Journal of Intelligent & Fuzzy Systems xx (20xx) x–xx
DOI:10.3233/JIFS-179280
IOS Press

A novel recurrent neural network algorithm with long short-term memory model for futures trading

Quan GU[a], Na LU[b,*] and Lin LIU[c]
[a]*School of Economics and Management, Tongji University, Shanghai, China*
[b]*College of Economics and Management, Xi'an Aeronautical University, Xi'an, Shaanxi, China*
[c]*School of Management, Wuhan University of Technology, Wuhan, China*

Abstract. This paper attempts to apply recurrent neural networks (RNN) to price forecasts and financial trading. Compared with previous neural networks models, the recurrent neural network can better use the previous information to infer subsequent events, which is more suitable for price time series analysis. Long Short-Term Memory (LSTM) has made structural changes to the RNN to avoid long-term dependency problems. The empirical research uses the 2010–2017 price panel data of four kinds of soybean futures in China's futures market, and confirms the model's improved predictive ability through statistical tests. The empirical analysis of futures trading verifies the practice of these model strategies in terms of risked return. This paper improves and expands the application of recurrent neural networks model, and provides a new idea for applying artificial neural network algorithm to futures trading.

Keywords: Deep learning, algorithmic trading, trend trading, commodity futures introduction

1. Introduction

In recent years, the emergence of artificial neural network algorithms has proposed a new idea for re-search on price prediction. The artificial neural net-work model is continuously iteratively trained to detect the relationship between the input variable and the target output variable, and can achieve approximation and fitting of any continuous function. Compared with traditional statistical methods, it has its unique advantages in predicting financial market price trends: First, the model can handle irregular data without setting rules; second, it can handle uncertain, incomplete, and insufficient data with rapid convergence speed; and third, it can find the data pattern and mine the law behind the nonlinear data.

With regard to the application of artificial neural networks to the analysis of financial market price forecasts, the early literature [1, 2, 3] verified the predictive effect of artificial neural networks on the direction of securities price fluctuations. Since then, research [4, 5] has developed a securities trading strategy based on the predictive results of the artificial neural network model. It is found that the securities are bought and sold according to the output of the forecasting model, and the gains are far superior to the simple stock holding strategy. Later studies [6, 7] developed a securities trading strategy based on the predictive results of the artificial neural net-work model, and the gains were far superior to the simple shareholding strategy. Regarding the improvement of the model, some studies [8, 9] used multilayer perception neural networks (MLP) models and adopted a learning training algorithm for backward error propagation, which can obtain relatively successful

*Corresponding author. Na LU, College of Economics and Management, Xi'an Aeronautical University, Xi'an, Shaanxi, China. E-mail: 42246992@qq.com.

predictive results. Although the MLP model is widely used in financial time series prediction, there are certain defects in this kind of neural network, which mainly lies in the absence of memory function, and thus it is not suitable for the scenario of timing dependency. Literatures [10] has verified that the recurrent neural networks (RNN) algorithm is very good at processing such time-dependent data. The nodes of the hidden layer are connected at different times, and the current output depends on the previous state, that is, the network memorizes the previous information and applies it to the current calculation. However, due to the gradient explosion and gradient disappearance, the parameters of the RNN model are difficult to train, making it difficult for RNN to model long-term time series dependencies, which limits the application of RNN model in financial market. Some study [11] has introduced the Long Short-Term Memory model to preserve historical information through the state of cells, and use different gates to dynamically let the network learn when to forget historical information and when to update cell status with new in-formation. However, it remains to be further studied whether this improved design can be applied to the promotion of models for trading. This paper intends to design a hybrid model integrating recurrent neural network model with LSTM to improve algorithm predictions and apply them to futures trading.

