends only on the current state S_t and the action A_t and is conditionally independent of the prior states, the state transitions of an MDP satisfy the Markov property, as the name itself suggests. There are two types of RL tasks (i.e., MDPs): non-episodic and episodic. Non-episodic tasks are never-ending and result in infinite trajectories, whereas episodic tasks end when a special state, called a *terminal state*, is reached.

A stochastic policy $\pi : \mathcal{A} \times \mathcal{S} \to [0, 1]$ expresses the probability of selecting action a in state s. A deterministic policy $\pi : \mathcal{S} \to \mathcal{A}$ maps states directly into actions. The sole goal of an RL agent is to maximize the expectation of the discounted cumulative sum of rewards — the return — which is given by:

$$G_t = R_{t+1} + \gamma R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}. \tag{1}$$

In non-episodic tasks $\gamma < 1$ to ensure convergence of (1). The RL agent searches for a policy — an optimal policy π^* — that maximizes the expected return:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi}[G_t]. \tag{2}$$

An optimal policy always exists but is not necessarily unique. Furthermore, let us express S_t as a feature vector $\mathrm{x}(S_t)$:

$$\mathrm{x}(S_t) = [x_1(S_t), x_2(S_t), \ldots, x_n(S_t)] \in \mathbb{R}^n. \tag{3}$$

Now, let us denote a parametrized (for example by a DNN) policy by π_θ and the objective function (e.g., the return) by $J(\theta)$. The goal now is to find an optimal parametrized policy π_θ^* that maximizes the objective:

$$\pi_\theta^* = \arg\max_{\pi_\theta} \mathbb{E}[J(\theta)]. \tag{4}$$

B. LIMIT ORDER BOOK

A *limit order book* [23] is a collection of outstanding (unexecuted) limit orders for a given security. A *limit order* is an offer to purchase or sell a certain amount of a given security at a specified or better price. After submission, a limit order joins the LOB at the specified price level until it is either canceled, modified, or executed against a market order. A *market order* is a request to immediately buy or sell a certain amount of a given security. A trader that sends a market order is referred to as a *market taker*. A market order is instantly executed against outstanding limit order(s), in accordance with the *price-time priority*, i.e., starting with the limit order with the most favorable price from the market taker's perspective and so forth. If there are multiple limit orders at the same price level, an order with an earlier submission time gets executed first. All prices are multiples of the *tick size*, the smallest price increment. The lowest (highest) price at which there exists an outstanding ask (bid) order is referred to as the *best ask* (*best bid*), the difference between the two is called the *bid-ask spread*, and the mean of the two is called the *mid-price*. A snapshot of a limit order book is provided in Fig. 1.

C. MARKET MAKING

MM refers to a trading strategy that relies on simultaneously quoting both bid and ask prices for the same asset. The goal of a market maker is to generate profits by repeatedly earning the difference between the quoted bid and ask prices, i.e., by capturing the quoted spread. For example, let us assume that, at $t = 0$, the market maker, holding no inventory, posts a bid limit order at \$99 and an ask limit order at \$101. If both

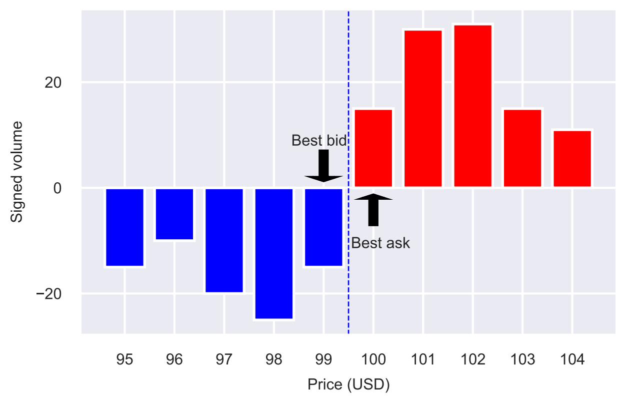


FIGURE 1. A snapshot of a limit order book showing five levels of depth. The current mid-price (denoted by the blue dashed line) is \$99.5, the best bid is \$99, the best ask is \$100, and the tick size is \$1. Bid orders are shown with negative volumes by convention.

limit orders get executed (i.e., if both a counterparty willing to sell at \$99 and a counterparty willing to buy at \$101 appear), the market-maker will earn the quoted spread of \$2. However, if only one of the limit orders gets executed, the market-maker will not only fail to capture the quoted spread but will also end up with a non-zero inventory, thereby exposing itself to the inventory risk. This risk stems from the uncertainties in the value of the held asset as well as the fact that trades happen at random times. When faced with a non-zero inventory, the market maker, being risk-averse, typically skews its quotes in order to reduce its inventory level. Moreover, the risk-averse market-maker should continuously update its orders while taking into account both its current inventory level and ideally also external market signals (like trend and volatility forecasts).

III. FORMULATION

A. MARKET MAKING PROCEDURE

We employ a MM procedure in the spirit of the AS framework [2] and formulate it as an episodic RL task. At the beginning of a time-step (at the time t) the MM agent observes the state of the environment S_t, consisting of the current inventory level and the auxiliary signals, and performs the action A_t — posts an ask and a bid limit order, both of unit size, at certain prices Q_t^{ask} and Q_t^{bid}. When the end of the time-step is reached at the time $t + \Delta t$, the agent receives a reward and posts a new pair of limit orders, unless the agent's inventory is equal to the minimum (I_{min}) or maximum (I_{max}) inventory constraint, in which case only a single limit order on the opposite side is posted, to prevent the inventory from exceeding the bounds. The procedure is iterated until the terminal time T when the episodic task ends. Ideally, in every time-step, both orders are executed and the agent earns a small profit. However, if only one of the orders gets executed, a change in the inventory I_t ensues, as well as in the amount of held cash X_t. We point out that a first-passage time (FPT) [24] execution model is assumed, according to which a bid (ask) limit order is executed as soon as a sell (buy) side trade takes place at a price that is lower (higher) or equal to the price at which the order is quoted. Additionally, instead of measuring time in units of physical time, we use non-physical time [25], as is commonly done in LOB modeling. Consequently, one unit of time corresponds to a single change on the top of the LOB, either in prices or volumes, making time discrete. Finally, we assume there are neither transaction fees nor rebates for market makers. The simplified summary of the MM procedure is depicted in Alg. 1.

