# Application of LSTM model optimized by individual-ordering-based adaptive genetic algorithm in stock forecasting

Yong He
*School of Computer, Dongguan University of Technology, Dongguan, China*
Xiaohua Zeng
*School of Economics and Trade, Guangzhou Xinhua University, Dongguan, China, and*
Huan Li and Wenhong Wei
*School of Computer, Dongguan University of Technology, Dongguan, China*

Received 19 April 2022
Revised 25 June 2022
Accepted 18 July 2022

## Abstract
**Purpose** – To improve the accuracy of stock price trend prediction in the field of quantitative financial trading, this paper takes the prediction accuracy as the goal and avoid the enormous number of network structures and hyperparameter adjustments of long-short-term memory (LSTM).
**Design/methodology/approach** – In this paper, an adaptive genetic algorithm based on individual ordering is used to optimize the network structure and hyperparameters of the LSTM neural network automatically.
**Findings** – The simulation results show that the accuracy of the rise and fall of the stock outperform than the model with LSTM only as well as other machine learning models. Furthermore, the efficiency of parameter adjustment is greatly higher than other hyperparameter optimization methods.
**Originality/value** – (1) The AGA-LSTM algorithm is used to input various hyperparameter combinations into genetic algorithm to find the best hyperparameter combination. Compared with other models, it has higher accuracy in predicting the up and down trend of stock prices in the next day. (2) Adopting real coding, elitist preservation and self-adaptive adjustment of crossover and mutation probability based on individual ordering in the part of genetic algorithm, the algorithm is computationally efficient and the results are more likely to converge to the global optimum.

**Keywords** Stock price prediction, Long-short-term memory, Adaptive genetic algorithm, Machine learning
**Paper type** Research paper

## 1. Introduction
It is a particularly challenging problem to predict the short-term trend of stock price. Many uncertain factors, such as news, policy and market sentiment, may lead to drastic changes in stock price in the short term. To minimize the investing risks in stock market, it is necessary to predict the up-and-down trend of stock market prices to maximize capital gains and minimize losses. Because the massive and complex stock data needs huge workload, the traditional non-artificial intelligence methods are difficult to achieve satisfactory results in stock price trend prediction. Based on the assumption of linearity among normally distributed variables, traditional statistical model assume that the time series under study is generated from a linear process and try to model the underlying process based on this assumption. These techniques include time series regression, exponential smoothing, auto-regressive integrated moving average (ARIMA) and its variations,

This work was supported by the Key Project of Science and Technology Innovation 2030 supported by the Ministry of Science and Technology of China (No. 2018AAA0101301), the Key Projects of Artificial Intelligence of High School in Guangdong Province (No. 2019KZDZX1011) and The High School innovation Project (No. 2018KTSCX222).

generalized auto regressive conditional heteroskedasticity (GARCH), etc. (Lin *et al.*, 2012; Wang *et al.*, 2012). However, financial time series present an abundance of uncertainty and noise this kind of data presents a nonlinear behavior and no identical statistical properties may be observed at each point of time. Besides, dynamic changes in the relationship between independent and dependent variables happen frequently in financial time series (Cavalcante *et al.*, 2016). Since financial markets are considered complex, evolutionary, noisy and nonlinear and non-parametric dynamic system (Huang and Tsai, 2009), more adaptive and flexible mechanisms are required to improve forecasting accuracy. In recent years, artificial intelligence technology is widely used to predict stock prices. Its prediction involves statistical methods, such as time series model and multivariate analysis. The change of stock price is regarded as a function of time series and solved as a regression problem (Bao and Yang, 2008). However, due to the chaos and high volatility of the stock market, it is better to regard the stock forecasting as a classification problem than a regression problem (Khaidem *et al.*, 2016).

Dai and Zhang (2013) carried out supervised model to predict stock prices. Logistic regression, Gaussian discriminant analysis, quadratic discriminant analysis and Support Vector Machine (SVM) were used to train the short-term model for predicting the stock price trend of the next day and the long-term model for predicting the stock price results in the next $n$ days. The accuracy of the prediction trend of the next day prediction model is exceptionally low, and the average accuracy of the four models is 50%. In 2011, Kara *et al.* (2011) compared SVM model and back propagation (BP) neural network to predict the trend of stock index and found that the prediction effect of BP neural network is better than SVM model. Krauss *et al.* (2016) introduced the random forest model in the integrated learning method into the stock forecasting problem. They compared the work efficiency of four models: random forest (RF), SVM, BP neural network and naive Bayes. They found that the random forest is the best in the overall performance.

Because it is very suitable for time series prediction with high randomness and can selectively learn useful hidden information from many complex historical data, long-short-term memory (LSTM) neural network can achieve excellent performance in the prediction of time series (Bao *et al.*, 2017). Although LSTM was rarely used for the time series prediction of stock prices, its features are in good agreement with the trend prediction of stock prices. Chen *et al.* (2015) used the LSTM network model to predict the stock return of China's stock market. Compared with other prediction method, the LSTM model improved the accuracy of stock return prediction from 14.3 to 27.2%. Fischer and Krauss (2018) used LSTM to predict the stock yields on S&P 500. Although the performance of this method is better than random forest, deep neural network and logistic regression, the accuracy is still relatively low, which is between 51 and 54%. Moghar and Hamiche (2020) used the LSTM model to predict the opening price of Google and NKE. In their model, the test errors are $4.97 \times 10^{-4}$ and $8.74 \times 10^{-4}$ in the case of small datasets. Nelson *et al.* (2017) predicted the future trend of stock prices based on price history and technical analysis indicators. According to the analysis results, the average accuracy is as high as 55.9%. Bathla (2020) compared LSTM and support vector regressor (SVR) by inputting nine indexes of five US stocks, the average prediction accuracy of LSTM is 66%. Ding *et al.* (2015) proposed a deep learning method to predict event driven stock price changes. Compared with the most advanced baseline method, their model can improve the prediction accuracy of S&P 500 index and individual stock by nearly 6%. Xie *et al.* (2019) evaluated the prediction performance of multi-layer LSTM neural network. The experimental results show that the average accuracy is 47.33%, the average precision is 49.83%, the average recall is 63.5% and the average F1-score is 55.5%. Staffini (2022), proposes a deep convolutional generative adversarial network, named DCGAN, for stock price forecasting. The author tests the empirical performance using the Financial Times Stock Exchange Milano Indice di Borsa and shows the proposed model achieve better performance, both in single-step and multi-step forecasting, improving over the benchmark study of GAN architecture proposed by Zhang *et al.* (2019). Mehtab and Sen (2020) apply multi convolutional neural network (CNN)s to augment regression models, obtaining good results for long-term forecasting. Gao *et al.* (2021), test two of the most commonly used deep learning models in time series forecasting LSTM and gated recurrent unit (GRU), finding a similar performance of the two methods.

