Neural Networks 165 (2023) 506–515

Contents lists available at ScienceDirect

Neural Networks

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

Modeling limit order trading with a continuous action policy for deep reinforcement learning

Avraam Tsantekidis *, Nikolaos Passalis, Anastasios Tefas

School of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 29 August 2022
Received in revised form 20 January 2023
Accepted 28 May 2023
Available online 2 June 2023

Keywords:
Financial trading
Limit orders
Policy gradient
Deep reinforcement learning

ABSTRACT

Limit Orders allow buyers and sellers to set a "limit price" they are willing to accept in a trade. On the other hand, market orders allow for immediate execution at any price. Thus, market orders are susceptible to slippage, which is the additional cost incurred due to the unfavorable execution of a trade order. As a result, limit orders are often preferred, since they protect traders from excessive slippage costs due to larger than expected price fluctuations. Despite the price guarantees of limit orders, they are more complex compared to market orders. Orders with overly optimistic limit prices might never be executed, which increases the risk of employing limit orders in Machine Learning (ML)-based trading systems. Indeed, the current ML literature for trading almost exclusively relies on market orders. To overcome this limitation, a Deep Reinforcement Learning (DRL) approach is proposed to model trading agents that use limit orders. The proposed method (a) uses a framework that employs a continuous probability distribution to model limit prices, while (b) provides the ability to place market orders when the risk of no execution is more significant than the cost of slippage. Extensive experiments are conducted with multiple currency pairs, using hourly price intervals, validating the effectiveness of the proposed method and paving the way for introducing limit order modeling in DRL-based trading.

1. Introduction

Trading in financial markets using Artificial Intelligence (AI) and Machine Learning (ML) has gained traction in past years. A large portion of the daily traded volumes in the markets originates from automated trading agents (Haynes & Roberts, 2015). The improved accessibility to financial markets and the proliferation of Crypto-Currencies have increased the use of Deep Learning (DL) methods to capture trading opportunities. New services provide easy access to Application Programming Interfaces, allowing for straightforward deployments of automated agents to trade in the market. The confluence of these developments will lead to further increases in the automated trading volume trend.

The research into automated methods to determine the optimal approach for extracting profits from financial markets has gone through several iterations. An important milestone of these strategies has been the use of technical indicators, which try to assign measurable attributes to price time series that can forecast some of the statistical properties of future price data points (Frankel & Froot, 1990; Taylor & Allen, 1992). These methods, although having some success when applied in the markets, have received criticism because they break the Efficient Market Hypothesis (EMH) which states that asset prices "fully reflect" all pertinent information that affects them (Malkiel & Fama, 1970). Thus EMH requires that any available information, such as past prices or their attributes, are not correlated with changes in the future. It is worth noting that the EMH has been scrutinized in some studies (Froot, Scharfstein, & Stein, 1992), since the agents participating in the market do not have long horizons and may rely on the same information to make short term decisions.

More recently, Machine Learning (ML) methods received research attention as a tool to model the price fluctuations in the markets (Jarusek, Volna, & Kotyrba, 2022; Tsantekidis et al., 2017; Zhang, Li, Wang, Fang, & Philip, 2019). Most of these methods directly model the expected price directions, in a supervised fashion, and capitalize on the correct predictions by placing the corresponding market orders. However, as described in Moody, Wu, Liao, and Saffell (1998), trying to use supervised learning methods in this setting comes with significant drawbacks, since the final objective is disconnected from the learning objective. Using more appropriate methods allows for directly optimizing the objective that is of interest, e.g., the final profit, without using intermediate proxies. The breakthroughs in the performance of recently proposed Reinforcement Learning (RL) algorithms have led many of them to be applied to new tasks. Many of those tasks were previously considered infeasible for

* Corresponding author.
E-mail addresses: avraamt@csd.auth.gr (A. Tsantekidis), passalis@csd.auth.gr (N. Passalis), tefas@csd.auth.gr (A. Tefas).

https://doi.org/10.1016/j.neunet.2023.05.051
0893-6080/

computers to compete against humans, yet the tables seem to have turned. New RL algorithms have been proposed combining traditional RL methods with Deep Learning, leading to powerful Deep Reinforcement Learning (DRL) formulations (Dilokthanakul, Kaplanis, Pawlowski, & Shanahan, 2019; Mnih et al., 2013; Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017; Wang, Elfwing, & Uchibe, 2021). Typical examples of these methods include Deep Q-Learning (Mnih et al., 2013, 2015) and Policy Gradient methods (Lillicrap et al., 2015; Schulman et al., 2017) which have surpassed previous performances. Such approaches have been employed with great success over a wide range of different tasks, such as surpassing human performance in games (Silver et al., 2016, 2017), emotion and sentiment analysis (Peng, Ma, Poria, Li, & Cambria, 2021; Zhang, Li, Wang, Cambria and Li, 2021), sound applications (Latif et al., 2022), adversarial attack design (Li, Pan, & Cambria, 2022) and even incorporating human feedback into model training (Christiano et al., 2017).

There are generally two types of orders in trading: *market orders* and *limit orders*. A market order is an order to buy or sell an asset immediately at the best available price. Market orders rely on the liquidity of the asset's order book to be executed. Such orders are executed against the currently available prices. As a result, price slippage typically occurs based on the current market conditions, e.g., liquidity and volatility. On the other hand, a trader can also specify a *limit price* for the buy or sell order submitted. This type of order is called *limit order*. By setting a limit it may either be executed immediately against an existing limit or it may be added to the open limit orders already on the market. Market participants have a view of the open orders and can react according to how the demand for each price level moves. Any open limit order may be executed against an incoming limit or market order, depending on its priority based on its price and order of arrival. Otherwise, the *limit order* is added to the order book awaiting execution from either an opposing limit or market order. Most of the recent research examines the performance of DRL algorithms using a somewhat limited scope as it pertains to the capability of a candidate trader model using market orders, while typically auxiliary actions, such as a *stay* action (do nothing) and an *exit* action (exit whatever position currently held) are also included.

To overcome this limitation, in this work we introduce a framework that enables the inclusion of limit orders in the action space of financial trading agents. To this end, we employ the Beta distribution (Bernardo & Smith, 2009) for modeling limit prices of the limit orders. Objective masking and reward shaping, as further presented in Section 3, are used for the limit price prediction part of the agent to improve the training stability. Furthermore, we allow the agent to select between placing a market order and a limit order, thus providing the same set of options that are available to human traders. Similar approaches, that decompose the task at hand to two or more tasks, are prevalent in the robotics and computer vision community. For example, robotics controllers separately learn to control multiple axes (Katyal, Wang, Burlina, et al., 2017; Li, Zhang, & Zhao, 2019; Sui, Pu, Yi, & Wu, 2020), object detection is typically tackled as solving both classification and regression tasks (Lin, Jia, Huang, & Gao, 2022; Liu et al., 2016), and multi-task learning is increasingly applied for a wide range of different domains (Dorado-Moreno et al., 2020; Yang et al., 2020). Therefore, the proposed framework provides a comprehensive alternative to the current status quo of DRL research into financial trading. It accurately models a more realistic set of actions, paving the way for including limit order modeling in many different DRL trading agents. Limit orders open an avenue for these agents for performing more efficient trades by reducing slippage losses, mainly incurred from market orders, at the cost of some additional risk of partially unfilled orders.

