A Novel Deep Reinforcement Learning Based Automated Stock Trading System Using Cascaded LSTM Networks

Jie Zou[a], Jiashu Lou[a], Baohua Wang[a,*]

[a]*College of Mathematics and Statistics, Shenzhen University, 518060, Guangdong, China*

Abstract

More and more stock trading strategies are constructed using deep reinforcement learning (DRL) algorithms, but DRL methods originally widely used in the gaming community are not directly adaptable to financial data with low signal-to-noise ratios and unevenness, and thus suffer from performance shortcomings. In this paper, to capture the hidden information, we propose a DRL based stock trading system using cascaded LSTM, which first uses LSTM to extract the time-series features from stock daily data, and then the features extracted are fed to the agent for training, while the strategy functions in reinforcement learning also use another LSTM for training. Experiments in DJI in the US market and SSE50 in the Chinese stock market show that our model outperforms previous baseline models in terms of cumulative returns and Sharp ratio, and this advantage is more significant in the Chinese stock market, a merging markets. It indicates that our proposed method is a promising way to build a automated stock trading system.

Keywords: Deep Reinforcement Learning, Long Short-Term Memory, Automated stock trading, Proximal policy optimization, Markov Decision Process

1. Introduction

In recent years, more and more institutional and individual investors are using machine learning and deep learning methods for stock trading and asset management, such as stock price prediction using Random Forests, Long short-term memory(LSTM) Neural Networks or Support Vector Machines[1], which help traders to get optimal online strategies and obtain higher returns than strategies using only traditional factors[2][3][4] .

However, there are three main limitations of machine learning methods for stock market prediction: (i) financial market data are filled with noise and are unstable, and also contain the interaction of many unmeasurable factors. Therefore, it is very difficult to take into account all relevant factors in complex and dynamic stock markets[5][6][7]. (ii) Stock prices can be influenced by many other factors, such as political events, the behavior of other stock markets or even the psychology of investors[8]. (iii) Most of the methods are performed based on supervised learning and require training sets that are labeled with the state of the market, but such machine learning classifiers are prone to suffer from overfitting, which reduces the generalization ability of the model[9].

Fundamental data from financial statements and other data from business news, etc. are combined with machine learning algorithms that can obtain investment signals or make predictions about the prospects of a company[10][11][12][13] to screen for good investment targets. Such an algorithm solves the problem of stock screening, but it cannot solve how to allocate positions among the investment targets. In other words, it is still up to the trader to judge the timing of entry and exit.

To overcome the main limitations listed above, in this paper we use a deep reinforcement learning approach to construct low-frequency automated stock trading strategies to maximize expected returns. We consider stock trading as a Markov decision process that will be represented by states, actions, rewards, strategies, and values in a reinforcement learning algorithm. Instead of relying on labels (e.g., up and down in the market) to learn, the reinforcement learning approach learns how to maximize the value function during the training phase. We mainly use the PPO algorithm to train the agent and

*Corresponding author

Email addresses: `ianzou2000@163.com` (Jie Zou), `loujiashu@163.com` (Jiashu Lou), `bhwang@szu.edu.cn` (Baohua Wang)

combine it with LSTM to extract time-series features for an initial state of a certain time window length, while the initial state is represented by the adjusted stock price, available balance, shares of the underlying asset plus some technical indicators: MACD, RSI, CCI and ADX, illustrated in Figure 1. We take the ensemble strategy proposed by Yang, Liu[14] as the baseline and further expand on their work, so the training environment, state space, behavior space and value function we use are consistent with it.

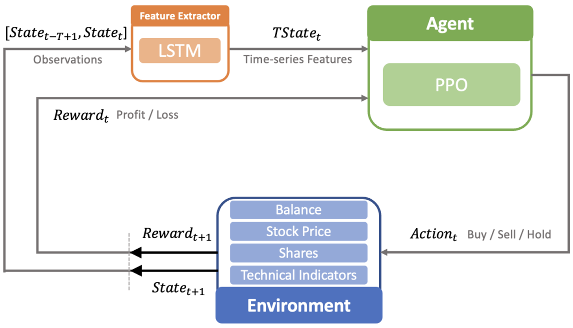

Figure 1: Agent-environment interaction in reinforcement learning

Experiments show that the automated stock trading strategy based on LSTM outperform, the ensemble strategy and the buy-and-hold strategy represented by the Dow Jones Industrial (DJI) index in terms of cumulative return,and also has better performance in Chinese stock market in terms of cumulative return and Sharp ratio.

The main contributions of this paper are two-fold: (i) In the literature, security's past price and related technical indicators are often used to represent state spaces,Instead of using that raw data,we use LSTM extract the time-series features from stock daily data to represent state spaces, since the memory property of the LSTM can discover features of the stock market that change over time and integrate information hidden in the time dimension, thus making it more likely that the POMDP is closer to the MDP. (ii) Different from the previous DRL based methods which use multi-layer neural networks or convolutional networks in agent training, we use LSTM as the training network because it is a type of recurrent neural network capable of learning order dependence in sequence prediction problems.

The rest of this paper is organized as follows. Section 2 is a brief introduction to the work related to stock trading using reinforcement learning, categorized by the training algorithm. Section 3 focuses on our algorithm, defining the necessary constraints in the environment and the framework of the agent. Section 4 presents the results and analysis of the experiments, covering the introduction of datasets, baseline models and evaluation metrics, process of finding the optimal parameters, and experiments in both Chinese and U.S. markets. In the conclusion, the paper is summarized and given some advice on future work.

2. Related Work

This section briefly summarizes the application of reinforcement learning and LSTM in quantitative trading, reviewing three learning methods in reinforcement learning that are frequently applied to financial markets and LSTM neural networks that are applied to predict stock prices. These three learning methods are: critical-only learning, actor-only learning and actor-critic learning.

