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

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

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

Abstract

Deep Reinforcement Learning (DRL) algorithms have been increasingly used to construct stock trading strategies, but they often face performance challenges when applied to financial data with low signal-to-noise ratios and unevenness, as these methods were originally designed for the gaming community. To address this issue, we propose a DRL-based stock trading system that leverages Cascaded Long Short-Term Memory (CLSTM-PPO Model) to capture the hidden information in the daily stock data. Our model adopts a cascaded structure with two stages of carefully designed deep LSTM networks: uses one LSTM to extract the time-series features from a sequence of daily stock data in the first stage, and then the features extracted are fed to the agent in the reinforcement learning algorithm for training, while the actor and critic in the agent also use LSTM network. We conduct experiments on stock market datasets from four major indices: Dow Jones Industrial index(DJI) in the US, Shanghai Stock Exchange 50(SSE50) in China, S&P BSE Sensex Index(SENSEX) in India, and Financial Times Stock Exchange 100(FTSE100) in the UK. We compare our model with several benchmark models, including: 1) Buy and hold indexes. 2) PPO model with Multilayer Perceptron(MLP) policy. 3) Some up-to-date models like MLP model, LSTM model, Light Gradient Boosting Machine(LGBM) model, and Histogram-based Gradient Boosting model. 4) An ensemble strategy model. The experimental results show that our model outperforms the baseline models in several key metrics, such as cumulative returns, maximum earning rate, and average profitability per trade. The improvements range from 5% to 52% depending on the metric and the stock index. This indicates that our proposed method is a promising way to build an automated stock trading system.

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

1. Introduction

Machine learning and deep learning methods have become increasingly popular for stock trading and asset management in recent years. For example, Fang et al. (2019a) used Random Forests, Long Short-Term Memory (LSTM) Neural Networks, and Support Vector Machines to predict stock prices, which helped traders to obtain higher returns than strategies using only traditional factors (Kuo, 1998; Baba and Kozaki, 1992; Cheng, 1994).

However, machine learning methods for stock market prediction face three main challenges: (i) financial market data are noisy, unstable, and influenced by many immeasurable factors, making it very difficult to account for all relevant factors in complex and dynamic stock markets (Bekiros, 2010; Zhang and Yang, 2017; Kim et al., 2017); (ii) stock prices are affected by many external factors, such as political events, the behavior of other stock markets, or even the psychology of investors (Zhang and Wu, 2009); and (iii) most of the methods rely on supervised learning and require labeled training sets that indicate the state of the market, but such machine learning classifiers tend to overfit the data, which reduces the generalization ability of the model (Carta et al., 2021).

In this paper, we address the problem of automatically trading a portfolio of dozens of stocks with the goal of maximizing expected returns. We propose a low-frequency automated stock trading system based on

*Corresponding author

Email addresses: ianzou2000@163.com (Jie Zou), loujiashu@163.com (Jiashu Lou), bhwang@szu.edu.cn (Baohua Wang), 18801117020@163.com (Sixue Liu)

the Proximal Policy Optimization (PPO) algorithm, a Deep Reinforcement Learning (DRL) approach that belongs to the branch of deep learning. We model stock trading as a Markov decision process that consists of states, transitions, actions, rewards, strategies, and values. Unlike supervised learning methods that rely on labels (e.g., market trends) to learn, the DRL approach learns to optimize the policy function and the value function during the training phase. The DRL methods, which were originally designed for the gaming community with image as input, take the raw data as features and use MLP or convolution neural network for training, but the stock data is a time series data with low signal-to-noise ratios and unevenness. Therefore, it is not a feasible way to use the raw data as features and use MLP or convolution network for training since they cannn't learn the temporal dependencies effectively. To capture the hidden information in time series data and learn temporal dependencies effectivel, we propose a PPO based DRL automated stock trading model using Cascaded LSTM networks (CLSTM-PPO), which first employs a LSTM to extract the features from daily stock data, and then feeds the extracted features to the agent based on PPO for training. The actor and critic in the agent also uses LSTM network, as illustrated in Figure 1.

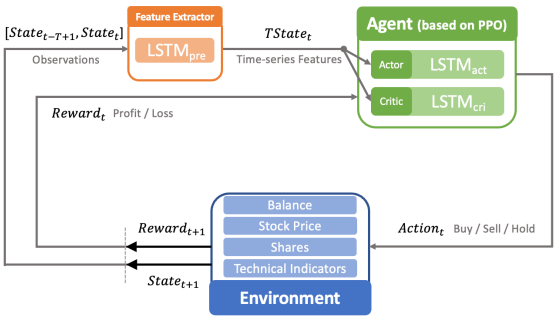

Figure 1: Agent-environment interaction in reinforcement learning

Our model is a novel approach to stock market prediction that leverages the advantages of LSTM which works better than other kinds of neural networks when data is sequential like time-series data and can learn long-term dependencies and PPO which is a state-of-the-art reinforcement learning algorithm. It adopts a cascaded structure with two stages of carefully designed deep LSTM networks: a feature extraction network and the agent training networks including an actor network and a critic network. The feature extraction network($LSTM_{pre}$) employs LSTM to capture the temporal dependencies and nonlinear patterns from the input time-series data, and outputs the extracted features as observations for the agent networks. The actor network($LSTM_{act}$) and critic network($LSTM_{cri}$) in the agent receive the observations and employs LSTM and bases on PPO to optimize policy function and value function. Therefore, partial observability and complex dynamics in the stock market can be handled. Cascaded structure is shown to be a feasible way for image captioning, face detection and alignment(Vinyals et al., 2015). The reason why we use the cascaded structure is because the financial time-series data is with low signal-to-noise ratios and unevenness, we need an feature extractor to capture the hidden information effectively and a smart agent with long short-term memory to make better investment decisions. Our experiments demonstrate superior performance of our cascaded structure compared to the single PPO model with LSTM networks in the actor and critic. We conduct extensive experiments to evaluate our approach on four different stock markets: US, UK, India, and China. We compare our approach with several baseline models, including some recent deep reinforcement learning methods for stock market prediction. The experimental results show that our approach achieves superior performance in all four markets, especially in the Chinese stock market where it significantly outperforms the other models in terms of various evaluation metrics. These findings demonstrate that our CLSTM-PPO approach is a robust and effective method for stock market prediction across different markets.

This paper makes two main contributions to the literature on DRL for stock trading: (i) We propose a novel state representation method that uses LSTM to extract time-series features from a sequence of stock daily data, instead of using the raw data or technical indicators as in previous studies. The advantage of this method is that the LSTM can capture the dynamic features of the stock market and incorporate hidden information in the time dimension, thus making the partially observable Markov decision process (POMDP) more similar to the Markov decision process (MDP). (ii) We adopt LSTM as the training network for the DRL agent, as it is a type of recurrent neural network that can learn order dependence in sequence prediction problems. This differs from the previous DRL methods that use multilayer neural networks or convolutional networks, which may not be able to handle temporal dependencies effectively. We also note that higher-order Markov models are another possible way to account for dependencies between a current state and a longer history of past states, which may improve the prediction accuracy in

some cases. However, we argue that higher-order models have several limitations and challenges, such as the high computational cost of training, the risk of overfitting, and the poor generalization to new data. Therefore, we suggest that LSTM-based models are more suitable and efficient for DRL-based stock trading.

