Received November 2, 2021, accepted January 3, 2022, date of publication January 14, 2022, date of current version January 26, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3143653

Market Making Strategy Optimization via Deep Reinforcement Learning

TIANYUAN SUN, DECHUN HUANG, AND JIE YU
Business School, Hohai University, Jiangning District, Nanjing 211100, China

Corresponding author: Jie Yu (yujiehhu@126.com)

This work was supported by the Innovative Team of Philosophy and Social Sciences in Jiangsu Higher Learning Institutions under Grant 2017ZSTD002.

ABSTRACT Optimization of market making strategy is a vital issue for participants in security markets. Traditional strategies are mostly designed manually, and orders are mechanically issued according to rules based on predefined market conditions. On one hand, market conditions cannot be well represented by arbitrarily defined indicators, and on the other hand, rule-based strategies cannot fully capture relations between the market conditions and strategies' actions. Therefore, it is worthwhile to investigate how to incorporate deep reinforcement learning model to address those issues. In this paper, we propose an end-to-end deep reinforcement learning market making model, i.e., Deep Reinforcement Learning Market Making. It exploits long short-term memory network to extract temporal patterns of the market directly from limit order books, and it learns state-action relations via a reinforcement learning approach. In order to control inventory risk and information asymmetry, a deep Q-network is introduced to adaptively select different action subsets and train the market making agent according to the inventory states. Experiments are conducted on a six-month Level-2 data set, including 10 stock, from Shanghai Stock Exchange in China. Our model is compared with a conventional market making baseline and a state-of-the-art market making model. Experimental results show that our approach outperforms the benchmarks over 10 stocks by at least 10.63%.

INDEX TERMS Deep reinforcement learning, LSTM, market making, stock market.

I. INTRODUCTION

Market making (MM) strategy is a kind of buy-side high-frequency trading strategy in stock market. It is usually used to enliven the market, stabilize market orders, improve liquidity of the stock and promote the development of the market. The profit of a MM agent is obtained by capturing the spread of the market, which is basically the volatility of the market and differences between best bid and ask prices. The agent frequently quotes its own bid and ask prices simultaneously, and makes profit by waiting both legs hit by other orders, and in some situations loses money due to price trending. Therefore, how to design the MM strategy and make it profitable becomes an important question.

Traditional MM strategies are designed by human experts. Trading rules are mechanically made based on experiences, and the MM strategy makes its trading actions according to those rules. Issues with this approach are 1) the traditional MM strategy cannot well describe or represent the market and strategy states and 2) the manually-designed rules cannot well capture the relations between states and proper trading actions. In recent years, deep reinforcement learning has been widely exploited in many research areas as well as industries. For example, the Alphastar trained by deep reinforcement learning algorithm in [1] exceeds 99.8% human players. Deep Q-network (DQN), which combines reinforcement learning with deep neural networks [2], uses deep neural network to extract data features, and realizes end-to-end optimization of complex decision problems through reinforcement learning. However, most of existing deep reinforcement learning tasks that are relevant to stock market focused on long- or mid-term stock trading, e.g., Deng *et al.* [3] used direct deep reinforcement learning method to represent real-time financial signals, and trained an agent for financial asset trading, and few work has been done on the buy-side market making strategies. Challenges in this task have three folds:

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

1) State representation. There are two categories of states that trading models need to capture i.e., internal state including strategies' inventory etc. and market state including whether it is trending or stable. 2) Decision frequency. MM agents need to deal with high-frequency market data, including several levels of bid and ask prices and related waiting orders in each level, and they need to make real-time decisions on how to place their own orders in the limit order book (LOB) and cancel orders when market is against the strategy. 3) State-action mapping. MM strategies need to learn the relations between massive states and different combinations of trading actions, which makes the learning process more difficult.

In this paper, we propose an end-to-end reinforcement learning MM trading model based on recurrent deep neural network representation, which is termed deep reinforcement learning MM model (DRLMM). The model is designed to have three modules: 1) Feature capture. A deep recurrent Q-network (DRQN) architecture [4] is exploited and applied to MM agent learning, where the DRQN is based on DQN and modified by replacing the first post-convolutional fully connected layer with a recurrent long short-term memory (LSTM) layer. The DRQN with the LSTM units automatically learns temporal market states from the LOB without any hand-designed features. 2) Action selection. The action space is different from many previous works that it is designed to be adaptive to strategy's internal state instead of a fixed action set. The action set contains several subsets, and at each time, the MM agent selects the appropriate subset as action space according to its internal state, such as inventory. Action selection policy is then learned by DRLMM. 3) Cross engine. A near-to-real market cross engine is designed to simulate order execution in stock exchange. Market data are fed into both strategies and the engine, and the engine executes strategies's orders based on predefined rules. Comparative experiments are conducted on a Level-2 data set of 10 stocks in Shanghai Stock Exchange of China. The experimental results show that our model is effective and can make more profits.

The key contributions of this paper are as follows:

- We design an end-to-end reinforcement learning MM trading model based on recurrent deep neural network representation. The model uses LSTM units to capture temporal market information and exploits DRQN to optimize the MM strategy.
- We design an adaptive action selection policy, which selects a subset of actions from the whole action space based on strategy's internal state. This mechanism makes the model training more efficient.
- We backtest the DRLMM model in Chinese stock market using Level-2 limit order book data, and experimental results show that DRLMM is better than the baseline strategies in many metrics.