2.1. Critic-only

The Critic-only approach, the most common of the three, uses only the action-value function to make decisions with the aim of maximizing the expected reward for each action choice given the current state. The action-value function Q receives the current state and the possible actions to be taken as input, then outputs an expected Q value as a reward. One of the most popular and successful approaches is Deep Q Network (DQN)[15] and its extensions[16]. Chen[17], Dang[18] and Jeong[19] used this method to train agents on a single stock or asset. Chen[17], Huang[20] used Deep Recurrent Q Network (DRQN) trained agents to achieve higher cumulative performance on quantitative trading than baseline models and DQN to achieve higher cumulative returns than baseline models. However, the main limitation of the method is that it performs better on discrete state spaces, however, the stock prices are continuous. If a larger number of stocks or assets are selected, the state space and action space will grow exponentially[21], which will weaken the performance of DQN.

2.2. Actor-only

The actor-only approach is able to learn policies directly and the action space can be considered as continuous. Therefore, the advantage of this approach is that it can directly learn the optimal mapping from a particular state to an action, which can be either discrete or continuous. Its disadvantages are that it requires a large amount of data for experiments and a long time to obtain the optimal strategy[22]. Deng[23] used this method and applied Recurrent Deep Neural Network to real-time financial trading for the first time. Wu[24] also explored the actor-only method in quantitative trading, where he compared deep neural networks (LSTM) with fully connected networks in detail and discussed the impact of some combinations of technical indicators on the daily data performance of the Chinese market, proving that deep neural networks are superior. The results of his experiments are mixed, and he shows that the proposed approach can yield decent profits in some stocks, but performs mediocrely in others.

2.3. *Actor-critic*

The actor-critic approach aims to train two models simultaneously, with the actor learning how to make the agent respond in a given state and the critic evaluating the responses. Currently, this approach is considered to be one of the most successful algorithms in RL, while Proximal Policy Optimization (PPO) is the most advanced actor-critic approach available. It performs better because it solves the well-known problems when applying RL to complex environments, such as instability due to the distribution of observations and rewards that constantly change as the agent learns[25]. In this paper, the baseline model[14] is constructed based on the actor-critic approach, using a combination of three DRL algorithms: PPO, A2C, and DDPG. However, the agent that learns only with PPO outperforms the resemble strategy in terms of cumulative return.

2.4. *LSTM in stock system*

Although Long Short-Term Memory Networks (LSTM)[26] are traditionally used in natural language processing, many recent works have applied them in financial markets to filter some noise in raw market data[41][42][43][44]. Stock prices and some technical indicators generated from stock prices are interconnected, so LSTM can be used as a feature extractor to extract potentially profitable patterns in the time series of these indicators. Zhang[22], Wu[24] have tried to integrate LSTM for feature extraction while training agent using DRL algorithm and the experiments have shown that it works better than baseline model. And the work of Lim[45] shows that LSTM delivers superior performance on modelling daily financial data.

3. Our method

3.1. *Stock Market Environment*

The stock market environment used in this paper is a simulation environment developed in [14] based on the OpenAI gym[28][29][30], which is able to give the agent various information for training, such as current stock prices, shareholdings and technical indicators. We use Markov Decision Process (MDP) to model stock trading[36], so the information that should be included in this multi-stock trading environment are: state, action, reward, policy and Q-value. Suppose we have a portfolio with 30 stocks.

3.1.1. *State Space*

A 181-dimensional vector consisting of seven parts of information represents the state space of the multi-stock trading environment: $[b_t, \boldsymbol{p}_t, \boldsymbol{h}_t, \boldsymbol{M}_t, \boldsymbol{R}_t, \boldsymbol{C}_t, \boldsymbol{X}_t]$. Each of these components is defined as follows.

1. $b_t \in \mathbb{R}_+$: available balance at current time step t.
2. $\boldsymbol{p}_t \in \mathbb{R}_+^{30}$: adjusted close price of each stock at current time step t.
3. $\boldsymbol{h}_t \in \mathbb{Z}_+^{30}$: shares owned of each stock at current time step t.
4. $\boldsymbol{M}_t \in \mathbb{R}^{30}$: Moving Average Convergence Divergence (MACD) is calculated using close price of each stock at current time step t. MACD is one of the most commonly used momentum indicators that identifies moving averages[32].
5. $\boldsymbol{R}_t \in \mathbb{R}_+^{30}$: Relative Strength Index (RSI) is calculated using close price of each stock at current time step t. RSI quantifies the extent of recent price changes[32].
6. $\boldsymbol{C}_t \in \mathbb{R}_+^{30}$: Commodity Channel Index (CCI) is calculated using high, low and close price. CCI compares current price to average price over a time window to indicate a buying or selling action[33].
7. $\boldsymbol{X}_t \in \mathbb{R}^{30}$: Average Directional Index (ADX) is calculated using high, low and close price of each stock at current time step t. ADX identifies trend strength by quantifying the amount of price movement[34].

3.1.2. *Action Space*

A set containing $2k + 1$ elements represents the action space of the multi-stock trading environment: $\{-k, ..., -1, 0, 1, ..., k\}$, where $k, -k$ represents the number of shares we can buy and sell at once. It satisfies some conditions as follows.

1. h_{max} represents the maximum number of shares we can be able to buy at a time.
2. The action space can be considered continuous since the entire action space is of size $(2k + 1)^{30}$.
3. The action space will next be normalized to $[-1, 1]$.