Algorithm 1: MM Procedure as an RL Environment

```
initialize $t, I_t, X_t \leftarrow 0$ ;
while $t < T$
    generate the auxiliary signals (the SGU outputs) ;
    observe state $S_t$ ;
    $Q_t^{\text{ask}}, Q_t^{\text{bid}} \leftarrow \infty, 0$ ;
    % perform action $A_t$ ;
    if $I_t < I_{max}$ then update $Q_{t,\text{bid}}$ ;
    else pass;
    if $I_t > I_{min}$ then update $Q_{t,\text{ask}}$ ;
    else pass;
    while True
        % process trades ;
        foreach trade from $[t, t+1]$ ;
         do update $I_t, X_t$ ;
        $t \leftarrow t + 1$ ;
        % break when the end of the time-step is
         reached ;
        if $t \bmod \Delta t = 0$ then break ;
    end while
    receive reward $R_{t+1}$ ;
end while
```

B. MARKET MAKING AGENT

1) STATE SPACE

At the time t, the MM agent observes the state given by:

$$S_t = [I_t, RR_t, TR_t], \tag{5}$$

where I_t denotes the agent's inventory whereas RR_t and TR_t denote the price range and trend predictions, i.e., the outputs of the SGUs, all at time t. Thus, the state space is three-dimensional.

2) ACTION SPACE

The action A_t represents a certain choice of bid/ask orders relative to the current best bid/ask. Therefore:

$$A_t = [A_{t,1}, A_{t,2}] = [Q_t^{\text{ask}} - Q_t^{\text{bask}}, Q_t^{\text{bbid}} - Q_t^{\text{bid}}], \tag{6}$$

where $A_{t,i} = k\Delta$ for $i \in \{1, 2\}$, $k \in \mathbb{Z}$, with Δ being the tick size, Q_t^{bask} (Q_t^{bbid}) the best ask (bid), all at time t. This action formulation is particularly convenient since the bid-ask spread is already implicitly encoded into it and hence there is no need for having an additional state space variable for it. Furthermore, it ensures that even a trivial policy for which $A_t = [0, 0]$ $\forall t$ provides a relatively strong benchmark

– namely, the zero tick offset benchmark. Also observe that larger (smaller) $A_{t,i}$ values correspond to more conservative (aggressive) quoting, with negative values representing limit orders posted inside the bid-ask spread. The action space is two-dimensional and continuous.

Lastly, we emphasize that the action A_t generally affects the next state S_{t+1} via its influence on the next inventory level I_{t+1}, which points to the intrinsic sequentiality of the optimal MM problem as formulated here and warrants the use of RL in lieu of supervised learning based-methods.

3) REWARDS

Finally, the reward function is defined as follows:

$$R_{t+1} = (Q_t^{\text{ask}} - M_{t+1})\mathbb{1}\{Q_t^{\text{ask}} \text{ exe}\} + (M_{t+1} - Q_t^{\text{bid}})\mathbb{1}\{Q_t^{\text{bid}} \text{ exe}\} - \lambda|I_{t+1}|, \quad (7)$$

where M_t denotes the mid-price at time t, $\mathbb{1}\{Q_t^{\cdot} \text{ exe}\}$ is the indicator function for whether the bid/ask order gets executed in the upcoming time-step and λ is a parameter that accounts for both volatility and risk aversion.

Such a reward function both incentivizes spread-capturing (making *round-trips*) and discourages holding inventory, in order to prevent inventory risk. It is inspired by utility functions containing a running inventory-based penalty, as found in [26] and [6]. However, instead of a quadratic penalty term, we use an absolute value inventory penalty term which has a convenient Value at Risk (VAR) interpretation. Furthermore, the first two terms of our reward function can be interpreted as the *symmetrically dampened PnL reward function* as introduced in [13], with maximum dampening. The idea is to strongly disincentivize the agent from trend chasing (trading) and direct it towards spread capturing (MM). However, note that if the market is, say, trending upwards, it is reasonable to expect that, on average, $|M_{t+1} - Q_t^{\text{ask}}| > |Q_t^{\text{bid}} - M_{t+1}|$ and therefore some small incentive to indulge in trend-chasing nevertheless remains.

Observe that in the original AS model, the market maker's value function takes into account the time remaining until the terminal time T — the end-time of the trading day or a trading session when all positions are terminated to avoid exposure to overnight risk. However, such a formulation is ill-suited when there is no natural notion for the terminal time, which is, for example, the case with cryptocurrencies, which are traded continuously. (For the same reason, the "remaining time" variable is not included in our state space.) Finally, we point out that a vacuous MM strategy that quotes extremely conservatively on both sides of the LOB, and under which no executions at all take place, achieves a return equal to zero.

C. ADVERSARY

In order to improve the MM agent's robustness to model uncertainty, we draw on ideas from adversarial reinforcement learning [20]. Ideally, this approach will reduce the sensitivity of the MM agent's learned policy to differences between training and testing conditions or model misspecification. Therefore, besides the MM agent, an additional RL agent — *the adversary* — is introduced. Its goal is to minimize the MM agent's return by strategically displacing its bid/ask quotes (actions). However, to prevent the adversary from completely hindering the MM agent, the amount of force available to the adversary, expressed as the maximum allowable sum of displacements, measured in ticks, is limited.

1) STATE SPACE

The state S_t' at the time t as observed by the adversary is given by:

$$S_t' = [I_t, \mathbb{1}\{Q_t^{\text{ask}} \text{ exe}\}, \mathbb{1}\{Q_t^{\text{bid}} \text{ exe}\}], \quad (8)$$

with the same notation as in (7). Note that the adversary is endowed with the "superpower" of "clairvoyance" — the adversary can perfectly forecast the immediate future and use this information to strategically displace the MM agent's quotes. We emphasize that the fact that the adversary's state space differs from the MM agent's does not preclude the existence of a global state space.

2) ACTION SPACE

The adversary's action A_t' corresponds to a certain displacement of the MM agent's quotes:

$$A_t' = [A_{t,1}', A_{t,2}'] = [A_{t,1}^{dis} - A_{t,1}, A_{t,2}^{dis} - A_{t,2}], \quad (9)$$

where $A_{t,i}' = l\Delta$ for $i \in \{1, 2\}$, $l \in \mathbb{Z}$, with $A_{t,1}^{dis}$ ($A_{t,2}^{dis}$) denoting displaced ask (bid) quotes. Observe that the action space is two-dimensional and continuous.

3) REWARDS

The reward function is the negative of the MM agent's reward function:

$$R_{t+1}' = -R_{t+1} = (M_{t+1} - Q_t^{\text{ask}})\mathbb{1}\{Q_t^{\text{ask}} \text{ exe}\} + (Q_t^{\text{bid}} - M_{t+1})\mathbb{1}\{Q_t^{\text{bid}} \text{ exe}\} + \lambda|I_{t+1}|, \quad (10)$$

with the same notation as in (7). It penalizes the captured spread and pushes the MM agent into accumulating large inventory levels, hence exposing it to significant inventory risks.

IV. IMPLEMENTATION

A. MARKET MAKING AGENT DETAILS

A feed-forward, fully-connected neural network (NN) is used as the function approximator, i.e., to represent the DRL agent's policy. It maps states directly to actions — bid/ask quotes expressed as offsets from the current best bid/ask. To ensure that the NN outputs are appropriately scaled and multiples of the tick size, we first multiply them by a scaling factor and then apply a round function. An NN architecture with two hidden layers, 32 neurons each, is selected. (Note that such relatively shallow but still formally deep architectures are commonly and successfully used in DRL [10].) The rectified linear unit (ReLU) is used as the activation function in all layers except the final layer where the linear activation function is used instead. This ensures that the agent is also able to post quotes inside the bid-ask spread. Moreover, the MM agent is even allowed to quote bid/ask pairs where the ask is lower than the bid, which is also the case with the AS approximations.