The rest of this paper is organized as follows. In Section II, we review the relevant works of reinforcement learning and deep reinforcement learning in the financial field. In Section III, we introduces the MM model framework based on deep reinforcement learning. In Section IV, the experiments are introduced, and, performances of DRLMM model and baseline are compared through market simulation. In Section V, we give our conclusions and future work directions.

II. RELATED WORK

There have been many research works that are related to MM and reinforcement learning. In recent years, with the development of deep learning and successful application of deep reinforcement learning, more and more attentions are given to this area by both researchers and practitioners. In this section, many previous works of market making strategy design, reinforcement learning and deep reinforcement learning are reviewed.

A. MARKET MAKING STRATEGY

MM strategy is concerned by many research areas, including both finance and machine learning. In finance and economics, MM is generally studied as an optimal control problem. Scholars use stochastic dynamic programming to study optimal bidding. For example, Avellaneda *et al.* [5] studied the pricing strategy in LOB. Guilbaud and Pham [6] considered the influence of execution priority. A large number of literatures have investigated the establishment of MM model. In the early stage, Ho and Stoll [7] proposed the classic model of single dealer. Since then, many researchers have extended the MM model. Das [8] expanded the scope of application of the model on the basis of glosten and Milgrom's market maker model [9]. However, these methods based on market microstructure modeling rely heavily on conditional assumptions and are not suitable for application in complex real markets.

B. REINFORCEMENT LEARNING

In recent years, reinforcement learning has been used to solve many kinds of financial problems. Chan and Shelton [10] applied reinforcement learning to the MM model for the first time to endow the MM agent with learning ability. Then, researchers applied various reinforcement learning methods to market makers. Spooner *et al.* [11] designed a market making agent based on time differential reinforcement learning, and applied return function to control inventory risk. Lim and Gorse [12] first proposed an optimized market making model based on reinforcement learning in high frequency trading, and used constant absolute risk aversion (CARA) as the final optimization function of the model. In the aspect of multi-agent, Patel [13] applied the multi-agent reinforcement learning framework to market making strategy, and made trades with the help of macro and micro agents. Ganesh *et al.* [14] established a multi-agent simulation system of a dealer market, and proposed a reinforcement learning reward method based on yield. Zhong *et al.* [20] collaborated with a market making firm and developed a market making strategy based on Q-learning, and they make the strategy become a lookup table that is easier to be implemented in real production. Spooner *et al.* [21] proposed an adversarial reinforcement learning based market making strategy and claimed that the agent can converge to Nash equilibrium in several special cases.

C. DEEP REINFORCEMENT LEARNING

Deep reinforcement learning can solve the problem of high-dimensional and dynamic strategy optimization by combining the high-dimensional input of deep learning with reinforcement learning. Kumar [23], in his extended abstract, proposed a deep reinforcement learning Q-network based market making strategy which is simulated and tested on his market simulator. Similarly, Gasperov *et al.* [22] proposed a deep learning market maker that utilizes trading signal generator to help predict market trends, and they test their strategy with one month bitcoin market tick data. Mnih *et al.* [2] and Alpha Go [15] have proved the practicability and effectiveness of DQN. Deep reinforcement learning's powerful representation ability and its end-to-end strategy optimization ability also attracted attentions of researchers in the field of quantitative trading. They tried to use DQN to solve the problem of investment decision-making under complex market conditions. Deng *et al.* [3] used direct deep reinforcement learning for financial asset trading. Ning *et al.* [16] established a fully connected neural network trained by experience replay and double DQN, and proved that its performance is better than the standard benchmark approach. Jia *et al.* [17] and others applied reinforcement learning based on LSTM to quantitative trading. Gueant and Manziuk [18] proposed a discrete-time model-based role criticism algorithm and compared it with the classical finite difference method.

III. DEEP REINFORCEMENT LEARNING MARKET MAKING

The reinforcement learning can be modeled as markov decision process (MDP) and represented by a tuple of four elements, i.e. (S, A, π, r), where in the tuple, S is a state space, A is an action space, π is policy, r is reward. In reinforcement learning algorithm, Q-learning is a representative value based algorithm. Its main goal is to build a two-dimensional Q-table to store Q-values of each pair of states and actions, and constantly optimize the values.

$$Q'(s,a) = Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)], \quad (1)$$

where α is learning rate, s' is the next state of s, a' is the action at state s', and γ is a discount factor.

Taking one step further, DQN uses a deep neural network instead of the Q-table in order to solve the problem of continuous state space or action space. The neural network is used to approximate the value function. The loss function of the network is defined as follows,

$$L_t(\theta_t) = \mathbb{E}_{(s,a,r,s')}[(y_t - Q_{\theta_t}(s,a))^2], \quad (2)$$

$$y_t = r + \gamma max_{a'} Q_{\theta_t}(s',a'), \quad (3)$$

where t is the current time step, and y_t is an estimate of the expected return.

DRQN [19] replaces the first post-convolutional fully connected layer with a recurrent LSTM on the basis of DQN. DRQN has been proved that it can deal with some observable information, and it is better handling loss of information than DQN. For the stock trading environment, many unknown variables in the time dimension can not be found well by DQN framework. So we use LSTM module in DRQN framework to construct market information representation. In this way, we can define the state closer to a comprehensive observation of the trading environment. The structure of the network is designed as follows: 1) a LSTM layer. It processes the market data in order to generate strategy's states; 2) a hidden layer. It is fully connected to the first layer and outputs four possible actions.