3.1.3. Reward

We define the reward value of the multi-stock trading environment as the change in portfolio value from state s taking action a to the next state s' (in this case two days before and after), with the training objective of obtaining a trading strategy that maximizes the return:

$$Return_t\,(s_t,\ a_t,\ s_{t+1}) = \left(b_{t+1} + \boldsymbol{p}_{t+1}^T \boldsymbol{h}_{t+1}\right) - \left(b_t + \boldsymbol{p}_t^T \boldsymbol{h}_t\right) - c_t \tag{1}$$

where c_t represents the transaction cost. We assume that the per-transaction cost is 0.1% of each transaction, as defined in [10]:

$$c_t = 0.1\% \ \times\ \boldsymbol{p}^T \boldsymbol{k}_t \tag{2}$$

3.1.4. Turbulence Threshold

We employ this financial index $turbulence_t$[14] that measures extreme asset price movements to avoid the risk of sudden events that may cause stock market crash[35], such as March 2020 stock market caused by COVID-19, wars and financial crisis:

$$turbulence_t = (\boldsymbol{y}_t - \boldsymbol{\mu})\,\boldsymbol{\Sigma}^{-1}\,(\boldsymbol{y}_t - \boldsymbol{\mu})' \in \mathbb{R} \tag{3}$$

Where $\boldsymbol{y}_t \in \mathbb{R}^{30}$ denotes the stock returns for current period t, $\boldsymbol{\mu}_t \in \mathbb{R}^{30}$ denotes the average of historical returns, and $\boldsymbol{\Sigma} \in \mathbb{R}^{30\times 30}$ denotes the covariance of historical returns. Considering the historical volatility of the stock market, we set the turbulence threshold to 90th percentile of all historical turbulence indexes. If $turbulence_t$ is greater than this threshold, it means that extreme market conditions are occurring and the agent will stop trading until the turbulence index falls below this threshold.

3.1.5. Other Parameters

In addition to defining the state space, action space and reward functions, some necessary constraints need to be added to the multi-stock trading environment.

1. Initial capital: \$1 million.
2. Maximum number of shares in a single trade h_{max}: 100.
3. Reward scaling factor: 1e-4, which means the reward returned by the environment will be only 1e-4 of the original one.

3.2. Stock Trading Agent

3.2.1. Framework

We introduce LSTM as a feature extractor to improve the model in [14], as shown in Figure 2.

At the current time step t, the environment automatically generates the current state S_t and passes it to the

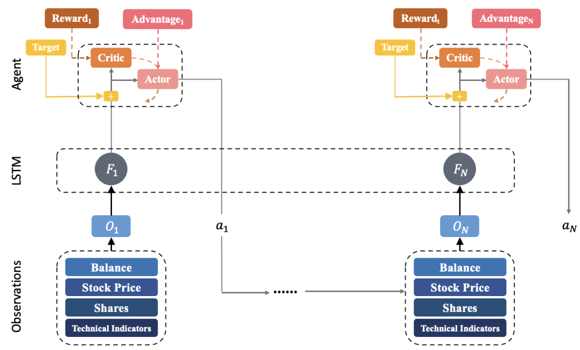

Figure 2: Overview of our model

LSTM network. It then remembers the state and uses its memory to retrieve the past T stock market states to obtain the state sequence $F_t = [S_{t-T+1}, ..., S_t]$. The LSTM analyzes and extracts the hidden time-series features or potentially profitable patterns in F_t, and then outputs the encoded feature vector F'_t and passes it to the agent, which is guided by the policy function $\pi(F'_t)$ to perform the optimal action a_t. The environment then returns the reward R_t , the next state S_{t+1} , and a boolean d_t to determine if the state is terminated according to the agent's behavior. Then, the obtained quintet $(S_t,\ a_t,\ R_t,\ S_{t+1},\ d_t)$ is stored in the experience pool. Actor computes A_t from the $targets_t$ computed by critic using the advantage function. After a certain number of steps, actor back-propagates the error through the loss function $J^{\theta_i}(\theta) = \sum_{t=1}^{T}\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_i}(a_t|s_t)} A_t^{\pi_\theta} - \beta KL(\theta, \theta_i)\right)$ and updates θ using the gradient descent method. And critic updates the parameters using the mean square error loss function. And the environment will keep repeating the process until the end of the training phase.

3.2.2. LSTM as Feature Extractor

Reinforcement Learning (RL) was initially applied to games, which have a limited state space, a limited action space, clear stopping conditions and a more stable environment, so there is room to improve the use of RL for stock trading. It is well known that financial markets are full of noise and uncertainty, and that the factors affecting stock prices are multiple and changing over time. This makes the stock trading process more like a partially observable Markov decision process (POMDP), since the states we use are not the real states in the stock trading environment. Therefore, we can use the memory property of the LSTM to discover features of the stock market that change over time. LSTM can integrate information hidden in the time dimension, thus making it more likely that the POMDP is closer to the

MDP[24][31].

In this paper, an LSTM-based feature extractor is developed using the customized feature extraction interface provided by stable-baselines3. The network structure of the LSTM feature extractor is shown in Figure 3.

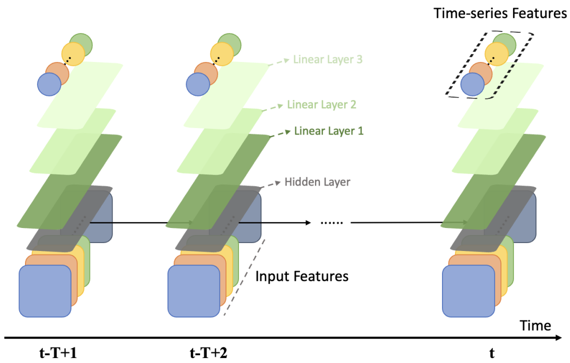

Figure 3: Overview of our model

As shown in the figure 3, each input to the LSTM is a state list of length T arranged in time order. Starting from the farthest state, the hidden layer of the LSTM remembers the information of the state and passes it to the next point in time. We use the feature vector of the most recent state, which is the current state, after one hidden layer of the LSTM with three linear layers. This feature vector F_t' will be used as input feature of goes into the PPO for the agent to learn.

