Proceedings of the 9th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'23)
Brunel University, London, United Kingdom - August 03-05, 2023
Paper No. CIST 121
DOI: 10.11159/cist23.121

Hybrid Deep Learning Architectures for Stock Market Prediction

Iren Valova[1], Natacha Gueorguieva[2], Thakkar Aayushi [2], Pulluri Nikitha[2], Hassan Mohamed[2]
[1]Computer and Information Science Department, University of Massachusetts Dartmouth
285 Old Westport Rd, Dartmouth, MA, USA
Iren.Valova@umassd.edu
[2]Department of Computer Science, College of Staten Island/City University of New York
2800 Victory Blvd, NY, USA
Natacha.Gueorguieva@csi.cuny.edu; Thakkar.Aayushi @cix.csi.cuny.edu;
Pulluri.Nikitha@cix.csi.cuny.edu; Hassan.Mohamed@cix.csi.cuny.edu

Abstract – Accurate stock market prediction is a challenging task due to the volatile and nonlinear nature of it which depends on numerous factors as local and global economic conditions, company specific performance etc. It is not possible to account all existing relevant factors which influence the stock market in order to make respective trading decisions without having appropriate algorithms and techniques. A recent development of deep learning for making trading decisions has been growing rapidly with numerous research papers addressing stock market forecasting as time series regression problem. In recent years, Long Short-Term Memory (LSTM) neural networks have become the state-of-the-art models for a variety of machine learning problems which differ significantly in scale and nature. The central idea behind the LSTM architecture is a memory cell which can maintain its state over time, and non-linear gating units which regulate the information flow into and out of the cell. Most modern studies incorporate many improvements that have been made to the LSTM architecture since its original formulation. Considering the complexity of financial time series, combining deep learning with financial market prediction is regarded as a very important topic of research. The experiments in this study are divided into two sets which use different topologies. For our first experimental set we created the following hybrid sequential nonlinear models Convolutional Neural Networks (CNN), stacked LSTM, BiDirectional LSTM (BiLSTM) and Gated Recurrent Unit (GRU) with BiLSTM. For our second experimental set we propose a new deep learning topology based on Attention CNN_BiLSTM for pretraining and Light Gradient Bosting Machine (LGBM) as a regressor. The evaluation of experimental results indicates that the last proposed model achieves better performance in predicting stock market when compared to the models proposed in the first experimental set.

Keywords**:** stock market prediction, deep learning, convolutional neural networks, long-short term memory, gated recurrent unit, attention mechanism, light gradient boosting machine

1. Introduction

Medium and large-sized companies play a significant role in the worldwide economy. Such companies make public offerings and gain a place on the stock exchange. Stocks, which dynamically update their price according to the company's current situation, also determine the company's value. The effect of human psychology on the movements of stocks cannot be underestimated. Technical stock interpretations have revealed that specific indicators affect the stock with various experiences from past to present.

Stock price prediction is one of four market categories defined in [1] which utilizes time-series data in order to calculate the anticipating values for stocks by researchers. Accurate stock market forecasting is considered as one of the most challenging problems among time series predictions due to the noise and high volatility associated with the data [2].

With the advancement of technology and science, there have been developments in artificial intelligence studies that have achieved success beyond human reasoning. In particular, linear or nonlinear relations using tabular data have become possible to be learned and applied by machine learning algorithms. In this context, the concept of machine learning can be considered a problem-solving method. Improved unique models, trained with the available datasets, can be implemented to real problems with high performance after their generalization and this applies also to solving stock price prediction.

Traditional machine learning methods such as Artificial Neural Networks (ANNs) [3], tree-based models [4, 5] and Support Vector Machines (SVM) [6] provided the necessary flexibly in modelling the existing nonlinear relationships. However, the performance of traditional machine learning methods largely depends on the selection of features, which

affects the prediction capabilities of the model to a certain extent. In addition, there are some problems in applying these methods, such as poor generalization ability and slow convergence speed.

More recent studies of stock market forecasting problem are based on the current advances in Deep Learning (DL) models and techniques. DL has the important ability to better represent the complex nonlinear topologies and extract robust features what results in capturing the necessary relevant information from complex data. The latter allows stock market prediction using DL approaches to achieve better performance when compared to traditional machine learning methods [7]. As deep learning models have improved, the methods used for predicting the stock market have shifted from traditional techniques to advanced deep learning techniques such as convolutional neural networks, Recurrent Neural Networks (RNNs), long short-term memory, gated recurrent units and Graph Neural Networks (GNNs).