The rest of the paper is structured as follows. First, the related work is briefly presented and discussed in Section 2. Then, the proposed method is introduced and analytically derived in Section 3. The experimental setup, along with the experimental evaluation using multiple currency pairs, are presented in Section 4. Finally, Section 5 concludes the paper.

2. Related work

Since the early days of electronic trading in the financial markets, indicators and algorithms were developed to profit from repeating patterns in the market. Technical Analysis (TA) was developed by measuring statistical attributes of the price movements or the trading behaviors of assets and creating rule based algorithms on how to use them (Taylor & Allen, 1992). More advanced non-linear statistical methods in the realm of Machine Learning (ML) exist that have been used in the past to make predictions based on ground truth annotations regarding the future behavior of an asset. Such methods can be regression-based or classification-based. In both cases, the ground truth annotations provide continuous or discrete targets, according to which the ML models are trained. More specifically, regression-based methods directly predict future attributes of a price time series, such as the price itself (Ariyo, Adewumi, & Ayo, 2014). On the other hand, classification-based methods create labels for each time step based on an intended action, such as buying or selling, and train a model on it (Kercheval & Zhang, 2015; Tsantekidis et al., 2017; Zhang et al., 2019). In both of those cases, if the resulting model is accurate enough, it is trivial to create a profitable strategy from its predictions.

However, it has been shown in Moody et al. (1998) that relying wholly on a supervised model for trading decisions leads to sub-optimal performance. Using regression to predict the future price and trading according to the predictions, leads to a bottleneck between the price forecaster and the trade emitter, since the information available to the trade emitter is just the price forecast. This situation is known as *forecasting bottleneck* (Moody et al., 1998). Similarly, directly predicting trading decisions as labels to bypass this bottleneck has its own problems. Static labels for each sample create a *temporal credit assignment*, because they do not consider the commission costs incurred, thus being most likely to underperform. In order to avoid defining supervised targets, one can optimize a model directly from the reward signal of the task using Reinforcement Learning (RL) as suggested by Moody and Saffell (2001) and Moody et al. (1998).

Directly optimizing a reward signal, allows a model to consider all effects on its Profit and Loss (PnL), allowing it to construct a reliable longer-term trading policy. Since the first published works on RL applied in trading such as Moody and Saffell (2001), newer Policy Gradient (PG) based RL algorithms have been developed, such as the Trust-Region Policy Optimization (TRPO) (Schulman, Levine, Abbeel, Jordan, & Moritz, 2015) which propose constraining the updates of the action distribution of its learned policy, ensuring that highly rewarding trajectories do not dominate exploration. This allows the agent to smoothly explore its environment and yield better results. One simplification of this work comes from Proximal Policy Optimization (Schulman et al., 2017), where the costly computation of TRPO bounding is removed and replaced with an additional objective that loosely attempts to enforce the same constraint. The improvement retains the same performance of TRPO while trimming the implementation and computational costs.

There have been many potential applications of such PG-based algorithms for financial trading recently, such as Tsantekidis, Passalis, and Tefas (2021), Tsantekidis et al. (2020) and Zhang, Jiang and Su (2021), where a PG-based algorithm is used to select

trading actions. All these approaches only consider market orders, whose consequences of their use are realized when a system is used in live trading and large slippage costs are accrued. Having an accurate model of costs can help with simulating the behavior of market orders. Employing limit orders can mostly alleviate slippage costs, but in return, they may cause orders to not be executed, so choosing one instead of the other has its trade-offs and one cannot claim only one should be applied when trading.

There have been some attempts to use RL to manage limit order placement such as Nevmyvaka, Feng, and Kearns (2006), where a Q-learning RL methodology, which enables training RL learning with discrete policies, was used for the limit price selection. Similar prediction methods for limit order price levels were carried out in He and Lin (2021) and Karpe, Fang, Ma, and Wang (2020). However, to the best of our knowledge, this is the first time a DRL framework for learning a continuous limit price policy is proposed for financial trading, along with the usual market order policy, bringing the capabilities of DRL agents closer to the ones of regular traders. The novelty of our work also extends to the fact that we are learning two distinct policies with different characteristics (one continuous and one discrete). We also show that applying reward shaping for noisy tasks such as the limit policy price selection can benefit the final agent. Please also note that optimal limit order pricing is extensively studied in the context of market making, ranging from classical works (Avellaneda & Stoikov, 2008) to reinforcement learning-based approaches (Ganesh et al., 2019; Spooner, Fearnley, Savani, & Koukorinis, 2018). However, the proposed method does not target market making, since the goal of the proposed approach is to allow a profitable limit price selection for order submissions regardless of the time horizon. Instead, the proposed method aims to simulate a regular investor on day-level (or similar horizon) decision-making instead of performing market making and maintaining a neutral position. Finally, the proposed method is also related to approaches that employ a hybrid action space, such as Li et al. (2021) and Neunert et al. (2020), since it employs a discrete action space for deciding the position on the market and a continuous action space for deciding the appropriate limits. However, to the best of our knowledge, this is the first time that such an approach has been used for limit order trading.

3. Proposed method

This paper focuses on financial trading using Deep Reinforcement Learning (DRL) and theoretically grounding the use of limit orders in the context of Policy Gradients (PG). In the following section, the PG approach is presented, which is an augmented version of the Proximal Policy Optimization (PPO). This PG approach is applied for training two separate policies, a *trade action policy* (or market policy) and a *limit price policy* (or limit policy). The proposed method is laid out in detail, with appropriate explanations about the choices made for the employed continuous distribution and the reward shaping scheme used to effectively train the limit price probability distribution.

3.1. Reinforcement learning methodology

In this work we utilize a Policy Gradient (PG) approach to train our agent and more specifically we use a modified version of the Proximal Policy Optimization (PPO) (Schulman et al., 2017). The objective of PG algorithms is to directly optimize the policy of an agent to maximize the reward received, by applying the Policy Gradient Theorem (Sutton & Barto, 2018). In particular it takes advantage of the ratio of policy probabilities as the policy changes:

$$r_t(\theta) = \frac{\pi_\theta(\alpha|s_t)}{\pi_{\theta_{\text{old}}}(\alpha|s_t)}, \tag{1}$$

where θ denotes the parameters of policy π, $\pi_\theta(\alpha|s_t)$ denotes the probability that policy π will select action α when the agent observes the environment state s_t, and t denotes the current time step. Following literature conventions, the parameterized notation $r_t(\theta)$ (Schulman et al., 2017) is used to refer to the ratio of policy probabilities, while the plain notation r_t is used to refer to the reward at time t. By employing a clipped ratio $r_t(\theta)$ around the value of 1 within ϵ, the policy exploration can be constrained to the close vicinity in the parameter space. The clipped policy ratio is defined as:

