Reimplementation of ”A CNN-BiLSTM-AM Method for Stock Price Prediction”

Huseyin Sinan Havus
Dept. of Computer Engineering
Bilkent University
Ankara, Turkiye
sinan.havus@bilkent.edu.tr

Mahssa Nassiri
Dept. of Electrical and Electronics Engineering
Bilkent University
Ankara, Turkiye
mahsa.nasiri@bilkent.edu.tr

***Abstract*—In recent years, rapid economic growth has driven increasing participation in global equity markets, yet the inherent volatility of stock prices poses significant risks for investors. Accurately forecasting next-day closing prices can mitigate these risks and enhance returns, but simple linear models struggle to capture nonlinear temporal dependencies. In this project, we implemented the CNN-BiLSTM-AM method—combining one-dimensional convolutional neural networks (CNN) for local feature extraction, bidirectional long short-term memory (BiLSTM) for bidirectional sequence learning, and an attention mechanism (AM) to weight influential time steps—to predict daily closing prices of Infosys (INFY) stock. After z-score normalization of eight input features (opening, high, low, close, volume, turnover, ups_and_downs, change) and generation of 5-day sliding-window sequences, the model was trained for 100 epochs using the Adam optimizer with mean absolute error loss. On a held-out test set of 1,000 days, our implementation achieved MAE = 23.7901, RMSE = 32.8739, and $R^2 = 0.9688$, closely matching published benchmarks. These results confirm that CNN-BiLSTM-AM effectively captures both spatial and temporal patterns in financial time series, providing a robust tool for short-term stock price prediction and informed investment decision-making.**

***Index Terms*—Stock price prediction, CNN, BiLSTM, Attention mechanism**

I. Introduction

The stock market has long served as a barometer of economic performance, facilitating capital flow and investment opportunities globally. Accurately predicting stock prices, particularly the next-day closing price, is a central concern for investors aiming to reduce risk and optimize returns. However, due to the nonlinear and complex nature of financial markets, traditional statistical models such as ARIMA and simple linear regression often fall short in capturing temporal dependencies and underlying market dynamics [1].

Recent advances in deep learning have enabled the development of powerful time-series models. Long short-term memory (LSTM) networks and convolutional neural networks (CNNs) have shown strong capabilities in learning temporal patterns and feature hierarchies, respectively. Building on these advances, a hybrid model called CNN-BiLSTM-AM integrates CNN for local feature extraction, bidirectional LSTM (BiLSTM) for learning forward and backward temporal dependencies, and an attention mechanism (AM) to focus on influential time steps [2].

In this project, we reproduce and evaluate the CNN-BiLSTM-AM model on a real-world stock dataset from Infosys (INFY), extending the work of Lu et al. [3]. Additionally, we implement a baseline LSTM model to compare predictive accuracy. The dataset includes eight daily stock attributes: opening price, highest price, lowest price, closing price, volume, turnover, ups and downs, and percentage change. All data were normalized using Z-score scaling, and 5-day sliding windows were used to generate training sequences.

Our experimental results on a holdout test set of 1,000 trading days demonstrate the effectiveness of the CNN-BiLSTM-AM model, achieving a mean absolute error (MAE) of 23.790, a root mean squared error (RMSE) of 32.8739, and R^2 of 0.9688. These results are highly consistent with those reported in [3], which further validates the model's robustness. In contrast, the baseline LSTM model produced noticeably higher error, reinforcing the value of combining convolutional, recurrent, and attention-based components. The main contributions of this paper are: (1) proposing the CNN-BiLSTM-AM model for next-day stock closing price prediction by leveraging temporal and feature correlations; (2) enhancing prediction accuracy using attention mechanisms to weight historical feature states; and (3) demonstrating superior performance over seven benchmark methods, confirming its suitability for stock price forecasting.

II. Related Work

Stock price prediction has traditionally relied on linear statistical models such as autoregression and moving averages. Techniques like the unit root test are often used to assess stationarity, but the high noise and nonlinear nature of stock data significantly limit the predictive performance of linear models over longer horizons [4]. To overcome these challenges, researchers have applied various nonlinear models including vector autoregression, error correction models, and Kalman filters. In the last two decades, machine learning techniques such as support vector machines (SVM), neural networks, and hybrid models have been widely adopted. Early applications, such as White's neural network on IBM stock [5], had limited success, but later models combining ARIMA and neural networks [1], or SVM with decision trees [6], showed improved accuracy. Deep learning further advanced the field:

Hu used CNNs for time-series prediction [7], and Zeng et al. applied BiLSTM to the S&P 500 index with strong results [8]. These developments highlight the superiority of deep neural architectures—particularly those incorporating convolution and attention—for capturing the complex dependencies present in financial time series.