Most of the prediction models belong to a supervised learning approach, when training set is used for the training and test set is used for evaluation. Models based on semi-supervised learning as well as on different deep learning topologies as transfer learning, Generative Adversarial Network (GAN) and Reinforcement Learning (RL) are still in development for being applied for stock market prediction [8]. The main tools used for implementation of DL models are Keras, TensorFlow, PyTorch, Theano, and scikit-learn [1].

Some researchers consider RL models as perspective and adaptable for stock market prediction because of the following two reasons: a) they do not need labeling of training data set; b) they use a reward function to maximize the return or minimize the risk. The latter however requires mathematical formulation of optimization objective criteria which needs to be formulated in accordance with the specific task.

In this research we are using stock market dataset of China described in Section 3 of this research. The experiments select the highest price, lowest price, opening price, closing price, trading volume, and adjusted closing price as features. In the training process, the numbers of different parameters such number of layers and neurons, batch size, optimization and activation functions, early stopping criteria etc. are adjusting in order to improve performance.

The remaining sections of this paper are organized as follows. The proposed models are explained in Section two as well as the evaluation criteria. Experiments and results are discussed in Section three. Section four concludes the study and outlines the future work.

2. CNN Based Models for Stock Market Prediction

Convolutional neural networks are most commonly used for classification of two-dimensional color images. They belong to a class of neural networks which consist of an arbitrary number of convolutional and pooling layers, followed by one or more fully connected layers that perform classification. The purpose of pooling layers is to further reduce the output volume of each layer by performing a simple operation such as *average*, *max* and *min*.

The convolution improves machine learning system in general because of its integrated capabilities: sparse interactions, sharing, and equivalent representations (translation invariance in image recognition framework). Sparse interaction is accomplished by using kernels (much smaller than the input) which occupy only a small fraction of pixels. Sharing of learned features by all neurons in a layer in the form of strides through the entire image leads to parameter sharing. The achievement of translation invariance originates from the identification of the learned features which is independent from their location [9].

The traditional convolutional layers use three-dimensional filters (kernels) and activation functions to process image features. However, in the stock prediction domain, CNNs are used to process time series data, which possess one dimensional features. To accommodate for this difference in data shape, CNNs for time series utilize a one-dimensional filter that slides over the time series with a stride determined by the dataset.

2.1. Gated RNNs for Sequence Modelling: LSTM and GRU

CNNs however are unable to learn sequential correlations as they do not relay on historical data what is the case of time sequence based models. In contrast, recurrent neural networks are specialized for sequential modelling but are unable to extract individual features in the same way as CNNs. RNNs are a family of neural networks for processing a sequence of values $x(1), x(2), \ldots, x(\tau)$ and some of them can also process sequences of variable length [10]. Recurrent computations in

RNN which involve mapping inputs and parameters are formalized with computational graphs because of their repetitive configuration which corresponds to a chain of Neural Network (NN) events. Unfolding these graphs results in sharing of parameters across the RNN structure. Traditional RNNs suffer from both vanishing and exploding gradients and may fail when longer contexts are required in order to fit the model. As a result, a form of RNN, the GRU and long short-term memory were created for more efficient memory usage [11].

While standard RNN computes hidden layer at next time step directly using the input vector $\boldsymbol{x}$, hidden state $\boldsymbol{h}$, weight matrices $\boldsymbol{W}$ and vector $\boldsymbol{b}$ as shown below:

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{W}_{hh}\,\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_{hx}\,\boldsymbol{x}^{(t)} + \boldsymbol{b}) \tag{1}$$

GRU first computes an *update gate* (another layer) which controls what parts of hidden state are updated vs. preserved:

$$\boldsymbol{u}^{(t)} = \sigma(\boldsymbol{W}_u\,\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_u\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_u) \tag{2}$$

The next computations are on *reset gate* which determines what parts of the previous hidden state will be used to compute the new content:

$$\boldsymbol{r}^{(t)} = \sigma(\boldsymbol{W}_r\,\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_r\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_r) \tag{3}$$

New hidden state content resets the gate selected useful parts of previous content:

$$\tilde{\boldsymbol{h}}^{(t)} = \tanh(\boldsymbol{W}_h\,(\boldsymbol{r}^{(t)} \circ \boldsymbol{h}^{(t-1)}) + \boldsymbol{U}_h\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_h) \tag{4}$$