The accuracy and convergence of neural networks are highly dependent on the combination of hyperparameters. Because the network topology is constructed by these hyperparameters according to experience or trial and error method, it is difficult to ensure to obtain a suitable topology that meets the practical application. Liashchynskyi and Liashchynskyi (2019) compared the accuracy of image recognition and the running time of neural networks optimized by grid search (GS), random search (RS) and genetic algorithm (GA) on CIFAR-10 dataset. The results show that evolutionary algorithm is the best choice, and convolution neural network optimized by GA improves the accuracy of image recognition from 76 to 86%. Bani-Hani *et al.* (2018) studied the image recognition of four diverse types of leukocytes. They optimized the hyperparameters of CNN through GA. The optimized CNN achieves 91 and 99% classification accuracy on the test set and training set, respectively. Loussaief and Abdelkrim (2018) conducted another image recognition research. An optimized CNN is designed to do the image recognition work, which accuracy is improved from 90% to more than 98%. Wei and You (2019) used the improved genetic algorithm to optimize a series of super parameters in the fully connected neural network in CNN. The accuracy of their model can reach 98.81%, which is higher than 98.4% of the official sample model. Wicaksono and Supianto (2018) showed that GA is almost as accurate as GS, but it can significantly reduce the computational cost, especially at large solution domain. They showed this meta-heuristic algorithm can reduce the processing time by 85%. Chung and Shin (2018) used GA to optimize the time window size and number of neurons of LSTM, and the MSE and RMSE is lower than the benchmark model. Junior and Yen (2019) used particle swarm optimization (PSO) to optimize CNN to find the best hyperparameters. The quick success to a good CNN network structure shows the superiority of its PSO-CNN model. Cai *et al.* (2007) combined PSO and EA with RNN to estimate missing values in time series data. Song *et al.* (2019) used PSO to optimize the LSTM neural network to reduce the root mean square error (RMSE) of the fitting curve and better fit the actual price.

In this paper, an adaptive genetic algorithm based on individual ordering will be used to optimize the network structure of LSTM model. The AGA-LSTM algorithm is used to generate a short-term model to predict the up and down trend of stock prices in the next day. The main contributions of this paper are: (1) The AGA-LSTM algorithm is used for inputting various hyperparameter combinations into GA to find the best hyperparameter combination. Compared with other models, it has higher accuracy in predicting the up and down trend of stock prices in the next day. (2) Adopting real coding, elitist preservation and self-adaptive adjustment of crossover and mutation probability based on individual ordering in the part of genetic algorithm, the algorithm is computationally efficient and the results are more likely to converge to the global optimum.

## 2. Method

### *2.1 LSTM*

LSTM is a neural network structure that connects the LSTM unit as a directed graph (Pascanu *et al.*, 2013). The characteristic of the LSTM neural network is that it can store the input of the neuron at the current time and the output at the previous time. LSTM can store the relationship between the current input of the neuron and the previous output (Dong and Li, 2015). By introducing the LSTM unit in the chain structure, LSTM has a complex internal structure, which can learn long-term calculation and memory information from input data. Nowadays LSTM has been widely utilized in the prediction of nonlinear time series data in recent years (Ma *et al.*, 2015). Figure 1 shows the internal structure of the LSTM calculation unit.

LSTM has three gates to protect and control the state of a single computing unit. The forget gate determines the information that needs to be discarded, which can be expressed as:

$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f). \tag{1}$$

where $W_f$ represents the connection weight of the previous output, $h_{t-1}$ is the previous output, $x_t$ is the current input, $b_f$ is the bias vector and $\sigma$ is the activation function. The input gate determines the information that needs to be updated. It can be obtained by multiplying the two vectors created by the input gate layer and the tanh layer, which can be described as:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{2}$$

and

$$\widetilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \tag{3}$$

The output gate updates the addition of the information of the input gate and the forget gate. It can be defined as:

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t. \tag{4}$$

The calculation unit outputs the result after the above operation. It consists of two steps. First, the sigmoid activation function determines the output part. Then, the cell state passes through the tanh layer to normalize the value to between $-1$ and 1 and performs dot multiplication. The output of this process determines the output part, which can be expressed as:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_0), \tag{5}$$

$$h_t = o_t * tanh(C_t). \tag{6}$$

Connecting each LSTM unit with a chain structure can get a LSTM neural network.

The optimization algorithm is used for gradient descent and the weight is continuously updated.

Adam calculates both the first-order moment estimation and second-order moment estimation so that each parameter can obtain an independent adaptive learning rate (Kingma

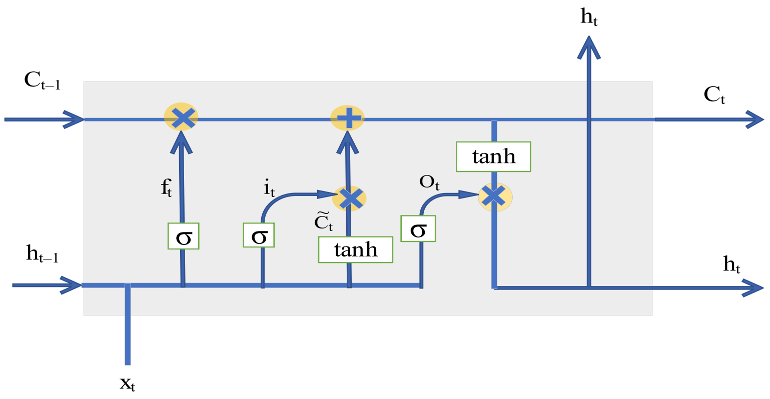

**Figure 1.**
The internal structure of the LSTM calculation unit

and Ba, 2014). As the number of network layers increases, it often leads to long training time and over-fitting problems. By adding a dropout rate, randomly omitting a certain ratio of feature detectors in each training case can greatly reduce the over-fitting phenomenon (Srivastava *et al.*, 2014).

### *2.2 Genetic algorithm*

*2.2.1 Genes code.* As shown in Figure 2, real number coding is adopted as the coding method of the genetic algorithm, which directly performs genetic operations on the phenotype of the solution. There are eight genes in the individual, the first gene represents the number of LSTM layers, 1–2, and the step size is 1. The second gene indicates the number of dense layers, 1–2, with a step size of 1. The third and fourth genes represent the number of neurons in each layer of the LSTM layer, 32–128, and the step size is 1. The seventh and eighth genes represent the dropout rate of dropout layer, 0.1–0.5, the step size is 0.1. Since our proposed model focus on predicting the direction of price movement, rather than predicting prices. More training epochs will fit a price closer to the predicted value, i.e. the root means square error will become lower. But there is no way to determine whether the specific predicted value is actually higher or lower than the true value. We conducted comparative experiments and tested the training epochs for 100, 200, 300, 400, 500 and 600 rounds and found that the price movement prediction accuracy was similar to that of the epochs 1–3 rounds. It may be that our experimental data and feature dimensions are relatively small, and more rounds are likely to cause overfitting problems. Therefore, based on the consideration of preventing overfitting and excessive consumption of computing resources, we finally set the number of training epochs to 1–3 and the step size is 1.

*2.2.2 Individual-ordering-based adaptive crossover and mutation method.* Canonical genetic algorithms use fixed crossover rate and mutation rate. In the early stage of evolution, individuals with poor fitness can be quickly screened out. These algorithms have small computation costs and the fast convergence speed, but there is premature convergence. Ho *et al.* (1992) proposed a sequential optimization method to quantitatively evaluate the optimization method from the perspective of probability (Ho *et al.*, 1992). The crossover rate and variation rate only depend on the ranking position of individual fitness value in the population, not the actual fitness value. Therefore, the sorting number is used for replacing the value comparison. This method improves the optimization ability of the algorithm, help the algorithm jump out of the local optimal solution and accelerate the convergence speed of the population to the optimal solution.

