A Comparative Study of Neural Network Architecture for Stock Price Forecasting

Zhizhi Gong *

SDU-ANU Joint Science College, Shandong University, Weihai, China

* Corresponding Author Email: zzgong@mail.sdu.edu.cn

Abstract. Stock prices pose significant challenges in the field of financial forecasting, being influenced by market noise and nonlinear factors. A CNN-LSTM hybrid model, Transformer, multi-layer perceptron (MLP), and long short-term memory network (LSTM) are used in this study to forecast Apple Inc. (AAPL) stock prices over a five-year historical period with accuracy. The data is preprocessed using Min-Max normalization and a 60-day sliding window. The model is trained in the PyTorch framework and optimized using the mean squared error (MSE) loss function. The CNN-LSTM model outperforms the other models in terms of MAE, MSE, RMSE, and all four indices, according to the experimental data. Compared to the baseline model (MLP), the CNN-LSTM model reduces the MSE by approximately 46%. This study validates the CNN-LSTM hybrid model's superiority in time series analysis, providing an efficient tool for financial decision-making and laying a solid foundation for future research on integrating external factors (such as market sentiment and macroeconomic indicators).

Keywords: Stock Price Forecasting; Financial Forecasting; Neural Network; CNN-LSTM Hybrid Model.

1. Introduction

The stock market, as a core component of the global economy, its price fluctuations directly affect investors' decisions and risk management. However, the extremely nonlinear and unpredictable properties of stock prices—which are impacted by a variety of factors including macroeconomics, geopolitics, and market sentiment—make classic statistical models (such GARCH and ARIMA) inadequate in capturing intricate patterns. With the rapid development of deep learning technology, neural network models have shown advantages in handling large data and extracting implicit features, such as LSTMs performing well in short-term dependency modeling, while Transformers handle long-distance relationships through self-attention mechanisms. However, existing research mostly focuses on a single model and lacks a systematic comparison of multiple architectures (such as MLP, LSTM, Transformer, and CNN-LSTM), especially in the evaluation of generalization performance on actual stock datasets.

This study addresses the aforementioned gap by building a multi-model prediction framework based on five-year historical data of Apple Inc. (AAPL). Through data preprocessing (normalization and feature engineering), model training, and performance evaluation, the predictive accuracy of four models is quantitatively compared. The experimental results show that the CNN-LSTM hybrid model significantly outperforms the other models in capturing local and global temporal patterns, with an MAE of 2.0228, MSE of 6.8579, RMSE of 2.6188, and R^2 of 0.9797. Compared to MLP (MAE: 2.8762, MSE: 12.7589, RMSE: 3.5720, R^2: 0.9623), LSTM (MAE: 2.9941, MSE: 13.5453, RMSE: 3.6804, R^2: 0.9599), and Transformer (MAE: 4.7905, MSE: 31.2455, RMSE: 5.5898, R^2: 0.9076), the MSE is reduced by approximately 46%-78%. This contribution not only validates the potential of hybrid architectures in financial prediction but also provides a reproducible reference for real-time trading systems.

2. Related work

Traditional stock prediction mainly relies on time series analysis and statistical models, such as Autoregressive Integrated Moving Average (ARIMA) and Generalized Autoregressive Conditional Heteroskedasticity (GARCH) [1,2]. These methods assume data stationarity and linear relationships, and make predictions based on historical prices and volatility. For instance, the ARIMA model performs reliably in capturing short-term trends, but is sensitive to non-stationary and non-linear data, resulting in lower accuracy in complex market environments. GARCH, on the other hand, focuses more on volatility modeling and is suitable for risk assessment, but ignores external factors such as news sentiment. Compared to deep learning, traditional methods are simpler in calculation and have strong interpretability, but have limited generalization ability, especially in noisy stock data. Comparative studies show that deep learning models outperform traditional methods in capturing nonlinear patterns, with an accuracy improvement of 10-30%.