This paper is structured as follows. In Section 2, we review the related work on stock trading using reinforcement learning, and classify them according to the training algorithm. In Section 3, we describe our proposed algorithm, and specify the environmental constraints and the agent's framework. In Section 4, we report and analyze the experimental results, which include the description of datasets, baseline models and evaluation metrics, the procedure of finding the optimal parameters, and the performance comparison in Chinese, Indian, UK and US markets. In Section 5, we conclude the paper and suggest some possible directions for future research.

2. Related Work

This section provides an overview of some machine learning methods used in investment, and evaluates the suitability of Geometric Brownian Motion for this domain. It then reviews the literature on reinforcement learning, LSTM, and some advanced models for quantitative trading, focusing on three reinforcement learning methods that are commonly applied to financial markets and LSTM neural networks that are used for stock price prediction. The three reinforcement learning methods are: critic-only learning, actor-only learning, and actor-critic learning.

2.1. Machine learning methods in Finance

Fundamental data from financial statements and other data from business news are combined with machine learning algorithms can obtain investment signals or make predictions about the prospects of a company(Yang et al., 2018; Fang et al., 2019b; Zhang and Skiena, 2010; Chen and Liu, 2020) 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.

Besides using some machine learning and deep learning methods to predict stock prices, there are some classical models like Geometric Brownian Motion(GBM). It is demonstrated that GBM model has a high accuracy in predicting stock prices and is proven with forecast Mean Absolute Percentage Error(MAPE) value $\leq 20\%$ in the Indonesian market(Agustinu et al., 2018). Also, it is a sufficiently simple and highly interpretable model that can be used to predict the stock price as long as we have the historical prices of the stock. However, it also has some limitations: 1. Returns of stocks should follow a log-normal distribution, which may not always be the case. 2. GBM may not capture the complex patterns and trends that can be observed in the financial markets, so its accuracy may be limited when the time span of historical stock price series is too long or when many stocks need to be traded simultaneously. Deep learning, although sometimes lacking in interpretability, is able to capture nonlinear patterns in financial time series, which may be difficult or impossible to model using traditional statistical methods. to solve the problem of automatically trading a portfolio containing dozens of stocks. 3. In contrast to the mathematical school of finance, which assumes that the price of an asset moves according to an i.i.d. random process with no underlying patterns, we believe that historical stock data contains actionable patterns that can be detected and exploited by traders. This belief is based on the observation that stock prices are influenced by a variety of factors, including fundamental information about the company, market sentiment, and investor behavior. These factors can create patterns in the price movements that are not captured by the simple i.i.d. model. Therefore, we introduce the LSTM method as a powerful tool to extract the underlying patterns from financial time series data.

Although LSTM(Fischer and Krauss, 2018) are traditionally used in natural language processing, many recent works have applied them in financial markets to filter some noise in raw market data(Bao et al., 2017; Di Persio and Honchar, 2016; Fischer and Krauss, 2018; Tsantekidis et al., 2017). 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. Bao et al. (2017) and Jia et al. (2019) 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. The work of Lim et al. (2019) shows that LSTM delivers superior performance on modelling daily financial data. These results show that LSTM can extract time-series features.

Machine learning methods have been applied to various innovative approaches in stock prediction. For example, Wu et al. (2021b) developed a system that can classify stocks into momentum- or contrarian-type strategies based on fuzzy analysis methods, and

achieved 1.5 times higher profitability on the Taiwan 50 dataset. Syu et al. (2022) proposed the TripleS, a stock selection system that uses fuzzy-set theory to link stocks and investment strategies. In addition to using LSTM to extract time-series features of stock price series, some researchers also represented the stock and its leading indicators (futures/options) price series as graph data, and applied CNN to extract features. Syu et al. (2022) presented a two-dimensional tensor input data and feature extraction method for training a CNN network to predict the stock market, which outperformed previous algorithms in reducing noise and overfitting. Wu et al. (2021a) designed a new framework based on CNN and LSTM to predict the direction of the stock market by aggregating multiple variables, automatically extracting features through CNN, and feeding them into LSTM. Hist(Xu et al., 2021) is a high-frequency trading simulator developed by Microsoft Research Asia in 2022, which provides a realistic environment for developing and testing high-frequency trading algorithms. Qlib(Yang et al., 2020b) is an open-source Python library developed by Microsoft Research Asia in 2020, which supports various deep learning, reinforcement learning, and traditional machine learning models. Researchers often decompose the time series into different spectra to extract features, from which empirical modal decomposition (EMD) and fully integrated empirical modal decomposition (CEEMD) algorithms are derived. Rezaei et al. (2021)constructed CEEMD-CNN-LSTM and EMD-CNN-LSTM hybrid algorithms based on this, then combined with LSTM models, and experiments showed that the hybrid models outperformed their individual counterparts. Biswas et al. (2021) used algorithms such as LSTM, XGBoost, linear regression, moving average, and end-value models to develop prediction models for more than 12 months of historical stock data, and observed that the LSTM method outperformed all other methods. Althelaya et al. (2021) combined deep learning techniques with multi-resolution analysis to predict stock prices, based on empirical wavelet transform, and demonstrated that the proposed model was much more effective than other models on the S&P 500 index and McKee-Glass time series. Jing et al. (2021) first used convolutional neural networks to classify the sentiment of stock investors and then analyzed the technical indicators of stocks using LSTM algorithm, which was experimentally validated on six major sectors of the Shanghai Stock Exchange, and the results showed that the hybrid algorithm outperformed the single model as well as the model without sentiment analysis.

2.2. Critic-only method

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)(Mnih et al., 2015) and its extensions. Van Hasselt et al. (2016), Chen and Gao (2019), Dang (2019), and Jeong and Kim (2019) used this method to train agents on a single stock or asset. Chen and Gao (2019), and Huang (2018) used Deep Recurrent Q Network (DRQN) and DQN to train agents and achieved higher cumulative performance on quantitative trading than baseline models.

2.3. Actor-only method

The actor-only method is a type of reinforcement learning that can learn policies directly from a continuous action space. This method has the advantage of learning the optimal mapping from a given state to an action, regardless of whether the action is discrete or continuous. However, this method also faces some challenges, such as requiring a large amount of data and a long time to achieve a high-performance strategy (Zhang et al., 2020). Deng et al. (2016) pioneered the application of this method to real-time financial trading using Recurrent Deep Neural Networks. Jia et al. (2019) further explored this method in quantitative trading, where he compared LSTM networks with fully connected networks and examined the effects of different combinations of technical indicators on the daily data performance of the Chinese market. The authors found that LSTM networks outperformed fully connected networks, but the results of his experiments were not consistent. He reported that his proposed method could generate reasonable profits for some stocks, but failed for others.

2.4. Actor-critic method

Actor-critic methods are a class of RL algorithms that aim to train two models simultaneously: an actor that learns how to choose an action in a given state, and a critic that evaluates the quality of the action. One of the most successful actor-critic methods is PPO, which is considered to be the state-of-the-art in RL. PPO achieves superior performance by addressing the well-known challenges of applying RL to complex environments, such as instability caused by the changing

distribution of observations and rewards as the agent learns (Pricope, 2021). In this paper, we use the baseline model proposed by Yang et al. (2020a), which is based on the actor-critic method and assembles three deep RL algorithms: PPO, Advantage Actor Critic (A2C), and Deep Deterministic Policy Gradient (DDPG). However, we find that the agent that learns only with PPO outperforms an ensemble strategy in terms of cumulative return.