Update simultaneously controls what is kept from previous hidden state and what is updated to the new hidden state:

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{u}^{(t)}) \circ \boldsymbol{h}^{(t-1)} + \boldsymbol{u}^{(t)} \circ \tilde{\boldsymbol{h}}^{(t)} \tag{5}$$

where $\boldsymbol{x}^{(t)}$ and $\boldsymbol{h}^{(t)}$ are the input and output vectors, $\tilde{\boldsymbol{h}}^{(t)}$ - candidate activation vector, $\boldsymbol{u}^{(t)}$ and $\boldsymbol{r}^{(t)}$ are the update and reset vectors; $\boldsymbol{W}$, $\boldsymbol{U}$ and $\boldsymbol{b}$ are weight matrices and bias vector parameters which need to be learned during training; σ is sigmoid activation function and $\circ$ denotes the Hadamard product.

LSTM were proposed as a solution to the vanishing gradients problem [12, 13]. On step t, there is a hidden state and a cell state presented by vectors with length n. The cell stores long-term information and the LSTM can erase, write and read information from the cell. The selection of which information is erased/written/read is controlled by the following three dynamic gates with length n: (a) *forget gate* ($\boldsymbol{f}^{(t)}$) controls what is kept vs forgotten, from previous cell state; (b) *input gate* ($\boldsymbol{i}^{(t)}$) controls what parts of the new cell content are written to cell; (c) *output gate* ($\boldsymbol{o}^{(t)}$) controls what parts of cell are output to hidden state. On each time step, each element of the gates can be open (1), closed (0), or somewhere in-between and their value is computed based on the current context. *New cell content* ($\tilde{\boldsymbol{c}}^{(t)}$) determines the new content to be written to the cell; *cell state* ($\boldsymbol{c}^{(t)}$) comes with two options: erase ("forget") some content from last cell state, and write ("input") some new cell content; *hidden state* ($\boldsymbol{h}^{(t)}$) reads ("output") some content from the cell. Gates are applied using the Hadamard product. Equations (6) represent the forward pass of LSTM [13].

$$\begin{aligned}
\boldsymbol{f}^{(t)} &= \sigma(\boldsymbol{W}_f\,\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_f\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_f) \\
\boldsymbol{i}^{(t)} &= \sigma(\boldsymbol{W}_i\,\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_i\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_i) \\
\boldsymbol{o}^{(t)} &= \sigma(\boldsymbol{W}_o\,\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_o\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_o) \\
\tilde{\boldsymbol{c}}^{(t)} &= \tanh(\boldsymbol{W}_c\,\boldsymbol{h}^{(t-1)} + \boldsymbol{U}_c\,\boldsymbol{x}^{(t)} + \boldsymbol{b}_c) \\
\boldsymbol{c}^{(t)} &= \boldsymbol{f}^{(t)} \circ \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tilde{\boldsymbol{c}}^{(t)} \\
\boldsymbol{h}^{(t)} &= \boldsymbol{o}^{(t)} \circ \tanh \boldsymbol{c}^{(t)}
\end{aligned} \tag{6}$$

where $\boldsymbol{x}^{(t)}$, $\boldsymbol{f}^{(t)}$, $\boldsymbol{i}^{(t)}$ and $\boldsymbol{o}^{(t)}$ are respectively input vector to LSTM unit, forget gate's activation vector, input/update gate's activation vector and output gate's activation vector; $\boldsymbol{h}^{(t)}$, $\tilde{\boldsymbol{c}}^{(t)}$ and $\boldsymbol{c}^{(t)}$ are in turn hidden state vector, cell input activation vector and cell state vector.

Stacked GRUs and LSTMs are used to overcome the gradient explosion and gradient vanishing problem in non-stationary financial time series data as in most cases the datasets which are considered contain long-term values. The purpose

of using stacked GRUs or LSTMs is to increase the depth of the model by adding respective multiple hidden layers in order to improve the model efficiency.

2.2. Hybrid Architecture: Convolutional NN Combined with LSTM

The combination of the CNN with the RNN-based LSTM allowed to enmesh two architectures together where the CNN feeds features directly into the LSTM and the LSTM learns higher order sequential features. CNN is able to learn local response from temporal or spatial data but lacks the ability of learning sequential correlations. LSTM model uses these sequences of higher level representations and further process them through its gates.

