Financial Engineering and Risk Management (2024)
Clausius Scientific Press, Canada

DOI: 10.23977/ferm.2024.070519
ISSN 2523-2576 Vol. 7 Num. 5

CNN-GRU-XGBoost stock price prediction model under hyperparameter-based optimisation

Yunyi Liu[1,a]

[1]*Hangzhou Dianzi University, Hangzhou, China*
[a]*232150028@hdu.edu.cn*

Keywords: Stock price prediction, convolutional neural network, XGBoost, Bayesian search, random search

Abstract: This paper addresses the limitations of existing stock price prediction methods, which often lack explanatory power and struggle with complex hyperparameter combinations. We propose a hyperparameter tuning approach based on the CNN-GRU-XGBoost model. By employing Bayesian tuning for the CNN-GRU and random search for XGBoost, we identify the optimal hyperparameters. The CNN extracts local features from the data, while the GRU captures long-term dependencies. The combined features are then input into the XGBoost model for accurate stock predictions. Testing on five years of Hang Seng Index data demonstrates significant improvements in prediction accuracy, reduced noise, and enhanced model interpretability compared to traditional single models.

1. Introduction

In the early days of stock price forecasting, traditional statistical models like ARMA, ARCH/GARCH, VAR, ARIMA, and Kalman Filtering were used to analyse historical price data, focusing on market trends and volatility. [1] These models typically assume linear market behaviour, which can limit their performance with non-linear and high-dimensional data. Advanced machine learning algorithms like SVMs, decision trees, random forests, and gradient boosting have been developed to better handle complex data relationships.

Researchers have advanced in using complex machine learning algorithms, including SVMs, decision trees, and gradient boosters, for their ability to manage nonlinear data and offer flexible modelling. The finance sector saw a rise in machine learning applications from the mid-2000s, thanks to big data and cloud computing. Kyoung-jae Kim notably used SVMs for financial forecasting. [2].

In recent years, deep learning models, especially deep neural networks, have gained traction in equities. These models trace back to Rosenblatt's 1958 perceptron model [3]. Hornik et al. demonstrated in 1989 that multilayer perceptrons can approximate smooth measurable functions, advancing deep learning [4]. Modern models usually have input, hidden, and output layers, varying mainly in the hidden layer. The CNN began with the LeNet model for digit recognition.[5] AlexNet, which won the 2012 ImageNet competition, further popularized deep learning in computer vision and finance [6].

Recurrent Neural Networks (RNNs), introduced by Hopfield in 1982, are designed for time

series data but can suffer from gradient issues. The LSTM model, developed by Hochreiter and Schmidhuber in 1997, overcomes this with a gated structure, excelling in NLP and time series analysis. LSTMs have seen a resurgence in finance since 2014, following their success in machine translation. [7] Feng, Yuxu, and Li, Yumei demonstrated that LSTMs provide more accurate predictions for the CSI 300 index than other models. [8]

In 2014, Cho et al. introduced the GRU, a streamlined variant of LSTM, well-suited for capturing time-scale dependencies in sequence data modelling. [9] The GRU model, simpler than LSTM and ideal for larger networks, was enhanced by Li-Qiong Gu et al. with an attention mechanism to improve temporal feature capture. This attention-based GRU model surpasses LSTM in simplicity and accuracy, particularly by highlighting key time points, showcasing the GRU's superior attention mechanism.[10] Umang Gupta, Vandana Bhattacharjee, and colleagues have developed a GRU-based data augmentation technique for the StockNet model, targeting the Hang Seng index's overall stock price, and tested it in the Indian market.[11] We introduce the CNN-GRU-XGBoost model, a new prediction model that captures the nonlinear dynamics, complex interactions, and volatility in stock price data, considering the time series continuity and data evolutionary direction for precise forecasting.

The CNN-GRU model excels in extracting spatial and temporal features from data for stock price predictions, with CNN reducing noise and GRU capturing temporal aspects. Despite potential feature oversight and noise, XGBoost is used to learn complex feature interactions and prevent overfitting. Hyper-parameter tuning employs Bayesian and random searches for the CNN-GRU and XGBoost components, respectively, for optimal predictions.