3. Our method

3.1. Stock Market Environment

This paper employs a simulation environment for the stock market, which is developed by Yang et al. (2020a) based on the OpenAI gym framework (Brockman et al., 2016; Dhariwal et al., 2017; Raffin et al., 2019). This environment provides various information for training the agent, such as current stock prices, shareholdings and technical indicators. We model stock trading as a Markov Decision Process (Ilmanen, 2011), which consists of the following components: state, action, reward, policy and Q-value. We select 30 stocks randomly from a stock index in a financial market as the objects of transaction. The agent only focuses on the information of these 30 stocks and performs actions such as buying, selling and holding.

3.1.1. State Space

A 181-dimensional vector consisting of seven parts in Yang et al. (2020a) represents the state space of the multi-stock trading environment for these 30 stocks: $[b_t, \boldsymbol{p}_t, \boldsymbol{h}_t, \boldsymbol{M}_t, \boldsymbol{R}_t, \boldsymbol{C}_t, \boldsymbol{X}_t]$. 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. $\boldsymbol{h}_t \in \mathbb{Z}_+^{30}$: number of 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(Chong et al., 2014). It measures the difference between two moving averages and uses this information to identify potential trends and changes in momentum in a security or asset.
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(Chong et al., 2014). It measures the strength of a security or asset's price action by comparing its average gains to its average losses over a specified period of time.
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(Maitah et al., 2016).
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(Gurrib, 2018).

In Yang et al. (2020a), this 181-dimensional vector is fed as a state directly into the reinforcement learning algorithm for learning. However, our approach gives T(T is the time window of LSTM) 181-dimensional vectors to the LSTM for learning first, and the feature vector generated by the LSTM are given to the agent for learning as our state.

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 the following conditions:

1. h_{max} represents the maximum number of shares we can be able to buy at a time.
2. The action space is a high-dimensional and large discrete action space since the entire action space is of size $(2k + 1)^{30}$, and it can be approximately considered as a continuous action space in the practice.
3. The action space will then 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 Yang et al., 2020a:

$$c_t = 0.1\% \ \cdot \ \left|\boldsymbol{p}^T \boldsymbol{k}_t\right| \tag{2}$$

3.1.4. Turbulence Threshold

We employ this financial index $turbulence_t$ (Yang et al., 2020a) that measures extreme asset price movements to avoid the risk of sudden events that may cause stock market crash (Kritzman and Li, 2010), such as March 2020 stock market caused by COVID-19, wars and financial crisis:

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

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 $\boldsymbol{\Sigma} \in \mathbb{R}^{30\times30}$ 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 use LSTM as a feature extractor to improve the model in Yang et al. (2020a), as shown in Figure 2. we propose a DRL based stock trading system using cascaded Long Short-Term Memory (**CLSTM-PPO Model**), which first uses LSTM to extract the time-series features from daily stock 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.

At the time step t, the environment gets the current state S_t and passes it to the LSTM network. It 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

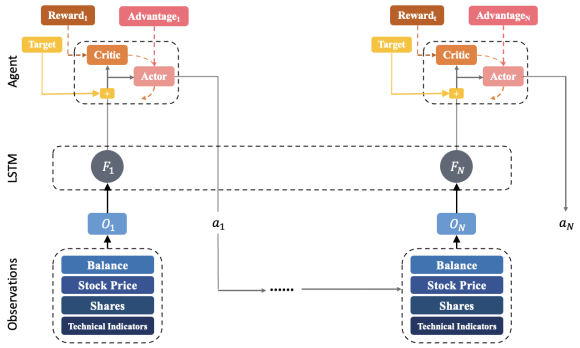

Figure 2: Overview of our CLSTM-PPO model

policy function $\pi(F'_t)$ to perform the 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. Some hyperparameters of the environment, such as the number of time steps, are usually used to control the end of the state. However, these hyperparameters need to be set in advance and cannot be adapted to the different needs of different environments. Therefore, a boolean d_t is introduced to determine whether to end the current state or not. Specifically, if the value of d_t is $True$, it means the current state is not finished and needs to continue; if the value of d_t is $False$, it means the current state is finished and needs to transition to the next state. Therefore, d_t plays a role to control the state transition. The value of d_t is used to determine whether the current state is finished and thus the next state is entered. In this way, the algorithm can adaptively control the end of the state and adapt more flexibly to the needs of different environments.

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 computed by critic using the advantage function. After a certain number of steps, actor back-propagates the error through the clipped surrogate objective function of PPO, then critic updates the parameters using the mean square error loss function. 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 action space, clear stopping conditions and a more stable environment. 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). 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(Jia et al., 2019; Lample and Chaplot, 2017).

In this paper, an LSTM-based feature extractor is developed using the customized feature extraction interface provided by stable-baselines3[1], which is a set of reliable implementations of reinforcement learning algorithms in PyTorch. The network structure of the LSTM feature extractor is shown in Figure 3.

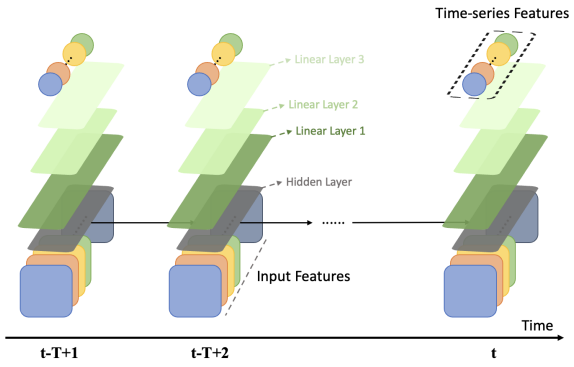

Figure 3: The LSTM feature extractor, which has as input a state list of length T arranged in time order and as output a 128-dimensional feature vector after T time steps

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, then be sent to the PPO for the agent to learn as input.

Algorithm 1 is the pseudo-code of our extractor. In practice, the defined LSTM can be connected to the reinforcement learning algorithm using the policy_kwargs interface provided by stable-baselines3, which enables the agent to directly receive the temporal features extracted by the LSTM.

3.2.3. Proximal Policy Optimization (PPO)

The PPO algorithm (Schulman et al., 2017) is a state-of-the-art policy-based reinforcement learning method

[1]Github repository: https://github.com/DLR-RM/stable-baselines3

Algorithm 1: one-day LSTM feature extractor

Input: hidden state of shape $h_0 = (num\_layers * num\_directions, N, hidden\_size)$, cell state of shape $c_0 = (num\_layers * num\_directions, N, hidden\_size)$

Output: N-day time-series feature

Get last N-day states list;
Initialize LSTM hidden and cell states: $h = h_0$, $c = c_0$;
for *n in range(N)* **do**
    Pass nth state into LSTM;
    Store the output and update the LSTM with (h, c) in output;
Extract features from the last LSTM layer;
Return features

that uses multiple epochs of stochastic gradient ascent to update the policy parameters at each iteration (Kingma and Ba, 2014). The PPO algorithm has shown superior performance in stock trading compared to other deep reinforcement learning algorithms, such as DQN and A2C, in the study by Yang et al. (2020a). This is one of the main reasons why we chose to apply the PPO algorithm to our problem. The PPO algorithm consists of two neural networks: an actor network and a critic network. The actor network takes the current state of the environment as input and outputs an action to be executed. The critic network takes the current state of the environment as input and outputs an estimate of the value of that state. The value function represents the expected cumulative reward that can be obtained by following the current policy from that state.