Using tensors allow further incorporation of CNN and LSTM for solving the forecasting of stock market. The data features are learned using convolution operations and the series of data over time are trained using LSTM what means the replacement of matrix multiplications in the input, forget and output gates of LSTM (6) with convolution operations denoted by * (7) [14]. Therefore the state of a particular cell in the grid is determined by the input and past states of its neighbours' states

$$\begin{aligned} \boldsymbol{i}^{(t)} &= \sigma(\boldsymbol{W}_i * \aleph^{(t)} + \boldsymbol{U}_i * \boldsymbol{H}^{(t-1)} + \boldsymbol{V}_i \circ \boldsymbol{C}^{(t-1)} + \boldsymbol{b}_i) \\ \boldsymbol{f}^{(t)} &= \sigma(\boldsymbol{W}_f * \aleph^{(t)} + \boldsymbol{U}_f * \boldsymbol{H}^{(t-1)} + \boldsymbol{V}_f \circ \boldsymbol{C}^{(t-1)} + \boldsymbol{b}_f) \\ \boldsymbol{o}^{(t)} &= \sigma(\boldsymbol{W}_o * \aleph^{(t)} + \boldsymbol{U}_o * \boldsymbol{H}^{(t-1)} + \boldsymbol{V}_o \circ \boldsymbol{C}^{(t)} + \boldsymbol{b}_o) \\ \boldsymbol{C}^{(t)} &= \boldsymbol{f}^{(t)} \circ \boldsymbol{C}^{(t-1)} + \boldsymbol{i}^{(t)} \circ \tanh(\boldsymbol{W}_c * \aleph^{(t)} + \boldsymbol{U}_c * \boldsymbol{H}^{(t-1)} + \boldsymbol{b}_c) \\ \boldsymbol{H}^{(t)} &= \boldsymbol{o}^{(t)} \circ \tanh(\boldsymbol{C}^{(t)}) \end{aligned} \tag{7}$$

where all inputs $\aleph^{(1)}, \aleph^{(2)}, ..., \aleph^{(t)}$, cell outputs $\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)}, ..., \boldsymbol{C}^{(t)}$, hidden states $\boldsymbol{H}^{(1)}, \boldsymbol{H}^{(2)}, ..., \boldsymbol{H}^{(t)}$, and gates $\boldsymbol{i}^{(t)}$, $\boldsymbol{f}^{(t)}$ and $\boldsymbol{o}^{(t)}$ are 3-dimensional.

2.3. Performance Indexes

In the experiments, the performance of the stock market prediction models is measured by four indicators Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Square Error (RMSE) and R-squared error (R^2). Their expressions are presented in Eqs. (8, 9, 10, 11)

$$MAE(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i| \tag{8}$$

$$MSE(\hat{y}, y) = \frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)^2 \tag{9}$$

$$RMSE(\hat{y}, y) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (\hat{y}_i - y_i)^2} \tag{10}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N} (y_i - \hat{y}_i)^2 / N}{\sum_{i=1}^{N} (\overline{y}_i - \hat{ty}_i)^2 / \mathrm{N}} \tag{11}$$

where N is the length of prediction, $\hat{y}_i$ and y_i represent the predicted and measured (observed) value for the i^{th} sample; $\overline{y}_i$ is the average value. The value range for R^2 is (0, 1). If the values of MAE and RMSE are close to 0, then the error between the respective predicted and real values is smaller and therefore the forecasting accuracy is higher. The fitting degree of the model is better if the value of R^2 is closer to 1.

3. Experiments

The experimental dataset used in this research is the stock market of China (601988.SH). It contains stock price data for the period of time January 1, 2007 to March 31, 2022 and is publicly available [15, 16]. It is shown in Fig. 1. It has 20 columns having daily information about parameters such as opening price, closing price, maximum price, minimum price,

trading volume, turnover rate and amount where the data in one day denotes a point of the sequence. For the purposes of stock market forecasting we selected the following limited part of dataset columns: 'open', 'high', 'low', 'close', and 'amount'. For all experiments *ModelCheckpoint* is used to save best model during the training. The lookback (*n_past)* value is selected for different models with *Adam* as an optimizer and number of training epochs is set to 50.