2. Medel specification

2.1. Modelling design

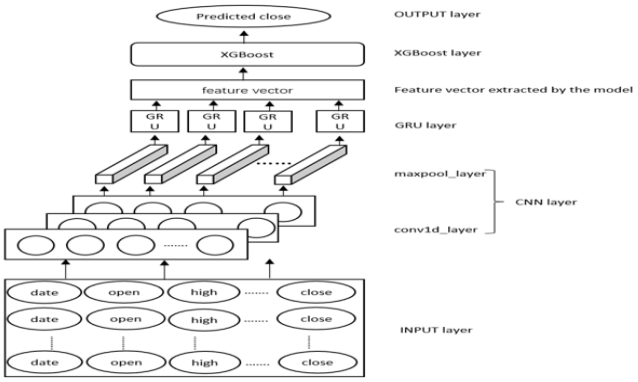

Figure 1: Structure of CNN-GRU-XGBoost model

This flowchart outlines my model, which takes raw stock data, processes it through CNN's convolution and pooling layers, then feeds the extracted features into GRU for extensive feature extraction. Finally, the data is input into the XGBoost model for the final prediction, as depicted in Figure 1.

2.2. CNN Layer

The model initiates with a CNN layer, which effectively captures local features for recognizing stock market patterns like short-term technical indicator fluctuations. CNN's parameter sharing lessens complexity and overfitting risks, vital for the noisy, nonlinear stock market. The network's

translation invariance in feature extraction improves robustness by identifying consistent patterns across different time windows.

2.3. GRU Layer

The second part of the process involves the GRU layer. Firstly, GRUs capture long-term dependencies in the stock market, such as the effects of macroeconomic factors on stock prices, which is valuable for investors. Secondly, GRUs excel in handling time series data, identifying key patterns like seasonality and cyclicality essential for stock price forecasting. Lastly, the gating unit in GRUs controls information flow and forgetting, helping filter noise and retain relevant data for predictions.

2.4. XGBoost Layer

XGBoost evaluates feature importance, revealing key stock price influencers. Its ability to model complex nonlinear data through multi-layer decision trees is vital for stock price analysis. Moreover, XGBoost resilience to outliers and missing data is significant in the financial sector.

2.5. Hyperparametric design of each part of the models

The models employed in this study include LSTM, CNN-LSTM, CNN-GRU, and XGBoost, each with a set of parameters. For the LSTM model, epochs are set within the range of 50 to 100, with units set within the range of 50 to 150, and a batch size ranging from 16 to 64, activation functions including relu, tanh, and sigmoid, and optimizers such as Adam, SGD, and RMSprop. The CNN-LSTM model shares similar epochs and units but adds filters ranging from 16 to 128, kernel sizes ranging from 3 to 10, pool sizes ranging from 2 to 5, and the same activation functions and optimizers. The CNN-GRU model follows the same structure as CNN-LSTM. For the XGBoost model, parameters include 50 to 200 estimators, a max depth of ranging from 3 to 10, learning rates of 0.001, 0.01, 0.1, 0.2, or 0.3, subsample rates of 0.6 to 1.0, colsample by tree values of 0.6 to 1.0, gamma values of 0, 1, or 5, and minimum child weights of 1, 5, or 10.

2.6. Overall steps of the model

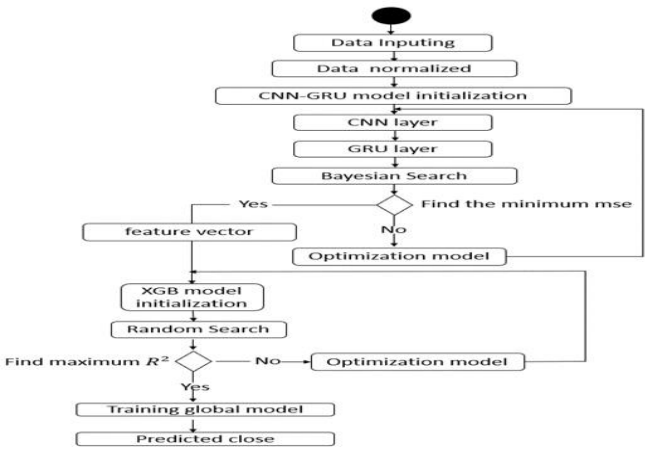

Figure 2: Step-by-step diagram of the CNN-GRU-XGBoost model

Step 1: Input historical stock data, including opening price, closing price, and turnover rate.
Step 2: Normalize the stock features to a range between (0, 1).