A. FORMULATION

The high-frequency market making strategy basically provides liquidity to the market by continuously quoting (both bid and ask sides) in the LOB. It processes incoming intra-day tick data, determines external (market) and its internal states, and takes instant actions simultaneously. It makes profits largely by capturing spreads in LOB, and loses money to inside traders in situations such as price trending. In general, market making strategies can issue new orders and wait for executions in order to capture more spreads, and cancel old orders that are placed in a risky position in order to avoid losses. In this paper, the state-determination and action-making are formulated in the DRQN framework, and we design a MM trading strategy based on DRQN, as shown in Figure 1.

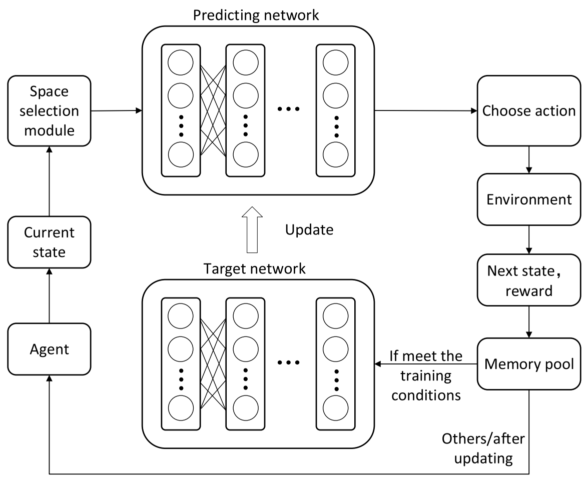


FIGURE 1. Deep reinforcement learning market making framework. The framework contains two networks, and state-determination and action-making are formulated in the DRQN framework.

Firstly, in each time step, the MM agent will send the state information as input to the neural network. Secondly, through several hidden full connection layers, the Q-network is used to estimate the possible value of each state-action. Thirdly, through the spatial selection module, the corresponding action is selected, and the real reward is returned through the cross engine simulator. Finally, the gradient descent method is used to update the θ parameters.

Our definition of the state, action and reward are as follows:

1) STATE *S*

The state set in the model consists of two parts: internal state and external state, $S = \{S_{in}, S_{out}\}$. The internal state $S_{in} = \{m, I, O\}$, including money m, inventory I, and remaining order information O; the external state S_{out} mainly contains the market information, and in contrast to many previous literatures that generally extracted data from LOB, construct features manually, and build a multi-dimensional state space, we directly choose entire LOB and let LSTM to build states for the agents. The entire LOB at each time stamp is a set of multi-dimensional data, including 10 bid/ask price levels (at the sell side $ask_{10} \ldots ask_1$ and the buy side $bid_{10} \ldots bid_{10}$, prices are highest at ask_{10} and lowest at bid_{10}, and ask_1 is the lowest sell price a.k.a best ask, and bid_1 is the highest buy price a.k.a best bid) and waiting order queues corresponding to each level. For each time stamp t, we select previous five snapshots of LOB data, and input it to LSTM iteratively to generate its external state at t. Therefore, the MM agent can retain the cognition of the previous market transaction data when the external state is constantly updated.

2) ACTION *A*

In this paper, we divide the fixed finite set of action space into many subsets, i.e. $A = \{A_1, A_2, A_3 \ldots, A_n\}$. At each time step t, the space selection module selects the appropriate action space A_t after analyzing the existing state s_t and transmits it to the network. DRQN traverses every action A in A_t and updates the corresponding Q-value continuously. Other actions not in A_t will be excluded from consideration, and the model will not select such actions for the state s_t. Specifically, A_t consists of three parts, i.e. $A = (D, P, N)$, where D is order side, P is order price, and N is the number of shares. The order side can be either 'buy' or 'sell'; the price ranges all the prices that can be placed in the market, e.g. any level from ask_{10} to bid_{10}; the number of shares can be adjusted according to the demand of placing orders, with a minimum of one lot size. Rules for action subset selection are as follows,

- If there is no open order, the action space includes issuing two new orders which are on the best bid and ask respectively.
- If there is only one open order, the action space includes: 1) wait for the order getting executed; 2) cancel the order and issue a new order with a new price.
- If there are two legs of open orders waiting in the LOB, the action space includes: 1) wait for the orders getting executed; 2) cancel either of them and issue a new order with a new price; 3) cancel both orders and issue two new orders.

3) REWARD *R*

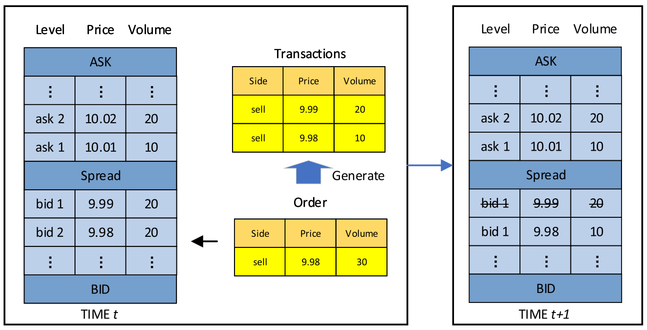
The reward function in reinforcement learning is generally defined as cumulative reward, i.e., $U_T = \sum_{t=1}^{T} R_t$, where R_t is the reward in each step, which is also the value function in the classic deep reinforcement learning framework. R_t is usually defined by the profit earned in many normal trading models. However, since the goal of MM agent is different from normal trading models, R_t in DRLMM needs to be defined from two major perspectives: 1) market liquidity. DRLMM needs to quote on both sides of the LOB simultaneously in order to provide liquidity to the market; 2) inventory risk. DRLMM needs to consider the risk of inside traders that have extra information and make market price trending. Therefore, the reward in DRLMM is defined as,

$$R_t = \begin{cases} 1, & \text{if 'buy' and 'sell' limit orders} \\ & \text{are executed,} \\ -0.5, & \text{if cross-spread order (market order)} \\ & \text{is executed,} \\ 0, & \text{others.} \end{cases} \quad (4)$$