In this paper, an improved recurrent neural network is used to establish a predictive model, which is applied to the trend trading of soybean futures in China's commodity market. The LTSM-RNN model is trained by historical data such as the opening price, the lowest price, the highest price, and the closing price, as well as other historical data, and the model is used for predicting and trading. The empirical results show that the recurrent neural network model is applicable to trend trading, and the model with long short-term memory is better. Although the performances of different futures strategies are different, the recurrent neural network algorithm model is feasible in China's futures market. The contribution of this paper is to improve the recurrent neural network model by using long short-term memory. And the model is applied to financial market price forecast analysis, which provides a new idea for trend trading.

The remainder of this paper is as follows: Section 2 is to illustrate the data of empirical research in this paper. Section 3 is the research method of the paper, mainly, the recurrent neural network algorithm with long short-term memory and the trend trading strategy based on this algorithm. Section 4 is the empirical result analysis, including statistical performances and trading performances of four futures trading strategies. Section 5 are the main conclusions of the paper.

2. Data

The research subject in this paper is the soybean futures listed on the Dalian Commodity Exchange of China (CDCE), which mainly includes four commodity futures: soybean 1, soybean 2, soy meal and soy oil. The details of futures contracts can be seen in following Table. The data-set covers the period from January 4, 2010 to December 29, 2017. The in-sample data-set covers the period from January 1, 2010 to December 31, and the out-of-sample data-set covers the period from January 4, 2016 to December 29, 2017. These panel data contain 11 explanatory variables required for artificial neural network model input, including price, transaction volume, etc. The specific information can be seen in Table 3.

3. Method

3.1. A survey of artificial neural network

Artificial Neural Network (ANN) has been a research focus in the field of artificial intelligence since the 1980s. It abstracts the human brain neuron network from the perspective of information processing, establishes a simple model, and forms different networks according to different connection methods. A neural network is an operational model consisting of a large number of neurons connected to each other. Each node represents a specific output function called an activation function. The connection between every two nodes represents a weighting value for passing the connection signal, called weight, which is equivalent to the memory of the artificial neural network. The output of the network varies depending on the connection method of the network, the weight value and the excitation function. The network itself is usually an approximation of an algorithm or function in nature, or it may be an expression of a logic strategy.

The basic form of a neuron is shown in the following mathematical expression. The D input variables and the offset terms are weighted and added. After the activation function, the output y is obtained. Commonly used activation functions are Sigmoid functions, tangent functions, and so on. The Sigmoid function and the neuron expression are as

Table 1
Commodity futures contract specifications

Contract Specifics	Soybean1	Soybean2	Soy meal	Soy oil
Code	A	B	M	Y
Size	10 tons	10 tons	10 tons	10 tons
Month	1,3,5,79,11	1,3,5,79,11	1,3,5,7, –89,11,12	1,3,5,7, –89,11,12
Tike size	¥1/ton	¥1/ton	¥1/ton	¥1/ton
Margin	5%	5%	5%	5%
Fees	¥4	¥4	¥3	¥6

Table 2
Data segregation for the full sample period

Trading Period	Days	Beginning	End
Total dataset	1947	2010/1/4	2017/12/29
Training dataset (in-sample)	970	2010/1/1	2013/12/31
Test dataset (in-sample)	489	2014/1/2	2015/12/31
Validation set (out-of-sample)	488	2016/1/4	2017/12/29

Table 3
Explanatory variables for the recurrent neural network models

Number	Variables
1	Opening price
2	Closing price
3	Highest price
4	Lowest price
5	Trading volume
6	Main purchase volume
7	Main sales volume
8	Main buyer's selling quantity ratio
9	Trading volume change rate
10	Main buying quantity change rate
11	Main selling quantity change rate

follows, where the weight coefficient w_D is the model parameter.

$$y = \sigma(X) = \frac{1}{1+e^{-x}} \tag{1}$$

$$y = \sigma\left(\sum_{i=1}^{D} w_i x_i + w_0\right) \tag{2}$$

Note: The delivery standard for the Soybean 1 futures contract is non-GM soybeans, and the delivery standard for the Soybean 2 futures contract is GM soybeans and non-GM soybeans.