The adaptive single-point crossover method is adopted, and a crossover point is randomly set in the individual. The adaptive crossover probability is expressed as:

$$p_c = \begin{cases} p_{c_1} \times \dfrac{1}{\left(p_{c_1} - p_{c_2}\right) + e^{\frac{N_1 - N_2}{N_3 - N_2}}} \;, N_1 \geq N_2, \\ 0.9, \;\; N_1 < N_2 \end{cases} \qquad (7)$$

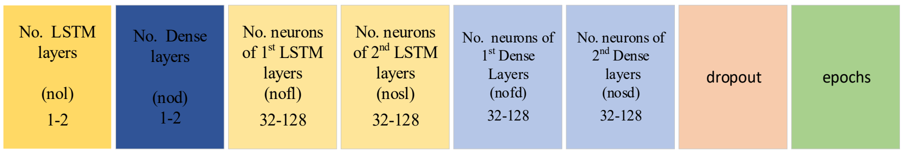
No. LSTM layers (nol) 1-2
No. Dense layers (nod) 1-2
No. neurons of 1st LSTM layers (nofl) 32-128
No. neurons of 2nd LSTM layers (nosl) 32-128
No. neurons of 1st Dense Layers (nofd) 32-128
No. neurons of 2nd Dense layers (nosd) 32-128
dropout
epochs

**Figure 2.** Chromosome of genetic algorithm

In Eq. (7), $N_1$ represents the ordering number of the current individual's fitness value, $N_2$ is the ordering number of the average fitness value and $N_3$ is the ordering number of the maximum fitness value. For individuals with high fitness, the smaller crossover rate is consistent with the "keep good individuals" setting, while the larger crossover rate for individuals with low fitness is consistent with the "change poorly adapted individuals" setting.

Here generally set $p_{c_1} = 0.9$ or 1, and $p_{c_2}$ takes the value in the interval [0.5, 1] to adjust the cross rate adaptively.

The mutation method used in this paper is an adaptive mutation single-point operator, which only adjusts the value of the individual after the two genes, and randomly selects the value according to the corresponding range, where the adaptive mutation probability is expressed as:

$$p_m = \begin{cases} p_{m_1} \times \dfrac{1}{\left(p_{m_1} - p_{m_2}\right) + e^{\frac{N_1 - N_2}{N_3 - N_2}}}, & N_1 \geq N_2 \\ 0.09, \quad N_1 < N_2 \end{cases} \tag{8}$$

Here generally set $p_{m_1} = 0.09$ or 0.1, and $p_{c_2}$ takes the value in the interval [0.05, 0.1] to adjust the cross rate adaptively.

*2.2.3 Elitist preservation.* Since the fitness of each individual in the experiment of this paper is similar, the canonical genetic algorithm uses proportional selection (the roulette method) based on the random number generated, which may incorrectly eliminate those individuals with high fitness. The idea is to replicate the best individuals (called elitist individuals) that have emerged so far in the population during evolutionary time into the next generation without crossover. The elitist is the individual with the highest fitness value searched for by the genetic algorithm in the evolution of the population, and it has the best genetic structure and good characteristics.

## 3. Methodology and analysis

### *3.1 Steps of the algorithm*

The process of experiments is described by following the steps of data download, data preprocessing, data denoising, divide the dataset and data normalization, AGA-LSTM model training, closing price prediction, model evaluation and experimental result recording.

According to the real results of the samples and the prediction results of the model, the samples are divided into four categories: TP (the number of positive class for both the sample and the predicted outcome of the model), FN (the number of positive class for the sample and the predicted outcome of the negative class), FP (the number of negative class for the sample and the predicted outcome of the positive class) and TN (the number of negative class for both the sample and the predicted outcome of the model). We use the accuracy of model prediction as the fitness function:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

The overall framework of AGA-LSTM prediction model established in this paper is shown in Figure 3, including data processing module, parameter optimization module and model evaluation module.

### *3.2 Data processing*

We use the stock data of Tesla (The time span is from 2010-06-29 to 2018-03-27), Google (The time span is from 2004-08-19 to 2018-03-27), Amazon, Apple and Microsoft (The time span is from 2000-11-01 to 2018-03-27) provided by the quandl API. According to Krauss' sample processing method, for each sample, six-dimensional data (opening price, closing price, highest price, lowest price, trading volume of a certain stock and the trend of the next day) of 50 days before the current time are used as input and the closing price of the next day as output (2016). The trend of the next day is a classification label calculated based on the closing price. $C_{t+1}$ represents the closing price on $day_{t+1}$, and the classification label on $day_t$ is calculated by

$$y_t = \begin{cases} 1, & C_{t+1} > C_t \\ -1, & C_{t+1} <= C_t \end{cases} \tag{10}$$

If the closing price of the next day is higher than the closing price of the day, it is set to rise and the value is set by 1; otherwise, it is set to fall and the value is set by $-1$. In order not to use the data having the information we are trying to predict to train a model, the trend for $day_{50}$ is changed to the same as $day_{49}$. Table 1 shows the data example.

### 3.3 Data noise reduction

Because the market dynamics are extraordinarily complex, these data contain infrequent noise, so the Pywt library in Python is used to perform wavelet transformation to remove data noise. Figures 4 and 5 are Google stocks before and after wavelet transformation.

### 3.4 Data normalization

Since features such as closing price and transaction volume are input as characteristic values at the same time, the value range is quite different. It is necessary to avoid excessively large transaction volume from having too much influence on the forecast results, and normalization processing is required. The range of data is processed and controlled between 0 and 1. The formula for data normalization is as follows:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{11}$$

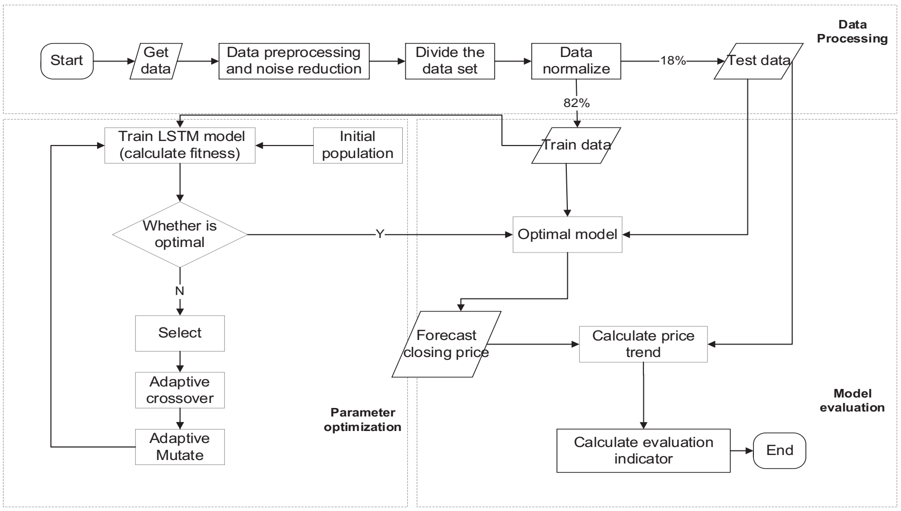

**Figure 3.**
The experiment processes

**Table 1.**
Example of the Google dataset (partial)

|  |  | Features |  |  |  |  |  |
|---|---|---|---|---|---|---|---|
| Google stock data (partial) | Date | Open | High | Low | Close | Volume | Trend |
| 1 | 2004/8/19 | 100.01 | 104.06 | 95.96 | 100.335 | 44,659,000 | 1 |
| 2 | 2004/8/20 | 101.01 | 109.08 | 100.5 | 108.31 | 22,834,300 | 1 |
| 3 | 2004/8/23 | 110.76 | 113.48 | 109.05 | 109.4 | 18,256,100 | −1 |
| 4 | 2004/8/24 | 111.24 | 111.6 | 103.57 | 104.87 | 15,247,300 | 1 |
| 5 | 2004/8/25 | 104.76 | 108 | 103.88 | 106 | 9,188,600 | 1 |
| 6 | 2004/8/26 | 104.95 | 107.95 | 104.66 | 107.91 | 7,094,800 | −1 |
| 7 | 2004/8/27 | 108.1 | 108.62 | 105.69 | 106.15 | 6,211,700 | −1 |
| 8 | 2004/8/30 | 105.28 | 105.49 | 102.01 | 102.01 | 5,196,700 | 1 |
| 9 | 2004/8/31 | 102.32 | 103.71 | 102.16 | 102.37 | 4,917,800 | −1 |
| 10 | 2004/9/1 | 102.7 | 102.97 | 99.67 | 100.25 | 9,138,200 | 1 |