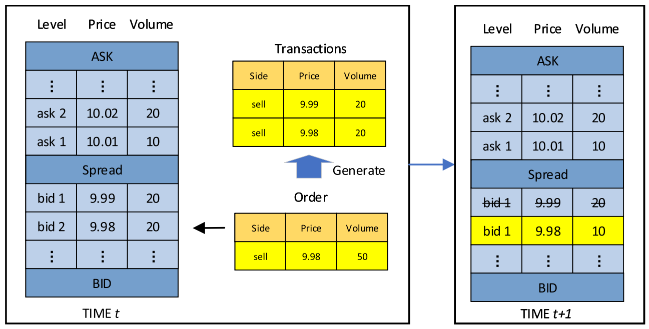
If two legs ('buy' and 'sell' limit orders) are executed at the same time or within a very short time period, the MM agent captures the LOB spread (makes profit) and also provides liquidity to the market. In this case, R_t is 1 (positive, two half-spreads). If one leg is executed first, the MM agent waits for a period of time and finds its inventory is at risk, the MM agent cancels the other leg and sends a new order that crosses the spread to get an immediate execution in order to get inventory at balance. In this case, the MM agent loses money, thus R_t is -0.5 (negative, one half-spread). In the other cases, the MM agent can be considered as waiting, so the R_t is 0.

B. CROSS ENGINE SIMULATOR

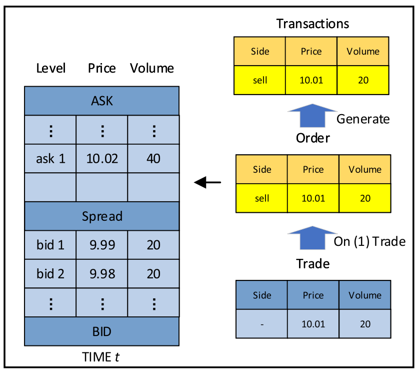
A near-to-real cross engine simulator is designed and implemented in order to assist the order transaction for our MM agents. The simulator is designed to accept historical Level-2 data and orders from the MM agents. Since there are only quotes and trades data in the historical data, and no order queue data is provided, we cannot accurately know the waiting time of the limit orders generated by the MM agents. Therefore, in this simulator, we use waiting time (a constant number suggested by practitioners) to simulate the waiting behaviors of limit orders, i.e., when other conditions do not change, the limit orders can be traded at the price level only when there has a constant number of trades happened. As shown in Figure 2, in detail, the matching mechanism in this simulator is summarized as follows:



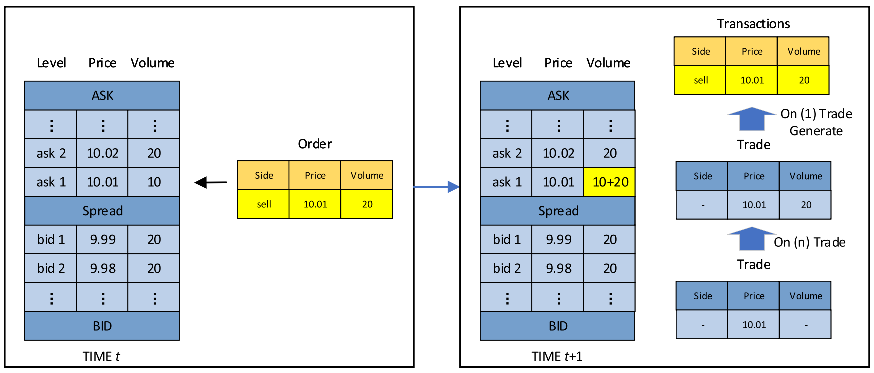
(a) A sell order with small volume crosses spread and reaches bid_2, which makes it a market order, and the order gets fully-filled and generates transactions immediately.



(b) A sell order with big volume crosses spread and reaches bid_2, which makes it a market order, and the order cannot get fully-filled where it generates transactions and leaves part of the order in the order book.



(c) A sell order's price is between bid_1 and ask_1, which makes it a limit order and wait in new ask_1 queue, if there is an incoming trade, the order will get fully-filled and generate transactions immediately.



(d) A sell order reaches ask_1, which makes it a limit order and wait in ask_1 queue, the order cannot get fully-filled immediately and it will wait for n incoming trades until it gets filled.

FIGURE 2. Cross engine.

- When buy order's price crosses the order book spread and touches the ask side, or sell order's price crosses the order book and touches the bid side, 1) if the order's quantity is less than the shares in the waiting queue, it will be executed immediately; 2) if the order's quantity is more than the shares in the waiting queue, part of it will be executed immediately and the rest part will wait in the queue.
- When buy order's price is on the best bid or sell order's price is on the best ask, the order will wait for a constant number of n trades and get fully executed if its quantity is less than the shares of the next trade, otherwise, the rest part of the order will remain in the queue.

IV. EXPERIMENTS AND DISCUSSIONS

In this section, the data set and setup used in the experiments are firstly introduced, and secondly, experimental results are presented and analyzed with discussions.