The actor and critic networks cooperate to improve the policy by providing feedback to each other. The critic network provides feedback to the actor network by evaluating how good the current policy is based on the value function. The actor network uses this feedback to adjust the policy parameters in a way that increases the expected reward.

Conversely, the actor network provides feedback to the critic network by sampling actions from the current policy. The critic network uses this feedback to update its value function estimates.

One of the challenges of reinforcement learning is to ensure stability and efficiency of learning, especially in high-dimensional and continuous action spaces. The PPO algorithm tackles this challenge by introducing a clipped surrogate objective function that limits the magnitude of the policy updates at each iteration. Let θ denote the policy parameters of the actor network. We define the probability ratio between the new policy and

the old policy as:

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

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

$$\begin{aligned} H^{CLIP}(\theta) = \hat{\mathbb{E}}[&\min(r_t(\theta)\hat{A}(s_t,\ a_t),\\ &clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}(s_t,\ a_t))] \end{aligned} \tag{5}$$

Where $r_t(\theta)\hat{A}(s_t,\ a_t)$ is the normal policy gradient objective, and $\hat{A}(s_t,\ a_t)$ is the estimated advantage function. The term 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, which helps to prevent the policy from changing too much at once and reduces the risk of the policy diverging. By limiting the amount of change that can occur, PPO is able to achieve more stable and reliable results.

We use a policy network based on LSTM (Long Short-Term Memory) to model the agent's behavior in the PPO (Proximal Policy Optimization) algorithm. The policy network takes the current state of the environment as input and outputs an action to execute. The LSTM policy has the advantage of being able to store and recall past observations, which can help the agent make better decisions in complex and dynamic environments.

To train the LSTM policy, we use the PPO algorithm, which computes a surrogate objective function that quantifies the difference between the old and new policies. The objective function is then optimized to update the policy parameters, subject to a clipping constraint that prevents excessive changes.

The LSTM network has an input dimension of 181 and an output dimension of 128. The hidden state size is also 128. The network consists of three linear layers, each followed by a Tanh activation function. The first linear layer has a weight matrix of size $(15 \times 128,\ 128)$, while the second and third linear layers have weight matrices of size $(128,\ 128)$.

Algorithm 2 shows the pseudo-code for training the agent using PPO with a cascaded LSTM network.

Algorithm 2: PPO with LSTM

Input: Initial state s_t ; Adam optimizer with learning rate α; Adam optimizer's first moment exponential decay rate β_1; Adam optimizer's second moment exponential decay rate β_2; Discount factor γ; Clipping range ϵ; Advantage estimate A_t;
Output: Trained actor network $\pi_\theta(a_t|s_t)$ and value network $V_\phi(s_t)$;

Initialize critic $V_\phi(s)$ and actor $\pi_\theta(a|s)$ networks with parameters ϕ and θ;
Initialize the replay buffer D;
for *each episode* **do**
  Initialize the environment with initial state s_0;
  for *each step t in the episode* **do**
    Receive state s_t from environment;
    Process s_t with LSTM to obtain a feature vector f_t;
    Compute the critic's value estimate $\hat{v}_t = V_\phi(f_t)$;
    Sample an action a_t from the policy $\pi_\theta(a_t|f_t)$;
    Execute a_t in the environment to receive the reward r_t and the next state s_{t+1};
    Compute the advantage estimate $A_t = r_t + \gamma\hat{v}_{t+1} - \hat{v}_t$;
    Add the transition (f_t, a_t, A_t) to the replay buffer D;
    if $t \mod T = 0$ **then**
      Update the critic by minimizing the MSE between the target $r_t + \gamma\hat{v}_{t+1}$ and the current estimate $\hat{v}_t$: $\phi \leftarrow \phi - \alpha_V \nabla_\phi (r_t + \gamma\hat{v}_{t+1} - \hat{v}_t)^2$;
      Update the actor using the PPO objective function: $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta L_{\text{PPO}}(\theta)$;
      Clear the replay buffer D;
    end
  end
end

4. Experimental Design

In this section, we first introduce the datasets used for the experiments, as well as show the training parameters of PPO, and then introduce the baseline models one by one. Finally, we introduce the evaluation measures used to assess the experimental results.

4.1. Description of Datasets

This paper selects 120 stocks as the stock pool, which consists of four groups of 30 stocks each: the Dow Jones Industrial Average (DJIA) constituent stocks, 30 stocks randomly selected from the SSE 50 Index of the Shanghai Stock Exchange, 30 stocks from the S&P BSE SENSEX Index of the Bombay Stock Exchange, and 30 stocks randomly selected from the FTSE 100 Index

of the London Stock Exchange. The DJIA stocks are identical to those used by Yang et al. (2020a) to facilitate comparison with their ensemble strategy, while the other 90 stocks are chosen to examine the applicability of the proposed model in a mature market as well as two emerging markets.

The daily data for backtesting spans from 01/01/2009 to 05/08/2020, and is divided into two periods: in-sample and out-of-sample. The agent is trained and tested using only the PPO algorithm, following the same setting as Yang et al. (2020a), to enable a fair comparison between models.

The dataset is split into two parts, with the in-sample data ranging from 01/01/2009 to 12/31/2015 and the out-of-sample data ranging from 01/01/2016 to 05/08/2020, as shown in Figure 4. To continuously improve the agent's performance, an expanding approach is adopted to dynamically add new out-of-sample data to the existing in-sample data.

As the in-sample data grows over time, it becomes more diverse and may contain data from different market states. This has important implications for how historical data is incorporated into the prediction, as the agent must balance the relevance of older data with the importance of more recent data.

If historical data is not weighted, the model can better utilize the information in the historical data because every data point is equally weighted. However, this approach may make the model overlook some new market dynamics. Weighting recent data more heavily has the advantage of better reflecting recent market dynamics, but it may also omit some important historical information.

Here, we do not weight the data, mainly because we want to discuss in our experiments how much the cascaded LSTM network will help the strategy, so we need to keep other factors as consistent as possible in this process, such as how the data is weighted.

It should be noted that in India and UK, it is not possible to ensure that all stocks are traded from 01/01/2009 because of the changes in the constituent stocks of the index, so after data pre-processing, there are a total of 7 years of data for the Indian market and close to 5 years of data for the UK market. The Indian market is trained from 02/26/2016 until 03/02/2022 and traded from 03/03/2022 until 03/03/2023. The UK market is trained from 09/19/2018 until 03/02/2022 and traded from 03/03/2022 until 03/03/2023.

4.2. Training Parameters of PPO

The training parameters of PPO are set as shown in the Table 1.

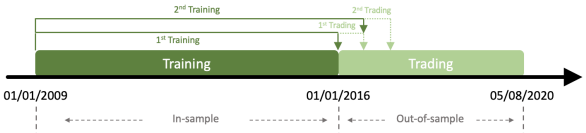

Figure 4: Illustration of stock data splitting: We used data from January 1, 2009 to January 1, 2016 in the first round of training, and used the next three months as trading tests; in the second round of training, the data from January 1, 2009 to March 1st, 2016 was used for training, and the following three months were also used as trading tests. It is worth noting that the data doesn't include the results from actively trading (e.g., data generated by the learned policy), but only contains the raw stock data.