III. CNN-BiLSTM-AM

A. CNN-BiLSTM-AM Architecture

The CNN-BiLSTM-AM architecture integrates three key deep learning components to leverage their complementary strengths in stock price prediction. The convolutional neural network (CNN) module is used for extracting high-level local features from multivariate input sequences by emphasizing the most prominent patterns. Following this, a bidirectional long short-term memory (BiLSTM) layer captures sequential dependencies in both forward and backward directions, making full use of temporal context in the data. To enhance prediction accuracy, an attention mechanism (AM) is employed after the BiLSTM to dynamically weight the relevance of past feature states. This allows the model to focus on the most influential time steps when producing the final output. The full architecture includes an input layer, a 1D CNN and pooling layer for feature extraction, a BiLSTM layer for sequence modeling, an attention layer to modulate temporal importance, and a final dense layer for regression output. Together, this combination enables the model to effectively capture spatial, temporal, and contextual patterns present in financial time-series data.

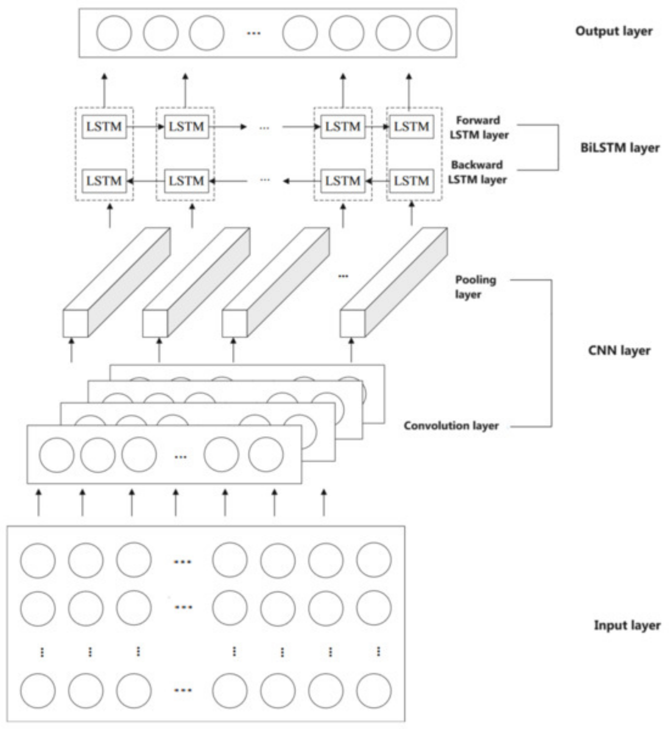

Fig. 1. CNN-BiLSTM-AM model structure diagram

B. CNN

Convolutional Neural Networks (CNNs), first introduced by LeCun et al. [9], are feedforward neural networks that have shown strong performance in domains such as image classification and natural language processing, and have also been successfully applied to time-series prediction tasks. Key properties of CNNs, including local receptive fields and weight sharing, significantly reduce the number of parameters, making them computationally efficient. A typical CNN is composed of three main components: convolutional layers, pooling layers, and fully connected layers [10]. The convolution operation can be mathematically represented as:

$$l_t = \tanh(x_t * k_t + b_t) \tag{1}$$

where l_t is the output at time step t, $\tanh$ is the activation function, x_t is the input vector, k_t is the convolutional kernel weight, and b_t is the bias term. After feature extraction by the convolution layer, a pooling layer is applied to reduce dimensionality and computational complexity [10].