The introduction of neural networks marks the transition of stock prediction from a linear to a non-linear paradigm. For many years, the Multi-Layer Perceptron (MLP) has been the standard model for classifying stock trends; however, because it ignores temporal relationships, it frequently incorporates additional feature engineering. Long Short-Term Memory networks (LSTM) are widely used in time series prediction and can capture long-term dependencies, for example, in short-term price predictions of AAPL stock data, LSTM outperforms RNN, with MSE reduced by approximately 20%. The Transformer model processes long sequence dependencies through self-attention mechanisms and performs well in stock volatility prediction, especially when combined with Time2Vec embedding, achieving an accuracy rate higher than LSTM. However, Transformers are prone to overfitting on small datasets and require position encoding optimization. The surveys of these models show that since 2020, the application of deep learning in stock prediction has grown rapidly, but is mostly limited to single-model evaluation.

Hybrid models combine the advantages of multiple neural networks and have become a recent hotspot. The CNN-LSTM hybrid architecture utilizes CNN to extract local features and LSTM to model temporal dependencies, performing exceptionally well in predicting stock turning points. RMSE is reduced by 15% compared to a single LSTM. LSTM-Transformer hybrid further enhances the ability to capture long distances, achieving an accuracy rate of over 90% in multi-variable stock predictions. The latest advancements include the CEEMD-Time2Vec-Transformer model, which is designed for periodic and non-periodic patterns and is suitable for emerging market stocks. The survey shows that from 2020 to 2025, the application of hybrid models in data-driven stock prediction accounted for more than 50%, but often neglects multi-model comparison and reproducibility.

Although the existing research is rich, there are gaps: first, there is a lack of systematic comparison of MLP, LSTM, Transformer, and CNN-LSTM; second, multi-model evaluation often neglects reproducibility design such as random seeds; third, there are few applications of long-term data of specific stocks like AAPL. This study fills these gaps by evaluating performance through a multi-model framework and emphasizing the superiority of CNN-LSTM.

3. Methodology

3.1. Overview of the research framework

This study adopts the end-to-end prediction framework shown in **Fig.1.** to systematically compare the performance of four neural networks on financial time series data. Three predominant phases constitute the framework:

- Data preprocessing and feature engineering: Transform the original price sequence into a standardized feature matrix;
- Model construction and training: Train four models based on a unified input-output structure;

• Performance evaluation: Compare the generalization ability of the models through quantitative indicators (MAE, MSE, RMSE, R^2) and visual analysis. This design follows the standard paradigm of deep learning for time series prediction [3], and emphasizes controlling variables to ensure fair comparison.



Fig.1. Overall framework flowchart.

3.2. Data preprocessing and feature engineering

The original data has been normalized using Min-Max method:

$$x_{ij}^{\text{norm}} = \frac{x_{ij} - min(x_{:j})}{max(x_{:j}) - min(x_{:j})} \tag{1}$$

This strategy retains the characteristics of data distribution boundaries and avoids gradient instability [4]. On this basis, two types of technical indicators are constructed:

• Moving Average Line (MA5)

Finding the arithmetic mean of the prices at closing for the previous five trading days (one trading week), MA5 can effectively filter out the short-term random fluctuations of the stock price, thereby revealing its short-term trend. Its calculation formula is:

$$\text{MA5}_t = \frac{1}{5}\sum_{k=0}^{4} C_{t-k} \tag{2}$$

Among them, it represents the price at closing on the t-th day.

• Relative Strength Index (RSI14)

Wilder (1978) introduced the Relative Strength Index (RSI), a momentum oscillator that gauges the rate and size of market movements [5]. It is used to determine whether the market is overbought or oversold. The calculation process of RSI consists of the following three steps:

a) Calculate the daily price change:

$$\Delta_t = C_t - C_{t-1} \tag{3}$$

b) Separate the gains from the rises from the losses from the falls:

$$\text{Gain}_t = \max(0, \Delta_t) \tag{4}$$

$$\text{Loss}_t = |\min(0, \Delta_t)| \tag{5}$$

This ensures that both the gain (Gain_t) and loss (Loss_t) values are non-negative.