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
Random Seed	9

To avoid the interference of the random noise present in the environment on the training PPO and the subsequent parameter tuning, we fixed the random seeds so that the random perturbations generated in the environment remain consistent for each run of the model.

4.3. Compared Models

Our model is compared with baseline models including:

- **Buy-And-Hold Strategy**: the typical Buy-And-Hold strategy for index in four markets including DJI, SSE50 Index, SENSEX Index, FTSE 100 Index, which means that the trader buys at the start of the trading period and holds to the end.

- **PPO model**: only use PPO with MLP Policy to train the agent.

- **Recurrent PPO model**: use PPO with LSTM Policy to train the agent.

- **MLP model**: a model offered by Qlib in 2020, which can predict the stock price using Multilayer Perceptron (MLP).
- **LSTM model**: a model offered by Qlib in 2020, which can use Long Short-Term Memory networks to predict the stock price.
- **Light GBM model**: Light Gradient Boosting Machine offered by Qlib in 2020 is a highly efficient gradient boosting framework that uses a novel decision tree algorithm based on histogram-based computation. It is designed to handle large-scale datasets and provides faster training speed and higher accuracy than other gradient boosting models.
- **Ensemble Strategy in Yang et al. (2020a)**: 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.
- **HIST model**(Xu et al., 2021): Histogram-based Gradient Boosting offered by Qlib in 2022 is another gradient boosting framework that is based on histogram-based algorithm. It is designed for high-dimensional and sparse data, and is able to handle datasets with a large number of features. HIST provides better performance than traditional gradient boosting models when dealing with high-dimensional data.

4.4. Evaluation Measures

- **Cumulative 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.

$$CR = \frac{P_{end} - P_0}{P_0} \tag{6}$$

- **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.

$$MER = \frac{max(A_t - A_0)}{A_0} \tag{7}$$

Where A_t is the total asset of the strategy at time t and A_0 is the initial cash.

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

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

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.

$$APPT = \frac{P_{end} - P_0}{NT} \tag{9}$$

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

- **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.

$$SR = \frac{E(R_P) - R_f}{\sigma_P} \tag{10}$$

5. Numerical Evaluation

In this section, we first tuned some parameters in the model, then used 30 Dow constituent stocks to evaluate our model, and performed robustness tests on 30 stocks in the SSE 50 in China, 30 stocks in SENSEX in India, and 30 stocks in FTSE 100 in UK. The data in the US market comes from Yang et al. (2020a), and the other come from Wind, which is a financial data and information provider based in China. The list of stock name has attached in the Appendix A.

5.1. Exploration of Well-performing Hyperparameters

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

5.1.1. Best Time Window of LSTM

For the time window of the LSTM, we tested the cases of TimeWindow (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 5, it can be seen that the agent at TW=30 is able to achieve

Figure 5: The trading results from U.S. market to find the time window that shows the best performance. The agent performs best when the time window of LSTM is 30

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%	**82.58%**	68.74%
MER	68.27%	63.46%	**92.32%**	79.32%
MPB	58.93%	**24.75%**	29.39%	37.01%
APPT	18.29	21.77	**33.57**	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 details: for TimeWindow=30, CR, MER, APPT and SR are much higher than the other options, but MPB does not perform well. Overall, TW = 30 is the best parameter for our experiment.

5.1.2. Best Hidden Size of LSTM in PPO

For the hidden size of the LSTM in PPO, we tested the cases of HiddenSize (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%	49.77%	**82.58%**	56.27%	79.04%
MER	84.29%	63.45%	**92.32%**	60.64%	92.04%
MPB	38.57%	29.92%	**29.39%**	30.39%	58.31%
APPT	28.07	30.79	**33.57**	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 details: for HiddenSize=512, CR, MER, APPT, MPB and SR are much higher than the other options. Therefore, HS = 512 is the best parameter for our strategy.

5.2. Performance in the US Market

The well-performing parameter set (TW=30, HS=512) is used as the parameters of the final model and the results of this model are compared with the trading results of the PPO model with LSTM in PPO and another with MlpPolicy and the Ensemble Strategy in Yang et al. (2020a), as shown in Figure 7.

Table 4 shows the details of trading results in the US markets. Our CLSTM-PPO model is the best in terms of profitability, but it doesn't perform well compared to the ensemble strategy in terms of risk control.

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

	CLSTM-PPO	PPO	RecurrentPPO	Ensemble	DJI
CR	**82.58%**	54.37%	49.77%	70.40%	50.97%
MER	**92.32%**	67.28%	63.45%	65.32%	63.90%
MPB	29.39%	28.30%	29.39%	**15.74%**	72.32%
APPT	**33.57**	20.02	22.84	28.54	N.A.
SR	1.1540	0.8081	0.6819	**1.3000**	0.4149

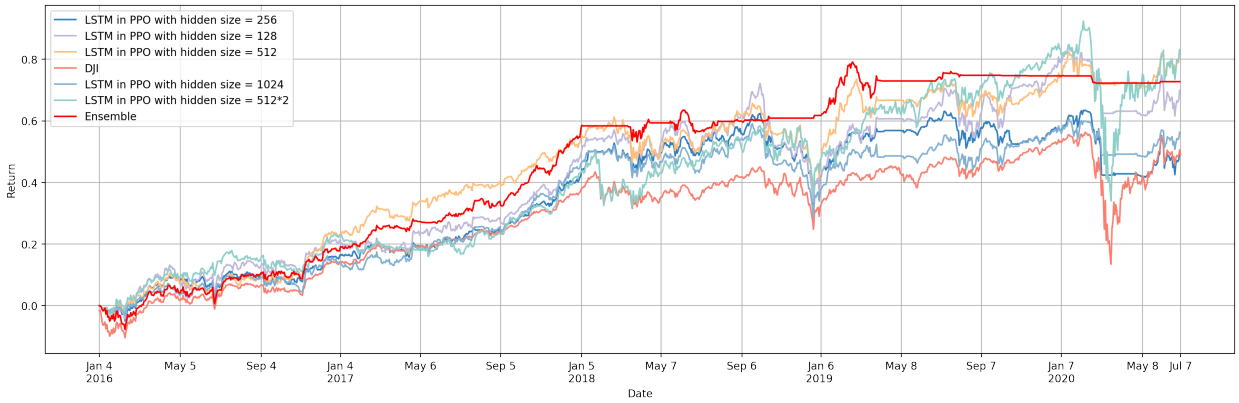

Figure 6: The trading results from U.S. market to find the hidden size of LSTM in PPO that shows the best performance. The agent performs best when the hidden size of LSTM in PPO (Recurrent PPO) is 512