C. LSTM

Long Short-Term Memory (LSTM), introduced by Hochreiter and Schmidhuber [13], addresses the vanishing and exploding gradient issues inherent in traditional RNNs. Unlike standard RNN units, LSTM cells include a memory mechanism composed of a forget gate, input gate, and output gate [11]. These gates regulate the flow of information and allow the network to retain long-term dependencies. The core equations governing the LSTM operation are as follows:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{3}$$
$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{4}$$
$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t \tag{5}$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{6}$$
$$h_t = o_t \cdot \tanh(C_t) \tag{7}$$

Here, x_t is the input, h_t is the hidden state, C_t is the cell state, and σ denotes the sigmoid function. This architecture enables effective modeling of sequential data with both short- and long-range dependencies.

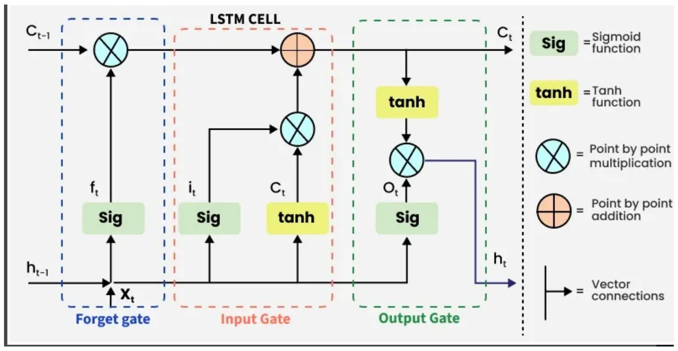

Fig. 2. Architecture of LSTM memory cell

D. Attention Mechanism (AM)

The Attention Mechanism (AM), inspired by human visual cognition [12], enables models to selectively focus on important parts of the input sequence while suppressing less relevant information. It improves performance by dynamically weighting the contribution of each hidden state h_t based on learned relevance scores. The attention score is computed as:

$$s_t = \tanh(W_h h_t + b_h) \tag{8}$$

where W_h and b_h are shared parameters. The attention weights are obtained via softmax normalization:

$$a_t = \frac{\exp(s_t^\top v)}{\sum_t \exp(s_t^\top v)} \tag{9}$$

The final context vector s is calculated as the weighted sum of hidden states:

$$s = \sum_t a_t h_t \tag{10}$$

This mechanism allows the network to adaptively highlight time steps that are most informative for prediction.

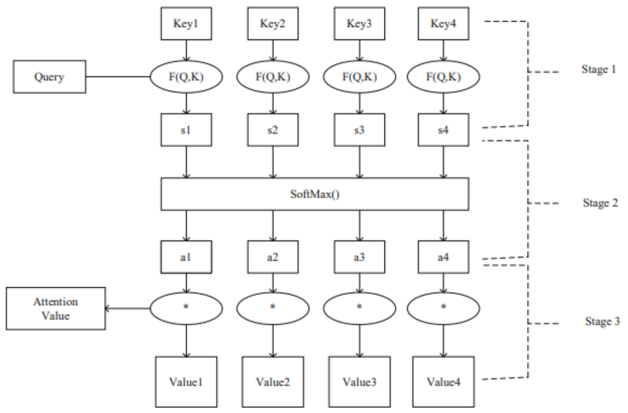

Fig. 3. AM process diagram

E. CNN-BiLSTM-AM Training Process

The training process of the CNN-BiLSTM-AM model involves several key steps to ensure accurate learning from sequential stock data. First, raw data is input and standardized using Z-score normalization to eliminate scale differences:

$$y_i = \frac{x_i - \bar{x}}{s} \tag{11}$$

where x_i is the input, $\bar{x}$ is the mean, and s is the standard deviation of the input feature. Next, the model parameters (weights and biases) are initialized. The input is passed through the CNN layer for local feature extraction, followed by BiLSTM for sequential modeling, and the AM layer for attention-weighted aggregation. The model's output is computed through a dense layer, and the prediction error is calculated by comparing predicted and true values. If stopping criteria such as minimal error or maximum epochs are not met, the error is backpropagated and weights are updated iteratively to optimize the model.