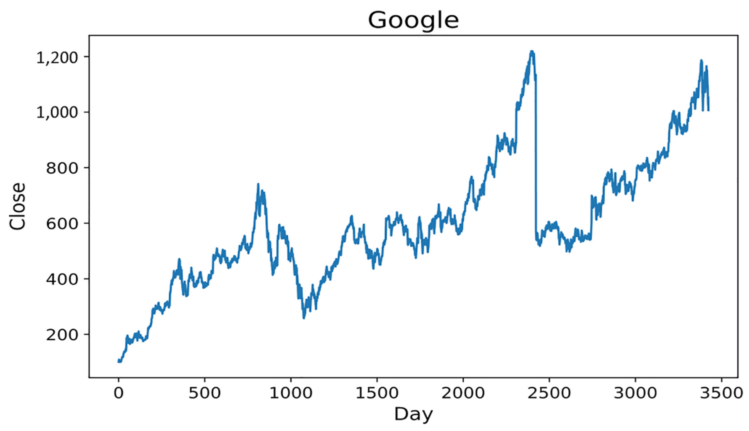

**Figure 4.**
Before wavelet transform

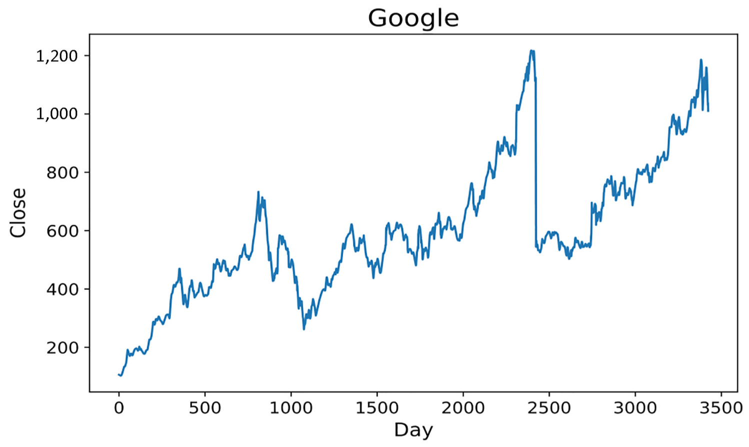

**Figure 5.**
After three-layer decomposition of wavelet transform

$x_{norm}$ is the converted value, $x_{min}$ is the minimum value of the sample and $x_{max}$ is the maximum value of the sample.

We first divide the dataset, then normalized the training data and save the normalized parameters (mean and variance). The test set is processed with the same parameters. Divided the original data sequence $D = \{d_1, d_2, \ldots, d_n\}$ into training set and test set are $d_{tr} = \{d_1, d_2, \ldots, d_m\}$ and $d_{te} = \{d_{m+1}, d_{m+2}, \ldots, d_n\}$. Normalize the training set to obtain a new converted data sequence $x = \{x_1, x_2, \ldots, x_m\}$. And set the timestamp of the data to 50 as S, the input of train data is:

$$\mathrm{X} = \{\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_s\} \tag{12}$$

$$X_i = \{x_i, x_{i+1}, \ldots, x_{m-s-1}\} \quad s.t \quad 1 \le i \le S; i, S \in N \tag{13}$$

*3.5 Evaluation indicator*

The actual and predicted closing prices are:

$$\mathrm{Y} = \{\mathrm{Y}_{m+1}, \mathrm{Y}_{m+2}, \ldots, \mathrm{Y}_n\} \tag{14}$$

$$y = \{\mathrm{y}_{m+1}, \mathrm{y}_{m+2}, \ldots, \mathrm{y}_n\} \tag{15}$$

and

$$\mathrm{Z}_t = \begin{cases} 1, & (y_{t+1} - y_t) * (Y_{t+1} - Y_t) > 0 \\ -1, & others \end{cases} \tag{16}$$

According to the actual results of the sample and the model prediction results, the sample is divided into TP, FN, FP and TN. Thus, the robustness of the model also be considered. AUC characterizes the classifier's ability to rank positive samples in front of negative samples. The larger the area of the AUC, the better the classification effect ($0.5 \le AUC \le 1.0$).The evaluation indicators used in this article to evaluate the predictive performance and robustness of the classifying model are Accuracy, Precision, Recall (also known as sensitivity), F1-Score and AUC. They can be calculated by the formulas below:

$$Precision = \frac{TP}{TP + FP} \tag{17}$$

$$Recall = \frac{TP}{TP + FN} \tag{18}$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{19}$$

$$AUC = \text{Area Under Curve ROC} \tag{20}$$

## 4. Experiment and result

The computer configuration and software environment used in the experiment are as follows: The processor is Intel(R) Core (TM) i5-9500 CPU, the memory is 8.0 GB; the system is Windows 10 (64-bit); the programming language version is Python 3.8.5; IDE is Jupyter 2021.8 in the Visual Studio Code extension. AGA-LSTM and LSTM are implemented in the Keras library with tensorflow as the backend. Take the Google dataset as an example. 3,500 trading days remain in the dataset. The entire dataset is divided into a training dataset (first 80%, 2,800 trading days) and a holdout dataset (last 20%, 700 trading days). The holdout dataset is reserved for the evaluation of the final optimal feature set and to compare performance comparison to the benchmark. The training dataset is split into the first 82% and the last 18% of 2,800 trading days. The first 82% is used to train for learning the proposed algorithm and the last 12% is used for validating the model. The parameters of the adaptive genetic algorithm are set as follows (see Table 2).

After the training results are shown in Figure 6, AGA-LSTM and GA-LSTM can quickly find the approximate optimal solution in the search space. Compared with GA, the effect of AGA is not much different in the early and mid-term, but it suddenly becomes higher in the later term. The results verify that the adaptive strategy proposed in this paper improves the crossover rate and mutation rate of the algorithm in the later stage of evolution.

For a comparison of the 30 optimal individual statistical parameters, as shown in Figure 7, the correlation coefficient of epochs and acc is 0.46, which has a low degree of correlation; the correlation coefficients of nofl, dropout and acc are −0.11, 0.12 respectively, which have a weak correlation; the correlation coefficients of nofd with acc is only −0.041, which is not very relevant.