B. ADVERSARY AGENT DETAILS

The same type of NN as in IV-A is used as the function approximator. Again, the NN maps states directly to actions — here displacements of the MM agent's quotes. However, a shallow architecture with only one hidden layer consisting of 12 neurons is employed. The ReLU is used as the activation function in the hidden layer whereas the linear activation is used in the outer layer. The same procedure as in IV-A is used to ensure the proper scale, as well as the discreteness of the NN outputs.

C. SIGNAL GENERATING UNITS

The first SGU is underpinned by a gradient boosting model and used for the prediction of the realized price range, a volatility-based measure. The second SGU is based on a long short-term memory (LSTM) NN and used for trend prediction. Details pertaining to the implementation of SGUs are provided in App. A and B.

D. TRAINING

We use a gradient-free approach - neuroevolution via genetic algorithms for training the RL agents. Therefore, we implicitly treat the search for an optimal policy as a black-box function optimization problem. Most state-of-the-art gradient-based RL algorithms rely on stochastic policies to ensure enough exploration. However, this introduces additional stochasticity and exacerbates the noisy gradient problem. This is particularly problematic when individual actions can have long-lasting consequences, or when tackling highly stochastic (or noisy) environments, as is often the case in financial applications, including MM. Genetic algorithms, on the contrary, are a gradient-free approach that can easily accommodate deterministic policies without detrimentally affecting exploration. Despite suffering from low sample efficiency, it is hypothesized [18] that their performance is improved due to temporally extended exploration. Furthermore, they generally have fewer hyperparameters to optimize, can easily tackle sparse rewards as well as "jump" over local minima in the parameter space, and since they are gradient-free, there is no need to resort to tricks such as gradient clipping to handle the problem of exploding gradients.

This choice is inspired by previous results [18], [27] that have shown that neuroevolution offers a viable alternative for training DNNs for DRL. Here we use the same approach as in [18], i.e., a simple population-based genetic algorithm, albeit with a small modification. Instead of using the Xavier initialization, we opt for the orthogonal initialization [28] (with the gain value set to 0.9) for the weights and constant initialization (with the value set to 0.05) for the biases. This particular initialization scheme is meticulously selected after thorough experimentation with different candidate initialization schemes, as it is shown to generate the most diverse policies.

We learn the adversary's policy while holding the MM agent's policy fixed and vice versa. For both the MM agent and the adversary, all input features are normalized with z-score normalization. Additional implementation details, including hyperparameter values, are given in App. C.

V. EXPERIMENTS

A. EVALUATION DATASET

The evaluation dataset comprises of historical tick-by-tick trades and quotes from the cryptocurrency exchange Bitstamp for the pair Bitcoin/US Dollar (ticker: BTC/USD), corresponding to the period from September 1, 2020, to September 30, 2020 — in total full 30 trading days (720 trading hours) worth of data. We chose Bitstamp as a result of a trade-off between maximizing both the liquidity and the average size of the bid-ask spread. Tick-by-tick trades data includes, among others, IDs, timestamps, prices, and amounts, pertaining to each realized trade. Quotes data encompasses, among others, timestamps, amounts, and prices for the best bid/ask quotes, recorded every time the top of the LOB has changed. Tick-by-tick trades and quotes data were collected from the exchange's real-time WebSocket trades and order book L2 data feeds, respectively.

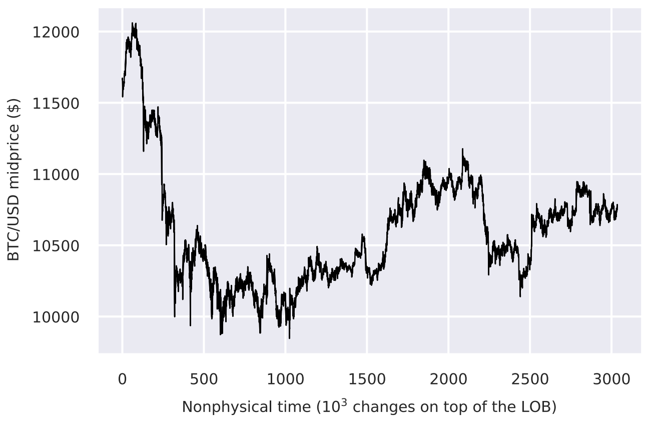


FIGURE 2. Evolution of the BTC/USD mid-price in the full dataset.

The total number of realized trades is 660, 337, which is tantamount to around 15.29 trades per minute. (We emphasize that the execution of a market order at multiple levels of LOB depth results in multiple separate trades, each with a unique ID.) The majority (54.94%) of trades are sells, whereas the majority of the traded volume is on the buy side (52.93%). The mean (median) size of a trade is 0.3194 (0.0921). There is a total of 3, 036, 073 quotes or approximately 70.28 quotes per minute, out of which 87.41% signify changes in prices on the top of the LOB. The tick size equals \$0.01. The price evolution in the full dataset is depicted in Fig. 2. We set the length of a single MM period to $\Delta t = 19$, which corresponds to approximately 16.22 s.

Using a 64/16/20 split, the dataset is divided into a training (S1), validation (S2), and testing set (S3). The S1 and S2 sets are used for training/validation of the SGUs, respectively, and SGU predictions are then made on the S3 set. The S3 set is further split into three subsets, for training, validation, and testing of the DRL unit.

B. BENCHMARKS

In order to benchmark our approach, we use the following two classes of benchmark strategies:

- **Fixed Offset with Inventory Constraints** — FOIC (N, M, c) — where N (M) is the offset, measured in ticks, from the current best bid (ask) and c is the inventory constraint. For example, FOIC(1, 1, 5) corresponds to the strategy that posts bid (ask) orders one tick below (above) the current best bid (ask) while adhering to the inventory constraint $-c = -5 \leq I_t \leq 5 = c$. In the case $I_t = c$ (or $I_t = -c$), only a single order on the opposite side is posted;
- **Linear in Inventory with Inventory Constraints** — LIIC(a, b, c) — where, if $-c < I_t < c$, ask (bid) orders at the time t are posted at the following prices:

$$Q_t^{ask} = M_t + a + bI_t, \tag{11}$$
$$Q_t^{bid} = M_t - a + bI_t, \tag{12}$$

where a and b are constants, and the remaining notation is the same as in III. If $I_t = c$ (or $I_t = -c$), only the order on the opposite side is posted. Clearly, the orders are posted around the *indifference price* $M_t^{ind} = M_t + bI_t$ with a constant spread equal to $2a$. This class of strategies, where the quotes depend linearly on the inventory is particularly important, since it also subsumes the Guéant–Lehalle–Fernandez-Tapia (GLFT) approximations [4], which are considered state-of-the-art and in which the optimal quotes, when the inventory is strictly within the constraints, are given by:

$$Q_t^{ask} = M_t + \frac{1}{\gamma}\log\left(1+\frac{\gamma}{k}\right) - \frac{2I_t - 1}{2}\sqrt{\frac{\sigma^2\gamma}{2kA}\left(1+\frac{\gamma}{k}\right)^{1+\frac{k}{\gamma}}}, \tag{13}$$

$$Q_t^{bid} = M_t - \frac{1}{\gamma}\log\left(1+\frac{\gamma}{k}\right) - \frac{2I_t + 1}{2}\sqrt{\frac{\sigma^2\gamma}{2kA}\left(1+\frac{\gamma}{k}\right)^{1+\frac{k}{\gamma}}}, \tag{14}$$

where γ is the risk aversion coefficient, σ the standard deviation, whereas k and A are liquidity-related parameters, as introduced in [2]. More precisely, A corresponds to the trading intensity while k accounts for the bid-ask spread and the shape of the LOB. Once again, if $I_t = c$ or $I_t = -c$, only a single order (on the opposite side) is posted. We will refer to this special case of LIIC(a, b, c) as GLFT$(\gamma$, c).

C. PERFORMANCE AND RISK METRICS

The following performance and risk metrics are employed:

- **Episode return**. The total episode return, G_0, as given by (1).
- **Terminal wealth**. The total wealth (portfolio value) at $t = T$.
- **Mean Absolute Position (MAP)**, defined as:

$$\text{MAP}(t) = \frac{1}{M}\sum_{k=1}^{M} |I_{k\Delta t}|, \tag{15}$$

where M is the number of completed time-steps until the time t. The MAP metric directly accounts for the inventory risk.
- **Maximum Drawdown (MDD)**, given by:

$$\text{MDD}(T) = \max_{\tau\in(0,T)}\left[\max_{s\in(0,\tau)} W_s - W_\tau\right], \tag{16}$$

where $W_t = M_t I_t + X_t$ denotes the total wealth at the time t, and the rest of notation is the same as before. The MDD can be understood as the maximum loss from a peak to a trough of a certain portfolio.
- **Rolling PnL-to-MAP Ratio (PnLMAP)**, a custom rolling metric that we define as follows:

$$\text{PnLMAP}(t) = \frac{W_t}{\text{MAP}(t)}, \tag{17}$$

which simultaneously considers both the profitability and the incurred inventory risk of the MM strategy up to the time t.

D. REINFORCEMENT LEARNING ENVIRONMENT DETAILS

We set $I_{max} = 2$, $I_{min} = -2$ and $\lambda = 0.15$. This choice of parameters corresponds to a highly risk-averse market maker.

E. RESULTS

Following training, the resulting DRL agent is evaluated on the testing set (*out-of-time*) and its performance is compared against multiple benchmarks from V-B. Fig. 3 depicts the PnLMAP diagram of the DRL agent compared against the agent variant trained without adversarial disturbances (DRL2), the FOIC(0, 0, 2) benchmark, and the GLFT(0.001, 2) benchmark. All GLFT parameters were calibrated in accordance with the procedure described in [29], except the risk aversion parameter γ which is generally chosen according to the risk preferences of the market maker. To obtain a strong benchmark for comparison, we select the value of γ that results in the maximum PnLMAP(T) value on the testing dataset. Furthermore, in order to ensure a fairer comparison, the inventory constraints for the benchmarks were set to be equal to the DRL agent's, i.e., to the values from V-D. More comparison details, for a slightly wider spectrum of benchmark strategies, are given in Table 1.

The results indicate that our DRL approach significantly outperforms the benchmarks and achieves not only by far the most favorable return-to-risk performance, but also the

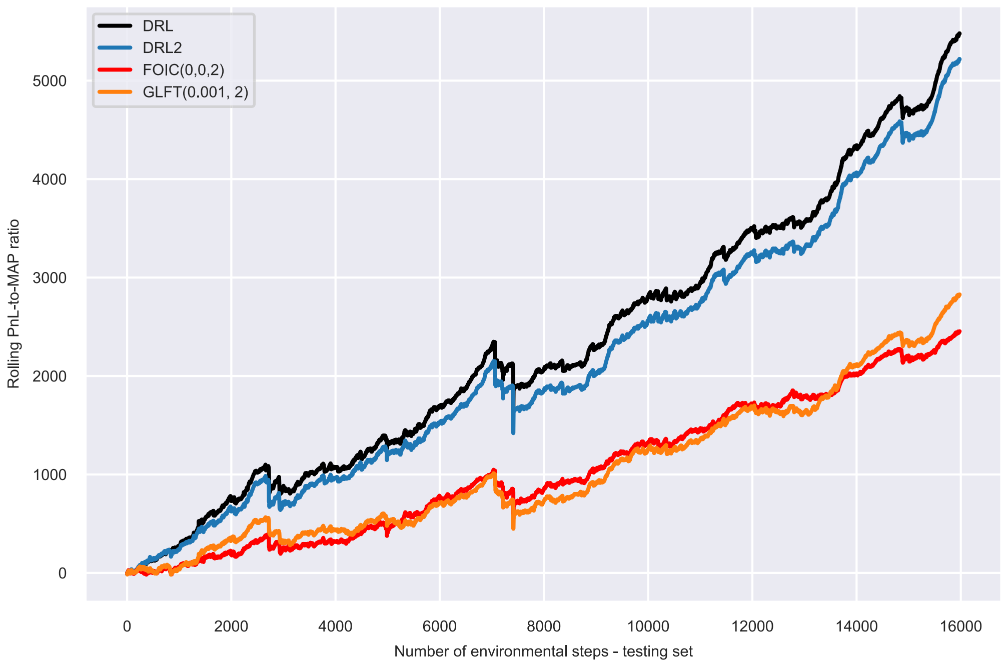


FIGURE 3. Rolling PnL-to-MAP ratio on the testing set for four of the considered strategies. The DRL variants significantly outperform the benchmarks, indicating more favorable return-to-risk performances.

TABLE 1. DRL agent vs benchmarks comparison.