A complete neural network model usually divides the nodes into several levels: input layer, output layer and hidden layer, as shown in the Fig. 1 above. The input layer is the given model input feature; the output layer is the content that is predicted by the neural network, such as function value of the sample; the hidden layer is equivalent to the intermediate state of the network system. For a recurrent neural network, the number of output layer nodes is the number of

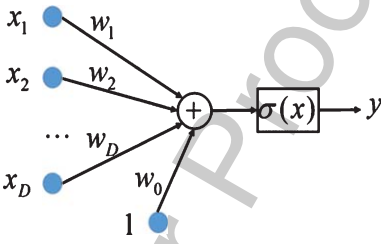

Fig. 1. The neuron of artificial neural network.

variables we want to predict. The $\sigma(X)$ and $h(X)$) are the activation functions of the output layer and the hidden layer respectively. The parameters of the neural network are the network coefficients w_{ij} of the layers, which can be collectively recorded as the vector W.

$$y_k = \sigma\left\{\sum_{j=1}^{M}\left(w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ij}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(1)}\right)\right\} \tag{3}$$

The learning of the neural network model is to use the input and output that we already have, and optimize the parameter W. So that the output given by the model is as close as possible to the real label of the sample, that is, the following prediction error (loss function) minimized:

$$E(W) = \sum_{n=1}^{N} E_n(W) = \sum_{n=1}^{N}\sum_{k=1}^{K}(y_{nk} - t_{nk})^2 \tag{4}$$

The optimization problem of the objective function is minimization mean square error. For the general neural network optimization problem, iterative optimization can be performed by the gradient descent method to obtain the optimal parameter W. It should be noted that α is the learning rate, indicating the step size of each iteration. In the nth iteration, the parameter of the n-1th iteration is moved by a certain step length in the gradient direction to obtain the latest parameter value.

$$w_{ij}^{(n)} = w_{ij}^{(n-1)} - \alpha \frac{\partial}{\partial wij} E(W) \tag{5}$$

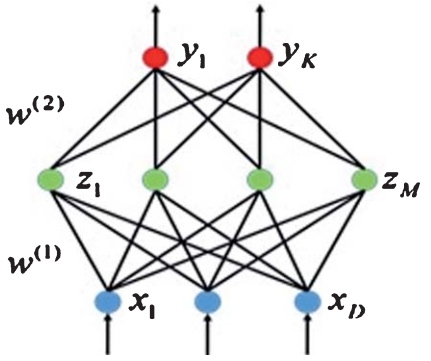

Fig. 2. The artificial neural network.

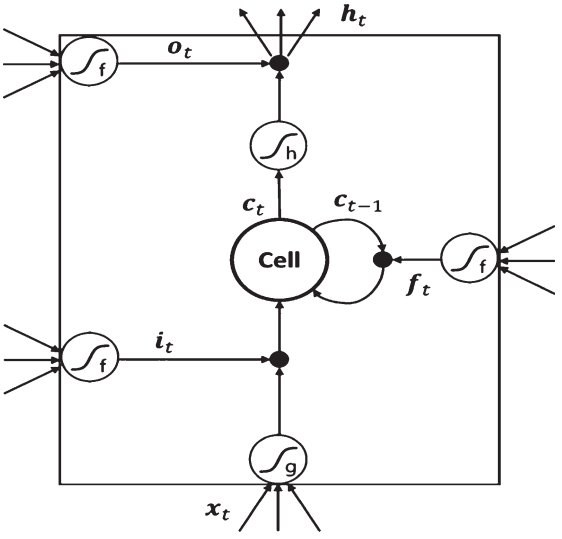

Fig. 3. The recurrent neural network.