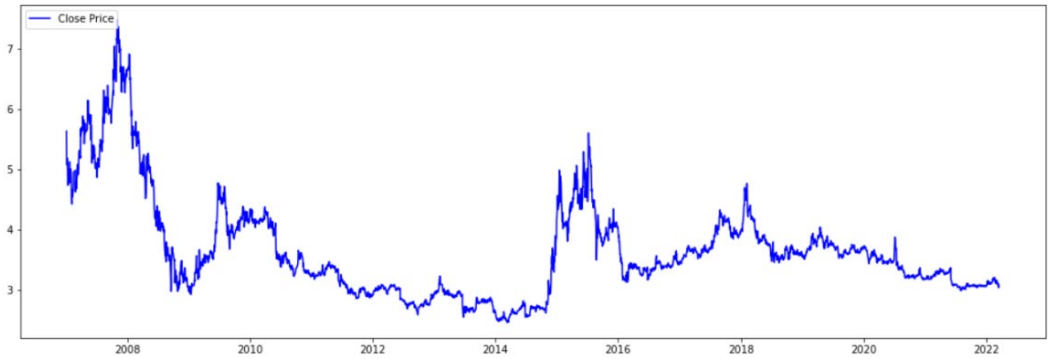

Fig. 1. Close prize data during the period January 1, 2007 - March 31, 2022

Applying the scaling process in financial datasets prevents the stock from predicting higher values in the future that are not in the dataset. Our experiments with implementation of *StandardScaler* and *MinMaxScaler* prove that the first one is preferable due to the high numbers in the amount column while the second one squeezes the numerical data in the columns into a particular range. *MinMaxScaler* preferred by some researchers improves the machine learning model's accuracy but when applied to real problems it would affect the model generalization. The dataset is split randomly into training (all dataset except the last six months) and test (last six months).

The time-series forecasting models require setting the lookback value t as a parameter before the training. For example $t = 20$ means that model's instantaneous close price is predicted based on the previous 20 days in respective dataset columns.

3.1. Experimental Models and Results

Our first experimental set for stock market forecasting uses models based on CNN_LSTM, Stacked_LSTM and GRU_BiLSTM where the lookback parameter t takes respectively values 5, 10 and 20. The lookback values 5-10-20 are chosen because in the financial world, companies are expected to have significant changes in their stocks during the week (it is 5 because there are 5 weekdays in a week), 2 weeks (it is 10 because there are 10 weekdays in 2 weeks), and 1 month (it is 20 because there are 20 weekdays in a month). Performance evaluation of the above models is shown in Table 1.

Convolutional operation of CNN_LSTM model is designed as a tensor operation (*n_samples*, *1*, *1*, *n_features*, *n_lookback*) which differs from the usual CNN tensor operation (*n_samples*, *n_lookback*, *n_features*). Convolutional LSTM with 64 neurons in the first layer is applied to the input dataset and second convolutional layer with 32 neurons is applied to its output. The extracted features are converted into a single long continuous linear vector by the flatten layer. The first layer of CNN Fully Connected Layer (FCL) is a dense layer with 16 neurons and the last one is a dense layer with 1 neuron (output).

For many sequence labeling tasks it is beneficial to have access to both past (left) and future (right) contexts. However, the LSTM's hidden state usually denoted h_t takes information only from past, knowing nothing about the future. Effective solution to this problem is Bi-directional LSTM (BiLSTM). The basic idea is to present each sequence forwards and backwards to two separate hidden states to capture past and future information, respectively. Then the two hidden states are merged to form the final output. A Bidirectional LSTM (BiLSTM), is a sequence processing model that consists of two LSTMs: one taking the input in a forward direction, and the other in a backwards direction [17].

To avoid overfitting while adjusting the depth of the model, two hidden layers containing 64 neurons and 32 neurons were used because of trial and error. The model is built sequentially where the input shape is added into the first hidden

layer containing the first 64 neurons with the feature offered by the TensorFlow library. While building the stacked LSTM, we avoided overfitting the train set by adding too many hidden LSTM layers.