Step 3: Initialize the CNN model to extract features from the standardized stock data.

Step 4: Feed the output data into the GRU layer to capture long-term dependencies in the time series features.

Step 5: Perform Bayesian search for hyperparameters, iterating 15 times to find the optimal MSE combination and extract feature vectors.

Step 6: Input the feature vectors into the XGBoost model and use random search to determine the optimal R^2 for its hyperparameters.

Step 7: Train the CNN-GRU-XGBoost model with the optimal hyperparameters and output the prediction results. The overall process is illustrated in Figure 2.

3. Empirical Analysis

3.1. Data preprocessing

The data employed in this article is the Hang Seng Index from 4 February 2019 to 2 February 2024, inclusive of the approximate open, close, turnover rate and other K-line data. Figure 3 below shows the k-line data of the Hang Seng Index for the past 5 years.

date	previous close	open	high	low	average price	rise and fall	percentage change	turnover	volume	turnover rate	close
2019-02-04	27,930.7400	27,985.4200	28,005.1500	27,847.4300	8.6596	59.4700	0.2129	47,460,563.0000	41,099,179,000.0000	0.5387	27,990.2100
2019-02-08	27,990.2100	27,708.1300	28,008.8200	27,534.2000	8.2291	-43.8900	-0.1568	98,765,189.0000	81,274,787,000.0000	1.1211	27,946.3200
2019-02-11	27,946.3200	27,927.4500	28,143.8400	27,847.8500	8.5808	197.5200	0.7068	102,721,151.0000	88,143,433,000.0000	1.1660	28,143.8400
2019-02-12	28,143.8400	28,093.3100	28,219.5900	27,983.4800	7.9795	27.4900	0.0977	120,294,799.0000	95,988,911,000.0000	1.3655	28,171.3300
2019-02-13	28,171.3300	28,184.8700	28,533.3600	28,160.4600	8.5020	326.2600	1.1581	139,555,520.0000	118,649,808,000.0000	1.5841	28,497.5900
2019-02-14	28,497.5900	28,396.4000	28,476.6500	28,275.1800	7.3319	-65.5400	-0.2300	138,652,260.0000	101,657,941,000.0000	1.5739	28,432.0500
2019-02-15	28,432.0500	28,241.4400	28,256.6900	27,845.8700	8.3353	-531.2100	-1.8683	122,712,898.0000	102,284,612,000.0000	1.3929	27,900.8400
2019-02-18	27,900.8400	28,186.7500	28,412.0800	28,186.7500	7.0382	446.1700	1.5991	135,066,900.0000	95,063,362,000.0000	1.5332	28,347.0100

Figure 3: Hang Seng Index k-line data

The dataset was split into training (80%) and test (20%) sets, then normalized using the min-max method to scale variables between 0 and 1, enhancing data comparability, speeding up network convergence, and boosting model prediction accuracy.

$$X_{\text{normalized}} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

Where $X_{\text{normalized}}$ is the normalised data and X is the original data.

Before building the model, time series data must be converted into a supervised learning format. This paper applies a sliding window approach with a width of 5 to predict the next day's closing price, creating samples by backward shifting the data by one unit at a time. The input T and predicted T+1 data from the sliding window serve as inputs, as depicted in Figure 4.