3.2. Recurrent neural network (RNN) model

Traditional artificial neural networks could not implement the function of inferring subsequent events using previous information. The recurrent neural network algorithm is very good at processing such time-dependent data. Recurrent neural networks contain a network of loops that allow information to be persisted. These loops allow information to be passed from the current step to the next step, but the transmission of this information from front to back has long-term dependencies. Recurrent neural networks can easily use the previous information. In the traditional forward neural network, information from the input layer to the hidden layer and the output layer, the layer is fully connected, but the nodes at different times are connected. Such a network can-

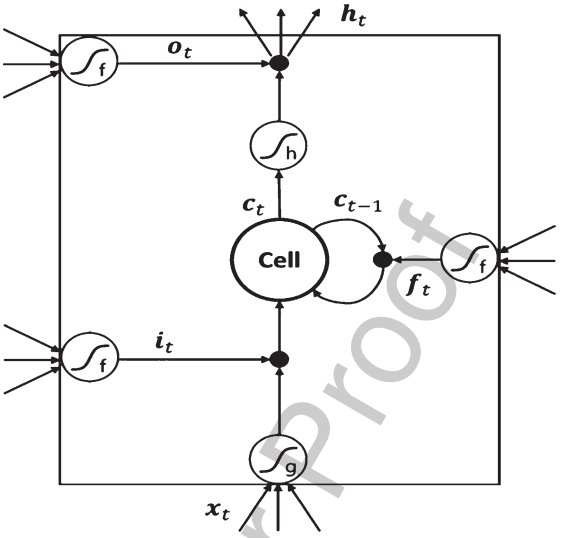

Fig. 4. The long short-term memory block.

not directly deal with time series. But in a recurrent neural network, nodes of the hidden layer at different times are connected, and the current output depends on the previous state. The network memorizes the previous information, and apply to the calculation of the current output.

RNN can be thought of as a neural network shared by weights in the time dimension. The model output depends on the hidden state h_t at t moment. It depends on the input x_t at t moment and the hidden state h_{t-1} at the $t-1$ moment. Since the hidden state has a timing dependency, the output of the RNN model is related to the input information at the previous time.

The output of the RNN model:

$$y_t = \sigma_0 (W_t h_t + b_0) \tag{6}$$

The hidden state of the RNN model:

$$h_t = \sigma_0 (W_x x_t + W_h h_{t-1} + b_h) \tag{7}$$

This can be used to analyze the relationship between the RNN model and the normal time series model. Considering the case of univariate, let $W_h = \beta$, $W_x = \alpha$, $y_t = h_t$, and let σ_h be an identity transformation, then the RNN model can be written as:

$$y_t = \alpha x_t + \beta y_{t-1} + b \tag{8}$$

This is a first-order autoregressive model with exogenous variables. Therefore, RNN is a complex nonlinear time series model. For parameter optimization of the above RNN model, we usually use the Back Propagation Through Time algorithm (BPTT) to find

the gradient. Since the hidden state h_t of the RNN is affected by the hidden state $h_{t-1}$of the previous time, the gradient of the RNN is related to time factors. The BPTT is used to expand it in time series. The formula for RNN gradient calculation is:

$$\frac{\partial e_t}{\partial W_x} = \frac{\partial e_t}{\partial h_t}\frac{\partial h_t}{\partial W_x} + \frac{\partial e_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial W_x} + \frac{\partial e_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}}\frac{\partial h_{t-2}}{\partial W_x} + \ldots \quad (9)$$

However, using the above methods, there are gradient disappearance and gradient explosion problems in parameter learning. This will make it difficult for RNN to model long-period timing dependencies.

3.3. Long short-term memory (LSTM)

The proposed Long Short-Term Memory (LSTM) effectively solves the problem of gradient explosion and gradient disappearance of simple recurrent neural networks and has been successful in many machine learning fields [11]. The key to the LSTM model is the introduction of a gating unit system that preserves historical information through the state of the cells. It uses different gates to dynamically learn when the network forgets historical information and when to update with new information.

LSTM uses doors to selectively filter a portion of the information. At time t, the internal memory unit records all historical information up to the current time and is controlled by three gates: The input gate determines the new information input to the internal memory unit; the forget gate determines that the internal memory unit needs to save the previous time information; the output gate determines the internal memory unit output information.