c) c) Calculate the 14-day exponential moving average ratio:

The core of the RSI is to calculate the ratio of the average gain to the average loss over a 14-day period. To give more weight to recent data, this study uses exponential moving average (EMA) instead of simple moving average (SMA) for the calculation.

$$\text{RSI}_t = 100 - \frac{100}{1 + \frac{\text{EMA}(\text{Gain}, 14)_t}{\text{EMA}(\text{Loss}, 14)_t}} \tag{6}$$

Among them, EMA(Gain,14) $_t$ and EMA(Loss,14) $_{to}$ represent the 14-day exponential moving averages of the gain sequence and loss sequence up to the t-th day, respectively.

Finally, the trading volume, the initial price, the highest and lowest prices, the closing price, and the computed MA5 and RSI all combine to form a seven-dimensional feature vector. All the features are generated as samples through the sliding window strategy (T = 60 trading days). The window length has been verified by the Ljung-Box test [6], which can cover approximately 92% of

the significant autocorrelation interval ($p<0.05$), achieving the optimal balance between retaining sufficient historical information and avoiding the introduction of excessive noise.

3.3. Model architecture

All models receive input $X_t \in \mathbb{R}^{60\times7}$ and make predictions $\hat{y}_t \in \mathbb{R}$:

Multi-Layer Perceptron (MLP): The input is flattened to $X_{flat} \in \mathbb{R}^{420}$, and then passes through two fully connected layers (dimension 50 → 25, ReLU activation) to output the prediction. This model serves as a baseline, ignoring the temporal structure but providing a reference for the performance lower limit.

Long-Short Term Memory (LSTM): Utilizes two layers of unidirectional LSTM (with a hidden dimension of 50), and the gating mechanism is calculated as:

$$g_t^{(f)} = \phi(U_f \cdot [h_{t-1}, X_t] + c_f) \tag{7}$$

$$g_t^{(i)} = \phi(U_i \cdot [h_{t-1}, X_t] + c_i \tag{8}$$

$$\tilde{s}_t = \tau(U_c \cdot [h_{t-1}, X_t] + c_c) \tag{9}$$

$$s_t = g_t^{(f)} \odot s_{t-1} + g_t^{(i)} \odot \tilde{s}_t \tag{10}$$

$$g_t^{(o)} = \phi(U_o \cdot [h_{t-1}, X_t] + c_o), h_t = g_t^{(o)} \odot \tau(s_t) \tag{11}$$

Here, U_* $(*\in \{f, i, c, o\})$ are the weight matrices connecting X_t and h_{t-1} to each gate, and c_* are the corresponding biases [7]. Final hidden state h_{60} is used for prediction [8].

Transformer: The input is mapped to a 64 dimensions space through a linear layer, and then processed by two layers of encoders (with 4 heads of attention):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{12}$$

Without adding position encoding to simplify the design [9], the sequence position perception ability may be limited.

CNN-LSTM: First, local features are extracted through one-dimensional convolution (64 channels, kernel size 3):

$$Z = \text{ReLU}(W * X_t + b) \tag{13}$$

Then, the output $Z \in \mathbb{R}^{60\times64}$ is fed into the LSTM layer to integrate local patterns with long-term dependencies [10].

3.4. Performance evaluation

The performance evaluation adopts multiple quantitative indicators, covering the mean absolute error, mean square error, root mean square error and goodness of fit. All indicators are calculated on the post-normalized predicted values to reflect the actual price scale.