The operation of LSTM in PPO is similar to it, which is equivalent to making the agent recall the behavioral information of the previous moment while receiving new data, so that the decision made by the agent at this moment is based on the previous decision.

In the LSTM network, the initial number of features is 181, the final number of output features is 128 and the hidden size is 128. Linear layer 1 is $(15 \times 128,\ 128)$ and then passes the Tanh activation function. Linear layer2 and 3 are the same two layers of $(128,\ 128)$, and then pass Tanh.

3.2.3. Proximal Policy Optimization (PPO)

PPO[37] is one of the most advanced of the current policy-based approaches which use multiple epochs of stochastic gradient ascent to perform each policy update[39], and also performs best in stock trading among the three DRL algorithms in [14], which is an important reason for our consideration of it. PPO is very closely related to Trust Region Policy Optimization (TRPO): it can be seen as a refinement of TRPO[38]. In PPO, parameters of actor(agent) is θ.

First, a notation is used to represent the probability ratio between the new policy and the old one:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \tag{4}$$

So we can get $r_t(\theta_{old}) = 1$ from it. The clipped surrogate objective function of PPO is:

$$\begin{aligned} H^{CLIP}(\theta) = \hat{\mathbb{E}}[\min(&r_t(\theta)\hat{A}(s_t,\ a_t), \\ &clip(r_t(\theta), r_t(\theta_{old}) - \epsilon, r_t(\theta_{old}) + \epsilon)\hat{A}(s_t,\ a_t))] \end{aligned} \tag{5}$$

That is,

$$\begin{aligned} H^{CLIP}(\theta) = \hat{\mathbb{E}}[\min(&r_t(\theta)\hat{A}(s_t,\ a_t), \\ &clip(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}(s_t,\ a_t))] \end{aligned} \tag{6}$$

Where $r_t(\theta)\hat{A}(s_t,\ a_t)$ is the normal policy gradient objective in TRPO, and $\hat{A}(s_t,\ a_t)$ is the estimated advantage function. Term $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ clips the ratio $r_t(\theta)$ to be within $[1-\epsilon, 1+\epsilon]$. Finally, $H^{CLIP}(\theta)$ takes the minimum of the clipped and unclipped objective.

PPO contains some additional main optimizations except the value function clipping[38]: (i) Reward scaling: Instead of feeding rewards directly from the environment to the target, the PPO implementation performs some sort of discount-based scaling scheme. (ii) Orthogonal initialization and layer scaling: Rather than using the default weight initialization scheme in the policy and value network, an orthogonal initialization scheme is used, with the proportions varying between the layers. (iii) Adam learning rate annealing: PPO sometimes anneals the learning rate for optimization[40].

4. Performance Evaluations

In this section, we first tuned some parameters in the model, then used 30 Dow constituent stocks to evaluate our model, and finally performed robustness tests on 30 stocks in the SSE 50.

4.1. Description of Datasets

In this paper, 60 stocks are selected as the stock pool: 30 Dow constituent stocks, and 30 stocks randomly selected from the SSE50 in Chinese stock market. The stocks from Dow Jones are the same as the pool in [14] to facilitate comparison with their ensemble strategy, while the 30 stocks from SSE 50 are used to explore the applicability of this paper's model in the Chinese stock market, a merging market.

The daily data for backtesting starts from 01/01/2009 and ends on 05/08/2020, and the data set is divided into two parts: in-sample period and out-sample period. The data in the in-sample period is used for training and validation, and the data in the out-sample period is used for trading. We only use PPO during the whole process.

The entire dataset is split as shown in the Figure 4. The training data is from 01/01/2009 to 09/30/2015, the validation data is from 10/01/2015 to 12/31/2015, and the trading data is from 01/01/2016 to 05/08/2020. In order to better exploit the data and allow the agent to better adapt to the dynamic changes of the stock market, the agent can continue to be trained during the trading phase.

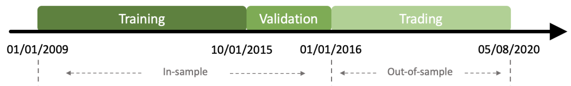

Figure 4: Stock Data Splitting

4.2. *Training Parameters of PPO*

The agent is trained using only the actor-critic based PPO method, and the training parameters of PPO are set as shown in the table 1.

Table 1: Training parameters of PPO

Parameter	**Value**
Reward Discount Factor	0.99
Update Frequency	128
Loss Function Weight of Critic	0.5
Loss Function Weight of Distribution Entropy	0.01
Clip Range	0.2
Maximum of Gradient Truncation	0.5
Optimizer	Adam
β_1	0.9
β_2	0.999
ϵ	1e-8
Learning Rate	3e-4

4.3. *Baseline Methods*

Our model is compared with baseline models including:

- **S&P 500**: the typical Buy-And-Hold strategy.
- **SSE50**: another Buy-And-Hold strategy in Chinese market.
- **Ensemble Strategy in [14]**: they train agents for three months simultaneously in the training phase using A2C, DDPG and PPO algorithms and then select the agent with the highest Sharpe ratio as the trader for the next quarter. This process is repeated until the end of the training.

4.4. *Evaluation Measures*

- **Comulative Return (CR)**: calculated by subtracting the portfolio's final value from its initial value, and then dividing by the initial value. It reflects the total return of a portfolio at the end of trading stage.

$$CR = \frac{P_{end} - P_0}{P_0} \tag{7}$$

- **Max Earning Rate (MER)**: the maximum percentage profit during the trading period. It measures the robustness of a model and reflects the trader's ability to discover the potential maximum profit margin.

$$MER = \frac{max(A_x - A_y)}{A_y} \tag{8}$$

Where A_x, A_y is the total asset of the strategy and $x > y$, $A_y < A_x$.