The input gate of LSTM:

$$i_t = \sigma\left(W^{xi}x_t + W^{hi}h_{t-1} + b^i\right) \quad (10)$$

The activation function σ is the sigmoid function, x_t is the input vector of the current time, h_{t-1} is the hidden state vector of the previous moment, W^{xi} and W^{hi} are parameters, and b is the bias term. When considering only one memory unit, i_t is a number between 0 and 1. When $i_t = 1$, the door is open and all information can be entered into the cell; when $i_t = 0$, the door is closed and information cannot be entered into the cell.

The forget gate of LSTM:

$$f_t = \sigma\left(W^{xf}x_t + W^{hf}h_{t-1} + b^f\right) \quad (11)$$

f_t is a number between 0 and 1. When $f_t = 1$, the door is opened, and the cell state is all input to the cell at the previous time; when $f_t = 0$, the door is closed, and the cell state is discarded at the previous time.

The output gate of LSTM:

$$o_t = \sigma\left(W^{xo}x_t + W^{ho}h_{t-1} + b^o\right) \quad (12)$$

o_t is a number between 0 and 1. When $o_t = 1$, the door is open and the cell state can be output; when $o_t = 0$, the door is closed and the cell state cannot be output. The internal memory unit status update formula is:

$$c_t = f_t c_{t-1} + i_t \tanh\left(W^{xc}x_t + W^{hc}h_{t-1} + b^c\right) \quad (13)$$

The former part is the information after the cell state of the previous moment is controlled by the forgetting gate, and the latter part is the information after the input information is controlled by the input gate. The output of the LSTM unit can express as follows:

$$h_t = o_t \tanh(c_t) \quad (14)$$

In the RNN model, the original RNN hidden state node is replaced by a different LSTM unit to construct an LSTM-based RNN network. In the LSTM-RNN network, the output layer of the RNN is added to the output of the LSTM unit, the output of the LSTM unit is used as the hidden layer state in the RNN model, or another LSTM layer is added to construct the multiple hidden layer RNN.

3.4. Proposing the trend trading strategy based on LSTM-RNN model

This paper studies the use of neural network algorithm to predict whether a trend trading strategy will be profitable. The specific strategy is as follows: When each trading day starts, the market data of the early trading is obtained, and the LSTM-RNN model is used to predict whether the daily trend strategy will be profitable. If it is judged that it will be profitable, the trend tracking will be carried out according to the trend of the early trading; if the model judges that it will not be profitable, the trading will not be opened on the same day. The predictive model needs to be trained before trading. In this paper, the model structure of "Time series input–Single label output" is adopted, that is, the input data is a multivariate

time series, and the output data is label data(0, 1).In the model training, it is necessary to mark the early market of the data in the sample, and mark the market as two categories: "suitable for trend trading" and "not suitable for trend trading". When trading data is quoted, we can use historical back-testing, that is use the opening range breakout strategy to test back every day to calculate whether the trend strategy is profitable. The profit is recorded as "1", which is suitable for trend trading; and on the contrary, it is recorded as "0", which is not suitable for trend trading.

The LSTM-RNN model used in this paper is designed as follows. The input layer has a total of 11 variables, including the opening price, closing price, highest price, lowest price, trading volume, main purchase volume, main sales volume, main buyer sales volume, and trading volume. rate of change, rate of change in main purchases, rate of change in main sales. The LSTM layer structure has a total of 100 layers, and the output layer is 1, that is, 0, 1 two label results, to determine whether it is suitable for transaction. The activation function is the Sigmoid function in the model:

$$S(X) = \frac{1}{1 + e^{-x}} \tag{15}$$

The Error Function to be minimize is

$$E\left(c, w_j\right) = \frac{1}{T}\sum_{t=1}^{T}\left(y_t - \widetilde{y}_t\left(w_k, c\right)^2\right. \tag{16}$$