A. DATA SET

We evaluate the DRLMM by real financial market data. The data set is Level-2 tick-by-tick data of A-shares from Shanghai Stock Exchange of China, including quotations and transactions. We selected 10 actively traded stocks, i.e., Shanghai Pudong Development Bank (600000.SS), Inner Mongolia Baotou Steel Union (600010.SS), Huaxia Bank (600015.SS), China Minsheng Bank (600016.SS), China Petroleum and Chemical Corporation(600028.SS), Citic Securities Company Limited (600030.SS), China Merchants Bank (600036.SS), Poly Developments and Holdings Group (600048.SS), China United Network Communications (600050.SS), and Tebian Electric Apparatus (600089.SS). Data from June 2014 to December 2014 (about 100 trading days) are used in the experiment, therefore, there are totally 4,739,610 quotes and 42,590,258 trades.

B. BASELINE

Two baseline methods are introduced in the experiments.

- **Rule-based market making strategy (RMM).** The first baseline is a traditional rule-based market making strategy. The strategy places buy and sell orders only on bid_1 and ask_1 respectively, and as the same with DRLMM, the order's quantity is set to one lot (100 shares) for all the stocks. Figure 3 demonstrates the rules used in RMM, and the detailed trading logic of RMM is summarized as follows,

- Place two orders (two legs) at the same time, one buy order on bid_1 and one sell order on ask_1;
- If both orders are executed (or closed), two new orders will be issued at the next time point. Accordingly, the prices will be updated to new bid_1 and ask_1;
- If only one of the two legs is executed, e.g., only the buy order is executed, the other leg will wait for three trades until the order gets executed, otherwise, the sell order will be canceled and a new sell order with bid_1 price is issued (cross the spread) and executed immediately.
- If the strategy is going to end market making (e.g., market is going to close), the strategy cancels any open order and closes open inventory by using market orders.

- **Reinforcement learning based market making strategy (RLMM)**. A modified version of the reinforcement learning based market making introduced by Lim *et al.* [12] is employed as the second baseline in our experiments. The inputs of the RLMM are snapshots of LOBs, which are vectors of prices and corresponding shares on each price levels. The LOB vectors are taken as states, and the RLMM uses Q-learning as its core algorithm to learn mappings between states and actions, where the action set is defined in the same way as the DRLMM.

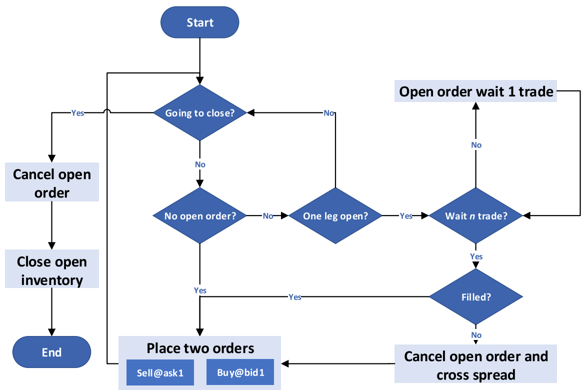


FIGURE 3. Trading logic of RMM: 1) if no open order, place two orders (two legs) at the same time, one buy order on bid_1 and one sell order on ask_1; 2) if both orders are executed (or closed), two new orders will be issued at the next time point, and their prices will be updated to new bid_1 and ask_1; 3) if only one of the two legs is executed, the other open leg will wait for three trades until the order gets executed, otherwise, the order will be canceled and a new order is issued (cross the spread) and executed immediately; 4) if the strategy is going to end market making, the strategy cancels any open order and closes open inventory by using market orders.

C. SETUP

We choose ϵ-greedy algorithm for model's action selection during training, and gradually reduce the probability of exploration along with the learning process of the agent. The action set of the MM agent contains only best bid and best ask, and the size of limit orders are set to be one lot (100 shares). At the beginning of each tranche, the inventory I ($I > 0$, enough to buy and short stocks) is set to be a constant. In the last minute of each tranche, if the inventory is not balanced, i.e., the agent did not close any live orders, the MM agent needs to take a series of done-for-day action: 1) stop market making, 2) cancel all the live orders, and 3) sell all holding positions, in order to keep the inventory balanced. In addition, without loss of generality, transaction cost is 0, since most market makers have market making licenses and do not need to pay any transaction fee (in some market they can even get rebate from market making).

The detailed experiment setup is summarized as follows,

- Data processing: each trading day is equally divided into 8 parts, and each part is 30 minutes (a tranche). Models are then trained and tested on the set of tranches. All the data are divided into a training set (first 80 days) and a test set (last 20 days).
- Model training: the network has one LSTM layer followed by one Linear layer. The structure of the network is set as Input -> LSTM(43, 20) -> (batch, 20) -> Linear(20, 6) -> (batch, 6) -> softmax -> (batch, 1) -> Output. one tranche in the training set is taken as a training episode. During the training, an episode is randomly selected to train the model, and it loops 10,000 times. The network is implemented in Python 3.7, Pytorch 1.2 and Pandas 1.1, trained on two Titan Xp.[1] ϵ in ϵ-greedy is set as 0.7, δ_ϵ is set as 0.95, discount factor γ is set as 0.9, batch size is set as 128, loss function is set as "MSELoss", optimization strategy is set as "Adam" and replay memory size is set as 10,000. There are two other parameters to be tuned, which is learning rate (lr) and number of output nodes in LSTM (n). lr is selected from $\{1e{-}5, 5e{-}5, 1e{-}4, 5e{-}4, 1e{-}3, 5e{-}3\}$, and n is selected from $\{10, 15, 20, 25, 30, 35, 40, 45, 50\}$. There are totally 54 combinations of those two parameters, and after tuning lr is set as $1e-3$ and n is set as 20.
- Model testing: after the training, the model is applied to the tranche in the test data set, and the DRLMM is then compared with baselines in terms of total PnL (Profit and Loss) and winning rate.