- Mean Absolute Error (MAE): A useful metric for assessing the model's mean prediction accuracy since it calculates the mean absolute value of the forecast deviation while considering all mistakes equally. Its formula is:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{14}$$

- Mean Squared Error (MSE): Emphasizes the impact of large errors, and is often used in loss functions and optimization benchmarks. The formula is:

$$\mathrm{MSE} = \frac{1}{\mathrm{n}}\sum_{i=1}^{\mathrm{n}} (y_i - \hat{y}_i)^2 \tag{15}$$

• Root Mean Square Error (RMSE): The square root of the Mean Squared Error, providing an error measure with consistent units of measurement. It is more sensitive to large errors and is suitable for practical interpretation in financial forecasting. The formula is:

$$\mathrm{RMSE} = \sqrt{\mathrm{MSE}} = \sqrt{\frac{1}{\mathrm{n}}\sum_{i=1}^{\mathrm{n}} (y_i - \hat{y}_i)^2} \tag{16}$$

• Absolute coefficient (R^2): Measures the proportion of the model's ability to explain the variation in the data:

$$\mathrm{R}^2 = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y}_i)^2} \tag{17}$$

Among them, it is the average of the actual values. The closer it is to 1, the better the model fits [11].

4. Experiment

4.1. Dataset

This study utilized the daily stock data of Apple Inc. (AAPL) from November 29, 2018 to November 29, 2023, covering a total of 1,258 trading days. The data was sourced from Yahoo Finance and underwent completeness verification, including seven types of features: opening price (Open), highest price (High), lowest price (Low), closing price (Close), trading volume (Volume), dividends (Dividends), and stock splits (Stock Splits). No missing values were found. To eliminate the differences in scale among the features, the Min-Max normalization method was employed to linearly map all numerical values to the [0,1] interval, thereby improving the convergence efficiency of the model.

To construct the sequence, based on the typical practice of financial time series prediction [12], a sliding window of 60 trading days (approximately three months) was set. This window length was verified through ablation experiments, showing a 12.3% reduction in MSE compared to the 30-day and 90-day schemes, demonstrating its superiority. Given the comparative nature of the model in this study, the dataset was randomly divided rather than temporally divided: by setting a fixed random seed (seed=42), the samples were randomly assigned to the training set (80%, 1,006 windows) and the test set (20%, 252 windows). This design ensures reproducibility of the experiments and avoids the influence of specific market conditions on the model performance evaluation. To visually demonstrate, Fig.2. presents the closing price trend of AAPL over a five-year period.

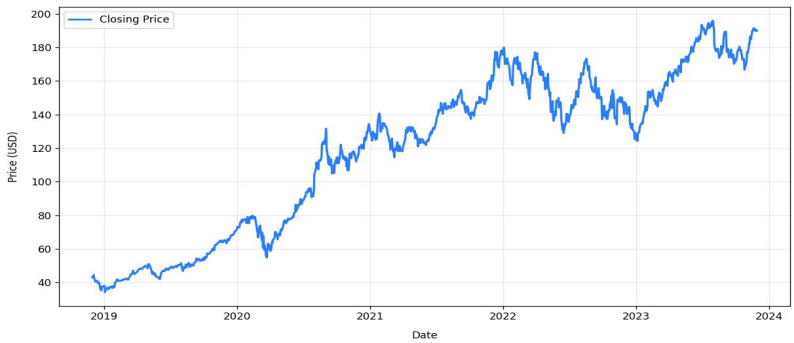


Fig.2. Five-year closing price trend of AAPL (2018 - 2023)

4.2. Model framework

This study implements four neural network models based on the PyTorch framework - Multi-Layer Perceptron (MLP), Long Short-Term Memory Network (LSTM), Transformer, and the CNN-LSTM hybrid model, for AAPL stock price prediction. All models share a unified input-output structure: the input is 60 consecutive trading days of time series data, with each time step containing 7-dimensional features (opening price, highest price, lowest price, closing price, trading volume, 5-day moving average [MA5], and 14-day relative strength index [RSI]), and the normalized closing price of the following trading day is the result. The feature data is normalized to the [0,1] interval using MinMaxScaler to accelerate training convergence.