Strategy	Ep. return	W_T	MAP	MDD	$\mathrm{PnLMAP}(T)$
DRL agent	2909.7	4066.7	0.742	-0.289	5478.1
DRL2 agent	2809.9	3838.8	0.736	-0.331	5217.8
FOIC(0, 0, 2)	-16254.6	3072.4	1.252	-0.475	2454.8
GLFT(0.01)	-12799.9	2388.8	0.992	-0.739	2407.3
GLFT(0.001)	-15028.9	3333.4	1.178	-0.557	2828.7
GLFT(0.0001)	-16065.4	3280.9	1.247	-0.554	2630.7

highest terminal wealth, as well as the most favorable MDD, all while having the second-lowest MAP. For example, notice that the DRL agent achieves more than 30% higher terminal wealth than the FOIC(0,0,2) while being exposed to less than 60% of its inventory risk. Similarly, the DRL agent by far outperforms the GLFT(0.001) benchmark with respect to all metrics. The difference in the performance between the DRL agents and the benchmarks seems to grow steadily (and apparently linearly) with time. Also notice that the DRL agent achieves both higher terminal wealth and lower maximum drawdown (MDD) than the DRL2 agent (the variant without AD). On the other hand, its MAP is slightly higher. This seems to suggest that the AD variant is better at deciphering *when* to keep relatively high inventory values without jeopardizing the PnL. Taken altogether, the DRL agent slightly outperforms the DRL2 agent while surpassing all other agents by a large margin. We remark that the performance drop that happens around the environmental step number 7300 corresponds to a sudden change in the BTC/USD value. Fig. 4 shows the agent's behavior on a small segment of the testing period. Generally, the inventory constantly fluctuates around zero, indicating MM behavior, as expected. Moreover, the inventory level is between -1 and 1 most of the time

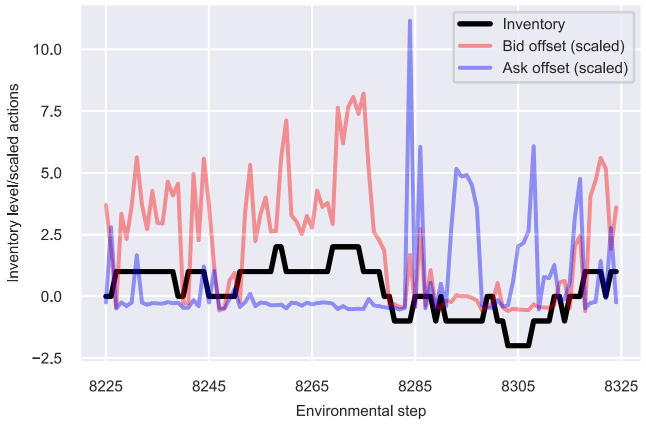


FIGURE 4. Typical DRL agent behavior. Observe that bid (ask) quotes are posted more conservatively (aggressively) when the inventory is positive, and vice versa. The actions are shown scaled (factor 0.2).

(89.06% of the time during the testing period), which clearly indicates high aversion to risk.

F. INTERPRETABILITY

In order to ameliorate the problem of poor interpretability generally plaguing deep learning approaches, in this section we interpret the behavior of the resulting DRL agent. To this end, we use partial dependence plots (PDPs) [31]. A PDP shows the marginal effect of an individual input variable x (or multiple input variables) on the model output $h(x)$, i.e., the dependent variable. Let us denote the set of all other input variables (besides x) by Y. Then the partial dependence is given by the expectation of g over the marginal distribution

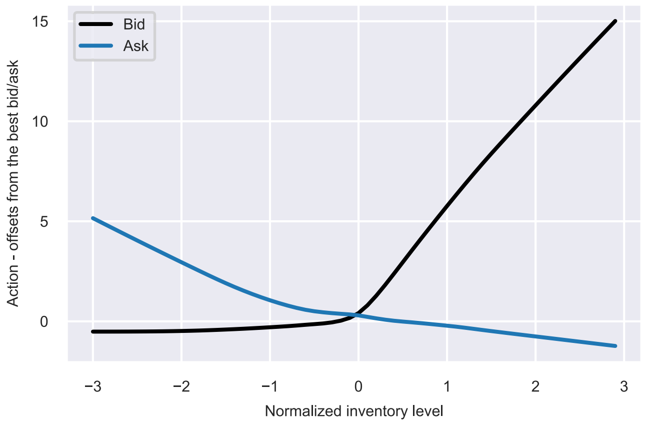


FIGURE 5. Dependence of the learned offsets from the best bid/ask (actions) on the normalized inventory level.

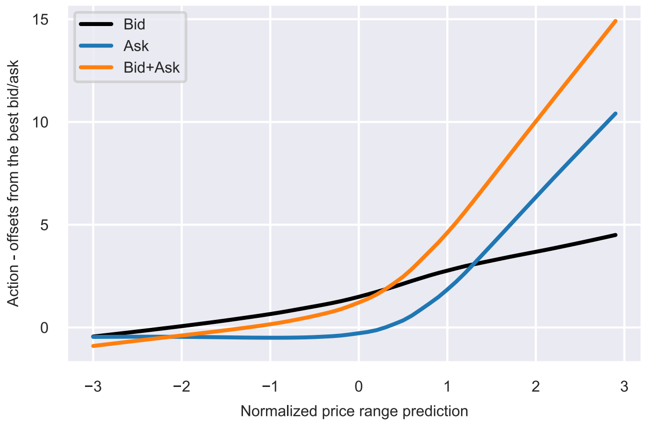


FIGURE 6. Dependence of the learned offsets from the best bid/ask (actions) on the realized price range predictions (the signal from the SGU1).

of all other input variables:

$$h_x = \mathrm{E}_Y \left[h(x, Y) \right] = \int h(x, Y)\, \mathrm{d}P(Y). \tag{18}$$

It can easily be estimated [39] by averaging over the training dataset ($\{(x_i, Y_i),\ i = 1, \ldots, n\}$) while keeping x fixed:

$$\hat{h}_x(x) = \frac{1}{n} \sum_{i=1}^{n} h(x, Y_i). \tag{19}$$

Fig. 5 depicts the dependence of the DRL agent's learned bid/ask offsets on the normalized inventory variable. Observe that the DRL agent's offsets from the best bid grow slowly and close to linearly when the inventory is negative and then again approximately linearly (but with a much larger slope) for positive inventory levels. Conversely, the offsets from the best ask decrease close to piecewise linearly as the inventory level is increasing. We also recall that, according to analytical GLFT approximations, the optimal bid-ask quotes depend linearly on the inventory, but it should be borne in mind that they, unlike our approach, do not take the current bid-ask spread into consideration. Overall, this behavior is expected since the DRL agent, after accruing a high level of inventory, posts ask orders aggressively to incentivize selling, and bid orders very conservatively to prevent further buying. On the other hand, if the DRL agent's inventory is negative, it posts bid orders aggressively to incentivize buying and ask orders very conservatively to prevent further selling. Interestingly, the slope for the bid curve seems to be significantly steeper. A possible explanation for both this and the difference in the curve shapes may partly lie in the asymmetry in the number of limit orders (i.e. the intensity of limit order arrivals) and their distribution on the two sides of the LOB. Lastly, note that the curves intersect very close to $I_t = 0$, as expected. Fig. 6 shows the dependence of the DRL agent's learned bid/ask offsets on the normalized realized price range predictions, similarly as before. Clearly, both quotes increase as realized price range predictions grow. Again, this is completely unsurprising, as higher realized price range predictions mean that the agent should indulge in more conservative quoting on both sides of the LOB, i.e., increase its quoted spread, since it has a higher probability of capturing a larger spread. The orange line in Fig. 6 represents the sum of the two offsets, i.e., the spread added by the DRL agent on top of the current bid-ask spread. This seemingly convex line also increases monotonously with the price range prediction signal, as was expected.