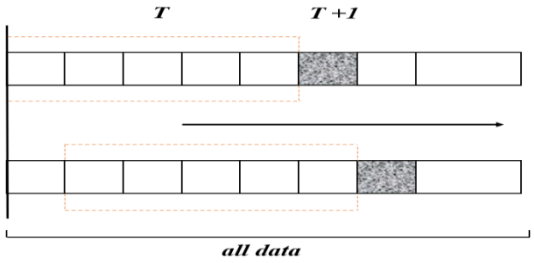

Figure 4: Structure of the sliding window

3.2. Bayesian optimisation and random search

Bayesian optimisation was conducted to find the minimum MSE, using a consistent random seed for reproducibility. Each optimiser performed 15 iterations, starting with five exploratory points and followed by ten hyper-parameter tuning iterations. The random state was set to 70 for consistent

experiment results, as shown in Figure 5.

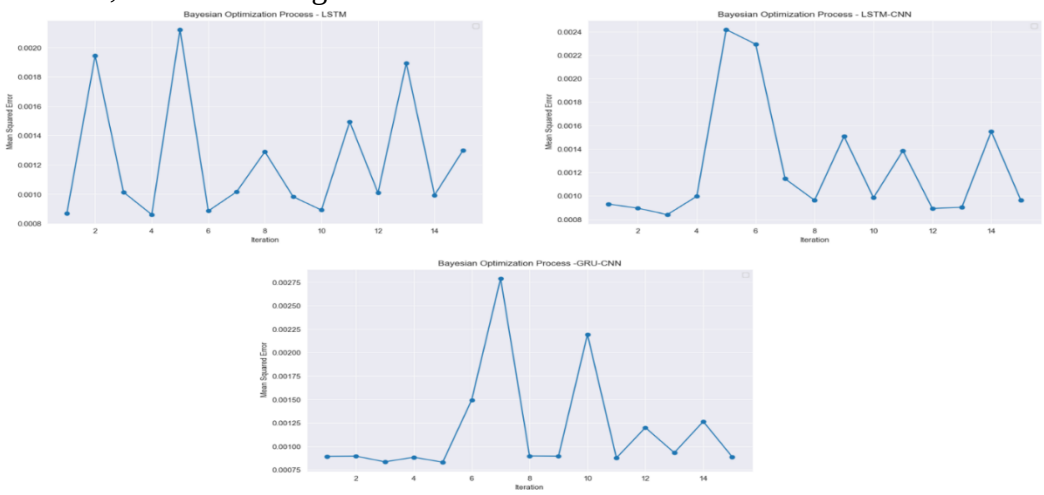

Figure 5: Process diagram for Bayesian optimisation of hyperparameters

The random searches were designed to identify the hyperparameter combination that yielded the largest R^2 . Each random search consisted of 10 iterations, during which 10 different hyperparameter combinations were randomly sampled from a given hyperparameter space. In terms of cross-validation, the XGBoost model searches used 5-fold cross-validation. This approach ensures that the performance of each model is evaluated on multiple data divisions, thereby providing evidence of the generalisation performance of the hyperparameter choices across different datasets. The random seed used in this process was set to 70. Figure 6 illustrates the random search process.

Figure 6: Random search optimisation hyperparameter process diagram

After the above search for hyperparameters. The finalised combination of hyperparameters is shown below: The optimal parameters for the models used in this study include LSTM, CNN-LSTM, CNN-GRU, and XGBoost. The LSTM model is configured with 85 epochs, 76 units, a batch size of 16, a tanh activation function, and the Adam optimizer. The CNN-LSTM model has 79 epochs, 93 units, a batch size of 57, 117 filters, a kernel size of 5, a pool size of 4, a tanh activation function, and utilizes the Adam optimizer. For the CNN-GRU model, the configuration includes 89 epochs, 102 units, a batch size of 16, 91 filters, a kernel size of 5, a pool size of 3, a ReLU activation function, and the Adam optimizer. Lastly, the XGBoost model is set with 164 estimators, a maximum depth of 4, a learning rate of 0.3, a subsample rate of 0.7, colsample by tree value of 0.6, a gamma value of 0, and a minimum child weight of 1.

3.3. Indicators for model evaluation

The test set's textual data is used to evaluate the model's performance, which is also benchmarked using XGBoost, LSTM, CNN-LSTM, CNN-GRU, CNN-LSTM-XGBoost, and CNN-GRU-XGBoost. The classification task employs precision, recall, and F-value as evaluation metrics based on the confusion matrix.