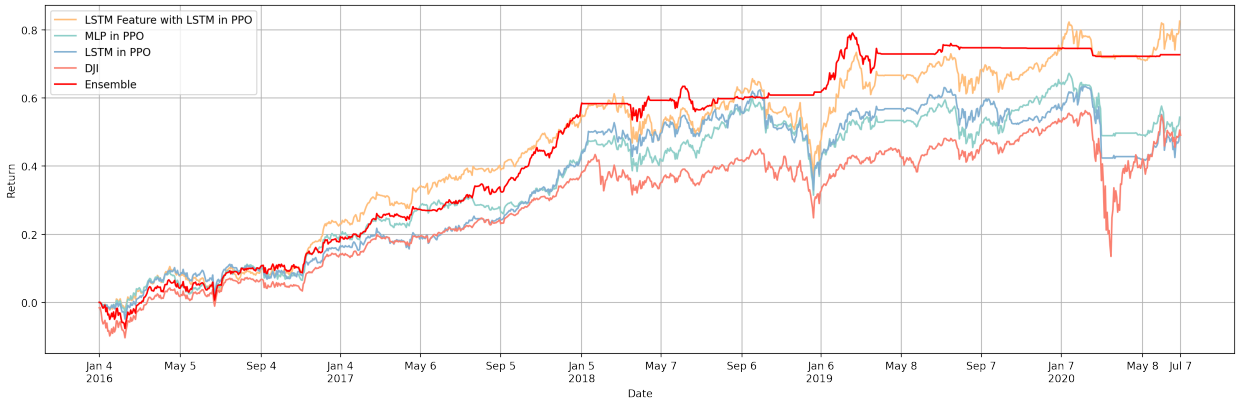

Figure 7: Our CLSTM-PPO (Recurrent PPO with LSTM feature extraction) performs better compared to Recurrent PPO, Ordinary PPO, Ensemble Strategy in Yang et al. (2020a), and Buy-And-Hold strategy on DJI in the US market

Our model (CLSTM-PPO) outperforms the ensemble strategy in Yang et al. (2020a) and other compared models on 30 Dow components, achieving the highest cumulative return of 82.58% and the maximum profitability of 92.32%. However, our model also has a higher maximum pullback (MPB) of 29.39%, compared to the ensemble strategy's MPB of 15.74%, indicating a lower risk tolerance and a weaker ability to identify down markets and stop trading. The ensemble strategy performs slightly better in this regard, but it also tends to avoid trading in both down and up markets, which can result in a significant loss of profit potential in the long run. Overall, our model demonstrates 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 assistance of other algorithms, which makes it much simpler and faster.

We further analyze the performance of our model in two phases that correspond to the DJI index: (i) Accumulation phase: until 06/2017, our model achieved stable growth with little difference in returns from the ensemble strategy. However, after that, our model quickly captured profits and increased its total returns rapidly. This phase lasted until 01/2018, when the cumulative return reached a level that was close to the final return. (ii) Volatility phase: Starting from 01/2018, our model adopted a more aggressive and courageous trading style, 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 after suffering a pullback on 01/2019.

5.3. Performance in Chinese Markets

The same model is used to trade the 30 stocks in the Chinese stock market, and the ensemble strategy is cho-

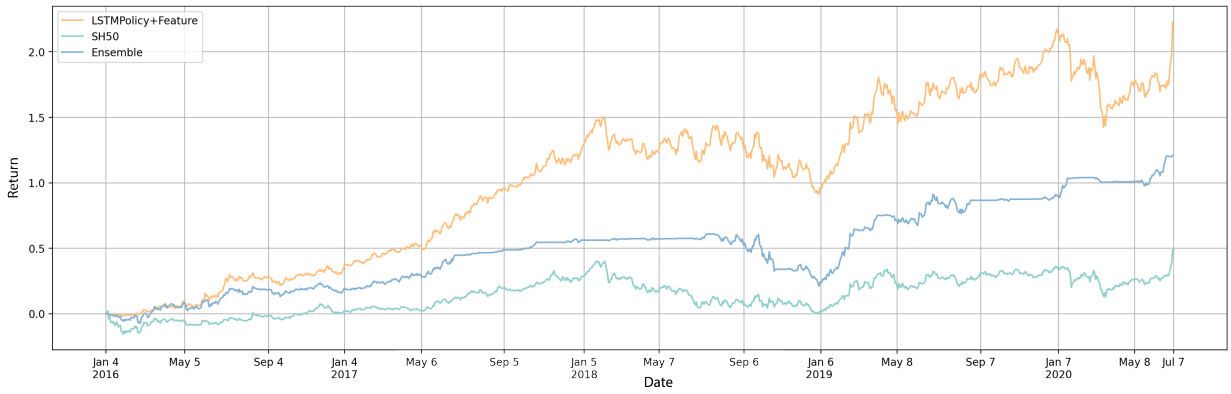

Figure 8: Our CLSTM-PPO (Recurrent PPO with LSTM feature extraction) performs better compared to Ensemble Strategy in Yang et al. (2020a) and Buy-And-Hold strategy on SSE50 in the Chinese market

sen as the most important compared model. We show the cumulative returns at the end of the trade as shown in Figure 8.

Table 5: Details of the trading results in Chinese Markets

	CLSTM-PPO	PPO	RecurrentPPO	Ensemble	SSE50
CR	**222.91%**	93.23%	102.48%	120.87%	51.46%
MER	**222.91%**	93.23%	102.48%	120.87%	51.46%
MPB	74.81%	32.48%	33.92%	**39.95%**	41.27%
APPT	**66.96**	39.44	42.38	47.55	25.78
SR	**2.3273**	1.5489	1.5977	1.6938	0.4149

Table 5 presents the trading performance of our model (CLSTM-PPO) and the ensemble strategy in the Chinese market. Compared to the ensemble strategy, our model demonstrates a significant advantage in this market. Our cumulative return (CR) is 222.91%, which is almost twice as high as the ensemble strategy's CR of 120.87%. Although our model also suffers from larger drawdowns, the volatility is mainly manifested in the increase of cumulative returns. Moreover, the Sharpe ratio (SR), which measures the risk-adjusted return, indicates that our model (SR=2.3273) outperforms the ensemble strategy (SR=1.6029) in the Chinese market. This result illustrates the superior performance of our model in emerging markets, which are typically characterized by higher volatility, and is consistent with our analysis of our model's features: the ability to capture returns quickly and accurately in volatile situations, and the positive correlation between returns and positive volatility.

Further analysis reveals that our model's cumulative return grew rapidly from 01/2016 to 02/2018, reaching almost three times that of the ensemble strategy. Afterwards, as the volatility of the SSE 50 index increased, our model's volatility also increased accordingly. From 02/2018 to 01/2019, the drawdowns of the two models were similar, but then our model captured nearly 80% of the return at 01/2019 within three months. Although it experienced a decline in 01/2020 due to the black swan event of Covid-19, it quickly recovered to its highest point six months later.

5.4. Performance in Indian and the UK markets

In this section, we conduct a robustness analysis of our proposed CLSTM-PPO model by testing it on two additional markets: the UK market and the Indian market. We also benchmark our model against several state-of-the-art trading models from Qlib (Yang et al., 2020b), a machine learning platform for quantitative investment. These models include MLP, LSTM, lightGBM and HIST. Table 6 shows the trading performance of our model and the baseline models in the four markets.

To use Qlib for prediction, we preprocess the input data to conform to the format required by the platform. Qlib then automatically computes alpha158, a library of 158 factors within Qlib, based on the data. These factors, along with the open price, close price, high price, low price and volume of the stock, are used as input features for the prediction models. For backtesting, we use the same train-test split as in Section 4. For trading, Qlib predicts the price movements of all stocks in a portfolio for the next day, sorts them in descending order by their expected returns, and buys the top ten stocks until the last trading day. Table 6 summarizes the trading performance of our model and the baseline models in the four markets.