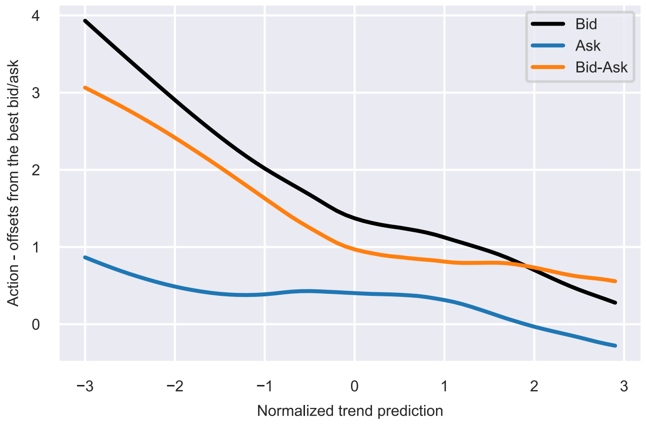


FIGURE 7. Dependence of the learned offsets from the best bid/ask (actions) on the trend predictions (the signal from the SGU2).

Finally, Fig. 7 depicts the dependence of the DRL agent's learned bid/ask offsets on the normalized trend predictions. First observe that the y-axis range is here smaller compared to Fig. 5 and 6. This stems from the fact that the trend signal is generally less informative and consequently less useful than the price range prediction signal, given that returns (and hence trends), unlike volatility, are notoriously difficult to predict [30]. For negative values of trend predictions, the mid-price is expected to drop, and therefore the ask orders should be posted much more aggressively than the bid orders, to increase the probability of the capture of the quoted spread. This is in accordance with Fig. 7. The difference in the aggressiveness between quoting on the bid and ask side is denoted by the orange line. As trend predictions increase, this difference tends to decrease, as expected, although it remains positive even for very large trend predictions. Again, an explanation probably partly lies in the shape of the LOB and its other microstructural properties. Furthermore, when the inventory level is low, which is commonly the case with a risk-averse MM agent, the inventory penalization term in the reward function incentivizes the agent to generally post quotes in such a way that the execution probabilities are similar for both the bid and the ask side, in order to prevent one side from getting filled more often, which would lead to larger inventory levels. Finally, it should be emphasized that the PDP simultaneously adjusts for other dependent variables as well, and hence any strong conclusions about the causal nature of the relationship are not warranted. However, the PDPs nevertheless provide an informative glimpse into the inner workings of the resulting DRL agent.

VI. CONCLUSION

We have developed a novel, comprehensive DRL-based framework for MM with signals incorporating signals from auxiliary standalone supervised learning SGUs. Unlike existing approaches, our model both incorporates additional predictive signals into the state space and uses a continuous, yet tick-based action space. The framework is underpinned by both ideas from gradient-free neuroevolutionary RL approaches and adversarial reinforcement learning. The resulting RL agent demonstrates superior performance over several standard MM benchmark strategies. Furthermore, our approach demonstrates the feasibility of using genetic algorithms (neuroevolution) for training DRL agents for MM. An additional emphasis is put on the often neglected, but due to the requirements of financial regulators highly important, issue of the interpretability of the obtained controls, via the use of PDPs. The main directions for future related research based on our work include the following. First, it would be interesting to generalize the framework to multi-asset MM, possibly with latent risk factors [38] as part of the state space. Second, price range and trend predictions could be incorporated not only in the state space but also into the reward function itself, perhaps as part of its inventory penalty term. Finally, quality-diversity optimization evolutionary algorithms such as MAP-Elites or other novelty search-based approaches, could possibly be used to find a wide variety of mutually diverse, high-quality MM strategies.

APPENDIX A
SIGNAL GENERATING UNIT–REALIZED PRICE RANGE

A. MODEL

Underpinning the SGU for realized price range prediction is a gradient boosting [31] model, a highly robust ensemble-based machine learning technique employing decision trees as weak learners. It is particularly suitable in the presence of mutually correlated predictors as it is unaffected by multicollinearity. The Python package XGBoost is used for the implementation of the unit.

B. LABELS

A (quite natural) candidate labeling method is given by the following. Let us denote the set of all prices at which the market buys (sells) took place during the i-th time-step by P_i^{buy} (P_i^{sell}). The labels are then given by:

$$y_i = \max\left(P_i^{\text{buy}}\right) - \min\left(P_i^{\text{sell}}\right). \tag{20}$$

This expression is quite similar to a measure of volatility called the realized (price) range [32] (also known as the high-low range), although it additionally accounts for the side of the LOB at which the trade takes place. From here forth we refer to it as *the modified realized price range*. Note that if either of the two sets is empty, which commonly occurs if Δt is short, the label is undefined. Now let us denote the set of all the best bid (ask) prices during the i-th time-step by P_i^{bid} (P_i^{ask}). We now define the labels in the following way:

$$y_i = \begin{cases} \max\left(P_i^{\text{buy}}\right) - \min\left(P_i^{\text{sell}}\right) & \text{if } P_i^{\text{buy}} \neq \emptyset, P_i^{\text{sell}} \neq \emptyset \\ \overline{P_i^{\text{ask}}} - \min\left(P_i^{\text{sell}}\right) & \text{if } P_i^{\text{buy}} = \emptyset, P_i^{\text{sell}} \neq \emptyset \\ \max\left(P_i^{\text{buy}}\right) - \overline{P_i^{\text{bid}}} & \text{if } P_i^{\text{sell}} = \emptyset, P_i^{\text{buy}} \neq \emptyset \\ \overline{P_i^{\text{ask}}} - \overline{P_i^{\text{bid}}} & \text{else} \end{cases}, \tag{21}$$

where $\overline{P_i}$ denotes the mean of P_i. Therefore, we simply replace the missing values by the mean best quote on the corresponding side of the LOB. Finally, we round the labels to two decimal places (since the tick size is \$0.01).

C. FEATURES

The following 23 features, derived from domain knowledge, are considered: the total number of trades in the previous p periods where $p \in \{1, 2, 3, 5, 10\}$, the bid-ask spread at the start of the time-step, the traded volume imbalance, the volume-weighted average price in the previous r periods where $r \in \{1, 3, 5\}$, the slope of the linear fit (OLS) of price versus time in the previous s periods where $s \in \{1, 3, 5\}$, the time of the day in hours, the total traded volume, the percentage of upticks, the total number of large buys, the total number of large sells, and lastly, the time-lagged labels (previous modified realized price ranges) with lags ranging from $L = 1$ to $L = 5$. Unless stated otherwise, features refer to the most recent previous period. The individual feature importance is then calculated, measured in terms of the gain. Our experimentation shows that removing $K = 3$ features with lowest gains improves the algorithm performance, and hence we continue training after omitting the following features: the percentage of upticks, the total number of large buys, the total number of large sells.