D. RESULTS AND DISCUSSIONS

In this section, we evaluate the DRLMM performance by comparing with the baselines. We use average profit (AP) over trading days and its standard deviation (std.) to evaluate the profitability and stability of the trading models. In Table 1, results of 10 stocks are presented, and numbers marked by bold font indicate that the model performs better than the other two models. It can be observed that the DRLMM wins in all the 10 stocks with smaller standard deviation.

[1]Resources are available at https://pan.baidu.com/s/1Fgb-Cg7woZ2lKZ-zi5nQKw

TABLE 1. Average profit and standard deviation of MM models.

Stock	600000.SS			600010.SS		
	RMM	RLMM	DRLMM	RMM	RLMM	DRLMM
AP	185.45	188.45	**218.32**	176.93	202.23	**217.97**
std.	442.57	161.74	124.89	134.24	110.65	98.73

Stock	600015.SS			600016.SS		
	RMM	RLMM	DRLMM	RMM	RLMM	DRLMM
AP	100.56	115.95	**128.27**	216.96	246.88	**274.54**
std.	465.56	177.43	99.02	287.00	158.79	146.77

Stock	600028.SS			600030.SS		
	RMM	RLMM	DRLMM	RMM	RLMM	DRLMM
AP	142.22	160.66	**180.67**	258.94	404.38	**471.31**
std.	171.98	142.64	127.49	1928.37	420.68	395.18

Stock	600036.SS			600048.SS		
	RMM	RLMM	DRLMM	RMM	RLMM	DRLMM
AP	147.14	187.24	**209.07**	77.61	133.06	**162.32**
std.	525.11	200.54	147.28	263.16	155.89	110.27

Stock	600050.SS			600089.SS		
	RMM	RLMM	DRLMM	RMM	RLMM	DRLMM
AP	88.37	122.16	**137.99**	92.33	94.82	**108.32**
std.	152.91	102.38	97.45	253.24	149.75	82.63

Since there are 8 tranche for each trading day, thus, there are totally 160 tranche in the test data set. In each trading period, e.g. 9:30-10:00, there are 20 tranche from 20 trading days in the test data set. In addition to the AP comparison between those models, we compare strategies's AP over different trading tranche, i.e., for each trading period, we calculate the AP of that period over 20 trading days, and compare their performances. In Table 2, numbers in bold font indicate the model performs better than the other two models. It can be observed that, the DRLMM wins 67 out of 80 trading tranche over 10 stocks, which indicates that the DRLMM performs better than the baselines in different trading periods. It can also be observed from the results that during the opening and closing hour of the market, strategies tend to make more profit than the time periods near the noon. It is because that the market is usually actively traded during the opening and closing hours, which gives MM strategies more chances to get both legs of orders executed and make profit.

To take one step further, we use *winning number* to compare the strategies in different trading periods. If a strategy gets more profits than the other strategy in one trading period, it gets 1 score. Therefore, the highest score that one strategy can get in one trading period is 20 in the test data set. Results of *winning number* in the test data set are shown in Table 3 and Table 4. Numbers in bold font indicate that the strategy performs better than the other strategy. It could be observed from the results that the DRLMM performed overwhelmingly better than the baselines in terms of *winning number*.

TABLE 2. Average profit over different tranches.

Stock	600000.SS			600010.SS		
	RMM	RLMM	DRLMM	RMM	RLMM	DRLMM
9:30-10:00	262.50	320.75	**345.90**	276.60	297.79	**306.05**
10:00-10:30	180.95	224.37	**250.50**	189.70	210.56	**232.45**
10:30-11:00	141.80	160.89	**207.76**	169.95	188.29	**204.75**
11:00-11:30	97.90	117.21	**155.65**	162.85	160.91	**168.10**
13:00-13:30	**233.50**	156.65	173.74	136.75	169.96	**172.35**
13:30-14:00	108.60	125.78	**167.50**	152.15	168.77	**194.82**
14:00-14:30	**280.75**	180.65	191.39	128.35	190.11	**205.90**
14:30-15:00	177.60	221.32	**254.07**	199.05	231.44	**259.35**

Stock	600015.SS			600016.SS		
	RMM	RLMM	DRLMM	RMM	RLMM	DRLMM
9:30-10:00	28.95	200.90	**244.28**	369.35	407.88	**415.50**
10:00-10:30	**183.95**	150.41	155.20	263.45	284.05	**304.20**
10:30-11:00	44.50	119.70	**122.02**	210.50	238.64	**247.35**
11:00-11:30	**139.65**	88.93	87.25	82.15	127.53	**208.91**
13:00-13:30	35.15	70.39	**75.39**	173.30	177.63	**215.39**
13:30-14:00	**252.90**	90.59	88.48	187.95	193.28	**235.00**
14:00-14:30	49.30	100.50	**113.80**	213.55	255.49	**258.20**
14:30-15:00	70.10	106.17	**139.76**	235.40	290.50	**311.80**