$$r_t^{\text{clip}}(\theta) = \text{clip}(r_t(\theta), 1 - \epsilon_{\text{ppo}}, 1 + \epsilon_{\text{ppo}}), \tag{2}$$

where ϵ_{ppo} constrains the update step of the policy to a limited range based on its value. The final objective function is defined as:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min(r_t(\theta)A_t, r_t^{\text{clip}}(\theta)A_t)\Big], \tag{3}$$

where A_t is the advantage estimate. The term advantage refers to the excess return expected for taking a specific action in a given state compared to the baseline return of that state. The return is the sum of discounted future rewards. The baseline return is usually set to the average return of the all possible trajectories stemming from the current state, weighted by the stationary distribution of the Markov chain of π_θ. When an action α results in a rewarding trajectory with positive A the objective encourages the policy $\pi_\theta(\alpha|s_t)$ to increase the assigned probability of the action. Similarly the same objective would decrease the probability of the action if it had a negative A. This results in the increase or decrease of the ratio defined in Eq. (1) since the "old" probability $\pi_{theta_{\text{old}}}$ stays the same. Taking the minimum between $r_t(\theta)A_t$ and $r_t^{\text{clip}}(\theta)A_t$ allows for large negative advantages that adversely impact performance to proportionately affect the objective, but limits the effect of large positive advantages. This constraint limits the changes in the policy based on actions that yield excessive positive rewards, but allows for larger changes that avoid excessive negative rewards. This results in a conservative exploration of the positively rewarded action paths without rushed policy updates towards the most obvious reward incline.

In order to take advantage of limit orders, the mechanism that we use is that of limit price regression using a continuous action policy (instead of discrete) in the context of DRL. Using PG-based methods, continuous actions can be sampled across multiple trials of the environment by the agent and based on the rewards received from each trial, the selected action will shift towards values that return higher rewards. Since the agent we present also outputs a discrete action, which determines the type of order to submit, a more elaborate framework must be used to combine the two actions and successfully increase the reward signal. Both the trade action policy, as well as the limit price policy are described in the subsequent subsections.

3.1.1. Trade action policy

The discrete action of the agent consists of 5 options and is chosen by the market policy of the agent $a_t^{(\tau)} \sim \pi^{(\tau)}(\cdot|s_t)$, where $\pi^{(\tau)}$ denotes a multinomial distribution with the number of actions set to $n = 5$. The discrete trade action space is $\mathcal{A} = \{-1_l, -1, 0, 1, 1_l\}$, where $(\cdot)_l \in \mathcal{A}$ represents the limit orders. The action denoted as $a_t^{(\tau)} = 0$ is the exit action, which uses a market order to exit the current position. Then actions $a_t^{(\tau)} = 1$ and $a_t^{(\tau)} = -1$ denote the market buy and sell orders respectively. Finally, $a_t^{(\tau)} = 1_l$ and $a_t^{(\tau)} = -1_l$ denote the limit buy and sell order respectively. When an agent selects $a_t^{(\tau)} \in \{-1, 0, 1\}$ a market order is executed on the open price of the following time step. If the agent selects $a_t^{(\tau)} \in \{1_l, -1_l\}$ then a limit order is submitted

along with a limit price selected by the continuous action policy of the agent. If the agent already has a long position and selects $a_t^{(\tau)} \in \{1, 1_l\}$ then the position is simply maintained without any new trade request submitted. The same thing also is true when the agent has a short position and selects $a_t^{(\tau)} \in \{-1, -1_l\}$, i.e., no trading action is executed and the position stays unchanged. If the agent has no position in the market and the agent selects the exit action $a_t^{(\tau)} = 0$ then the agent remains out of the market. The discrete action network employs a Neural Network (NN) model that is equipped with 5 output neurons and uses the softmax activation function to define a probability distribution over the actions. Note that the softmax activation function is used to normalize the output values of a neural network producing a discrete probability distribution.

3.2. Limit price policy

The trade action policy can select between the five discrete actions, two of which are the limit buy and limit sell actions. When these actions are selected a limit price must be set for the order to be submitted. The limit price selection policy is defined for this purpose. This policy must emit a continuous probability distribution over the possible limit prices the agent may sample from. Such distribution will also facilitate the calculation of the PG objective function which requires a measure of action probability, or cumulative distribution function in the case of continuous distributions. We define two separate continuous actions for the limit order price policy, to determine the bid and ask prices, i.e., $(a^{(1_l)}, a^{(-1_l)}) \sim \pi^{(l)}(\cdot|s_t)$. The "bid price" is the price offered to purchase an asset, while the "ask price" is the price requested to sell an asset, hence the names bid and ask price. Absolute price values have different scales across assets compared to price percentage changes, so we opt for our limit price policy to model a percentage distance from a reference price to place the limit order. The reference price we use is the market execution price. When the agent selects a limit bid price with 0.1% distance from the reference price, the order is placed 0.1% below the open price. Please note that we do not perform high frequency trading and we do not directly utilize any orderbook information as input to the model. Instead, we infer the status of our open orders based on the price of the executions in the OHLC candle of the next time step. If the execution prices cross our order price we consider our order executed. As a result, we do not need to reconstruct all the order book information and the simulation environment can be supported using widely available data.

The Beta distribution is employed as the continuous distribution since it has been shown to work better in RL settings (Chou, Maturana, & Scherer, 2017) by providing a smoother bounding for the continuous action possible range of values. By bounding the possible values of the continuous limit price action, we avoid the noise caused by sampling the long tails of distributions such as the normal or exponential. The inherent boundaries of the Beta distribution sample values are (0, 1). We could directly use the (0, 1) range of the Beta distribution for our percentage distances, which would mean that the percentage distances for the orders would range from 0% to 100%. This would waste most of the distribution bounds since price movements and limit orders rarely exceed single digits in an hourly interval. We scale the Beta distribution range by applying a linear transformation, so the emitted values are in the range $(\rho_{\min}, \rho_{\max})$. This range is chosen based on the quantile statistic of the percentage distance measure between the high p_h and low p_l price to the market execution price. We only use the prices from the training set of our data. The calculation of $\rho_{\min}$ and $\rho_{\max}$ is done at the beginning of training and it has the same range for all our experiments. The quantile that we have chosen is the 0.6 quantile of the observed values in the training set. To transform the values $a^{(1_l)}$ and $a^{(-1_l)}$ emitted from the Beta distribution to the intended $(\rho_{\min}, \rho_{\max})$ we do:

$$\widehat{a^{(1_l)}} = a^{(1_l)}(\rho_{\max} - \rho_{\min}) + \rho_{\min}, \tag{4}$$

$$\widehat{a^{(-1_l)}} = a^{(-1_l)}(\rho_{\max} - \rho_{\min}) + \rho_{\min}. \tag{5}$$

This type of Beta sample value transformation is not new and it is similar to the one described in Chou et al. (2017). However, the proposed quantile ranges offer more flexibility. This transformation helps to bring the distribution values closer to the average expected values for the limit orders.