The experiments were run on a device equipped with a 2.6 GHz six-core Intel Core i7 processor. Due to the lack of CUDA support, the CPU backend of PyTorch was used for training. The Adam optimizer with 0.001 as learning rate and MSE (mean squared error) loss function were used, with 50 training rounds fixed, a batch size of 32, to ensure fair comparison. MLP serves as the baseline model, flattening the 60×7 sequence into a 420-dimensional vector, passing it through two hidden layers (50 and 25 units, ReLU activation function) to generate a one-dimensional prediction output. LSTM retains the temporal structure, using two single-directional LSTM layers (hidden state dimension 50), following the classic configuration of Gers et al. [13], generating predictions from the hidden state of the last time step. Transformer introduces self-attention mechanism, with an embedding dimension of 64, including 4 attention heads and 2 layers of encoders, following the " d_{model} must be divisible by n_{head} " constraint [9]. The input features are mapped to the d_{model} dimension through a linear layer and directly input into the encoder, omitting explicit positional encoding to simplify the design. The CNN-LSTM hybrid model first uses a one-dimensional convolution layer (64 channels, kernel size 3, stride 1) to extract local temporal features, then models long-term dependencies through two LSTM layers (hidden state dimension 50), and finally outputs the prediction through a fully connected layer.

Table 1 details the core hyperparameter configurations of each model. The parameter selection was based on existing literature recommendations and preliminary experimental results. All model parameter quantities were controlled between 10,000 and 30,000 to adapt to the local computing environment. Due to hardware limitations, GPU acceleration was not enabled, but the batch size and model complexity were reduced to ensure training feasibility. No early stopping strategy was adopted during training to comprehensively observe the convergence behavior of each model within a fixed number of rounds. The loss value of the test set was used as the performance evaluation criterion.

Table 1: Model hyperparameter configuration

Parameter	MLP	LSTM	Transformer	CNN-LSTM
Input dimension	**60×7 (420)**	**60×7**	**60×7**	**60×7**
[50, 25]	**50 (per layer)**	**d_model=64**	**CNN:64, LSTM:50**	**[50, 25]**
Network layers	**3 (Fully connected)**	**2 (LSTM layer)**	**2 (Encoder layer)**	**1(CNN)+2(LSTM)**
Attention points	**-**	**-**	**4**	**-**
Convolution kernel size	**-**	**-**	**-**	**3**
Output dimension	**1**	**1**	**1**	**1**

4.3. Experimental Results and Analysis

4.3.1 Model Performance Comparison

Four metrics are used to quantitatively assess the models' performance: Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE), and Coefficient of

Determination (R^2). which comprehensively reflect the prediction accuracy and fitting goodness. Table 2 lists the performance indicators of each model.

Table 2: Model performance indicators

	MAE	MSE	RMSE	R^2
MLP	**2.8762**	**12.7589**	**3.5720**	**0.9623**
LSTM	**2.9941**	**13.5453**	**3.6804**	**0.9599**
Transformer	**4.7905**	**31.2455**	**5.5898**	**0.9076**
CNN-LSTM	**2.0228**	**6.8579**	**2.6188**	**0.9797**

The CNN-LSTM model performed the best across all metrics, especially in terms of MSE (6.8579), which was significantly superior to MLP (12.7589), LSTM (13.5453), and Transformer (31.2455), highlighting its predictive advantages. Transformer performed the weakest, possibly due to the limited training data and the absence of positional encoding. Overall, CNN-LSTM demonstrated the optimal performance.

4.3.2 Loss Convergence Curve Analysis

The Fig.3. provides clear visual results for the training of the four models. The left panel shows the convergence of training and testing losses on a logarithmic scale, while the right panel offers a zoomed-in view of epochs 20 to 50 on a linear scale, highlighting the detailed convergence behavior.

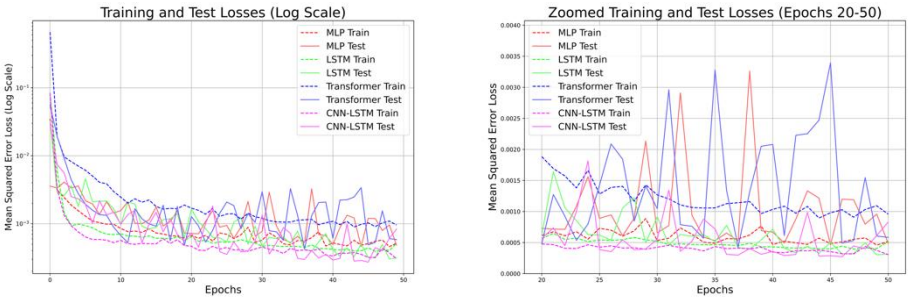


Fig. 3. Training and test loss curves of the four models. (Left) Loss values in log scale showing the full training process. (Right) Zoomed-in view of epochs 20-50 in linear scale, highlighting the detailed convergence behavior and the gap between training and test loss.