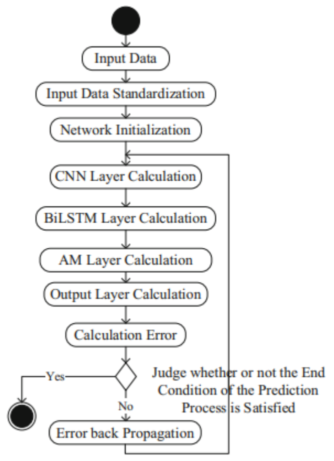

Fig. 4. 4 Activity diagram of CNN-BiLSTM-AM training process

F. CNN-BiLSTM-AM Prediction Process

The prediction process assumes the CNN-BiLSTM-AM model has already been trained. First, the input features are provided and standardized using the Z-score normalization described in Equation (11). The standardized input is then passed through the trained CNN-BiLSTM-AM model to obtain the predicted standardized output. To convert the model's output back to the original scale, the reverse transformation is applied using:

$$x_i = y_i \cdot s + \bar{x} \tag{12}$$

where x_i is the restored prediction in the original scale, y_i is the model output (standardized), s is the standard deviation, and $\bar{x}$ is the mean of the input feature. Finally, the restored prediction is presented as the model's output.

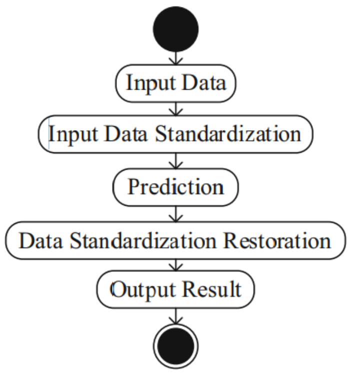

Fig. 5. Activity diagram of CNN-BiLSTM-AM prediction process

IV. Experiments

To evaluate the effectiveness of the proposed CNN-BiLSTM-AM model, we compare it with several baseline models: MLP, CNN, RNN, LSTM, BiLSTM, CNN-LSTM, CNN-BiLSTM, and BiLSTM-AM. All models are implemented in Python using the Keras library with a TensorFlow backend and run on a machine with an Intel i7-4700HQ 2.6GHz processor, 12GB RAM, and Windows 10.

A. Dataset

In this study, we use daily stock data from Infosys Limited (INFY) as the experimental dataset. The data include 8 key features: opening price, highest price, lowest price, closing price, volume, turnover, ups and downs, and percentage change. A total of 7,083 trading days were originally available, from which we selected 6,083 days for training and the most recent 1,000 days for testing.

The features are defined as follows: the *opening price* is the first traded price of the day; the *highest* and *lowest prices* are the extremes observed during the trading day; the *closing price* is the weighted average just before the final transaction; *volume* is the total number of shares traded; *turnover* is the monetary value of all transactions; *ups and downs* represent the price difference from the previous close; and *change* is the percentage change in closing price compared to the previous day.

B. Evaluation Metrics

The predictive performance of all models is evaluated using three commonly used metrics: mean absolute error (MAE), root mean square error (RMSE), and the coefficient of determination (R^2). The formulas are defined as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{13}$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{14}$$

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

Here, y_i is the true value, $\hat{y}_i$ is the predicted value, $\bar{y}$ is the mean of the true values, and n is the number of samples. Lower MAE and RMSE indicate better predictive accuracy, while R^2 values closer to 1 suggest a better model fit.

C. Model Implementation

The CNN-BiLSTM-AM model and all comparative methods are implemented using Python and Keras with a TensorFlow backend. To ensure fairness in comparison, all models are trained under the same hyperparameter settings: 100 epochs, a batch size of 64, a learning rate of 0.001, and a time step of 5. The Adam optimizer is used with the Mean Absolute Error (MAE) as the loss function. The parameters used in CNN-BiLSTM-AM, including the convolution filter size and BiLSTM units, are shown in the configuration table below.

D. Results

All models are trained using the processed training data and evaluated on the same test set. The predictions of each method are compared against the real stock closing prices. The visual results are illustrated in the prediction plots below. Among the tested models, the CNN-BiLSTM-AM method demonstrates the highest degree of alignment between the predicted and actual values, while models like MLP show the weakest fit.

Quantitative evaluation metrics, including MAE, RMSE, and R^2, are computed for each method and summarized in the comparison table below. The results indicate that CNN-BiLSTM-AM achieves the best performance with the lowest MAE and RMSE and the highest R^2 score. Specifically, it outperforms all other methods, including BiLSTM-AM and CNN-BiLSTM. Each addition—such as CNN layers before BiLSTM or the AM mechanism after—incrementally improves predictive accuracy.