The transformed values are then treated as the percentage change from the market execution price, which we define as the open price for the market agent. We calculate the distance for the limit price as follows:

$$p^{(1_l)}(t) = (1 - \widehat{a^{(1_l)}})p_o(t), \tag{6}$$

$$p^{(-1_l)}(t) = (1 + \widehat{a^{(-1_l)}})p_o(t), \tag{7}$$

where p_o is the open price, while $p^{(1_l)}(t)$ and $p^{(-1_l)}(t)$ are the prices selected for the buy and sell limit orders, respectively.

The limit price policy head of our model controls the parametrization of the Beta distribution, shifting the probability density towards the estimated profitable value ranges. The limit price policy head has a 4-neuron output emitting 4 values, which parametrize the two Beta distributions. The first two values are the α and β for the bid limit price Beta distribution and the last two values are the α and β for the ask limit price Beta distribution. During training, the limit price actions are sampled from the distribution and the probability ratio used in the PPO objective is derived from the same distribution's cumulative density function. During the evaluation, instead of sampling the distribution in a stochastic manner, we directly select the mean of the distribution in a deterministic fashion. Therefore, the agent always samples the highest probability action from its policy distribution instead of stochastically sampling it. To this end, during evaluation and back-testing, we use the mean of the Beta distribution $\text{Beta}(\alpha, \beta)$. The mean is defined as $\frac{\alpha}{\alpha+\beta}$, where α and β are the limit logits outputs of the continuous policy network as shown in Fig. 1.

The objective function to train the limit price policy is separate for each of the bid price and ask price since it is two separate distribution:

$$L^{(l)} = L_{\text{bid}} + L_{\text{ask}}, \tag{8}$$

where L_{bid} and L_{ask} are the objective of each limit head and are calculated according to Eq. (3). Under a mean reverting market assumption no weighing is required between them, which holds for the FOREX use case we present. However one can claim that this assumption does not hold true for stock markets and thus a different weighting might be required between the bid and ask parameters. This is because a market with year-over-year average gains will disproportionately sample buying actions and thus have imbalanced training without weighing.

3.3. Objective masking

Objective masking is the process of creating a mask over individual loss values of each sample and time step which will selectively nullify their contribution to the final training signal. It is used to stop noisy or unhelpful samples, or pieces of samples, from tainting a training step. During the sampling of the simulated trading environment, the agent produces limit prices that do not affect the reward received. One such case is the bid limit price, which is completely ignored by the simulated environment if any action other than the limit buy action is selected. In theory,

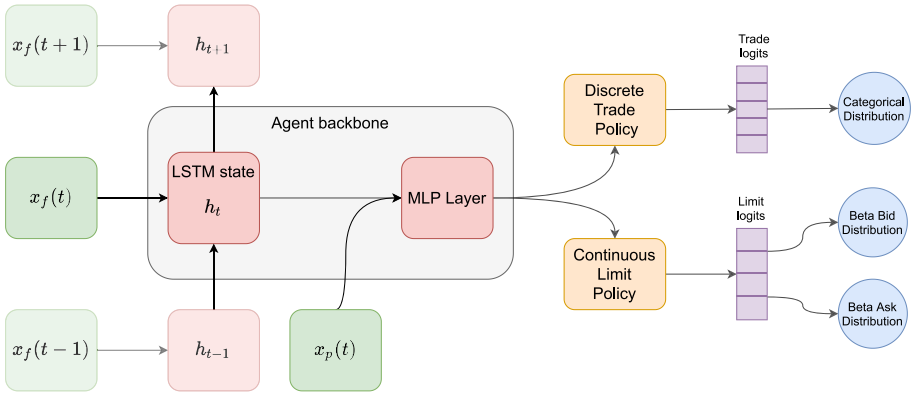

Fig. 1. Diagram of the limit-capable agent. The backbone network processes the input time series to extract useful representations. Then, its output feeds into the discrete trade policy and the continuous limit policy. The limit logits are the parameters used for the Beta distribution, which is used to sample the limit price.

RL allows actions that do not contribute to the reward and given infinite samples, the model should eventually converge. However, allowing the agent to train without addressing this issue will cause additional oscillations in the training process and lead to insufficient exploration of the possible limit actions, diminishing the achieved performance.

Therefore, to stop backpropagating gradients from limit price samples when the trade action is not a limit order, masking must be applied. A binary masking vector for each trajectory is defined as:

$$\mathbf{m} = (m_0, \ldots, m_t, \ldots, m_T), \tag{9}$$

where m_t is the mask value at time step t and T is the total length of the trajectories. The elements of the vector are activated based on whether the trade action $a^{(\tau)}$ is a limit order action and whether the limit order type (buy or sell) can affect the current position.

$$m_t = \begin{cases} 1, & \text{if } \text{sign}(a_t^{(\tau)}) \neq \text{sign}(b_t) \text{ and } a_t^{(\tau)} \in \{-1_l, 1_l\} \\ 0, & \text{if } \text{sign}(a_t^{(\tau)}) = \text{sign}(b_t) \text{ or } a_t^{(\tau)} \notin \{-1_l, 1_l\} \end{cases}, \tag{10}$$

where b_t is the current position (or balance) of the agent, sign() is the function that returns the sign of the position (long or short) and the sign of the trade action. For example, if a limit sell order -1_l is submitted while the current position is $b_t = -1$, then the mask element will be set to 0, whereas if the position is $b_t \geq 0$ then it will be set to 1. Also, the position sizes of our environment are of "constant lot" size. Also, the agent does not execute a trade in cases where the trade action has the same direction as the current position. One example of that is when the agent has previously bought an asset, i.e., has a *long* position, and selects either the market-buy or limit-buy trade action. In such a case, no trade will be submitted and the position will be maintained.

In order to ignore such episode steps for the limit policy, the training objective introduced in Section 3.1 is masked using the masking vector $\mathbf{m}$. Therefore, the final training objective of PPO is formulated as:

$$L^{\text{MASK}}(\theta) = \mathbb{E}_t\left[m_t \min(r_t(\theta)A_t, r_t^{\text{clip}}(\theta)A_t)\right], \tag{11}$$

by multiplying the binary masking vector with the objective vector of each trajectory. The masking is such that the objective has nonzero values for the trajectory steps that are actually affected by the limit price policy selection. The intended result of masking is the prevention of parameter update oscillations due to a noisy objective and consequently inaccurate gradients. The market policy can potentially affect the reward signal at any time step, so it is not masked using the described method. Masking is applied separately to the bid L_{bid} and ask L_{ask} objectives.