This paper uses a simple indicator to determine whether the trend strategy is profitable. The definition of the trend strategy profit indicator is:

$$R = \frac{|p_{close} - p_{open}|}{p_{high} - p_{low}} \tag{17}$$

When the R value is large, the day trend strategy is easy to make profit; otherwise, the trend strategy is not easy to profit. In this paper, the trend strategy profit status of different trading dates is divided into two categories according to R>0.5 and R<0.5, and used as positive and negative samples to train the machine learning model. When the trading strategy is stetted, use the model to predict the trend strategy profit indicator of the day, get the profit probability p, and calculate the mean MA30(p) of the probability at T time in the past 30 trading days. If p>MA30(p), it indicates that the trend of the day's trend is relatively high, and it is possible to conduct trend trading. After opening a position, the position is closed until the close of the day, and the position will be immediately closed when the stop loss is triggered. In the specific simulation transaction, it should be noted that we set the futures margin to 50%, that is, consider a 2x leverage. The transaction cost is 0.002% bilaterally, and the stop loss is 25% fixed loss. In addition, in order to verify the improvement effect of the model, we will empirically compare the traditional RNN model with the LSTM-RNN model.

3.5. Model evaluation

The evaluation of the model is divided into two aspects, one is about the predictive effect of the model, and the other is about the performance of the model applied to the trading. When evaluating model predictions, the most common evaluative indicators includes Root mean square error (RMSE), Mean absolute error (MAE), Mean absolute percentage error (MAPE), which measure the difference between the predicted value and the true value of a variable. The specific explanation of these measures can be seen in Table 4. In Table 5, we list the measurement of trading performance, mainly including the measurement of the return value, on the one hand, the absolute return, including the Annualized return and Cumulative return, and on the other hand, the return volatility, including the Annualized volatility and the Maximum drawdown. On the basis of this, there is also a measure of risk return. This paper uses Shape ratio and Calmar ratio.

Table 4
Statistical performance measures

MAE	$MAE = \sqrt{\sum \lvert y_t - \widetilde{y_t}\rvert / T}$
MAPE	$MAPE = {}^{1}\!/_{T}\sum \lvert y_t - \widehat{y_t}\rvert / y_t$
RMSE	$RMSE = \sqrt{\sum \left(y_t - \widehat{y_t}\right)^z / T}$

Table 5
Trading performance measures

Annualized return	$R^A = 252 * {}^{1}\!/_{N}\sum_{t=1}^{N} R_t$
Cumulative return	$R^c = {}^{1}\!/_{N}\sum_{t=1}^{N} R_t$
Annualized volatility	$\sigma^A = \sqrt{{}^{252}\!/_{N-1} * \sum_{t=1}^{N}\left(R_t - \overline{R}\right)^z}$
Shape ratio	$SR = {}^{R^A}\!/_{\sigma^A}$
Maximum drawdown	$MaxDD = \min\left[R_t - \max(\sum_{t=1}^{N} R_t\right]$
Calmar ratio	$CR = {}^{R^A}\!/_{MaxDD}$

4. Results

4.1. Statistical performance

Before simulating the transaction, we need to perform a quantitative analysis of the model fitting effect to confirm the prediction effect of the neural network model. The statistical performance test is mainly divided into the following four indicators: Mean absolute error, mean absolute percentage error, and Root mean square error. They are all indicators that measure the difference between the predicted and true values of a variable. According to previous literature studies [9, 12], we could conclude that the smaller the value, the more accurate the prediction.

It can be seen from the data in Tables 6 and 7 that the overall predictive effect of LSTM-RNN is better than that of T-RNN, indicating the improvement effect of the long short-memory term. From the comparison, the predictive effect in the sample is stronger than the prediction effect out-of-sample. This is also in line with previous research [10]. The prediction effect in the sample is better, but there may be an over-fitting effect. From the perspective of commodities, soy meal and soy oil have better predictive effects than soybean 1 and soybean 2, which may be closely related to the larger trading volume and more flexible price volatility. In terms of specific indicators, the MAE is the mean absolute values of the deviations of all individual observations from the arithmetic mean. The MAPE is expressed as a percentage of MAE. These two indicators can accurately reflect the actual predictive errors, and also show that the LSTM-RNN model algorithm is better than the T-RNN model algorithm. The RSME is sensitive to very large or very small errors in a set of measurements and is a good reflection of the precision of the model measurements. In this indicator comparison, the difference between LSTM-RNN and T-RNN prediction is not so obvious as former indicators.