Table 1: Confusion Matrix

		Practicality	
		Rise	**Fall**
Projection	**Rise**	TP	FP
	Fall	FN	TN

Table 1 shows the confusion matrix, detailing samples correctly predicted as up (TP) or down (TN) and those incorrectly predicted (FP and FN). Precision measures the proportion of true positive predictions among all predicted ups, while recall measures true positives among actual ups. The F-value combines precision and recall into an average. Precision is the ratio of correctly classified samples to total samples. The formula is as follows:

$$precision = \frac{TP}{TP + FP} \tag{2}$$

$$recall = \frac{TP}{TP + FN} \tag{3}$$

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} \tag{4}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

In the meantime, the Mean Absolute Error, Root Mean Squared Error and Goodness of Fit R^2 were selected as the evaluation indexes of the models, and the prediction effect of the models was comprehensively analysed. Figure 7 shows the results of the confusion matrix of each model.

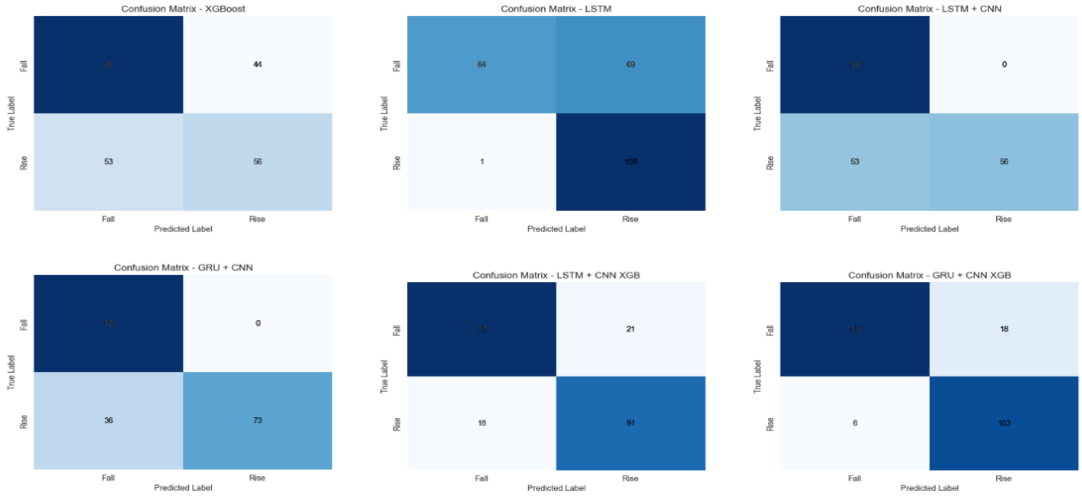

Figure 7: Confusion matrix plot for each model

4. Results and discussion

The optimal combination of hyperparameters for the model was identified through experimentation on the processed data. The confusion matrix was classified to assess the prediction results. The four metrics of precision, recall, F-measure, and accuracy were computed, and metrics were recorded for each model. The results of the experiments are presented in Tables 2 and 3.

Table 2: Comparison of the performance of the models

Models	Precision	Recall	F - measure	Accuracy
XGBoost	0.56	0.5138	0.5359	0.5992
LSTM	0.6102	0.9908	0.7552	0.7107
CNN-LSTM	1.0	0.5138	0.6788	0.7810
CNN-GRU	1.0	0.6698	0.8022	0.8512
CNN-LSTM-XGBoost	0.8125	0.8349	0.8235	0.8388
CNN-GRU-XGBoost	0.8512	0.9450	0.8957	0.9008

Table 3: Comparison of evaluation indicators for each model

Models	Indicators for model evaluation		
	MAE	**RMSE**	R^2
XGBoost	0.0293	0.0015	0.8157
LSTM	0.0104	0.0127	0.9798
CNN-LSTM	0.0101	0.0112	0.9842
CNN-GRU	0.0070	0.0072	0.9925
CNN-LSTM-XGBoost	0.0059	0.0084	0.9912
CNN-GRU-XGBoost	0.0040	0.0053	0.9964