3.4. Reward shaping

Reward shaping is the process by which a reward signal is augmented to better incentivize an RL agent to the correct action sequence, thus speeding up the model convergence (Ng, Harada, & Russell, 1999). In this work we apply reward shaping to guide the limit price policy to emit price distribution more likely to execute while still considering the final reward. Limit orders, unlike market orders, may not get executed if their limit price is not within the upcoming price movements. It can be both beneficial for scenarios where we do not want to execute a trade if the price does not reach the intended level, but it can also be detrimental if a trading opportunity is missed when a limit order is left unfulfilled. This means that an unexecuted limit order can generate positive rewards. The PnL reward accurately depicts the contribution of the trade policy, but not the limit price policy. Since the limit price policy has a large continuous space of actions to search for every single simulation step, the reward signal should be the direct result of the continuous action to encourage the exploration of the best value ranges. In the presented trading scenario, a misplaced and unexecuted limit price can result in positive or negative rewards originating from a position started on previous time steps. In the long, this would not matter given sufficient exploration, but existing methods might not be able to overcome this reward noise. To remedy these cases we shape the reward to punish the limit price policy when a sampled limit price is placed outside the execution price range of the next time step.

$$r_t^{(l)} = \begin{cases} r_t + \epsilon_{(\tau)}, & \text{if } a_t^{(\tau)} \in \{1_l, -1_l\} \text{ and} \\ & p_l(t+1) < p^{((\cdot)_l)}(t) < p_h(t+1) \\ r_t - \epsilon_{(\tau)}, & \text{if } a_t^{(\tau)} \in \{1_l, -1_l\} \text{ and} \\ & p^{((\cdot)_l)}(t) < p_l(t+1) \text{ or} \\ & p^{((\cdot)_l)}(t) > p_h(t+1) \\ r_t, & \text{if } a_t^{(\tau)} \notin \{1_l, -1_l\}, \end{cases} \tag{12}$$

where $r_t^{(l)}$ is the reward to be used for the limit policy, r_t is the base reward signal received from the trading simulation environment, $(\cdot)_l \in \{1_l, -1_l\}$ is the limit trade action, and $\epsilon_{(\tau)}$ is the reward punishment that tries to bias the agent towards selecting limit order prices in the executable range. The price notations $p_l(t)$ and $p_h(t)$ represent the low and high prices at time step t, which will be further explained in Section 4.1. The price range from $p_l(t+1)$ to $p_h(t+1)$ is the executable price range for limit orders placed at time step t. If the limit price $p^{(1_l|-1_l)}(t)$ selected by the limit policy for the corresponding trade action $a_t^{(\tau)}$ is outside the aforementioned executable range, then the policy is punished, otherwise it is rewarded.

Each policy is assigned a different reward signal. Although both the limit and the trade policy have the same reward signal source, we only shape the reward applied to the limit price policy. The trade policy reward $r_t^{(\tau)}$ remains the same $r_t^{(\tau)} = r_t$. This approach is applicable in our case since the different rewards are aligned or at least adjacent. In different tasks where policies might have adversarial conflict in the rewarding schemes such an approach could result in training oscillations severely delaying convergence. From preliminary experiments, it was noted that when applying the described reward shaping to the trade policy $\pi^{(\tau)}$, the agent sampled fewer limit orders during sampling causing diminished exploration of the limit price action space. This degraded the performance compared to the simple market-only agent. Thus, the reward signals were separated to prevent the limit exploration fall-off.

Since each policy "head" of the agent receives a separate reward, a separate objective is required per head. The objectives, for the trade policy and the limit price policies, are combined by weighted simple addition:

$$L^{(\text{total})} = L^{(l)} + L^{(\tau)} \tag{13}$$

where $L^{(l)}$ and $L^{(\tau)}$ are the individual objective that are calculated using Eqs. (3) and (11) and with their Advantage values computed from each of the $r_t^{(\tau)}$ and $r_t^{(l)}$ rewards. Similarly, with Eq. (8), we have not applied any weighing between the objective terms. Some multi-objective optimization tasks are very prone to having a different scale in their gradients from the two objectives and end up optimizing only a single objective sufficiently. This has not been the case for our approach since in our results we observe significant improvement over a baseline single-objective task. However, we note that there might be gains to be had by tweaking the objective weights

4. Experiments

In this section, the dataset and the feature extraction methods are presented, followed by the model architecture and experimental setup. Then, the performance improvements of the limit-capable agents are investigated.

4.1. Dataset

The dataset used to drive the market simulation that interacts with the RL agents consists of Foreign Exchange (FOREX) trading data. The data includes multiple currency trading pairs, such as EUR/USD, EUR/GBP, and GBP/JPY. The trade data is subsampled using the Open-High-Low-Close (OHLC) price level technique, which reduces it into 4 values for the specified time intervals. This subsampling technique gives a clear picture of the price movements across time intervals. We have used a time interval of 1 h as our simulation stepping interval and as the OHLC subsampling interval.

At each simulation step, the environment provides an observation of the agents, which has two components. These are preprocessed feature matrices extracted from the OHLC prices and the current position held in the market. The preprocessed price features are the following:

1. $\frac{p_h(t)}{p_c(t)} - 1$
2. $\frac{p_l(t)}{p_c(t)} - 1$
3. $\frac{p_c(t)}{p_c(t-1)} - 1$
4. $\frac{p_h(t)}{p_h(t-1)} - 1$
5. $\frac{p_l(t)}{p_l(t-1)} - 1$

The high, low, and close prices are denoted as $p_h(t)$, $p_l(t)$, $p_c(t)$ respectively, where t denotes the time step. The high and low prices are the maximum and minimum prices of trades that occurred during the interval at time t while the close price is the price of the last trade that occurred during said interval. These features are constructed with the intention to avoid large distribution differences between asset pairs. This allows us to train the agents across multiple currency pairs and gain insight into the agent's ability to learn useful patterns from all of them.

The features achieve balanced distributions across different asset pairs because they are all derived from percentage distances between the sampled price values within the same interval or across intervals. Features 1 and 2 represent the price range within a single candle in terms of its fraction to the close price of that candle. Features 3, 4 and 5 represent the change of the close, high, and low price between each consecutive candle. Features 1–5 are concatenated into a vector and defined as $\mathbf{x}_f(t)$ for each time step t. This vector is calculated across the whole dataset once and the mean and standard deviation of the training set is used to normalize each feature of the vector using standardization.

The second component of the observation contains the position the agent holds since the last time-step. The position can be long, short or none, which means that he has previously bought, sold or exited the market respectively. This is represented by a one-hot vector of size 3, where [1, 0, 0] means that no position is currently taken, [0, 1, 0] means that a long position is currently active and [0, 0, 1] means a short position is currently active. The position one-hot vector is defined as $\mathbf{x}_p(t)$ for each time step t.

To change the position of the agent the environment provides five possible actions $a^{(\tau)}$ as described in Section 3.1. When a limit long 1_l or limit short action -1_l is selected an additional price value $p^{(1_l)}$ or $p^{(-1_l)}$ is accepted as input that sets the limit price for the order submitted. The price value selected is derived from the limit price policy action $a^{(-1_l)}$ defined in Section 3.1 The limit action is executed during the simulation step $t \rightarrow t+1$ if the selected price is within the price range $(p_l(t+1), p_h(t+1))$.