Table 1. Performance Evaluation of Stock-Prediction Models

Architecture	*MSE*	*RMSE*	*MAE*	*R2*
Conv_LSTM_5	0.00030	0.01742	0.01310	0.83128
Conv_LSTM_10	0.00029	0.01714	0.01271	0.84110
Conv_LSTM_20	0.00031	0.01766	0.01329	0.83851
Stacked_LSTM_5	0.00028	0,01670	0.01228	0.84492
Stacked_LSTM_10	0.00029	0.01693	0.01260	0.84498
Stacked_LSTM_20	0.00029	0.01697	0.01264	0.85094
GRU_BiLSTM_5	0.00028	0.01685	0.01289	0.84208
GRU_BiLSTM_10	0.00028	0.01676	0.01268	0.84799
GRU_BiLSTM_20	0.00036	0.01909	0.01462	0.81139

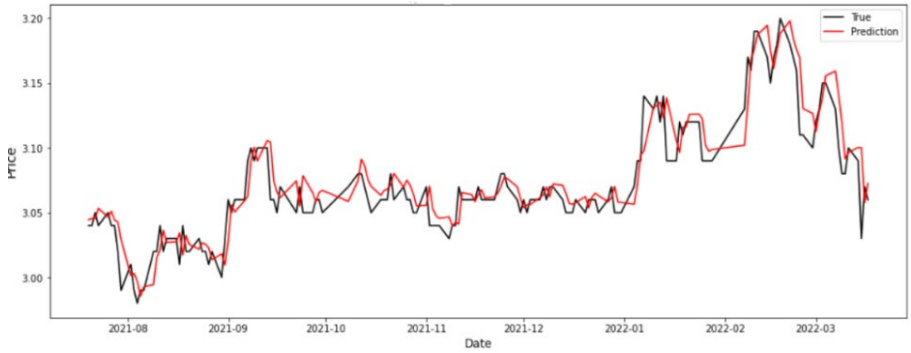

Fig. 2. Stacked_LSTM_20 prediction and true values

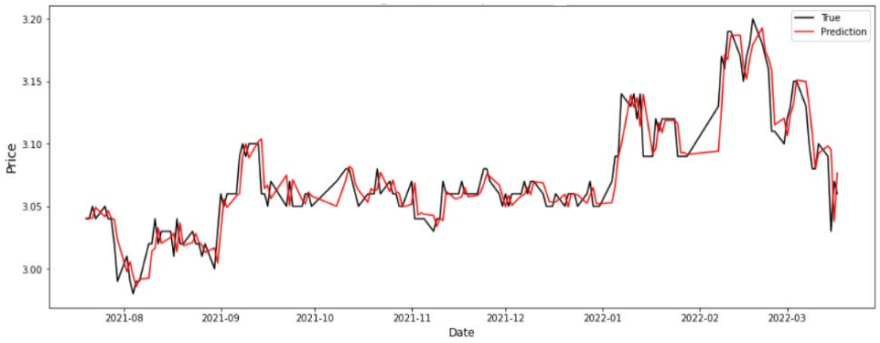

Fig. 3. GRU_BiLSTM_20 prediction and true values

Analysis of performance results shown in Table 1 and Fig. 2and Fig. 3 demonstrate that the Stack_LSTM_20 model has the highest accuracy of 0.85094 among the rest of the models used in the first experimental set but it is much lower than the one reported in [18] (0.88342). GRU_BiLSTM model has the lowest performance with an R^2 score of 0.81139 when the lookback value is set to 20 - Table 1, Fig. 3.

With our second experimental setting we propose the pretraining hybrid model Attention-based CNN_BiLSTM with lookback of 20, where tuning of parameters is done by using Light Gradient Boosting Machine (LGBM). Our model is based on sequence-to-sequence framework, where the Attention-based CNN is the encoder, and the Bidirectional LSTM is the decoder. The deep features of the dataset are extracted by convolution operations and respectively LSTM gets the long-

term data series features. Light gradient boosting regressor (LGBM) performs fine tuning, while fully extracting the information of the stock market in multiple periods. The LightGBM algorithm utilizes two novel techniques called Gradient-Based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB) which allow the algorithm to run faster while maintaining a high level of accuracy.

In Table 2 we present the evaluation results of stock market dataset with the proposed model and the one in [18] and in Fig. 4 some final experimental results.

Table 2. Evaluation performance of the proposed model and the one in [18]

Model	MSE	R^2 score	MAE	RMSE
Attn_CNN_BiLSTM_LGBM	0.00011	0.93909	0.00789	0.01030
ACNN_BiLGBM/XGBoost	0.00020	0.88342	0.01126	0.01424

Analysis of MSE values for both approaches in solving stock market prediction forecasting (Table 2) shows a decrease of almost 50% when using the proposed model. Similarly, the R^2 value, which indicates the pattern of the regression values, is much higher in proposed model. Therefore, it predicts the trends of the stock more accurately when compared to the rest of our models suggested in this paper as well as the one in [18].