Table 6
In-sample statistical performance

	LSTM-RNN			
	Soybean1	Soybean2	Soymeal	Soy oil
MAE	0.0784	0.0412	0.0409	0.0321
MAPE	23.19%	22.12%	19.25%	24.57%
RMSE	0.0878	0.0965	0.0954	0.0875
	T-RNN			
	Soybean1	Soybean2	Soy meal	Soy oil
MAE	0.0965	0.0765	0.654	0.0761
MAPE	25.61%	26.43%	22.71%	21.34%
RMSE	0.0932	0.0871	0.0912	0.0813

Table 7
Out-of-sample statistical performance

	LSTM-RNN			
	Soybean1	Soybean2	Soy meal	Soy oil
MAE	0.0871	0.0785	0.0546	0.0459
MAPE	25.17%	26.13%	22.15%	25.12%
RMSE	0.0976	0.0941	0.0892	0.0931
	T-RNN			
	Soybean1	Soybean2	Soy meal	Soy oil
MAE	0.0697	0.0817	0.0675	0.0617
MAPE	24.17%	19.12%	23.19%	29.12%
RMSE	0.0956	0.0896	0.0914	0.0959

4.2. Trading performance

In this section, we examine the effects of the two models of LSTM-RNN and T-RNN applied to trend trading. For the measurement of trading performance, we must consider the absolute value of the return, but also the volatility of the income. Therefore, the average annual return and cumulative return measure the performance of the strategy, while the average annual volatility measures the volatility of the return of the strategy, especially the extreme risk of the tail of the futures trading needs to be specifically measured. And risk-return is an important criterion for comparison of evaluations.

From the data in Tables 8 and 9 above, it can be concluded that the LSTM-RNN model performs better than the T-RNN, but the performance in different futures is quite different. In the soybean oil trend trading and soybean meal trend trading, the effect of using long short-term memory models to improve the neural network is particularly obvious. It may consistent with the previous analysis, soy meal and soy oil futures with large trading volume, price are intensive, the price data distribution is continuous. And there is no price gap caused by the small transaction volume, which will not cause the deviation of the model training. Compared with the empirical results of the model in-sample and out-of-sample, the model has better fitting results because of the more training data, so the model performs better in the sample than out-of-sample. From the perspective of specific trading performance metrics, the absolute returns of the two models are not significantly different, but in terms of the annualized volatility and the maximum drawdown, the LSTM-RNN model is better than the T-RNN model. From the aspect of risk-return metrics, the LSTM-RNN model is better than the T-RNN model, which is particularly evident in the calmar

Table 8
In-sample trading performance

LSTM-RNN				
	Soybean1	Soybean2	Soy meal	Soy oil
Annualized return	13.17%	17.16%	28.82%	21.02%
Cumulative return	44.29%	23.12%	39.12%	45.62%
Annualized volatility	54.12%	49.71%	39.84%	42.45%
Shape ratio	0.2433	0.3452	0.7233	0.4952
Maximum drawdown	65.19%	77.14%	43.12%	37.84%
Calmar ratio	0.2020	0.2224	0.6684	0.5555
T-RNN				
	Soybean1	Soybean2	Soy meal	Soy oil
Annualized return	24.18%	11.87%	21.12%	18.27%
Cumulative return	51.12%	18.19%	29.13%	36.68%
Annualized volatility	64.12%	54.28%	38.29%	43.71%
Shape ratio	0.3771	0.2187	0.5516	0.4182
Maximum drawdown	48.73%	51.79%	29.27%	44.71%
Calmar ratio	0.4962	0.2292	0.7216	0.4089