The reward the agent receives is the profit or loss that is incurred by the position currently held. When the agent changes the current position held, a commission is incurred to make the change, which is set to be similar to the FOREX brokerage IB, which is 0.002% for a 100k lot in USD. The simulation environment is an approximation of a true trading environment, with some simplifications. This leaves some effects unmodeled such as price changes caused by the trader and incurred costs from the bid–ask spread. However, this is without loss of our generality, since any additional effects would equally apply to all presented approaches.

The total number of available currency pairs is 27 and their available trading data spans from 2008 to the end of 2019. In our experiments we utilize the range 2008–2016 for training and validation, and 2016–2018 to test the resulting agents. In total, the dataset contains about 2.5 million data points, each representing one subsampled hour of trading data.

4.2. Experimental setup

The agent architecture consists of a Long-Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) processing layer (Greff, Srivastava, Koutník, Steunebrink, & Schmidhuber, 2016) for the time series features, followed by an MLP and the different policy "heads". The LSTM input is the feature time series $\mathbf{x}_f(t)$ described in Section 4.1 and is sequentially processed. The LSTM hidden state $\mathbf{h}_t$ at each time is then concatenated with the current position vector $\mathbf{x}_p(t)$ and then fed into the MLP layer to construct a representation vector $\mathbf{h}'_t$ of the current state. The representation $\mathbf{h}'_t$ is then transformed by the discrete trade policy head $\pi^{(\tau)}$ and the continuous limit policy head $\pi^{(l)}$, into a categorical distribution and a Beta distribution respectively. The proposed model architecture is depicted in Fig. 1.

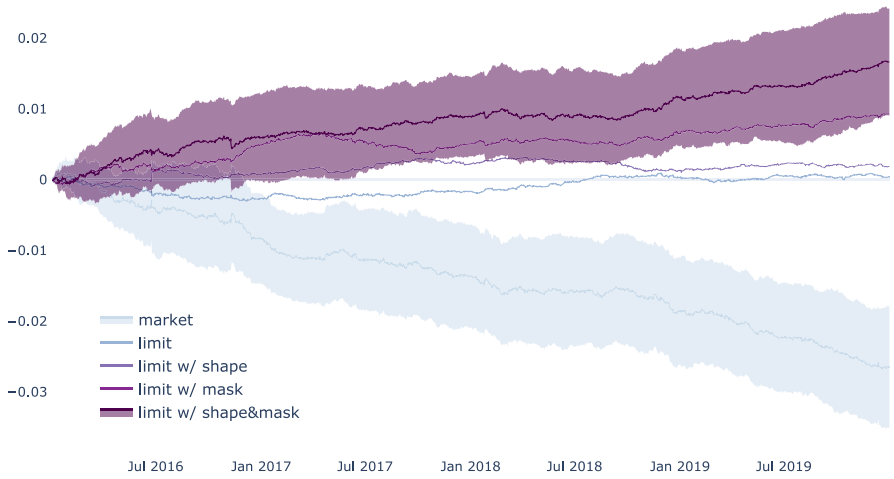

Fig. 2. Mean-normalized average PnL across all trading pairs. Lines are the means of the cumulative PnL in time and shaded areas are standard deviations, both of which values are calculated across the multiple experiments of each group.

Fig. 3. Sensitivity analysis of the limit punishment $\epsilon_{(\tau)}$ parameter. Each dot is a separate experiment and the green line is the rolling average. Performance of the reward shaping improves the performance evenly across the different values of its parameter, ensuring that there is not a narrow criterion for it to work.

All agent models have the same backbone for processing the input features and only differ in their policy layers, where the market-only agents have a discrete policy head, while the limit capable agents have both a discrete policy head for the trade action as well as a limit policy head for the price level of limit orders.

The environment sampling process used to train the agent is similar to previous works such as Schulman et al. (2017) and Tsantekidis et al. (2021). Training is split into 2 alternating phases. During the first phase, trajectories are sampled and appended into a short-term buffer. In the second phase, the sampled trajectories in the buffer are used to calculate the objective and train an agent. The first phase continues until a total of 2048 trajectories are simulated and saved to the buffer, at which point the second phase starts. Trajectories are sampled from the buffer in batches of 16 and the current agent distributions are calculated for them. The distributions are used according to the PPO algorithm (described in Section 3.1) to calculate the objective $L^{(\text{total})}$, which is then back-propagated and used to execute a gradient descent step. This process repeats for all trajectories in the buffer until every single one has been used.

To encourage the exploration of the action space for both the trade and the limit price actions we use the entropy bonus S for both policy heads. The entropy bonus adds to the objective the entropy of each policy distribution encouraging the policy to avoid saturating a single action or value. The final objective for the limit agent is:

$$J(\theta) = L^{(\text{total})}(\theta) + \eta_{\text{entropy}}(S[\pi_\theta^{(\tau)}] + S[\pi_\theta^{(l)}]), \tag{14}$$

where $L^{(\text{total})}$ is the PPO objective described in Section 3, η_{entropy} is the coefficient for the entropy bonus and $S[\cdot]$ is the entropy of each policy distribution. For the market-only agents the objective is:

$$J(\theta) = L^{(\tau)}(\theta) + \eta_{\text{entropy}} S[\pi_\theta^{(\tau)}], \tag{15}$$

where $L^{(\tau)}$ in this case includes just the market policy objective, since the market-only agent is only able to emit market order signals.

Table 1
Average Annualized PnL of evaluated agents by LSTM network size.

Agent type	32 neurons	64 neurons	128 neurons
Market	$14.7 \pm 0.9\%$	$14.5 \pm 1.0\%$	$13.1 \pm 0.7\%$
Limit	$16.6 \pm 0.8\%$	$15.6 \pm 1.0\%$	$15.1 \pm 1.0\%$
Limit (w/shape)	$16.3 \pm 0.9\%$	$15.9 \pm 1.1\%$	$15.6 \pm 0.8\%$
Limit (w/mask)	$\mathbf{17.2 \pm 1.1\%}$	$16.8 \pm 0.4\%$	$17.1 \pm 1.0\%$
Limit (w/mask&shape)	$\mathbf{17.2 \pm 1.0\%}$	$\mathbf{17.4 \pm 0.7\%}$	$\mathbf{17.4 \pm 0.9\%}$

Table 2
Comparing the proposed method to (a) a typical buy and hold baseline, where the agent holds a long position during the whole evaluation, as well as (b) a state-of-the-art DRL-based method that performs price trailing (Tsantekidis et al., 2020). The average annualized PnL is reported.

Method	PnL
Buy and hold	−0.01%
DRL with price trailing (Tsantekidis et al., 2020)	6.2%
Proposed	**17.4%**

To speed up the sampling we use a vectorized environment that simultaneously samples 128 separate simulations and allows the vectorized processing of the observations. Each trajectory has a total of 50 time steps, with each one corresponding to one trading hour. The gradient descent steps are not directly applied to the model parameters, but they are passed through a momentum optimizer, which in our case is RMSProp (Tieleman & Hinton, 2012) with a learning rate of 10^{-3}. The learning rate is decayed exponentially during training to 10^{-4}. During back-propagation, the Frobenius norm (2-norm) of the gradients is clipped to a maximum value of 1, to ensure smoother updates to the momenta of RMSProp. Note that the optimization concerns solely the parameters of the neural network that implements the policy and is used to control the Beta distribution. The total number of parameters of the networks used for the conducted experiments ranges from 2.5k (smallest architecture used) to 22k (largest architecture used).