A detailed comparison of the statistical parameters of the top 30 individuals and a statistical description of the prediction accuracy are shown in Table 3. The search range of the eight

**Table 2.**
Detailed settings of model parameters

| Details of the parameters set | |
|---|---|
| Train_test_split | 0.82 |
| Batch_size | 32 |
| Optimizer | Adam |
| DNA_SIZE_MAX | 8 |
| POP_SIZE | 20 |
| N_GENERATION | 50 |
| $P_{c_1}$ | 0.9 |
| $P_{c_2}$ | 0.6 |
| $P_{m_1}$ | 0.1 |
| $P_{m_2}$ | 0.5 |

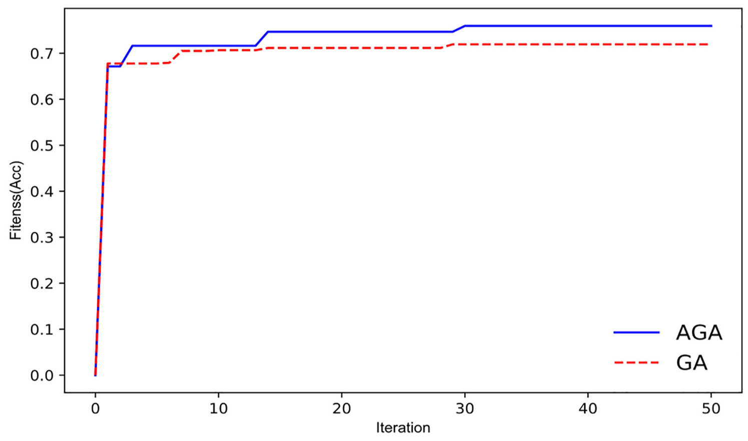

**Figure 6.**
Fitness curve of Google dataset

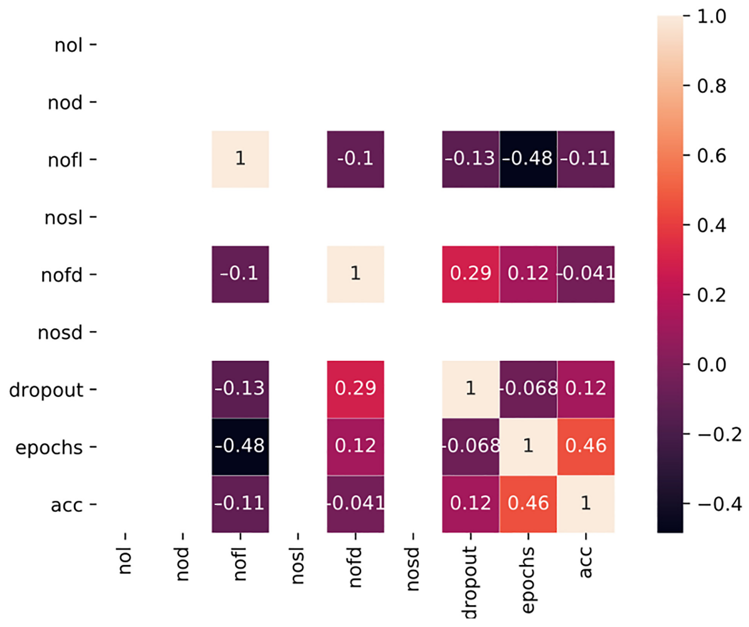

**Figure 7.**
30 optimal combination parameter correlation

**Table 3.**
Statistical description of 30 optimal parameter combination information

| | Nol | Nod | Nofl | Nosl | Nofd | Nosd | Dropout | Epochs | Acc |
|---|---|---|---|---|---|---|---|---|---|
| Count | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| Mean | 1 | 1 | 90.333 | 0 | 82.233 | 0 | 0.31 | 1.967 | 0.707 |
| STD | 0 | 0 | 24.718 | 0 | 21.367 | 0 | 0.135 | 0.718 | 0.021 |
| Min | 1 | 1 | 34 | 0 | 45 | 0 | 0.1 | 1 | 0.676 |
| 25% | 1 | 1 | 77.5 | 0 | 64 | 0 | 0.2 | 1.25 | 0.691 |
| 50% | 1 | 1 | 95 | 0 | 78.5 | 0 | 0.3 | 2 | 0.705 |
| 75% | 1 | 1 | 105 | 0 | 102 | 0 | 0.4 | 2 | 0.716 |
| Max | 1 | 1 | 126 | 0 | 121 | 0 | 0.5 | 3 | 0.761 |

parameters is close to the full search space, and the accuracy obtained is approximately the optimal solution. The maximum accuracy is 0.761, and the average is 0.707.

In this paper, the data of five stocks are used to show how the prediction by AGA-LSTM fit the real data. Figures 8–12 show the comparison between the true value and the predictive value of the closing prices of Google, Amazon, Apple, Microsoft and Tesla, respectively. It is clear from these figures that the algorithm can fit the real data well, and the model produces robust performance and high prediction accuracy.

The experimental results of the AGA-LSTM, GA-LSTM, LSTM, KNN, random forest and SVM used in this paper to predict stock trends are shown in Tables 4–8. Among the forecasts of five different stock data, AGA-LSTM produces the largest ACC, F1-Score and AUC. The AGA-LSTM model achieves the greatest accuracy, 80.3%, in the Apple dataset. AGA-LSTM has the greatest advantage in the prediction of Google and Microsoft data samples, showing its superiority. The AGA-LSTM model is slightly better than the GA-