The cascaded LSTM can effectively capture the temporal features of the market and enhance the agent's profitability, as evidenced by the superior performance

Table 6: Comparison of experimental results with baselines

Datasets	Metrics	CLSTM-PPO	PPO	RecurrentPPO	Index	MLP	LSTM	lightGBM	HIST	Ensemble
USA	**CR**	**82.58%**	54.37%	49.77%	50.97%	51.27%	45.55%	36.68%	87.27%	70.40%
	MER	92.32%	67.28%	63.45%	63.90%	65.77%	61.18%	47.67%	**94.33%**	65.32%
	MPB	29.39%	28.30%	29.39%	72.32%	29.19%	25.97%	18.33%	20.72%	**15.74%**
	APPT	**33.57**	20.02	22.84	N.A.	16.02	23.06	17.48	33.22	28.54
	SR	1.154	0.8081	0.6819	0.4149	0.4368	0.8471	0.7787	**1.4884**	1.3116
China	**CR**	**222.91%**	93.23%	102.48%	51.46%	54.32%	79.93%	39.62%	147.57%	120.87%
	MER	**222.91%**	93.23%	102.48%	51.46%	54.32%	79.93%	39.62%	147.57%	120.87%
	MPB	74.81%	32.48%	33.92%	41.27%	25.56%	23.12%	**15.83%**	29.86%	39.95%
	APPT	**66.96**	39.44	42.38	N.A.	28.41	32.43	21.07	58.58	47.55
	SR	**2.3273**	1.5489	1.5977	0.6482	0.6922	1.0866	0.4658	2.1283	1.6938
Indian	**CR**	**16.74%**	7.30%	8.33%	8.65%	6.91%	9.85%	9.44%	14.35%	13.81%
	MER	**20.03%**	10.18%	12.74%	13.75%	7.28%	10.77%	11.28%	18.24%	18.96%
	MPB	9.72%	10.03%	11.30%	**6.79%**	12.54%	13.28%	13.16%	10.12%	9.98%
	APPT	**27.96**	16.99	18.86	N.A.	14.31	19.03	17.52	23.31	25.53
	SR	**1.0839**	0.4853	0.5506	0.5709	0.4606	0.6470	0.6210	0.9323	0.8981
UK	**CR**	16.84%	8.83%	9.02%	9.74%	10.32%	14.02%	**18.60%**	18.15%	14.26%
	MER	**22.59%**	8.83%	9.02%	14.68%	11.96%	15.51%	20.35%	20.77%	17.68%
	MPB	27.54%	18.59%	17.18%	**2.30%**	30.65%	39.78%	38.75%	39.22%	18.24%
	APPT	**36.03**	19.87	21.51	N.A.	21.38	26.12	35.77	33.61	32.41
	SR	1.0318	0.5572	0.5685	0.6111	0.6455	0.8647	**1.1360**	1.1094	0.8789

of our CLSTM-PPO model over all the benchmark models in terms of MER and APPT metrics across the four markets. However, our model does not achieve the highest cumulative return in the UK market, ranking third among the models. A possible reason for this is the insufficiency of the dataset for training a deep reinforcement learning agent, which requires more than seven years of data to learn the diverse characteristics of the market for daily trading, as shown by our experiments. The dataset for the UK market has a shorter trading period than the other markets, which may limit the agent's performance.

Our model performs moderately in terms of maximum drawdown, without a clear advantage over the benchmark models. The ensemble strategy has a relatively stable performance in controlling the drawdown, which suggests that using multiple agents trained simultaneously can reduce the risk of making large losses, as reported by Yang et al. (2020a). However, this also implies a trade-off between risk and return in the investment world, as higher returns usually entail higher risks.

6. Conclusion

In this paper, we propose a novel reinforcement learning model that leverages cascaded LSTM networks to capture the temporal dependencies of financial data. We apply our model to four different stock markets: US, China, India, and UK, and compare its performance with several benchmark models. The experimental results show that our model can generate higher profits than the benchmarks, especially in the Chinese market. However, our model also involves higher risks of drawdowns. We analyze the reasons behind this phenomenon and suggest that our model is more suitable for markets with smoother and less volatile trends, where the returns can be more steadily accumulated. For example, the A-share market in China has exhibited such characteristics in recent years. This implies that there is a latent return pattern in the stock market, and that LSTM as a time-series feature extractor plays a crucial role. Furthermore, it indicates that the Chinese market is a favorable environment for developing quantitative trading strategies. Our contributions to the literature are threefold: (1) We develop a novel CLSTM-PPO model for quantitative investment, which adopts a cascaded structure with two stages of carefully designed deep LSTM networks to extract features and train the agent; (2) We extend the work of Yang et al. (2020a) by using the features generated by LSTM network instead of the raw stock times series data, and by using LSTM to train the agent instead of the simple MLP network; (3) We evaluate our model on more datasets from different countries than Yang et al. (2020a), and test its robustness across different markets. Our research advances the field of quantitative investment using RL, and highlights the importance of considering the unique properties of financial data in developing effective models.

For future research, we propose the following potential enhancements: (i) The size of training data. Since PPO requires a large amount of historical data to achieve satisfactory learning outcomes, increasing the

size of training data may help to improve the performance. (ii) Reward function. There are some alternative reward functions for stock trading that can enhance the stability of the algorithm. For example, we can adopt the Sharpe ratio approach, which balances risk and return and controls drawdown. The Sharpe ratio is calculated by dividing the mean of the return series by the standard deviation of the return series. Furthermore, if the reward signal has a very large or very small range, it can cause numerical instabilities during training. In this case, we can normalize the reward function to a smaller range (e.g., between -1 and 1) to make it more stable.

Appendix A. Stock List

This is the list of stock name we use in the experiments including four different stock markets in the world, including US, UK, India and China.

US	China	UK	India
AXP	600036.XSHG	AAL.L	500010.BO
AAPL	600048.XSHG	ABF.L	500034.BO
VZ	600104.XSHG	ADM.L	500112.BO
BA	600111.XSHG	AHT.L	500114.BO
CAT	600196.XSHG	ANTO.L	500124.BO
JPM	600276.XSHG	AUTO.L	500180.BO
CVX	600309.XSHG	AV_.L	500209.BO
KO	600346.XSHG	AZN.L	500247.BO
DIS	600436.XSHG	BARC.L	500325.BO
DD	600438.XSHG	BATS.L	500470.BO
XOM	600519.XSHG	BA_.L	500510.BO
HD	600570.XSHG	BDEV.L	500520.BO
INTC	600585.XSHG	BEZ.L	500696.BO
IBM	600588.XSHG	BKG.L	500790.BO
JNJ	600690.XSHG	BLND.L	500820.BO
MCD	600745.XSHG	BNZL.L	500875.BO
MRK	600809.XSHG	BP_.L	507685.BO
MMM	600837.XSHG	BRBY.L	524715.BO
NKE	600887.XSHG	BT_A.L	532174.BO
PFE	600893.XSHG	CCH.L	532187.BO
PG	600900.XSHG	CNA.L	532215.BO
UNH	601088.XSHG	CPG.L	532281.BO
RTX	601166.XSHG	CRDA.L	532454.BO
WMT	601318.XSHG	CRH.L	532500.BO
WBA	601398.XSHG	CTEC.L	532538.BO
MSFT	601601.XSHG	DCC.L	532540.BO
CSCO	601628.XSHG	DGE.L	532555.BO
TRV	601857.XSHG	III.L	532755.BO
GS	601899.XSHG	STAN.L	532898.BO
V	601919.XSHG	ULVR.L	532978.BO

References

Agustinu, W. F., Widiastuti, E., Nursetyo, A. A.and Tjahjono, A. E., and Hidayat, F. (2018). Stock price prediction using stochastic volatility model and markov switching autoregressive. *Journal of Physics: Conference Series*.