4.3. Experimental evaluation

To investigate the proposed method, the following experiments were conducted. Each experiment was run 20 times with random initializations to ensure the noisy nature of DRL and financial simulation will not affect the significance of the results. These experiments build on top of the baseline achieved in previous works such as Tsantekidis et al. (2020) where the highest achieved test PnL was $11.44\% \pm 0.96\%$.

The main premise of our work is the comparison between a vanilla agent that executes only market orders (market-only) and an agent capable of submitting limit orders (limit-capable). In the case of the limit-capable agent, the action space includes both the trade-action and the limit price action. In our experiments, both types of agents achieve positive PnL on average in the tested configurations, but the limit-capable configurations outperform the market-only ones, as demonstrated.

First, we evaluate the market-only agents with multiple independent runs to establish a baseline for the limit-capable agents to compare with. Then we performed an ablation study, which enables us to independently evaluate different components in the proposed method to ensure that incremental improvements are provided by each one. More specifically, we evaluated the effect of using the proposed reward shaping ("w/shape"), masking ("w/mask"), as well as their combination ("w/shape&mask"). These results are reported in Table 1, along with the results for the baseline agent (market agent) for different network sizes. Using an agent capable of performing limit orders improves the performance of the market agent. Furthermore, using the proposed reward shaping, as well as the objective masking (mask) further improvements are obtained. When all components of the framework are combined, then the highest average performance is reached. Note that all agents manage to achieve similar standard deviations across experiments showing the same relative stability between them. The performance uplift between the market and limit agents is substantial because the limit agent can develop different trading strategies which are not available to the market agent due to inheriting restrictions on execution prices.

Furthermore, the back-testing PnL for the whole testing period is depicted in Fig. 2. Again, the proposed method leads to significant improvements over the baseline market agent, while both the proposed reward shaping and masking approaches always improve the results. The effectiveness of the proposed reward shaping approach is also further validated in Fig. 3, where a sensitivity analysis on the punishment parameter $\epsilon_{(\tau)}$ is provided. We also compared the proposed method to additional methods by using (a) a typical buy and hold (BnH) baseline, where the agent holds a long position during the whole evaluation, as well as (b) a state-of-the-art DRL-based method that performs price trailing (Tsantekidis et al., 2020). For the DRL agent, we used the same architecture as in the proposed method (128 neurons were used for the LSTM). The obtained results are provided in Table 2, further validating the benefits that can be obtained using the proposed method.

The proposed method and its components also improve the ratio of executed limit orders. In Tables 3 and 4 we present the statistics of trading signals emitted throughout training, their average profit as well as the execution success of these signals. Our approach clearly shows an increase of emitted limit order signals, since the agents trained as we suggest have explored the most rewarding regions of the limit price policy. Although the execution percentage of limit orders seemingly with all the proposed components, the total number of executed limit order is much greater. Specifically, in Table 3 the full model manages a 42% execution with a submitted total of 247k limit orders resulting in an average of 104k limit order executions compared to the 81k executions of the baseline limit model.

5. Conclusion

In this work, we proposed a framework for learning agents capable of placing continuous limit orders in financial trading and we demonstrated its advantages compared to the classical market order only approach, which is prevalent in the current literature. Choosing a limit order is not straightforward and has caveats, such as failing order execution. We presented methods that are capable of overcoming such issues and investigated their performance, including an ablation study between the proposed framework components. To highlight the potential of the proposed method, we also design an experiment with partial future information regarding measures related to volatility, and it was demonstrated that a limit-capable agent can more readily take advantage of this information to achieve higher profits compared to market agents.

Several interesting future research directions exist. First, while we used a large FOREX dataset, different agent behaviors might be observed in the market regimes of assets, such as stocks, bonds, or cryptocurrencies, or even different periods of time for the same assets. Therefore, extending and evaluating the proposed method in such assets can further highlight the importance of designing and using agents that can place limit orders. Therefore, extending and evaluating the proposed method in such assets can further highlight the importance of designing and using agents that can place limit orders. Another future direction for this

Table 3
Agent signal statistics.

Agent	Trade signals	Trade signal %	Limit signals	Limit signal %	Executed limits %
Market	172k ± 6k	0.25 ± 0.01	–	–	–
Limit	268k ± 16k	0.40 ± 0.02	167k ± 17k	0.25 ± 0.03	0.49 ± 0.02
Limit w/punishment	266k ± 15k	0.39 ± 0.02	165k ± 17k	0.24 ± 0.03	0.49 ± 0.03
Limit w/mask	334k ± 23k	0.50 ± 0.03	240k ± 23k	0.36 ± 0.03	0.43 ± 0.02
Limit w/punish&mask	340k ± 14k	0.50 ± 0.02	**247k ± 12k**	0.37 ± 0.02	0.42 ± 0.01

Table 4
Agent trade statistics.

Agent	Total trades	Profit ‰ per trade	Limit trades	Profit ‰ per limit trade
Market	172k ± 6k	0.08 ± 0.01	–	–
Limit	182k ± 7k	0.09 ± 0.01	81k ± 7k	0.34 ± 0.03
Limit w/punishment	182k ± 7k	0.09 ± 0.01	81k ± 8k	0.33 ± 0.03
Limit w/mask	196k ± 9k	0.09 ± 0.01	102k ± 8k	**0.35 ± 0.03**
Limit w/punish&mask	197k ± 9k	0.09 ± 0.01	**104k ± 6k**	**0.35 ± 0.04**

research would be the inclusion of an order size policy, selecting the quantity of the limit order while also supporting a variable lot size giving the agents the ability to accumulate their positions. Throughout the limit experiment, the scaling of the limit price action we used was set at a more conservative maximum of 0.6 quantile of the best limit price executions possible in the training set. In future research this could likely be explored to ensure to additional gains are left on the table. On a similar note, comparisons using rule-based and technical analysis-based scalping strategies could also provide an insight into their difference with Deep Learning approaches such as ours.

Declaration of competing interest

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

Data availability

The data that has been used is confidential

Acknowledgments

This research has been co-financed by the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH - CREATE - INNOVATE (project code: T2EDK-02094). Avraam Tsantekidis was funded by a scholarship from the State Scholarship Foundation (IKY) according to the "Strengthening Human Research Resources through the Pursuit of Doctoral Research" act, with resources from the "Human Resources Development, Education and Lifelong Learning 2014–2020" program. We kindly thank Speedlab AG for providing their expertise on the matter of FOREX trading and the comprehensive dataset of 28 FOREX currency pairs.

References

Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock price prediction using the ARIMA model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation* (pp. 106–112).