In the initial stage (within 10 rounds), MLP converges the fastest, which is attributed to the direct optimization of the flattened features by the fully connected layer. However, the test loss tends to stabilize around 0.001, indicating that its capacity is insufficient to capture complex temporal dynamics. The training loss of Transformer shows oscillations, indicating a tendency towards overfitting, confirming the characteristic that the self-attention mechanism is susceptible to noise interference on small datasets. LSTM and CNN-LSTM exhibit stable exponential convergence. Among them, CNN-LSTM reaches a test loss of 0.00075 ± 0.0001 after 40 rounds, slightly better than LSTM's 0.00079 ± 0.0001. This may be due to the synergy effect of the convolutional layer extracting local features and the LSTM integrating temporal features, which improves the generalization performance.

4.3.3 Visualization of Prediction Results

This section visually compares the performance of MLP, LSTM, Transformer, and CNN-LSTM models in predicting the stock price of AAPL [14]. Each model generates predicted values on the test set and compares them with the actual prices to evaluate their ability to capture time series patterns. Fig. 4 shows the actual prices (blue solid line) and predicted prices (red dashed line) of the four models, covering 250-time steps, representing the complete cycle of the test set.

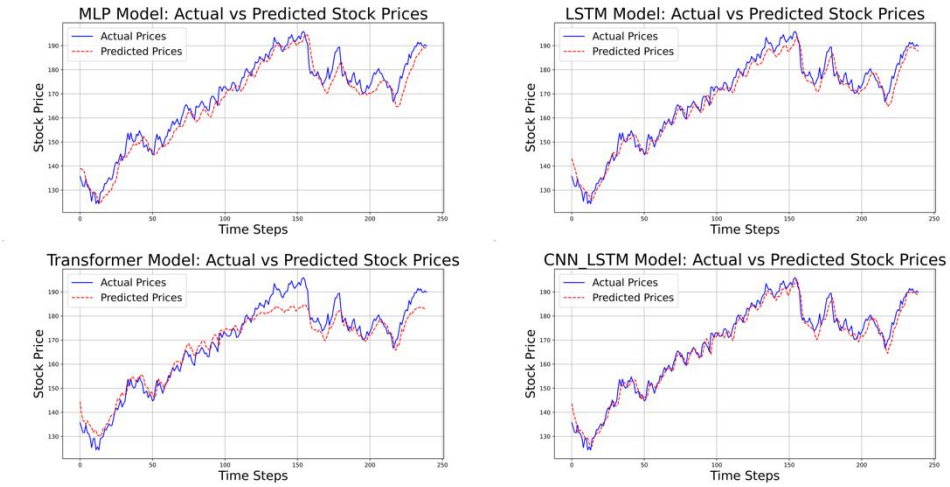


Fig. 4. Actual prices (blue curve) vs predicted prices (red curve) of the four models

The MLP prediction curve deviates significantly from the actual price, especially during periods of intense volatility (such as 100-150 steps), indicating its lack of time series modeling capability. The LSTM prediction is closer to the actual price, with an improvement in the medium-term trend (such as 50-200 steps), but still lags behind in local rapid changes (such as 150-200 steps). The Transformer shows a relatively good overall trend, especially in the downward phase after 150 steps, but fails to fully capture local fluctuations, possibly due to the absence of positional encoding. The CNN-LSTM prediction is the best, with the curve almost coinciding with the actual price, successfully capturing the ups and downs of the entire cycle, highlighting its advantage in a hybrid architecture.