Comparatively, LSTM shows improved performance over RNN, reducing MAE and RMSE while slightly increasing R^2. Further improvements are seen in BiLSTM, followed by CNN-BiLSTM, and finally CNN-BiLSTM-AM, which achieves the best results overall. The performance ranking from highest to lowest is as follows: CNN-BiLSTM-AM, BiLSTM-AM, CNN-BiLSTM, CNN-LSTM, BiLSTM, LSTM, CNN, RNN, and MLP. These results confirm that the proposed CNN-BiLSTM-AM architecture provides the most accurate next-day stock closing price predictions and can serve as a valuable tool for financial decision-making.

TABLE I
TEST-SET PERFORMANCE COMPARISON

Model	MAE	RMSE	R^2
MLP	47.7402	82.6017	0.8033
CNN	30.5819	42.3451	0.9483
RNN	42.4742	64.3114	0.8888
LSTM	25.8196	40.5924	0.9525
BiLSTM	25.7535	38.0903	0.9581
CNN–LSTM	26.0955	44.4272	0.9431
CNN–BiLSTM	26.7014	39.4487	0.9551
BiLSTM–AM	24.5657	33.7425	0.9672
Our CNN–BiLSTM–AM	**23.7901**	**32.8739**	**0.9688**

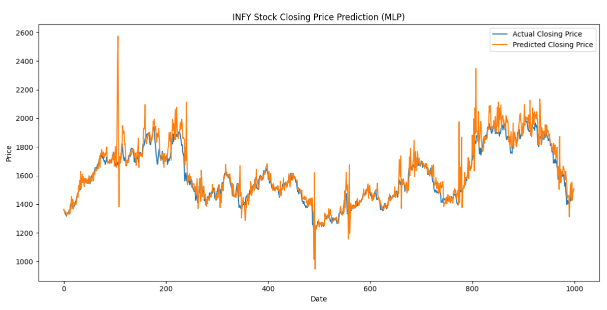

Fig. 6. Comparison of MLP predicted value and real value

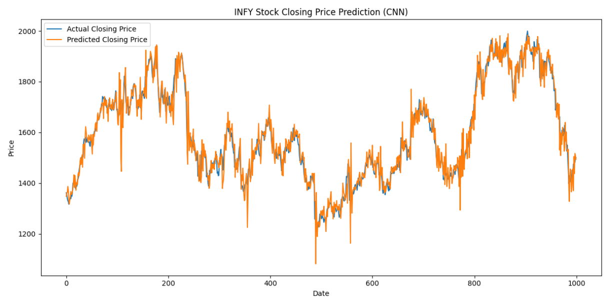

Fig. 7. Comparison of CNN predicted value and real value

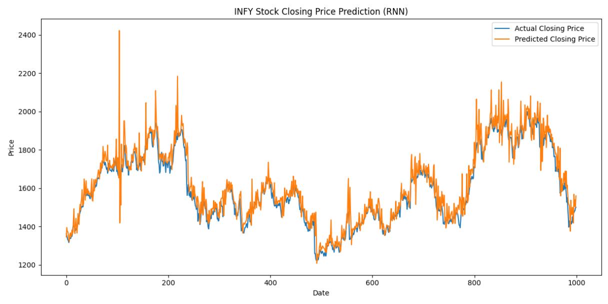

Fig. 8. Comparison of RNN predicted value and real values

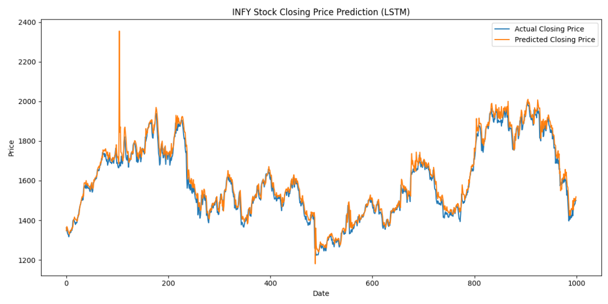

Fig. 9. Comparison of LSTM predicted value and real value

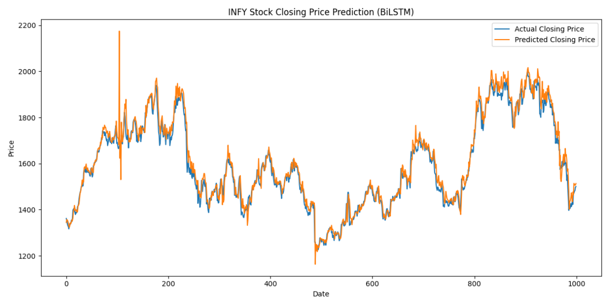

Fig. 10. Comparison of BiLSTM predicted value and real value