The following presents the prediction effect of each model on the test set, including XGBoost, LSTM, CNN-LSTM, CNN-GRU, CNN-LSTM-XGBoost, and CNN-GRU-XGBoost, among others. As illustrated in Figure 8, despite the high degree of consistency observed in the prediction curves of the various models, an analysis of the metrics reveals that the CNN-GRU-XGBoost model continues to exhibit superior performance compared to the other models

Figure 8: Plot of prediction results for each model

5. Conclusions

The experimental results show that the CNN-GRU-XGBoost model outperforms five alternative

models, including XGBoost, LSTM, and CNN-LSTM-XGBoost, in prediction efficacy and stability. Incorporating a convolutional neural network (CNN) enhances the predictive capacity of the long short-term memory (LSTM) model, reducing the mean absolute error (MAE), root mean square error (RMSE), and coefficient of determination (R^2) from 0.0104, 0.0127, and 0.9798 to 0.0101, 0.0112, and 0.9842, respectively. The CNN-LSTM model had a lower F-value than LSTM, possibly due to data complexity and hyper-parameter ranges. However, replacing LSTM with GRU in the CNN-GRU model showed superior performance, with F-value and accuracy improving from 0.6788 and 0.7810 to 0.8022 and 0.8512. Additionally, MAE, RMSE, and R^2 values improved to 0.0070, 0.0072, and 0.9842. When examined in isolation, the XGBoost model's performance and evaluation indices were weaker than those of deep learning models. However, combining deep learning with XGBoost significantly enhances results. The CNN-LSTM-XGBoost model outperformed both CNN-LSTM and XGBoost in F-value and accuracy, indicating effective integration. After switching from LSTM to GRU, the CNN-GRU-XGBoost model further improved, with MAE, RMSE, and R^2 values decreasing to optimal levels. The highest F-value and accuracy were 0.8957 and 0.9008, respectively, while MAE and R^2 were also the best among all models. Overall, the GRU-XGBoost model, following hyperparameter optimization, demonstrates stability and strong performance in stock price prediction.

References

[1] Bhardwaj., Swanson, N.R. (2006), An Empirical Investigation of the Usefulness of ARFIMA Models for Predicting Macroeconomic and Financial Time Series. Journal of Econometrics, 131 (1-2), 539-578;

[2] Kyoung-jae Kim,(2003) Financial time series forecasting using support vector machines,Neurocomputing,Volume 55, Issues 1–2,2003, Pages 307-319, ISSN 0925-2312

[3] Rosenblatt F. (1958) The perceptron: a probabilistic model for information storage and organization in the brain [. Psychological Review, 1958, 65:386-408.

[4] Hornik K, Stinchcombe M, White H. (1989) Multilayer feedforward networks are universalapproximators.Neural networks, 1989, 2(5):359-366.

[5] Lecun, Y., & Bottou, L. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[6] Krizhevsky, A., Sutskever, I., & Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25(2).

[7] Hochreiter S, Schmidhuber J. (1997) Long Short-Term Memory[J]. Neural Computation, 19979(8):1735-1780.

[8] Feng Yuxu, Li Yumei.(2019) Research on the prediction model of CSI 300 index based on LSTM neural network[J]. Practice and Understanding of Mathematics, 2019, 49(07):308-315.

[9] Cho K, Van Merriënboer B, Gulcehre C, etc.(2014) Learning phrase representations using RNNencoder-decoder for statistical machine translation [J. arXiv preprint arXiv:1406.1078, 2014

[10] Liqiong Gu, Yunjie Wu, Jinhui Feng,(2020) GRU stock prediction model based on Attention mechanism[J]. Systems Engineering, 2020, 38(05):134-140.sed on Bayesian optimization," Intelligent Automation & Soft Computing, vol. 29, no.3, pp. 855–868, 2021

[11] Umang Gupta, Vandana Bhattacharjee, Partha Sarathi Bishmu,(2022) StockNet-GRU based stock index prediction, Expert Systems with Applications, Volme 207,2022,117986,ISSN 0957-4174, https:/idoi.org/10.1016f. eswa. 2022.117986.