Avellaneda, M., & Stoikov, S. (2008). High-frequency trading in a limit order book. *Quantitative Finance*, *8*(3), 217–224.

Bernardo, J. M., & Smith, A. F. (2009). *Bayesian theory, Vol. 405*.

Chou, P.-W., Maturana, D., & Scherer, S. (2017). Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *Proc. int. conf. on machine learning* (pp. 834–843). PMLR.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Proceedings of the advances in neural information processing systems, Vol. 30*.

Dilokthanakul, N., Kaplanis, C., Pawlowski, N., & Shanahan, M. (2019). Feature control as intrinsic motivation for hierarchical reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, *30*(11), 3409–3418.

Dorado-Moreno, M., Navarin, N., Gutiérrez, P. A., Prieto, L., Sperduti, A., Salcedo-Sanz, S., et al. (2020). Multi-task learning for the prediction of wind power ramp events with deep neural networks. *Neural Networks*, *123*, 401–411.

Frankel, J. A., & Froot, K. A. (1990). Chartists, fundamentalists, and trading in the foreign exchange market. *The American Economic Review*, *80*(2), 181–185.

Froot, K. A., Scharfstein, D. S., & Stein, J. C. (1992). Herd on the street: Informational inefficiencies in a market with short-term speculation. *The Journal of Finance*, *47*(4), 1461–1484.

Ganesh, S., Vadori, N., Xu, M., Zheng, H., Reddy, P., & Veloso, M. (2019). Reinforcement learning for market making in a multi-agent dealer market. arXiv preprint arXiv:1911.05892.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232.

Haynes, R., & Roberts, J. S. (2015). Automated trading in futures markets. In *CFTC white paper*.

He, X., & Lin, S. (2021). Reinforcement learning and evolutionary equilibrium in limit order markets. Available at SSRN 3131347.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Jarusek, R., Volna, E., & Kotyrba, M. (2022). FOREX rate prediction improved by Elliott waves patterns based on neural networks. *Neural Networks*, *145*, 342–355.

Karpe, M., Fang, J., Ma, Z., & Wang, C. (2020). Multi-agent reinforcement learning in a realistic limit order book market simulation. arXiv preprint arXiv:2006.05574.

Katyal, K., Wang, I., Burlina, P., et al. (2017). Leveraging deep reinforcement learning for reaching robotic tasks. In *Proc. IEEE conf. on computer vision and pattern recognition workshops* (pp. 18–19).

Kercheval, A. N., & Zhang, Y. (2015). Modelling high-frequency limit order book dynamics with support vector machines. *Quantitative Finance*, *15*(8), 1315–1329.

Latif, S., Cuayáhuitl, H., Pervez, F., Shamshad, F., Ali, H. S., & Cambria, E. (2022). A survey on deep reinforcement learning for audio-based applications. *Artificial Intelligence Review*, 1–48.

Li, Y., Pan, Q., & Cambria, E. (2022). Deep-attack over the deep reinforcement learning. *Knowledge-Based Systems*, Article 108965.

Li, B., Tang, H., Zheng, Y., Hao, J., Li, P., Wang, Z., et al. (2021). Hyar: Addressing discrete-continuous action reinforcement learning via hybrid action representation. arXiv preprint arXiv:2109.05490.

Li, H., Zhang, Q., & Zhao, D. (2019). Deep reinforcement learning-based automatic exploration for navigation in unknown environment. *IEEE Transactions on Neural Networks and Learning Systems*, *31*(6), 2064–2076.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Lin, Z., Jia, J., Huang, F., & Gao, W. (2022). Feature correlation-steered capsule network for object detection. *Neural Networks*, *147*, 25–41.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., et al. (2016). Ssd: Single shot multibox detector. In *Proc. European conf. on computer vision* (pp. 21–37).

Malkiel, B. G., & Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, *25*(2), 383–417.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., et al. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

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

Moody, J., Wu, L., Liao, Y., & Saffell, M. (1998). Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting*, *17*(5–6), 441–470.

Neunert, M., Abdolmaleki, A., Wulfmeier, M., Lampe, T., Springenberg, T., Hafner, R., et al. (2020). Continuous-discrete reinforcement learning for hybrid control in robotics. In *Conference on robot learning* (pp. 735–751).

Nevmyvaka, Y., Feng, Y., & Kearns, M. (2006). Reinforcement learning for optimized trade execution. In *Proc. 23rd int. conf. on machine learning* (pp. 673–680).

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, *Vol. 99* (pp. 278–287).

Peng, H., Ma, Y., Poria, S., Li, Y., & Cambria, E. (2021). Phonetic-enriched text representation for Chinese sentiment analysis with reinforcement learning. *Information Fusion*, *70*, 88–99.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *Proc. int. conf. on machine learning* (pp. 1889–1897).

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv preprint arXiv:1712.01815.

Spooner, T., Fearnley, J., Savani, R., & Koukorinis, A. (2018). Market making via reinforcement learning. arXiv preprint arXiv:1804.04216.

Sui, Z., Pu, Z., Yi, J., & Wu, S. (2020). Formation control with collision avoidance through deep reinforcement learning using model-guided demonstration. *IEEE Transactions on Neural Networks and Learning Systems*.

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

Taylor, M. P., & Allen, H. (1992). The use of technical analysis in the foreign exchange market. *Journal of International Money and Finance*, *11*(3), 304–314.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*.

Tsantekidis, A., Passalis, N., & Tefas, A. (2021). Diversity-driven knowledge distillation for financial trading using Deep Reinforcement Learning. *Neural Networks*, *140*, 193–202.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., & Iosifidis, A. (2017). Using deep learning to detect price change indications in financial markets. In *Proc. European signal processing conf.* (pp. 2511–2515).

Tsantekidis, A., Passalis, N., Toufa, A.-S., Saitas-Zarkias, K., Chairistanidis, S., & Tefas, A. (2020). Price trailing for financial trading using deep reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, *PP*, http://dx.doi.org/10.1109/tnnls.2020.2997523.

Wang, J., Elfwing, S., & Uchibe, E. (2021). Modular deep reinforcement learning from reward and punishment for robot navigation. *Neural Networks*, *135*, 115–126.

Yang, M., Huang, W., Tu, W., Qu, Q., Shen, Y., & Lei, K. (2020). Multitask learning and reinforcement learning for personalized dialog generation: An empirical study. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 49–62.

Zhang, H., Jiang, Z., & Su, J. (2021). A deep deterministic policy gradient-based strategy for stocks portfolio management. In *Proc. IEEE int. conf. on big data analytics* (pp. 230–238).

Zhang, K., Li, Y., Wang, J., Cambria, E., & Li, X. (2021). Real-time video emotion recognition based on reinforcement learning and domain knowledge. *IEEE Transactions on Circuits and Systems for Video Technology*, *32*(3), 1034–1047.

Zhang, X., Li, Y., Wang, S., Fang, B., & Philip, S. Y. (2019). Enhancing stock market prediction with extended coupled hidden Markov model over multi-sourced data. *Knowledge and Information Systems*, *61*(2), 1071–1090.