Stock	600028.SS			600030.SS		
	RMM	RLMM	DRLMM	RMM	RLMM	DRLMM
9:30-10:00	217.80	233.86	**248.81**	599.05	879.02	**931.86**
10:00-10:30	94.75	159.98	**188.95**	**557.90**	527.82	532.05
10:30-11:00	**194.80**	169.04	172.85	**590.75**	409.90	415.06
11:00-11:30	97.50	110.43	**134.50**	-514.00	27.31	**324.78**
13:00-13:30	113.10	114.07	**152.40**	284.95	323.98	**379.00**
13:30-14:00	125.65	139.48	**165.70**	225.45	311.68	**341.84**
14:00-14:30	135.20	166.89	**172.40**	-43.95	279.14	**349.08**
14:30-15:00	158.95	191.54	**209.75**	371.35	476.19	**496.82**

Stock	600036.SS			600048.SS		
	RMM	RLMM	DRLMM	RMM	RLMM	DRLMM
9:30-10:00	224.35	333.04	**345.12**	185.15	259.67	**273.60**
10:00-10:30	2.70	203.64	**231.63**	131.70	171.69	**180.23**
10:30-11:00	120.40	180.46	**194.81**	94.05	144.31	**158.96**
11:00-11:30	83.55	91.84	**159.86**	62.25	75.07	**120.25**
13:00-13:30	97.85	122.45	**160.14**	-19.25	25.78	**105.65**
13:30-14:00	**233.70**	149.28	152.25	78.50	96.43	**133.70**
14:00-14:30	**272.35**	199.14	200.11	-0.70	109.84	**137.35**
14:30-15:00	142.25	218.08	**228.66**	89.15	181.68	**188.80**

Stock	600050.SS			600089.SS		
	RMM	RLMM	DRLMM	RMM	RLMM	DRLMM
9:30-10:00	125.80	201.78	**225.05**	140.15	200.19	**206.56**
10:00-10:30	109.00	150.49	**154.30**	**172.80**	104.87	110.75
10:30-11:00	84.00	123.45	**125.47**	**160.00**	90.42	90.04
11:00-11:30	5.90	24.97	**86.50**	33.50	34.04	**69.10**
13:00-13:30	**107.80**	90.34	94.90	39.50	53.45	**72.55**
13:30-14:00	67.80	89.54	**109.70**	50.80	74.89	**87.09**
14:00-14:30	96.75	130.24	**139.90**	56.35	79.78	**92.70**
14:30-15:00	109.90	166.43	**168.09**	85.55	120.90	**137.80**

In summary, the performance of DRLMM in our experiments is better than the baselines. It could reliably generate more profits in the volatile market. From the experimental results, it could be observed that the LSTM can bring better market state representations than manually feature engineering. In addition, the deep reinforcement learning can

TABLE 3. Winning number, RMM v.s. DRLMM.

Stock	600000.SS		600010.SS		600015.SS		600016.SS		600028.SS	
	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM
9:30-10:00	0	20	2	18	1	19	0	20	1	19
10:00-10:30	0	20	0	20	1	19	2	18	0	20
10:30-11:00	0	20	0	20	0	20	0	20	2	18
11:00-11:30	0	20	1	19	2	18	1	19	0	20
13:00-13:30	3	17	0	20	0	20	0	20	2	18
13:30-14:00	0	20	0	20	2	18	0	20	0	20
14:00-14:30	2	18	1	19	0	20	0	20	1	19
14:30-15:00	0	20	0	20	0	20	0	20	0	20
Total	5	**155**	4	**156**	6	**154**	3	**157**	6	**154**

Stock	600030.SS		600036.SS		600048.SS		600050.SS		600089.SS	
	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM
9:30-10:00	2	18	1	19	0	20	1	19	0	20
10:00-10:30	1	19	0	20	1	19	0	20	1	19
10:30-11:00	3	17	0	20	0	20	0	20	1	19
11:00-11:30	2	18	1	19	0	20	0	20	0	20
13:00-13:30	2	18	0	20	1	19	1	19	1	19
13:30-14:00	6	14	1	19	1	19	0	20	0	20
14:00-14:30	4	16	1	19	0	20	0	20	1	19
14:30-15:00	4	16	0	20	0	20	0	20	0	20
Total	24	**136**	4	**156**	3	**157**	2	**158**	4	**156**

TABLE 4. Winning number, RLMM v.s. DRLMM.

Stock	600000.SS		600010.SS		600015.SS		600016.SS		600028.SS	
	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM
9:30-10:00	7	13	7	13	9	11	5	15	9	11
10:00-10:30	9	11	7	13	8	12	7	13	9	11
10:30-11:00	6	14	7	13	6	14	6	14	9	11
11:00-11:30	1	19	1	19	2	18	3	17	3	17
13:00-13:30	1	19	2	18	1	19	2	18	3	17
13:30-14:00	5	15	6	14	5	15	7	13	7	13
14:00-14:30	9	11	8	12	7	13	9	11	8	12
14:30-15:00	9	11	8	12	7	13	8	12	8	12
Total	47	**113**	46	**114**	45	**115**	46	**114**	56	**104**

Stock	600030.SS		600036.SS		600048.SS		600050.SS		600089.SS	
	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM	RMM	DRLMM
9:30-10:00	9	11	8	12	6	14	6	14	7	13
10:00-10:30	9	11	7	13	6	14	6	14	7	13
10:30-11:00	9	11	7	13	7	13	5	15	8	12
11:00-11:30	5	15	3	17	1	19	1	19	1	19
13:00-13:30	5	15	2	18	1	19	1	19	2	18
13:30-14:00	9	11	7	13	4	16	4	16	7	13
14:00-14:30	8	12	7	13	7	13	6	14	8	12
14:30-15:00	9	11	7	13	7	13	6	14	7	13
Total	63	**97**	48	**112**	39	**121**	35	**125**	47	**113**

learn a better mapping between strategy states and actions, and make smarter actions to obtain more profits and lower risks.