- **Maximum Pullback (MPB)**: the maximum percentage loss during the trading period. It measures the robustness of a model.

$$MPB = \frac{max(A_x - A_y)}{A_y} \tag{9}$$

Where A_x, A_y is the total asset of the strategy and $x > y$, $A_y > A_x$.

- **Average Profitability Per Trade (APPT)**: refers to the average amount that you can expect to win or lose per trade. It measures the trading performance of the model.

$$APPT = \frac{P_{end} - P_0}{NT} \tag{10}$$

Where $P_{end} - P_0$ means the returns at the end of trading stage, and NT is the number of trading.

- **Sharpe Ratio (SR)**: calculated by subtracting the annualized risk free rate from the annualized return, and the dividing by the annualized volatility. It considers benefits and risks synthetically and reflects the excess return over unit systematic risk.

$$SR = \frac{E(R_P) - R_f}{\sigma_P} \tag{11}$$

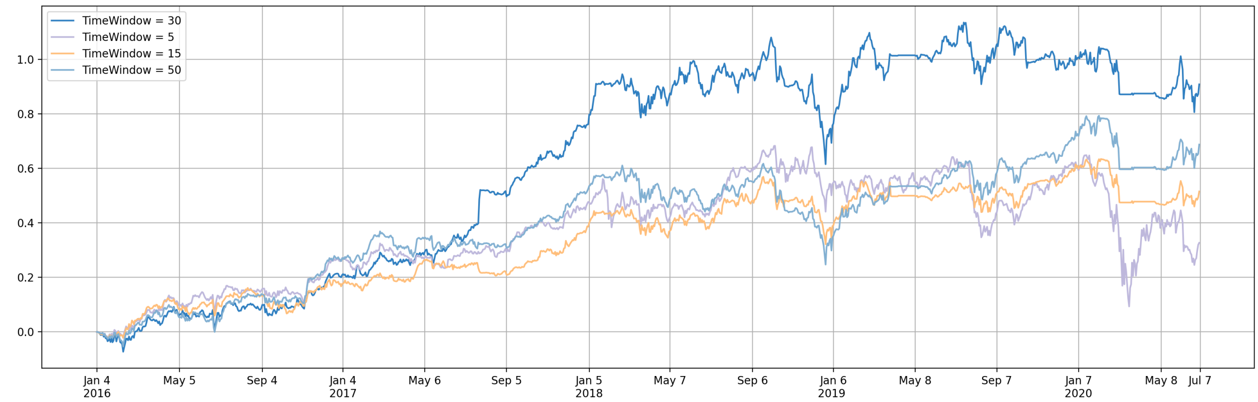

Figure 5: Trading results of different time windows in LSTM

4.5. Exploration of Optimal Hyperparameters

We performed parameter tuning on two important parts of the model: (i) the time window size of the LSTM as a feature extractor. (ii) the hidden size of LSTM in PPO training.

4.5.1. Best Time Window of LSTM

For the time window of the LSTM, we tested the cases of TW=5,15,30,50, and then show the trading results of the model in Figure 5. (hidden size of LSTM in PPO is 512) From the figure, it can be seen that the agent at TW=30 is able to achieve the highest cumulative return during the trading period, ahead of the agent at TW=50 by more than 20% and a figure that exceeds 40% in agent without LSTM extraction. This verifies the feasibility of our model: the stock price movements in the stock market are correlated with their past trajectories, and the LSTM is able to extract the time-series features of them.

Table 2: Comparison of different time windows in LSTM

	TW=5	TW=15	TW=30	TW=50
CR	32.69%	51.53%	**90.81%**	68.74%
MER	68.27%	63.46%	**113.50%**	79.32%
MPB	58.93%	**24.75%**	46.51%	37.01%
APPT	18.29	21.77	**35.27**	23.31
SR	0.2219	0.7136	**1.1540**	0.9123

The data in the table 2 shows the difference between these options in more detail: for TimeWindow=30, CR, MER, APPT and SR are much higher than the other options, but MPB does not perform well. On balance, TW = 30 is the optimal parameter.

4.5.2. Best Hidden Size of LSTM in PPO

For the hidden size of the LSTM in PPO, we tested the cases of HS=128, 256, 512, 1024, 512*2(two hidden layers) and then show the trading results of the model in Figure 6. (time window of LSTM is 30)

As can be seen from the figure 6, when hidden size=512, the cumulative yield is significantly higher than the other choices. It has a smaller drawback compared to hidden size=512*2 and was able to stop trading in the big drawback in March 2020, indicating that the agent can be a smart trader under the right training conditions of DRL.

Table 3: Comparison of different hidden sizes of LSTM in PPO

	HS=128	HS=256	HS=512	HS=1024	HS=512*2
CR	69.94%	72.58%	**90.81%**	56.27%	89.13%
MER	84.29%	82.32%	**113.50%**	60.64%	92.34%
MPB	38.57%	**29.92%**	46.51%	30.39%	58.31%
APPT	28.07	30.79	**35.27**	23.96	33.26
SR	0.9255	1.0335	**1.1540**	0.8528	0.8447

The data in the table 3 shows the difference between these options in more detail: for HiddenSize=512, CR, MER, APPT and SR are much higher than the other options, but MPB does not perform well. On balance, HS = 512 is the optimal parameter.

4.6. Performance in U.S. Markets

The optimal parameters found (TW=30, HS=512) were used as the parameters of the final model and the results of this model were compared with the trading results of the PPO model with LSTM in PPO and another with MlpPolicy and the Ensemble Strategy in [14], as shown in Figure 7.

Table 4 shows the details of trading results in U.S. Markets.