D. MODEL AND TRAINING DETAILS

In order to find suitable hyperparameters, we perform cross-validation on the combined training and validation set. Key hyperparameters are selected via stepwise grid search and additional hyperparameters are set to common, sensible

TABLE 2. Extreme gradient boosting hyperparameter values and grid search ranges.

Hyperparameter	Range	Value
`max_depth`	{2,3,...,7}	4
`min_child_weight`	{1,2,...,5}	4
`subsample`	{0.6,0.7, ...,1.0}	1.0
`col_sample_bytree`	{0.6,0.7, ...,1.0}	1.0
`learning_rate`	{0.3,0.2,0.1,0.05,0.01,0.005}	0.01
`regularization_term`	{0.1, 1.0, 5.0, 10.00}	10.00
`early_stopping_rounds`	-	5
`num_boost_round`	-	1000
`n_fold`	-	5

values. All hyperparameter values as well as the ranges used for the grid search are given in Table 2. After hyperparameter optimization, we train the model again on the combined training and validation set. The mean square error (MSE) is used as the evaluation metric.

APPENDIX B SIGNAL GENERATING UNIT–TREND

A. MODEL

The trend prediction SGU is based on a long short-term memory (LSTM) NN, a special type of recurrent neural network (RNN) capable of grasping long-term dependencies. Due to their ability to take into account sequential dependencies, LSTMs are extensively used for time series forecasting, including financial time series prediction, and hence provide a natural choice here.

B. LABELS

We employ the following labeling method. First, we define the "pseudo-mid-prices":

$$m_i = \begin{cases} \frac{1}{2}\left(\max\left(P_i^{\text{buy}}\right) + \min\left(P_i^{\text{sell}}\right)\right) & \text{if } P_i^{\text{buy}} \neq \emptyset, P_i^{\text{sell}} \neq \emptyset \\ \frac{1}{2}\left(\overline{P_i^{\text{ask}}} + \min\left(P_i^{\text{sell}}\right)\right) & \text{if } P_i^{\text{buy}} = \emptyset, P_i^{\text{sell}} \neq \emptyset \\ \frac{1}{2}\left(\max\left(P_i^{\text{buy}}\right) + \overline{P_i^{\text{bid}}}\right) & \text{if } P_i^{\text{sell}} = \emptyset, P_i^{\text{buy}} \neq \emptyset \\ \frac{1}{2}\left(\overline{P_i^{\text{ask}}} + \overline{P_i^{\text{bid}}}\right) & \text{else ,} \end{cases} \tag{22}$$

using identical notation as in App. A-B. Then the labels are simply given as simple financial returns:

$$y_i = \frac{m_i - m_{i-1}}{m_{i-1}}. \tag{23}$$

The resulting labels can be interpreted as simple "pseudo-returns" based on "pseudo-mid-prices".

C. FEATURES

Lagged pseudo-returns (labels) with lags ranging from $L = 1$ to $L = 10$ are used as predictors. We use z-score normalization as a feature scaling technique:

$$x' = \frac{x - \bar{x}}{\sigma_x}, \tag{24}$$

where $\bar{x}$ and σ_x denote the mean and the standard deviation of the input variable x, respectively.

D. MODEL AND TRAINING DETAILS

It should be noted that the NN consists of a single hidden layer with 10 LSTM units with the tanh activation function. The output layer is a dense layer with one neuron and the linear activation function. The validation set is used to find the suitable hyperparameter values, listed in Table 3. The mean squared error (MSE) is selected as the metric, whereas the optimization process is performed with the Adam optimizer with its default parameters.

TABLE 3. Hyperparameter values.

Hyperparameter	Value
Hidden layer bias regularizer type	L2
Hidden layer bias regularizer value	0.01
Hidden layer kernel regularizer type	L2
Hidden layer kernel regularizer value	0.01
Dropout rate	0.8
Number of epochs	20
Batch size	128

APPENDIX C REINFORCEMENT LEARNING TRAINING DETAILS

For the implementation of the main RL unit, the open-source deep learning framework PyTorch is used. The selected hyperparameters are given in Table 4. We note that early stopping on the validation set is used as a regularization technique. Additionally, the use of adversarial perturbations and (relatively) shallow DNNs also serves to prevent overfitting.

TABLE 4. RL training hyperparameters.

Hyperparameter	Value
Discount rate γ	1
MM agent NN output scaling factor	5
Adversary NN output scaling factor	0.05
Population size	50
Initial mutation rate	0.02
Mutation rate decay criterion	no improvement in 15 generations
Mutation rate decay type	exponential
Mutation rate decay factor	0.5
Terminal mutation rate	0.0025
Truncation size	5
Adversary power	500

ACKNOWLEDGMENT

The authors would like to thank the three anonymous reviewers for their suggestions and comments which proved helpful in enhancing the quality of the article.

REFERENCES

[1] B. Hagströmer and L. Nordén, "The diversity of high-frequency traders," *J. Financial Markets*, vol. 16, no. 4, pp. 741–770, Nov. 2013.
[2] M. Avellaneda and S. Stoikov, "High-frequency trading in a limit order book," *Quant. Finance*, vol. 8, no. 3, pp. 217–224, Apr. 2008.
[3] T. Ho and H. R. Stoll, "Optimal dealer pricing under transactions and return uncertainty," *J. Financial Econ.*, vol. 9, no. 1, pp. 47–73, Mar. 1981.

[4] O. Guéant, C.-A. Lehalle, and J. Fernandez-Tapia, "Dealing with the inventory risk: A solution to the market making problem," *Math. Financial Econ.*, vol. 7, no. 4, pp. 477–507, Sep. 2013.

[5] P. Fodra and M. Labadie, "High-frequency market-making with inventory constraints and directional bets," HAL, Bengaluru, India, Tech. Rep. hal-00675925, Jun. 2012.

[6] F. Guilbaud and H. Pham, "Optimal high-frequency trading with limit and market orders," *Quant. Finance*, vol. 13, no. 1, pp. 79–94, Jan. 2013.

[7] Á. Cartea, S. Jaimungal, and J. Ricci, "Algorithmic trading, stochastic control, and mutually exciting processes," *SIAM Rev.*, vol. 60, no. 3, pp. 673–703, Jan. 2018.

[8] Á. Cartea, R. Donnelly, and S. Jaimungal, "Algorithmic trading with model uncertainty," *SIAM J. Financial Math.*, vol. 8, no. 1, pp. 635–671, Jan. 2017.

[9] Á. Cartea, R. Donnelly, and S. Jaimungal, "Enhancing trading strategies with order book signals," *Appl. Math. Finance*, vol. 25, no. 1, pp. 1–35, Jan. 2018.

[10] V. Mnih, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Oct. 2015.