Althelaya, K. A., Mohammed, S. A., and El-Alfy, E.-S. M. (2021). Combining deep learning and multiresolution analysis for stock market forecasting. *IEEE Access*, 9:13099–13111.

Baba, N. and Kozaki, M. (1992). An intelligent forecasting system of stock price using neural networks. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 1, pages 371–377. IEEE.

Bao, W., Yue, J., and Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PloS one*, 12(7):e0180944.

Bekiros, S. D. (2010). Fuzzy adaptive decision-making for boundedly rational traders in speculative stock markets. *European Journal of Operational Research*, 202(1):285–293.

Biswas, M., Shome, A., Islam, M. A., Nova, A. J., and Ahmed, S. (2021). Predicting stock market price: A logical strategy using deep learning. In *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, pages 218–223. IEEE.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.

Carta, S., Ferreira, A., Podda, A. S., Recupero, D. R., and Sanna, A. (2021). Multi-dqn: An ensemble of deep q-learning agents for stock market forecasting. *Expert systems with applications*, 164:113820.

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

Chen, Q. and Liu, X.-Y. (2020). Quantifying esg alpha using scholar big data: an automated machine learning approach. In *Proceedings of the First ACM International conference on AI in finance*, pages 1–8.

Cheng, S. (1994). A neural network approach for forecasting and analyzing the price-volume relationship in the taiwan stock market. *Master's thesis, National Jow-Tung University, Taiwan, ROC*.

Chong, T. T.-L., Ng, W.-K., and Liew, V. K.-S. (2014). Revisiting the performance of macd and rsi oscillators. *Journal of risk and financial management*, 7(1):1–12.

Dang, Q.-V. (2019). Reinforcement learning in stock trading. In *International conference on computer science, applied mathematics and applications*, pages 311–322. Springer.

Deng, Y., Bao, F., Kong, Y., Ren, Z., and Dai, Q. (2016). Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. (2017). Openai baselines.

Di Persio, L. and Honchar, O. (2016). Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International journal of circuits, systems and signal processing*, 10:403–413.

Fang, Y., Chen, J., and Xue, Z. (2019a). Research on quantitative investment strategies based on deep learning. *Algorithms*, 12(2):35.

Fang, Y., Liu, X.-Y., and Yang, H. (2019b). Practical machine learning approach to capture the scholar data driven alpha in ai industry. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2230–2239. IEEE.

Fischer, T. and Krauss, C. (2018). Deep learning with long short-term memory networks for financial market predictions. *European journal of operational research*, 270(2):654–669.

Gurrib, I. (2018). Performance of the average directional index as a market timing tool for the most actively traded usd based currency pairs. *Banks and Bank Systems*, 13(3):58–70.

Huang, C. Y. (2018). Financial trading as a game: A deep reinforcement learning approach. *arXiv preprint arXiv:1807.02787*.

Ilmanen, A. (2011). *Expected returns: An investor's guide to harvesting market rewards*, volume 535. John Wiley & Sons.

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

Jia, W., Chen, W., Xiong, L., and Hongyong, S. (2019). Quantitative trading on stock market based on deep reinforcement learning. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Jing, N., Wu, Z., and Wang, H. (2021). A hybrid model integrating deep learning with investor sentiment analysis for stock price prediction. *Expert Systems with Applications*, 178:115019.

Kim, Y., Ahn, W., Oh, K. J., and Enke, D. (2017). An intelligent hybrid trading system for discovering trading rules for the futures market using rough sets and genetic algorithms. *Applied Soft Computing*, 55:127–140.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kritzman, M. and Li, Y. (2010). Skulls, financial turbulence, and risk management. *Financial Analysts Journal*, 66(5):30–41.

Kuo, R. J. (1998). A decision support system for the stock market through integration of fuzzy neural networks and fuzzy delphi. *Applied Artificial Intelligence*, 12(6):501–520.

Lample, G. and Chaplot, D. S. (2017). Playing fps games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31.

Lim, B., Zohren, S., and Roberts, S. (2019). Enhancing time-series momentum strategies using deep neural networks. *The Journal of Financial Data Science*.

Maitah, M., Prochazka, P., Cermak, M., and Šrédl, K. (2016). Commodity channel index: Evaluation of trading rule of agricultural commodities. *International Journal of Economics and Financial Issues*, 6(1):176–178.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., and Ostrovski, G. (2015). Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533.

Pricope, T.-V. (2021). Deep reinforcement learning in quantitative algorithmic trading: A review. *arXiv preprint arXiv:2106.00123*.

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N. (2019). Stable baselines3.

Rezaei, H., Faaljou, H., and Mansourfar, G. (2021). Stock price prediction using deep learning and frequency decomposition. *Expert Systems with Applications*, 169:114332.

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

Syu, J.-H., Lin, J. C.-W., Wu, C.-J., and Ho, J.-M. (2022). Stock selection system through suitability index and fuzzy-based quantitative characteristics. *IEEE Transactions on Fuzzy Systems*, 31(1):322–334.

Tsantekidis, A., Passalis, N., Tefas, A., Kanniainen, J., Gabbouj, M., and Iosifidis, A. (2017). Using deep learning to detect price change indications in financial markets. In *2017 25th European signal processing conference (EUSIPCO)*, pages 2511–2515. IEEE.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator.

Wu, J. M.-T., Li, Z., Herencsar, N., Vo, B., and Lin, J. C.-W. (2021a). A graph-based cnn-lstm stock price prediction algorithm with leading indicators. *Multimedia Systems*, pages 1–20.

Wu, M.-E., Syu, J.-H., Lin, J. C.-W., and Ho, J.-M. (2021b). Effective fuzzy system for qualifying the characteristics of stocks by random trading. *IEEE transactions on fuzzy systems*, 30(8):3152–3165.

Xu, W., Liu, W., Wang, L., Xia, Y., Bian, J., Yin, J., and Liu, T.-Y. (2021). Hist: A graph-based framework for stock trend forecasting via mining concept-oriented shared information. *arXiv preprint arXiv:2110.13716*.

Yang, H., Liu, X.-Y., and Wu, Q. (2018). A practical machine learning approach for dynamic stock recommendation. In *2018 17th IEEE international conference on trust, security and privacy in computing and communications/12th IEEE international conference on big data science and engineering (TrustCom/BigDataSE)*, pages 1693–1697. IEEE.

Yang, H., Liu, X.-Y., Zhong, S., and Walid, A. (2020a). Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance*, pages 1–8.

Yang, X., Liu, W., Zhou, D., Bian, J., and Liu, T.-Y. (2020b). Qlib: An ai-oriented quantitative investment platform. *arXiv preprint arXiv:2009.11189*.

Zhang, W. and Skiena, S. (2010). Trading strategies to exploit blog and news sentiment. In *Proceedings of the international AAAI conference on web and social media*, volume 4, pages 375–378.

Zhang, Y. and Wu, L. (2009). Stock market prediction of s&p 500 via combination of improved bco approach and bp neural network. *Expert systems with applications*, 36(5):8849–8854.

Zhang, Y. and Yang, X. (2017). Online portfolio selection strategy based on combining experts' advice. *Computational Economics*, 50:141–159.

Zhang, Z., Zohren, S., and Stephen, R. (2020). Deep reinforcement learning for trading. *The Journal of Financial Data Science*.