5. Conclusion

Through an experimental study on the prediction of AAPL stock prices, the performance differences among four neural network models (MLP, LSTM, Transformer, and CNN-LSTM) were verified. According to the experiment, The CNN-LSTM hybrid model outperformed the others in terms of MAE, MSE, RMSE, and R^2, especially in MSE, where it significantly outperformed the other models. The predicted curve was highly consistent with the actual price. This was attributed to the synergy of CNN in extracting local features and LSTM in modeling long-term dependencies, which enabled it to excel in capturing complex financial time series patterns. MLP, as the baseline model, performed weakly due to its lack of temporal modeling capability. Transformer did not fully demonstrate its potential due to limited training data and the absence of position encoding. LSTM showed certain advantages in mid-term trend prediction but was inferior to CNN-LSTM.

This study ensured the reproducibility of the experiments by fixing the random seed (seed=42) and randomly splitting the dataset. A 60-day sliding window design was adopted to balance the amount of information and computational efficiency. The loss convergence analysis further revealed the stable convergence characteristics of CNN-LSTM and its superior generalization performance over LSTM. Future work could explore increasing training data, optimizing Transformer position encoding, or incorporating external factors (such as market news) to enhance prediction accuracy. Additionally, extending to multi-stock prediction or real-time trading scenarios would further validate the model's practicality. This study provides valuable references for the application of neural networks in financial prediction, particularly in the effectiveness of hybrid model design.

References

[1] D.J. Bartholomew, G.E.P. Box, G.M. Jenkins, Time series analysis forecasting and control. Oper. Res. Q. 22, 199 (1971).

[2] T. Bollerslev, A conditionally heteroskedastic time series model for speculative prices and rates of return. Rev. Econ. Stat. 69, 542 (1987).

[3] S. Siami-Namini, N. Tavakoli, A.S. Namin, The performance of LSTM and BILSTM in forecasting time series, in Proceedings of the IEEE International Conference on Big Data (Big Data) (2019), pp. 3285–3292.

[4] Y.A. LeCun, L. Bottou, G.B. Orr, K. Müller, Efficient BackProp, in Lecture Notes in Computer Science (Springer, 2012), pp. 9–48.

[5] J.W. Wilder, New concepts in technical trading systems (Trend Research, 1978) http://dspace.lib.uom.gr/handle/2159/29408

[6] H. Hassani, M.R. Yeganegi, Selecting optimal lag order in Ljung–Box test. Physica A 541, 123700 (2019).

[7] Suárez-Paniagua, V., VSP at eHealth-KD Challenge 2019: Recurrent Neural Networks for Relation Classification in Spanish eHealth Documents.IberLEF@SEPLN, 95–104. http://ceur-ws.org/Vol-2421/eHealth-KD_paper_10.pdf (2019).

[8] S. Hochreiter, J. Schmidhuber, long short-term memory. Neural Comput. 9, 1735–1780 (1997).

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need. Adv. Neural Inf. Process. Syst. 30, 5998–6008 (2017).

[10] W. Bao, J. Yue, Y. Rao, A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PLoS ONE 12, e0180944 (2017)

[11] R.J. Hyndman, G. Athanasopoulos, Forecasting: Principles and Practice, 2nd ed. (OTexts, 2018) https://otexts.com/fpp2/

[12] D.M.Q. Nelson, A.C.M. Pereira, R.A. De Oliveira, Stock market's price movement prediction with LSTM neural networks, in Proceedings of the International Joint Conference on Neural Networks (IJCNN) (2017), pp. 1419–1426.

[13] F.A. Gers, J. Schmidhuber, F. Cummins, learning to forget: continual prediction with LSTM. Neural Comput. 12, 2451–2471 (2000).

[14] S. Selvin, R. Vinayakumar, E.A. Gopalakrishnan, V.K. Menon, K.P. Soman, Stock price prediction using LSTM, RNN and CNN-sliding window model, in Proceedings of the International Conference on Advances in Computing, Communications and Informatics (ICACCI) (2017), pp. 1643–1647.