[11] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Singapore, May 2017, pp. 3389–3396.

[12] Y. S. Lim and D. Gorse. (2018). *Reinforcement Learning for High-Frequency Market Making*. [Online]. Available: https://discovery.ucl.ac.uk/id/eprint/10116730/

[13] T. Spooner, J. Fearnley, R. Savani, and A. Koukorinis, "Market making via reinforcement learning," in *Proc. 17th Int. Conf. Auton. Agents Multiagent Syst.*, 2018, pp. 1–8. [Online]. Available: https://smallake.kr/wp-content/uploads/2019/01/1804.04216.pdf

[14] J. Sadighian, "Deep reinforcement learning in cryptocurrency market making," 2019, *arXiv:1911.08647*. [Online]. Available: http://arxiv.org/abs/1911.08647

[15] P. Kumar, "Deep reinforcement learning for market making," in *Proc. 19th Int. Conf. Auto. Agents MultiAgent Syst.*, Auckland, New Zealand, 2020, pp. 1892–1894.

[16] O. Guãant and I. Manziuk, "Deep reinforcement learning for market making in corporate bonds: Beating the curse of dimensionality," *Appl. Math. Finance*, vol. 26, no. 5, pp. 387–452, Sep. 2019.

[17] S. Ganesh, N. Vadori, M. Xu, H. Zheng, P. Reddy, and M. Veloso, "Reinforcement learning for market making in a multi-agent dealer market," 2019, *arXiv:1911.05892*. [Online]. Available: http://arxiv.org/abs/1911.05892

[18] F. Petroski Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," 2017, *arXiv:1712.06567*. [Online]. Available: http://arxiv.org/abs/1712.06567

[19] Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning," in *Proc. Intell. Syst. Conf.*, London, U.K., Sep. 2017, pp. 905–913.

[20] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, 2017, pp. 2817–2826.

[21] T. Spooner and R. Savani, "Robust market making via adversarial reinforcement learning," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Yokohama, Japan, Jul. 2020, pp. 4590–4596.

[22] L. Leal, M. Laurière, and C.-A. Lehalle, "Learning a functional control for high-frequency finance," 2020, *arXiv:2006.09611*. [Online]. Available: http://arxiv.org/abs/2006.09611

[23] M. D. Gould, M. A. Porter, S. Williams, M. McDonald, D. J. Fenn, and S. D. Howison, "Limit order books," *Quant. Finance*, vol. 13, no. 11, pp. 1709–1742, Nov. 2013.

[24] C. Yingsaeree, "Algorithmic trading: Model of execution probability and order placement strategy," Ph.D. dissertation, Dept. Comp. Sci., UCL Univ. College London, London, U.K., 2012.

[25] I. M. Toke, "'Market making' behaviour in an order book model and its impact on the bid-ask spread," in *Econophysics of Order-driven Markets*. Milan, Italy: Springer, 2011.

[26] Ä. Cartea, S. Jaimungal, and J. Penalva, "Market making," in *Algorithmic High-Frequency Trading,* 1st ed. Cambridge, U.K.: Cambridge Univ. Press, 2015, ch. 10, pp. 246–272.

[27] A. Sehgal, H. La, S. Louis, and H. Nguyen, "Deep reinforcement learning using genetic algorithm for parameter optimization," in *Proc. 3rd IEEE Int. Conf. Robotic Comput. (IRC)*, Naples, Italy, Feb. 2019, pp. 596–601.

[28] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," in *Proc. ICLR*, 2014, pp. 1–5. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.747.8097

[29] J. F. Tapia, "Modeling, optimization and estimation for the on-line control of trading algorithms in limit-order markets," Ph.D. dissertation, General Math., Univ. Pierre et Marie Curie-Paris VI, Paris, France, 2015.

[30] G. Zhou, "How much stock return predictability can we expect from an asset pricing model?" *Econ. Lett.*, vol. 108, no. 2, pp. 184–186, Aug. 2010.

[31] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[32] M. Martens and D. van Dijk, "Measuring volatility with the realized range," *J. Econometrics*, vol. 138, no. 1, pp. 181–207, May 2007.

[33] X. Gao and Y. Wang, "Optimal market making in the presence of latency," *Quant. Finance*, vol. 20, no. 9, pp. 1495–1512, Sep. 2020.

[34] R. Li, Z. Zhao, Q. Sun, C.-L. I, C. Yang, X. Chen, M. Zhao, and H. Zhang, "Deep reinforcement learning for resource management in network slicing," *IEEE Access*, vol. 6, pp. 74429–74441, 2018.

[35] N. D. Nguyen, T. Nguyen, and S. Nahavandi, "System design perspective for human-level agents using deep reinforcement learning: A survey," *IEEE Access*, vol. 5, pp. 27091–27102, 2017.

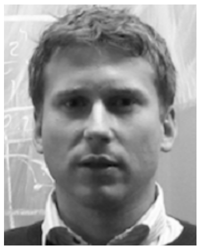
[36] H. Buehler, L. Gonon, J. Teichmann, and B. Wood, "Deep hedging," *Quant. Finance*, vol. 19, no. 8, pp. 1271–1291, Aug. 2019.

[37] B. Gasperov, F. Saric, S. Begusic, and Z. Kostanjcar, "Adaptive rolling window selection for minimum variance portfolio estimation based on reinforcement learning," in *Proc. 43rd Int. Conv. Inf., Commun. Electron. Technol. (MIPRO)*, Opatija, Croatia, 2020, pp. 1098–1102.

[38] S. Begusic and Z. Kostanjcar, "Cluster-specific latent factor estimation in high-dimensional financial time series," *IEEE Access*, vol. 8, pp. 164365–164379, 2020.

[39] Q. Zhao and T. Hastie, "Causal interpretations of black-box models," *J. Bus. Econ. Statist.*, vol. 39, no. 1, pp. 272–281, Jan. 2021.

BRUNO GAŠPEROV received the M.Sc. degree in quantitative finance from the Vienna University of Economics and Business, in 2016. He is currently pursuing the Ph.D. degree in computer science with the Faculty of Electrical Engineering and Computing, University of Zagreb. He is currently a Research Associate with the Laboratory for Financial and Risk Analytics, Faculty of Electrical Engineering and Computing, University of Zagreb. His main research interests include deep reinforcement learning, genetic algorithms, and portfolio optimization.



ZVONKO KOSTANJČAR (Member, IEEE) received the Dipl.-Ing. degree from the Faculty of Electrical Engineering and Computing, University of Zagreb, Zagreb, Croatia, in 2002, the degree in financial mathematics from the Faculty of Science, in 2008, and the Ph.D. degree from the University of Zagreb, in 2010. He is currently an Associate Professor with the Faculty of Electrical Engineering and Computing, University of Zagreb, and the Head and the Founder of the Laboratory for Financial and Risk Analytics. He has served as the President for the Signal Processing Chapter, IEEE Croatia Section. His main research interests include statistical and machine learning methods for finance.