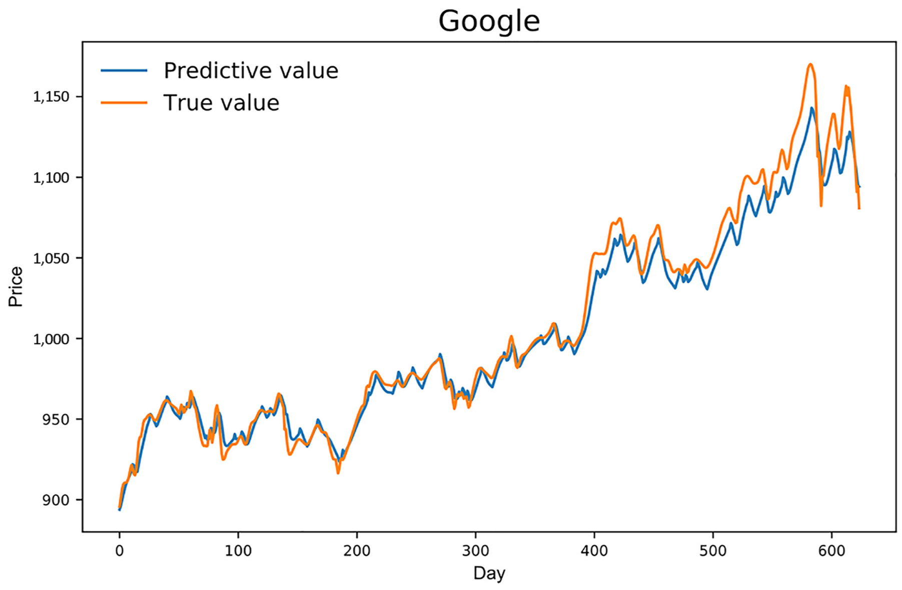

**Figure 8.**
Google stock price forecast results of AGA-LSTM model

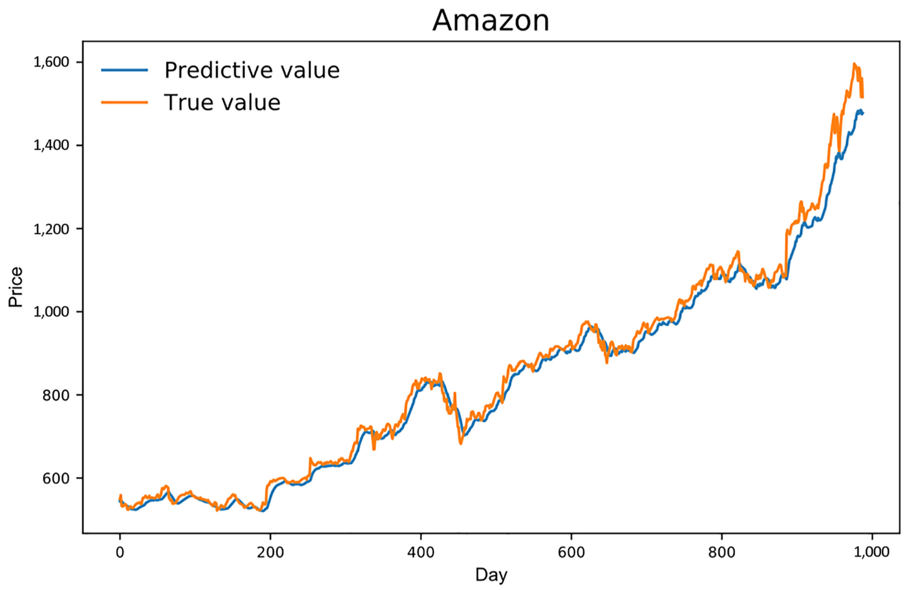

**Figure 9.**
Amazon stock price forecast results of AGA-LSTM model

LSTM model, and they are much better than other models. They have good predictive performance in each stock data, showing the good applicability and scalability of the model (see Figure 13).

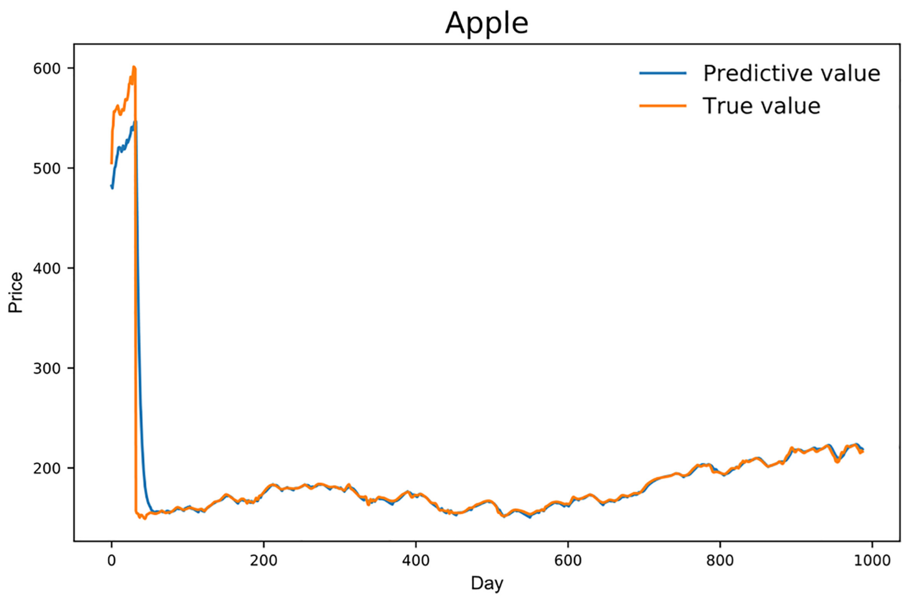

**Figure 10.**
Apple stock price forecast results of AGA-LSTM model

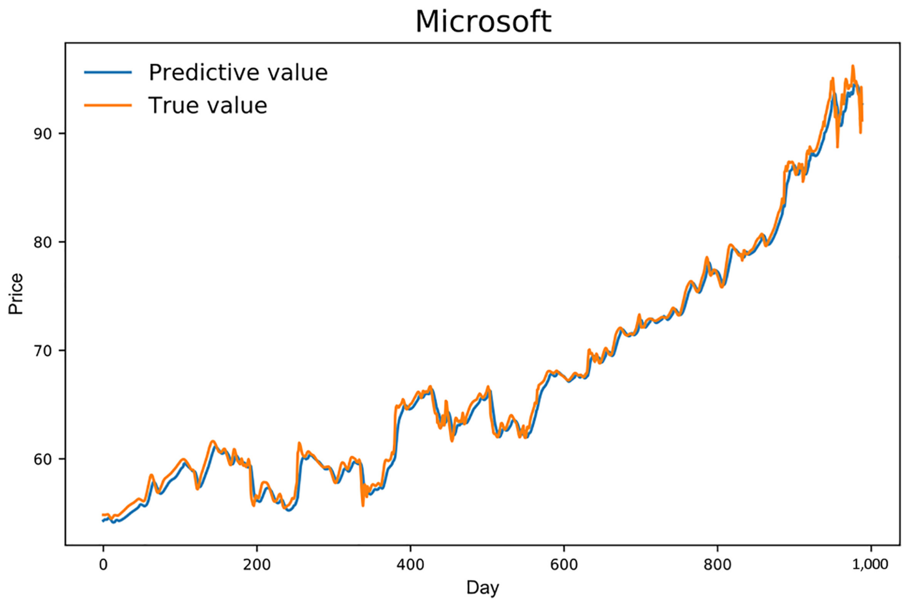

**Figure 11.**
Microsoft stock price forecast results of AGA-LSTM model

To find the appropriate hyperparameters to adapt the machine learning model to a specific problem or dataset, a grid search (GS) or random search (RS) method is usually used. If the time of each model training is set to *t*, GS can be regarded as an exhaustive search, and the model

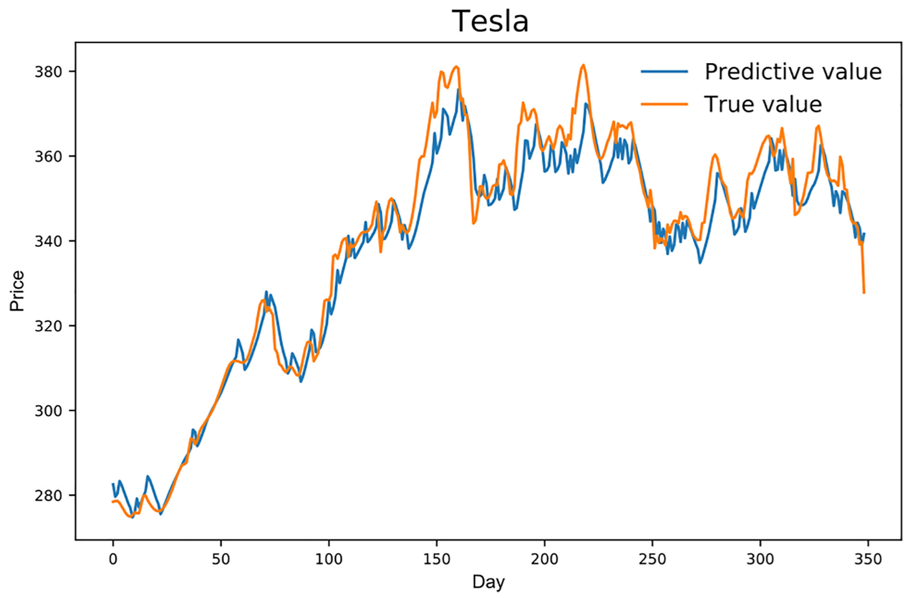

**Figure 12.**
Tesla stock price forecast results of AGA-LSTM model

**Table 4.**
Results of AGA-LSTM model

| AGA-LSTM | Google | Amazon | Apple | Microsoft | Tesla | Average |
|---|---|---|---|---|---|---|
| Accuracy | 0.742 | 0.655 | 0.803 | 0.762 | 0.601 | 0.713 |
| Precision | 0.825 | 0.699 | 0.866 | 0.793 | 0.641 | 0.765 |
| Recall | 0.727 | 0.701 | 0.761 | 0.794 | 0.357 | 0.668 |
| F1-score | 0.773 | 0.700 | 0.810 | 0.794 | 0.459 | 0.707 |
| AUC | 0.746 | 0.646 | 0.808 | 0.756 | 0.589 | 0.709 |