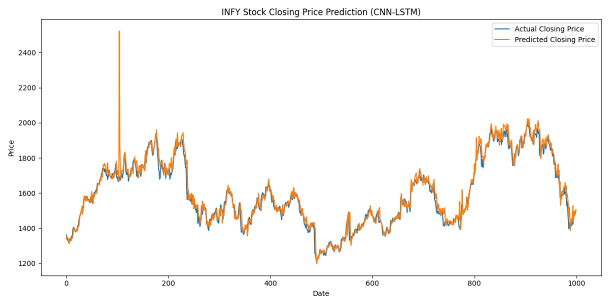

Fig. 11. Comparison of CNN-LSTM predicted value and real value

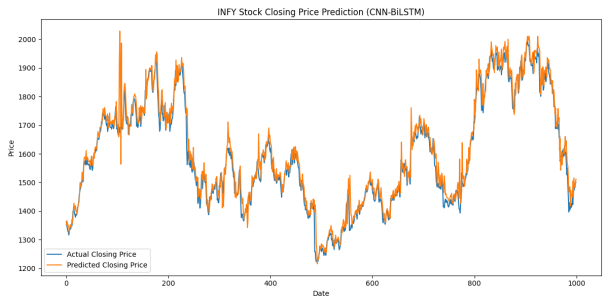

Fig. 12. Comparison of CNN-BiLSTM predicted value and real value

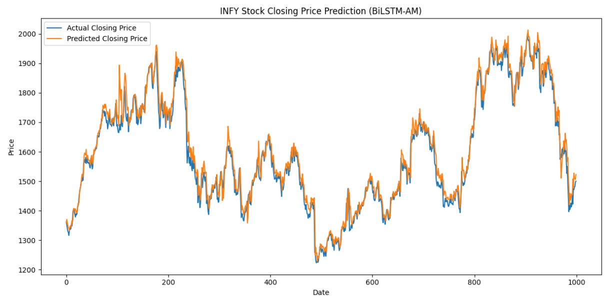

Fig. 13. Comparison of BiLSTM-AM predicted value and real value

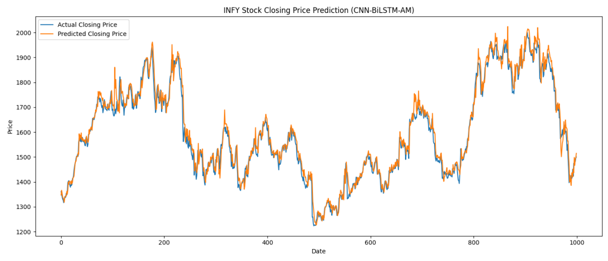

Fig. 14. Comparison of CNN-BiLSTM-AM predicted value and real value

E. Hyperparameter Tuning

TABLE II
HYPERPARAMETER TUNING RESULTS

Experiment Description	Configuration	MAE	RMSE	R^2
More CNN filters	Number of filters = 128	22.1031	30.1031	0.9736
Larger kernel size	Kernel size = 3	24.5857	37.9640	0.9585
More LSTM units	Number of units = 128	21.7605	35.4400	0.9638
With dropout	Dropout rate = 0.2	30.8255	40.9564	0.9426
Larger batch size	Batch size = 128	21.3441	35.5936	0.9635
Smaller learning rate	Learning rate = 0.005	22.7100	31.9436	0.9706
Base model	-	23.7901	32.8739	0.9688

The table above presents the results of hyperparameter tuning based on three evaluation metrics: MAE, RMSE, and R^2.

The base model achieved an MAE of 23.79 and a R^2 score of 0.9688. In particular, increasing the number of LSTM units and using a larger batch size resulted in significant improvements, with the latter achieving the lowest MAE (21.34). Increasing the number of CNN filters also improved performance, achieving the highest R^2 value of 0.9736.