Table 9
Out-of-sample trading performance

LSTM-RNN				
	Soybean1	Soybean2	Soy meal	Soy oil
Annualized return	23.41%	16.12%	22.78%	19.21%
Cumulative return	29.45%	34.17%	36.15%	38.27%
Annualized volatility	43.21%	55.29%	43.74%	39.28%
Shape ratio	0.5418	0.2916	0.5208	0.5020
Maximum drawdown	58.17%	61.34%	47.21%	50.12%
Calmar ratio	0.4024	0.2628	0.4825	0.3833
T-RNN				
	Soybean1	Soybean2	Soy meal	Soy oil
Annualized return	37.28%	27.18%	18.34%	32.16%
Cumulative return	46.21%	47.39%	51.28%	63.28%
Annualized volatility	41.34%	34.43%	48.78%	40.06%
Shape ratio	0.9018	0.7894	0.3760	0.5532
Maximum drawdown	48.39%	74.25%	68.76%	61.23%
Calmar ratio	0.7704	0.3661	0.2668	0.3619

ratio. It also shows that the LSTM is used to improve the robustness of the RNN model, especially in the futures trading.

5. Conclusions

In this paper, we have designed a recurrent neural network model based on the improvement of long short-term memory, which was applied to futures price forecasting and futures trend trading. Taking the historical futures price panel data as input, the model predicts the price increase or decline trend, and then conducted simulated trading. The empirical research results show that the model prediction effect is sound and it is effective in the futures trend trading strategy. This paper has for the first time illustrated the application of the improved LSTM-RNN model in futures trend trading. Of course, the model's application also has an over-fitting problem, which is a common problem in artificial neural networks. In the future, the hidden layer node self-generation method can be used to shorten the network training, reduce the network structure, and improve the model's accuracy.

References

[1] C. Ntungo and M. Boyd, Commodity futures trading performance using neural network models versus ARIMA models, *Journal of Futures Markets* **18.8** (1998), 965–983.

[2] P.C. Chang, C.Y. Fan and C.H. Liu, Integrating a Piecewise Linear Representation Method and a Neural Network Model for Stock Trading Points Prediction, *IEEE Transactions on Systems Man & Cybernetics Part C* **39.1** (2008), 80–92.

[3] P.C. Chang, et al., A neural network with a case based dynamic window for stock trading prediction., *Expert Systems with Applications An International Journal* **36.3** (2009), 6889–6898.

[4] G. Sermpinis, et al., Forecasting and trading the EUR/USD exchange rate with stochastic Neural Network combination and time-varying leverage, *Decision Support Systems* **54.1** (2012), 316–329.

[5] C. Evans, K. Pappas and F. Xhafa, Utilizing artificial neural networks and genetic algorithms to build an algo-trading model for intra-day foreign exchange speculation, *Mathematical & Computer Modelling* **58.5–6** (2013), 1249–1266.

[6] A. Rodríguez González, et al., CAST: using neural networks to improve trading systems based on technical analysis by means of the RSI financial indicator, *Expert Systems with Applications* **38.9** (2011), 11489–11500.

[7] W.L. Tung and C. Quek, Financial volatility trading using a self-organising neural-fuzzy semantic network and option straddle-based approach, *Expert Systems with Applications* **38.5** (2011), 4668–4688.

[8] B. Vanstone and G. Finnie, An empirical methodology for developing stockmarket trading systems using artificial neural networks, *Expert Systems with Applications* **36.3** (2009), 6668–6680.

[9] G. Sermpinis, J. Laws and C.L. Dunis, Modelling and trading the realised volatility of the FTSE100 futures with higher order neural networks, *The European Journal of Finance* **19.3** (2013), 165–179.

[10] C. Quek, M. Pasquier and N. Kumar, A novel recurrent neural network-based prediction system for option trading and hedging, *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies* **29.2** (2008), 138–151.

[11] H.Y. Kim and C.H. Won, Forecasting the Volatility of Stock Price Index: A Hybrid Model Integrating LSTM with Multiple GARCH-Type Models, *Expert Systems with Applications* **103** (2018), 25–37.

[12] C.L. Dunis, J. Laws, P.W. Middleton and A. Karathanasopoulos, Trading and hedging the corn/ethanol crush spread using time-varying leverage and nonlinear models, *The European Journal of Finance* **21.4** (2015), 352–375.