**Table 5.**
Comparison of different models of ACC

| Stock/ACC | AGA-LSTM | GA-LSTM | LSTM | KNN | Random forest | SVM |
|---|---|---|---|---|---|---|
| Google | *0.742* | 0.690 | 0.528 | 0.456 | 0.530 | 0.454 |
| Amazon | *0.655* | 0.653 | 0.515 | 0.487 | 0.504 | 0.464 |
| Apple | *0.803* | 0.795 | 0.521 | 0.492 | 0.475 | 0.545 |
| Microsoft | *0.762* | 0.642 | 0.581 | 0.483 | 0.595 | 0.480 |
| Tesla | *0.601* | 0.581 | 0.486 | 0.487 | 0.522 | 0.496 |

**Table 6.**
Comparison of different models of F1-SCORE

| Stock/F1-SCORE | AGA-LSTM | GA-LSTM | LSTM | KNN | Random forest | SVM |
|---|---|---|---|---|---|---|
| Google | *0.773* | 0.721 | 0.673 | 0.447 | 0.599 | 0.453 |
| Amazon | *0.700* | 0.699 | 0.627 | 0.471 | 0.571 | 0.519 |
| Apple | *0.810* | 0.808 | 0.329 | 0.454 | 0.527 | 0.700 |
| Microsoft | *0.794* | 0.724 | 0.695 | 0.488 | 0.647 | 0.461 |
| Tesla | *0.459* | 0.262 | 0.356 | 0.408 | 0.513 | 0.430 |

training time is $(2 \times 2 \times 96 \times 96 \times 96 \times 96 \times 5 \times 3)\ t = 5096079370t$, this time consuming is very huge. If the RS method is used, in a sufficiently large configuration space, although it can avoid excessive training time in a small area with deficient performance, it does not use the previously well-performing parameter combination. The AGA-LSTM algorithm has strong early local search capabilities and adaptively adjusts the crossover and mutation probability so that it is not easy to fall into the local extreme value in the early stage and fine-tuned to maintain good individuals in the later stage. As shown in Figure 7, a satisfactory solution has been found in the 50 generations of iteration. To obtain a parameter combination close to the optimal solution, the model training time required is only $(50 \times 20) \times t = 1000t$, which is $1.96 \times 10^{-7}$ of the time required by the grid search algorithm.

| Stock/AUC | AGA-LSTM | GA-LSTM | LSTM | KNN | Random forest | SVM |
|---|---|---|---|---|---|---|
| Google | *0.746* | 0.698 | 0.455 | 0.480 | 0.516 | 0.476 |
| Amazon | *0.646* | 0.645 | 0.480 | 0.504 | 0.491 | 0.457 |
| Apple | *0.808* | 0.795 | 0.556 | 0.505 | 0.469 | 0.497 |
| Microsoft | *0.756* | 0.611 | 0.536 | 0.492 | 0.587 | 0.496 |
| Tesla | *0.589* | 0.559 | 0.477 | 0.509 | 0.532 | 0.516 |

**Table 7.** Comparison of different models of AUC

| Average | AGA-LSTM | GA-LSTM | LSTM | KNN | Random forest | SVM |
|---|---|---|---|---|---|---|
| Avg of accuracy | *0.713* | 0.672 | 0.526 | 0.481 | 0.525 | 0.488 |
| Avg of precision | *0.765* | 0.753 | 0.580 | 0.572 | 0.591 | 0.563 |
| Avg of recall | *0.668* | 0.623 | 0.571 | 0.377 | 0.555 | 0.514 |
| Avg of F1-score | *0.707* | 0.643 | 0.536 | 0.454 | 0.571 | 0.513 |
| Avg of AUC | *0.709* | 0.662 | 0.501 | 0.498 | 0.519 | 0.488 |

**Table 8.** Comparison of average evaluation indicators of different models

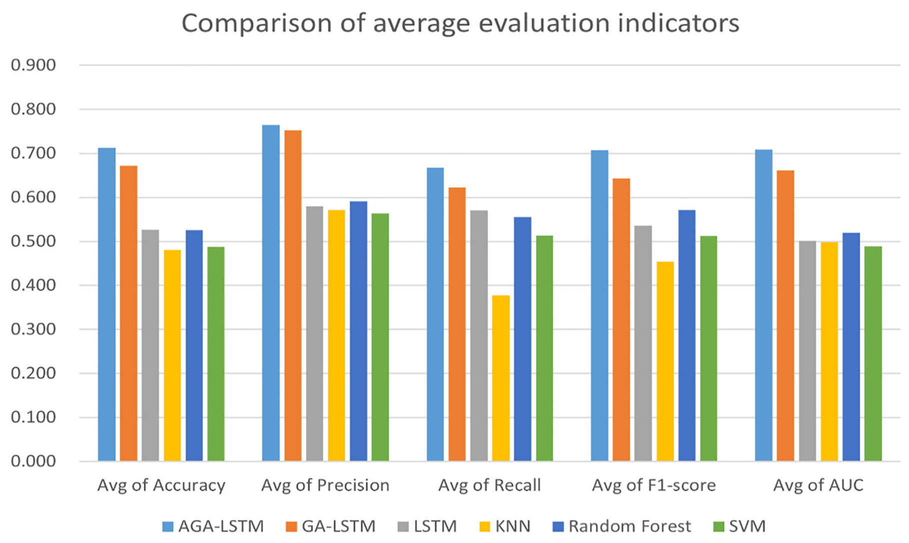

**Figure 13.** Comparison of average evaluation indicators

## 5. Conclusions

In conclusion, we conducted a quantitative analysis of the five stocks. The results demonstrate that our model has better performance than LSTM, and other machine learning models. Our approach includes the use of actual codes to directly find the optimal solution in the solution space; the implement of adaptive crossover and mutation method ensures the acceleration of finding the optimal solution. Therefore, the parameter combinations can achieve better prediction goals. Furthermore, because the difference of each fitness is small, the adaptive algorithm based on individual fitness ranking can be used to improve the speed of finding the optimal solution; using the elitist preservation to ensure the optimal individuals that emerge during the evolutionary process will not lost or destroyed by selection, crossover and mutation operations, improving the ability of the algorithm to converge to the global optimum. What is more, compared with grid search (GS) method and random search (RS) method, the efficiency of parameter tuning is higher. In the future research, the use of more features of big data, e.g. the emotional characteristics of stock market funds and technical indicators, might be an excellent choice for yielding a better prediction accuracy. With the further application of deep learning, multi-source data mining and time series prediction considering the influence of multi-state interleaving will also become the next research direction.

## References

Bani-Hani, D., Khan, N., Alsultan, F., Karanjkar, S. and Nagarur, N. (2018), "Classification of leucocytes using convolutional neural network optimized through genetic algorithm", *Proceedings of the 7th Annual World Conference of the Society for Industrial and Systems Engineering*.

Bao, D. and Yang, Z. (2008), "Intelligent stock trading system by turning point confirming and probabilistic reasoning", *Expert Systems with Applications*, Vol. 34 No. 1, pp. 620-627.

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*, Vol. 12 No. 7, p. e0180944.

Bathla, G. (2020), "Stock Price prediction using LSTM and SVR", *2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pp. 211-214, doi: 10.1109/PDGC50313.2020.9315800.