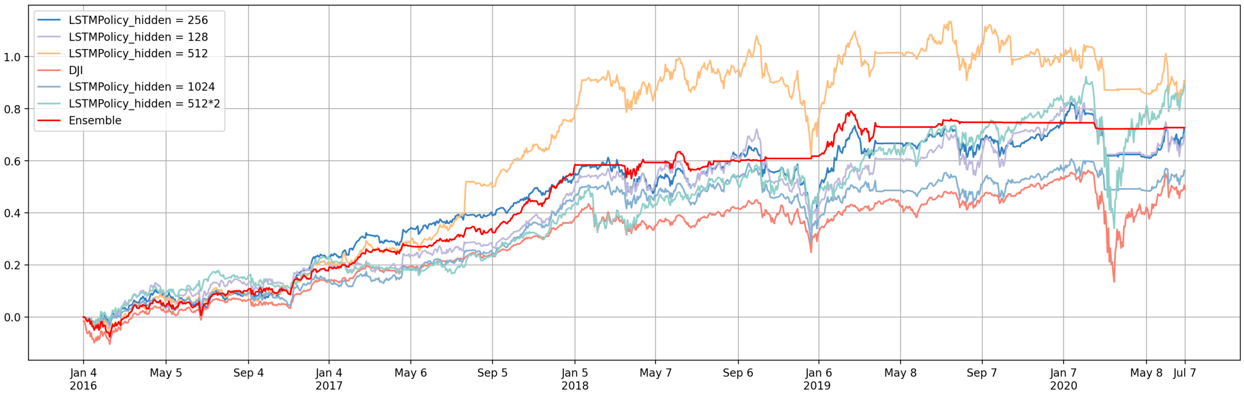

Figure 6: Trading results of different hidden sizes of LSTM in PPO

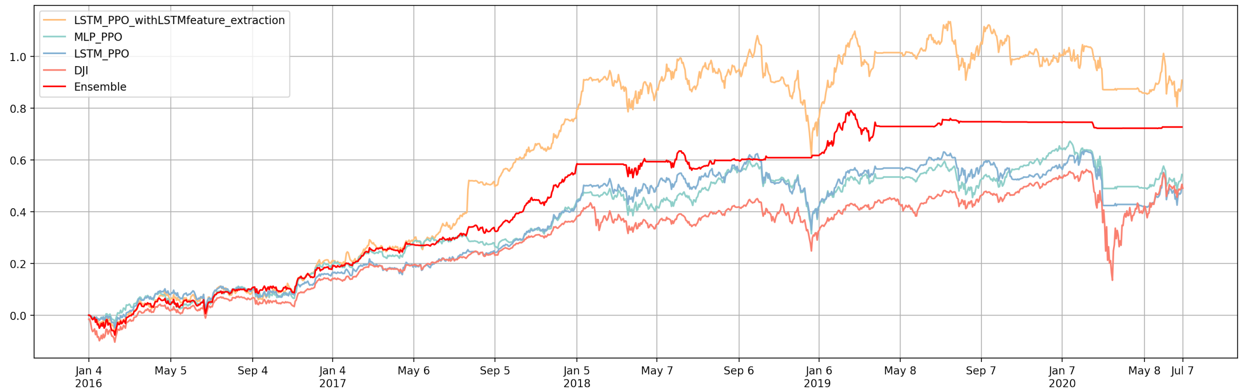

Figure 7: Trading results by agents with LSTM in PPO plus LSTM feature extraction, LSTM in PPO, Ordinary PPO, Ensemble Strategy in [14] and Buy-And-Hold strategy on DJI

Table 4: Details of the trading results in U.S. Markets

	PPO	LSTM in PPO	Our Model	Ensemble	DJI
CR	54.37%	49.77%	**90.81%**	70.40%	50.97%
MER	67.28%	63.45%	**113.50%**	65.32%	63.90%
MPB	28.30%	29.39%	46.51%	**15.74%**	72.32%
APPT	20.02	22.84	**35.27**	28.54	N.A.
SR	0.8081	0.6819	1.1540	**1.3000**	0.4149

Our model obtains the highest cumulative return of 90.81% and the maximum profitability of 113.5% on 30 Dow components, better than the ensemble strategy in [14], the baseline model. However, in terms of maximum pullback, our model has an MPB=45.51% compared to DJI's MPB=63.90%, indicating the possession of risk tolerance and the ability to identify down markets and stop trading. In this respect, ensemble strategy does a little better. But for a long time the agent of ensemble strategy has a negative attitude towards investing, choosing not to trade whether the market is down or up. This can lead to the loss of a large profit margin in the long run. Overall, our model has the strongest profit-taking ability, excels at finding profits within volatile markets, and recovers quickly after pullbacks. In terms of the Sharpe ratio, our strategy is very close to the baseline model, but does not require the help of other algorithms, which is much easier and faster.

Analyzing in more detail, all strategies can be divided into two phases which are consistent with DJI index: (i) Accumulation phase: until 06/2017, our strategies were able to achieve stable growth with little difference in returns from the integrated strategies. However, after that, our agent quickly captured profits and was able to grow total returns rapidly. This phase lasted until 01/2018, when the cumulative return had reached a level where the difference with the final return was not significant. (ii) Volatility phase: Starting from 01/2018, our agent's trading style became very aggressive and courageous, as reflected in the large fluctuations in returns. The returns were generally more stable during this phase and were able to bounce back quickly within two months