V. CONCLUSION

The market making strategy optimization is an attractive topic for both researchers and practitioners. With the development and successful application of deep reinforcement learning models, how to use deep reinforcement learning model to market making strategy becomes an interesting research problem. In this paper, we propose an end-to-end reinforcement learning market making strategy based on recurrent deep network representation, DRLMM. It exploits LSTM network to extract temporal patterns of the market directly from the LOBs, and it learns state-action relations via a reinforcement learning approach. In order to control inventory risk and information asymmetry, a deep Q-network is introduced to adaptively select different action subsets and train the market making agent according to the inventory states.

Experiments are conducted on a six-month Level-2 data set, including 10 stock, from Shanghai Stock Exchange in China. Our model is compared with two baseline market making strategies. Experimental results show that: 1) the DRLMM performs better than the benchmark MM strategies; 2) using DRQN to directly extract market information and construct market features can make the state representation in DRLMM better than manually made features; 3) the adaptive action space can improve the training process of DRLMM as well as the profitability of the MM strategy.

In future work, DRLMM can be extended to a multi-agent setting, where many agents with different parameters are trained to learn market making and a meta-learning mechanism could be further introduced to select agent in order to build a more profitable MM strategy.

REFERENCES

[1] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, Mar. 2017.

[4] L. Chen and Q. Gao, "Application of deep reinforcement learning on automated stock trading," in *Proc. IEEE 10th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Oct. 2019, pp. 29–33.

[5] M. Avellaneda and S. Stoikov, "High-frequency trading in a limit order book," *Quant. Finance*, vol. 8, no. 3, pp. 217–224, Apr. 2008.

[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] 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.

[8] S. Das, "A learning market-maker in the Glosten–Milgrom model," *Quant. Finance*, vol. 5, no. 2, pp. 169–180, Apr. 2005.

[9] L. R. Glosten and P. R. Milgrom, "Bid, ask and transaction prices in a specialist market with heterogeneously informed traders," *J. Financial Econ.*, vol. 14, no. 1, pp. 71–100, Mar. 1985.

[10] N. T. Chan and C. R. Shelton, "An electronic market-maker," in *Proc. Conf. Soc. Comput. Econ.*, Jan. 2001, pp. 1–43.

[11] T. Spooner, J. Fearnley, R. Savani, and A. Koukorinis, "Market making via reinforcement learning," 2018, *arXiv:1804.04216*.

[12] Y.-S. Lim and D. Gorse, "Reinforcement learning for high-frequency market making," in *Proc. ESANN*, Apr. 2018, pp. 521–526.

[13] Y. Patel, "Optimizing market making using multi-agent reinforcement learning," 2018, *arXiv:1812.10252*.

[14] 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*.

[15] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[16] B. Ning, F. H. T. Lin, and S. Jaimungal, "Double deep Q-learning for optimal execution," 2018, *arXiv:1812.06600*.

[17] J. Wu, C. Wang, L. Xiong, and H. Sun, "Quantitative trading on stock market based on deep reinforcement learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[18] 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.

[19] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," 2015, *arXiv:1507.06527*.

[20] Y. Zhong, Y. Stone, and A. Ward, "Data-driven market-making via model-free learning," in *Proc. IJCAI*, 2020, pp. 4461–4468.

[21] T. Spooner and R. Savani, "Robust market making via adversarial reinforcement learning," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 1–7.

[22] B. Gasperov and Z. Kostanjcar, "Market making with signals through deep reinforcement learning," *IEEE Access*, vol. 9, pp. 61611–61622, 2021.

[23] P. Kumar, "Deep reinforcement learning for market making," in *Proc. AAMAS*, 2020, pp. 1892–1894.

TIANYUAN SUN received the master's degree in business administration from Nanyang Technological University, Singapore, and the E.M.B.A. degree from Shanghai Jiao Tong University, China. He is currently pursuing the Ph.D. degree with Hohai University, Nanjing, China. After graduating with a master's degree, he worked in several listed companies, engaging in asset management and securities investment related works. His research interests include stock market and deep reinforcement learning.

DECHUN HUANG received the bachelor's degree in grain engineering from Jiangnan University, Wuxi, China, in 1989, and the M.S. and Ph.D. degrees in management from Hohai University, Nanjing, China, in 1999 and 2003, respectively. He is currently a Professor with the Department of Finance, Business School, Hohai University. He presided over and participated in many national research projects funded by NSSFC, NSFC, and other institutions. He is the author of more than ten books, publishes more than 100 papers in journals and conferences, and wins some awards. His research interests include enterprise strategic investment, risk management, industrial economy, and investment economy. He was a fellow of the Institute of Applied Technology, Fraunhofer, Germany, in 2015, and the Deputy Director of the International Federation of East Asian Management Associations (IFEAMA), in 2014.

JIE YU was born in Nantong, China, in 1992. He received the B.S.E. degree in engineering management from Yangzhou University, Yangzhou, China, in 2011. He is currently pursuing the Ph.D. degree with Hohai University, Nanjing, China. He has participated in several research projects funded by the NSFC and NSSFC. He is the coauthor of one book on investment risk and coauthored papers in journals and conferences. His research interests include corporate finance, strategic management, and water-energy-food nexus.