Cai, X., Zhang, N., Venayagamoorthy, G.K. and Wunsch, D.C. II (2007), "Time series prediction with recurrent neural networks trained by a hybrid PSO -EA algorithm", *Neurocomputing*, Vol. 70 Nos 13-15, pp. 2342-2353.

Cavalcante, R.C., Brasileiro, R.C., Souza, V.L.F., Nobrega, J.P. and Oliveira, A.L.I. (2016), "Computational intelligence and financial markets: a survey and future directions", *Expert Systems with Applications*, Vol. 55, pp. 194-211.

Chen, K., Zhou, Y. and Dai, F. (2015), "A LSTM-based method for stock returns prediction: a case study of China stock market", *2015 IEEE International Conference on Big Data (Big Data)*, pp. 2823-2824, doi: 10.1109/BigData.2015.7364089.

Chung, H. and Shin, K.-S. (2018), "Genetic algorithm-optimized long short-term memory network for stock market prediction", *Sustainability*, Vol. 10, p. 3765. doi: 10.3390/su10103765.

Dai, Y. and Zhang, Y. (2013), "Machine learning in stock price trend forecasting", Stanford University, available at: http://cs229.stanford.edu/proj2013/DaiZhang-MachineLearningInStockPriceTrendForecasting.pdf (Erişim Tarihi 21 (2013): 2021).

Ding, X., Zhang, Y., Liu, T. and Duan, J. (2015), "Deep learning for event-driven stock prediction", *International Conference on Artificial Intelligence*, pp. 2327-2333.

Dong, Y. and Li, D. (2015), *Automatic Speech Recognition*, Springer-Verlag, London, p. 237.

Fischer, T. and Krauss, C. (2018), "Deep learning with long short-term memory networks for financial market predictions", *European Journal of Operational Research*, Vol. 270 No. 2, pp. 654-669.

Gao, Y., Wang, R. and Zhou, E. (2021), "Stock prediction based on optimized LSTM and GRU models", *Scientific Programming*, Vols 1-8, p. 2021.

Ho, Y.C., Sreenivas, S.R. and Vakili, P. (1992), "Ordinal optimization of discrete event dynamic systems", *Discrete Event Dynamic Systems (DEDS)*, Vol. 2 No. 2, pp. 61-88.

Huang, C.L. and Tsai, C.Y. (2009), "A hybrid SOFM-SVR with a filter-based feature selection for stock market forecasting", *Expert Systems with Applications*, Vol. 36 No. 2, pp. 1529-1539.

Junior, F.E.F. and Yen, G.G. (2019), "Particle swarm optimization of deep neural networks architectures for image classification", *Swarm and Evolutionary Computation*, Vol. 49, pp. 62-74.

Kara, Y., Boyacioglu, M.A. and Baykan, Ö.K. (2011), "Predicting direction of stock price index movement using artificial neural networks and support vector machines: the sample of the Istanbul Stock Exchange", *Expert Systems With Applications*, Vol. 38 No. 5, pp. 5311-5319.

Khaidem, L., Saha, S., Basak, S., Kar, S. and Dey, S. (2016), "Predicting the direction of stock market prices using random forest", *arXiv preprint*, arXiv:1605.00003, pp. 1-20.

Kingma, D. P. and Ba, J. (2014), "Adam: a method for stochastic optimization", *International Conference on Learning Representations*, Piscataway, NJ, IEEE Press, p. 1572.

Krauss, C., Do, X.A. and Huck, N. (2016), "Deep neural networks, gradient-boosted trees, random forests: statistical arbitrage on the S&P 500", *European Journal of Operational Research*, Vol. 259 No. 2, pp. 689-702.

Liashchynskyi, P. and Liashchynskyi, P. (2019), "Grid search, random search, genetic algorithm: a big comparison for NAS", available at: https://arxiv.org/abs/1912.06059.

Lin, C.S., Chiu, S.H. and Lin, T.Y. (2012), "Empirical mode decomposition–based least squares support vector regression for foreign exchange rate forecasting", *Economic Modelling*, Vol. 29 No. 6, pp. 2583-2590.

Loussaief, S. and Abdelkrim, A. (2018), "Convolutional neural network hyper-parameters optimization based on genetic algorithms", *International Journal of Advanced Computer Science and Applications*, Vol. 9 No. 10, pp. 252-266, doi: 10.14569/IJACSA.2018.091031.

Ma, X., Tao, Z., Wang, Y., Yu, H. and Wang, Y. (2015), "Long short-term memory neural network for traffic speed prediction using remote microwave sensor data", *Transportation Research C Emerging Technologies*, Vol. 54, pp. 187-197.

Mehtab, S. and Sen, J. (2020), "Stock price prediction using convolutional neural networks on a multivariate timeseries", *Conference: Proceedings of the 3rd National Conference on Machine Learning and Artificial Intelligence*, February, 2020, New Delhi, India, pp. 1-7.

Moghar, A. and Hamiche, M. (2020), "Stock market prediction using LSTM recurrent neural network", *Procedia Computer ScienceNeurocomputing*, Vol. 170, pp. 1168-1173, [CrossRef], doi: 10.1016/j.procs.2020.03.049, networks trained by a hybrid PSO–EA algorithm. Neurocomputing 2007, 70, 2342–2353.

Nelson, D.M.Q., Pereira, A.C.M. and de Oliveira, R.A. (2017), "Stock market's price movement prediction with LSTM neural networks", *International Joint Conference on Neural Networks (IJCNN)*, pp. 1419-1426, doi: 10.1109/IJCNN.2017.7966019.

Pascanu, R., Gulcehre, C., Cho, K. and Bengio, Y. (2013), "How to construct deep recurrent neural networks", *Computer Science, arXiv preprint*, arXiv:1312.6026, pp. 1-11.

Song, G., Zhang, Y., Bao, F. and Qin, C. (2019), "Stock prediction model based on particle swarm optimization LSTM", *Journal of Beijing University of Aeronautics and Astronautics*, Vol. 45 No. 12, pp. 2533-2542, (in Chinese).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. (2014), "Dropout: a simple way to prevent neural networks from overfitting", *Journal of Machine Learning Research*, Vol. 15 No. 1, pp. 1929-1958.

Staffini, A. (2022), "Stock price forecasting by a deep convolutional generative adversarial network", *Frontiers in Artificial Intelligence*, Vol. 5, 837596, doi: 10.3389/frai.2022.837596.

Wang, B., Huang, H. and Wang, X. (2012), "A novel text mining approach to financial time series forecasting", *Neurocomputing*, Vol. 83, pp. 136-145.

Wei, X. and You, Z.N. (2019), "Neural network hyperparameter tuning based on improved genetic algorithm", *Proceedings of the 8th International Conference on Computing and Pattern Recognition*, pp. 17-24, doi: 10.1145/3373509.3373554.

Wicaksono, A.S. and Supianto, A.A. (2018), "Hyper parameter optimization using genetic algorithm on machine learning methods for online news popularity prediction", *International Journal of Advanced Computer Science and Applications*, Vol. 9 No. 12, pp. 263-267, doi: 10.14569/IJACSA.2018.091238.

Xie, Q., Cheng, G. and Xu, X. (2019), "Research based on stock predicting model of neural networks ensemble learning", *Computer Engineering and Applications*, Vol. 55 No. 8, pp. 238-243.

Zhang, K., Zhong, G., Dong, J., Wang, S. and Wang, Y. (2019), "Stock market prediction based on generative adversarial network", *Procedia Computer Science*, Vol. 147, pp. 400-406.

**Corresponding author**
Huan Li can be contacted at: lihuan@dgut.edu.cn