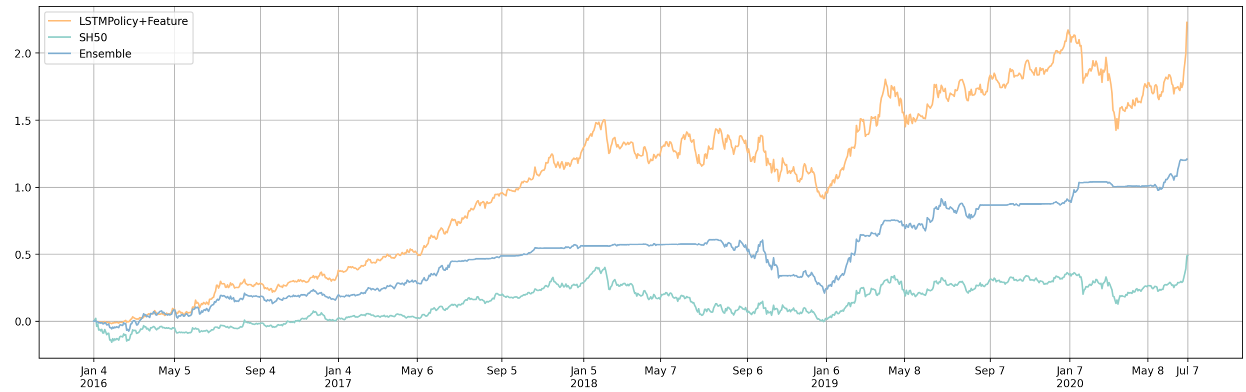

Figure 8: Trading results by agents with LSTM in PPO plus LSTM feature extraction, Ensemble Strategy in [14] and Buy-And-Hold strategy on SSE50

after suffering a pullback on 01/2019.

4.7. Performance in Chinese Markets

The same model is used to trade the samples in the Chinese stock market, as Figure 8 shown, and the ensemble strategy is chosen as the most important baseline model. We show the cumulative returns at the end of the trade as follows.

And table 5 shows the details of trading results in Chinese Markets. In the Chinese market, the advantages of our model are greater than the ensemble strategy. Our CR=222.91% is nearly twice that of ensemble strategy (Ensemble_CR=120.87%). Although our pullbacks are greater, the volatility is more reflected in the rise in cumulative returns. Also, the Sharpe ratio tells us that our model (SR=2.3273) is better in the Chinese market when combining return and risk. This illustrates our model's superior performance in emerging markets, which typically have greater volatility, and is consistent with our analysis of our model: the ability to capture returns in volatility quickly and accurately, and the returns obtained are positively correlated with positive volatility.

Table 5: Details of the trading results in Chinese Markets

	Our Model	Ensemble	SSE50
CR	**222.91%**	120.87%	51.46%
MER	**222.91%**	120.87%	51.46%
MPB	74.81%	**39.95%**	41.27%
APPT	**66.96**	47.55	25.78
SR	**2.3273**	1.6938	0.4149

On further analysis, the cumulative return of our model increased rapidly from 01/2016 to 02/2018, finally reaching almost three times that of the integrated strategy. Subsequently, as the volatility of the SSE 50 increased, the volatility of our model increased accordingly. Within 02/2018 to 01/2019, the pullbacks of the two models were similar, but then our model captured nearly 80% of the return at 01/2019 within three months. Even though it suffered a decline in 01/2020 due to the black swan event of Covid-19, it quickly bounced back to its highest point six months later.

5. Conclusion

In this paper, we propose a PPO model using cascaded LSTM networks and compare the ensemble strategy in [14] as a baseline model in the U.S. market and the Chinese market, respectively. The results show that our model has a stronger profit-taking ability, and this feature is more prominent in the Chinese market. However, according to the risk-return criterion, our model is exposed to higher retracement risk while obtaining high returns. Finally, we believe that the advantages of our model can be more fully realized in markets with smoother overall trends and less volatility (such as the A-share market in China in recent years). This suggests that there is indeed a potential return pattern in the stock market, and the LSTM as a time-series feature extractor plays an active role. Also, it shows that the Chinese market is a suitable environment for developing quantitative trading.

In the subsequent experiments, improvements can be made in the following aspects: (i) The amount of training data. The training of PPO requires a large amount of historical data to achieve good learning results, so expanding the amount of training data may help to improve the results. (ii) Reward function. Some improved reward functions for stock trading have emerged, which can enhance the stability of the algorithm.

References

[1] Fang, Y.-C. et al. "Research on Quantitative Investment Strategies Based on Deep Learning." Algorithms 12 (2019): 35.

[2] Kuo, Ren Jie. "A Decision Support System for the Stock Market Through Integration of Fuzzy Neural Networks and Fuzzy Delphi." Appl. Artif. Intell. 12 (1998): 501-520.

[3] N.BaBa and M. Kozaki, "An intelligent forecasting system of stock price using neural networks," Proc. IJCNN, Baltimore, Maryland, 652-657, 1992.

[4] S . Cheng, A Neural Network Approach for Forecasting and Analyzing the Price-Volume Relationship in the Taiwan Stock Market, Master's thesis, National Jow-Tung University, 1994.

[5] Stelios D. Bekiros, "Fuzzy adaptive decision-making for boundedly rational traders in speculative stock markets," European Journal of Operational Research, vol. 202, no. 1, pp. 285–293, 2010.

[6] Yong Zhang and Xingyu Yang, "Online portfolio selection strategy based on combining experts' advice," Computational Economics, vol. 50, 05 2016.

[7] Youngmin Kim, Wonbin Ahn, Kyong Joo Oh, and David Enke, "An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms," Applied Soft Computing, vol. 55, pp. 127–140, 02 2017.

[8] Zhang, Y., & Wu, L, Stock market prediction of s&p 500 via combination of improved bco approach and bp neural network. Expert Systems with Applications, 36, 8849–8854, 2009.

[9] Carta, S.M., Ferreira, A., Podda, A.S., Recupero, D.R., & Sanna, A. ,Multi-DQN: An ensemble of Deep Q-learning agents for stock market forecasting. Expert Syst. Appl., 164, 113820, 2021.