However, applying dropout with a rate of 0.2 led to the poorest performance, suggesting underfitting. A lower learning rate slightly improved the metrics compared to the base model. In contrast, using a larger kernel size degraded performance, likely due to loss of local temporal detail.

Overall, these results indicate that increasing model capacity and using larger batches benefit model accuracy, while regularization techniques like dropout should be carefully tuned in this context.

V. CONCLUSION

In this work, we designed and implemented a hybrid deep learning model, CNN-BiLSTM-AM, to forecast next-day stock closing prices using financial time series data from Infosys (INFY). The model integrates one-dimensional Convolutional Neural Networks (CNNs) for spatial feature extraction, Bidirectional Long Short-Term Memory (BiLSTM) networks for temporal sequence modeling, and an Attention Mechanism (AM) to enhance the focus on critical time steps during prediction.

The input to the model consists of eight key market indicators: opening price, highest price, lowest price, closing price, volume, turnover, ups and downs, and percentage change. These features were standardized using Z-score normalization to ensure consistent scaling before feeding into the model.

Our implementation adopts a five-day time window and is trained using the MAE loss function, Adam optimizer, and a batch size of 64 over 100 epochs. On the test set consisting of 1,000 trading days, the model achieved a MAE of 23.79, RMSE of 32.87, and an R^2 score of 0.9688—demonstrating strong predictive performance and alignment with prior benchmark results.

Compared to baseline models including MLP, CNN, RNN, LSTM, BiLSTM, CNN-LSTM, and BiLSTM-AM, the proposed CNN-BiLSTM-AM achieved superior results in all three evaluation metrics. This confirms that the integration of convolutional layers, bidirectional temporal learning, and attention-weighted aggregation is particularly effective for capturing both local patterns and long-range dependencies in stock price sequences.

The results also highlight that hybrid architectures significantly outperform standalone models. Whereas traditional RNN or CNN models showed reasonable but limited accuracy, the use of sequential attention and bidirectionality allowed our model to better learn dynamic price patterns and reduce generalization error.

This architecture is not only applicable to stock forecasting but also generalizable to other financial and temporal domains. Future work may include hyperparameter optimization using automated search, ensemble learning to boost robustness, and adapting the model for other time series applications such as commodity pricing (e.g., gold or oil), climate forecasting, and earthquake modeling. Additionally, exploring explainability techniques for model interpretability could enhance its practical utility for financial analysts and institutional investors.

REFERENCES

[1] Zhang, G. Peter. "Time series forecasting using a hybrid ARIMA and neural network model." Neurocomputing 50 (2003): 159-175.

[2] Xiao, Chenglin, Weili Xia, and Jijiao Jiang. "Stock price forecast based on combined model of ARI-MA-LS-SVM." Neural Computing and Applications 32.10 (2020): 5379-5388.

[3] Lu, Wenjie, et al. "A CNN-BiLSTM-AM method for stock price prediction." Neural Computing and Applications 33.10 (2021): 4741-4753.

[4] Dunea, Daniel, and Stefania Iordache. "TIME SERIES ANALYSIS OF AIR POLLUTANTS RECORDED FROM ROMANIAN EMEP STATIONS AT MOUNTAIN SITES." Environmental Engineering Management Journal (EEMJ) 14.11 (2015).

[5] White, Halbert. "Economic prediction using neural networks: The case of IBM daily stock returns." ICNN. Vol. 2. 1988.

[6] Li, Jing, Shuxiao Pan, and Lei Huang. "A machine learning based method for customer behavior prediction." Tehnički vjesnik 26.6 (2019): 1670-1676.

[7] Hu, Y. "Stock market timing model based on convolutional neural network–a case study of Shanghai composite index." Finance Economy 4 (2018): 71-74.

[8] Zeng, An, and Wenjun Nie. " LSTM (Stock Recommendation System Based on Deep Bidirectional LSTM)." 46.10 (2019): 84-89.

[9] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[10] Kamalov, Firuz. "Forecasting significant stock price changes using neural networks." Neural Computing and Applications 32.23 (2020): 17655-17667.

[11] Treisman, Anne M., and Garry Gelade. "A feature-integration theory of attention." Cognitive psychology 12.1 (1980): 97-136.

[12] Kamalov, Firuz. "Forecasting significant stock price changes using neural networks." Neural Computing and Applications 32.23 (2020): 17655-17667.

[13] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.