[10] Hongyang Yang, Xiao-Yang Liu, and Qingwei Wu, "A practical machine learning approach for dynamic stock recommendation," in IEEE TrustCom/BiDataSE, 2018., 08 2018, pp. 1693–1697.

[11] Yunzhe Fang, Xiao-Yang Liu, and Hongyang Yang, "Practical machine learning approach to capture the scholar data driven alpha in ai industry," in 2019 IEEE International Conference on Big Data (Big Data) Special Session on Intelligent Data Mining, 12 2019, pp. 2230–2239.

[12] Wenbin Zhang and Steven Skiena, "Trading strategies to exploit blog and news sentiment.," in Fourth International AAAI Conference on Weblogs and Social Media, 2010, 01 2010.

[13] Qian Chen and Xiao-Yang Liu, "Quantifying esg alpha using scholar big data: An automated machine learning approach," ACM International Conference on AI in Finance, ICAIF 2020, 2020.

[14] Yang, Hongyang, Xiao-Yang Liu, Shan Zhong, and Anwar Walid. "Deep reinforcement learning for automated stock trading: An ensemble strategy." In Proceedings of the First ACM International Conference on AI in Finance, pp. 1-8. 2020.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. nature, 518(7540):529-533, 2015.

[16] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double qlearning. In Proceedings of the AAAI conference on artificial intelligence, volume 30, 2016.

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

[18] Quang-Vinh Dang, "Reinforcement learning in stock trading," in Advanced Computational Methods for Knowledge Engineering. ICCSAMA 2019. Advances in Intelligent Systems and Computing, vol 1121. Springer, Cham, 01 2020.

[19] Gyeeun Jeong and Ha Kim, "Improving financial trading decisions using deep q-learning: predicting the number of shares, action strategies, and transfer learning," Expert Systems with Applications, vol. 117, 09 2018.

[20] Chien Yi Huang. Financial trading as a game: A deep reinforcement learning approach. arXiv:1807.02787, 2018.

[21] Zhuoran Xiong, Xiao-Yang Liu, Shan Zhong, Hongyang Yang, and Anwar Walid. Practical deep reinforcement learning approach for stock trading. arXiv preprint arXiv:1811.07522, 2018.

[22] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deep reinforcement learning for trading. The Journal of Financial Data Science, 2(2):25-40, 2020.

[23] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. IEEE transactions on neural networks and learning systems, 28(3):653-664, 2016.

[24] WU Jia, WANG Chen, Lidong Xiong, and SUN Hongyong. Quantitative trading on stock market based on deep reinforcement learning. In 2019 International Joint Conference on Neural Networks (IJCNN), pages 1-8. IEEE, 2019

[25] Pricope, Tidor-Vlad. "Deep reinforcement learning in quantitative algorithmic trading: A review." arXiv preprint arXiv:2106.00123, 2021.

[26] Sepp Hochreiter and J¨urgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735-1780, 1997.

[27] A. Ilmanen, "Expected returns: An investor's guide to harvesting market rewards," 05 2012.

[28] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, "Openai gym," 2016.

[29] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.

[30] Ashley Hill, Antonin Raffin, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, Rene Traore, Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.

[31] Lample, Guillaume, and Devendra Singh Chaplot. "Playing FPS gameswith deep reinforcement learning." Thirty-First AAAI Conference on Artificial Intelligence. 2017.

[32] Terence Chong, Wing-Kam Ng, and Venus Liew, "Revisiting the performance of macd and rsi oscillators," Journal of Risk and Financial Management, vol. 7, pp. 1–12, 03 2014.

[33] Mansoor Maitah, Petr Prochazka, Michal Cermak, and Karel Sredl, "Commodity channel index: evaluation of trading rule of agricultural commodities," International Journal of Economics and Financial Issues, vol. 6, pp. 176–178, 03 2016.

[34] Ikhlaas Gurrib, "Performance of the average directional index as a market timing tool for the most actively traded usd based currency pairs," Banks and Bank Systems, vol. 13, pp. 58–70, 08 2018.

[35] Mark Kritzman and Yuanzhen Li, "Skulls, financial turbulence, and risk management," Financial Analysts Journal, vol. 66, 10

2010.
[36] A. Ilmanen, “Expected returns: An investor’s guide to harvesting market rewards,” 05 2012.
[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov, “Proximal policy optimization algorithms,” arXiv:1707.06347, 07 2017.
[38] Engstrom, Logan, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. ”Implementation matters in deep policy gradients: A case study on PPO and TRPO.” arXiv preprint arXiv:2005.12729, 2020.
[39] Y. Gu, Y. Cheng, C. L. P. Chen, and X. Wang, “Proximal Policy Optimization With Policy Feedback”, IEEE Transactions on Systems, Man, and Cybernetics: Systems, pp. 1–11, 2021.
[40] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. CoRR, abs/1412.6980, 2014.
[41] Wei Bao, Jun Yue, and Yulei Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one, 12(7):e0180944, 2017.
[42] Luca Di Persio and Oleksandr Honchar. Artificial neural networks architectures for stock price prediction: Comparisons and applications. International journal of circuits, systems and signal processing, 10:403–413, 2016.
[43] Thomas Fischer and Christopher Krauss. Deep learning with long short-term memory networks for financial market predictions. European Journal of Operational Research, 270(2):654–669, 2018.
[44] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning to detect price change indications in financial markets. In 2017 25th European Signal Processing Conference (EUSIPCO), pages 2511–2515. IEEE, 2017.
[45] Bryan Lim, Stefan Zohren, and Stephen Roberts. Enhancing time-series momentum strategies using deep neural networks. The Journal of Financial Data Science, 2019.