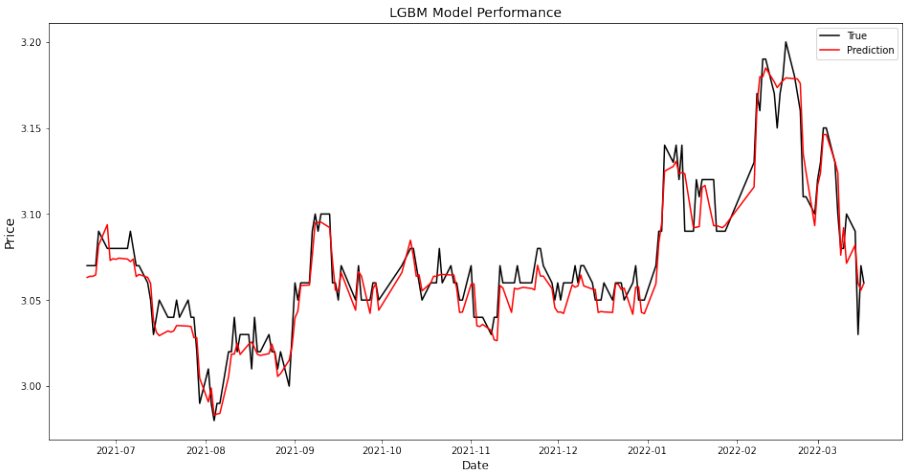

Fig. 4. Attn_CNN_BiLSTM_LGBM_20 prediction and true values

4. Conclusion

The stock market forecasting problems are mostly handled as time series task due to the specific time frame of the prediction which may range from minute level to day level. In this paper we conducted research using several hybrid models which combine a deep learning approach by employing a Convolutional LSTM, stack of LSTM and GRU_BiLSTM to study their effectiveness and applicability on the historical China stock market dataset. Our experiments show that Stack_LSTM_20 model captures better the stock market complex data and improves the forecasting accuracy what differ from the previous research studies. We then propose a hybrid research model which includes attention techniques, CNN, BiLSTM and light gradient boosting machine (Attn_CNN_BiLSTM_LGBM_20). The experimental results indicate that the proposed model achieves better performance than the previous one, and this hybrid approach performs better in predicting the stock market. As evaluation measures we calculated the commonly used for regression models error-based metrics such as MAE, RMSE and R-squared error. In our next research we will be concentrating on using different ensemble techniques for solving stock market prediction problems.

5. References

[1] J. Zou, Q. Zhao *et al*, " Stock Market Prediction via Deep Learning Techniques. A Survey", 2023. arXiv:2212.12717 [q-fin.GN].

[2] M. Ballings, D. Van den Poel, N. Hespeels, and R. Gryp, "Evaluating multiple classifiers for stock price direction prediction" in *Expert systems with Applications* 42, 20 (2015), pp. 7046–7056.

[3] Z. Guo, H. Wang, Q. Liu, J. Yang, " A Feature Fusion Based Forecasting Model for Financial Time Series", *Plos One***9**(6): 172-200.

[4] A.Rupesh Kamble, " Short and long term stock trend prediction using decision tree", in *International Conference on Intelligent Computing and Control Systems (ICICCS),* IEEE, 2017, pp. 1371–1375.

[5] F.X. Satriyo D Nugroho, T. Bharata Adji and S. Fauziati, " Decision support system for stock trading using multiple indicators decision tree", in *First International Conference on Information Technology, Computer, and Electrical Engineering*, IEEE, 2014, pp. 291–296.

[6] S. Prasadas, S. Padhy, "Support Vector Machines for Prediction of Futures Prices in Indian Stock Market. *International Journal of Computer Applications,* 2012*,* **41**(3): pp. 22-26.

[7] Y. Bengio, A. Courville, P. Vincent P., " Representation Learning: A Review and New Perspectives", *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**(8): 2013, pp. 1798-1828.

[8] J. Weiwei, " Applications of Deep Learning in Stock Market Prediction: Recent Report", *Expert Systems* 184 (2021), 115537. https://arxiv.org/pdf/2003.01859.pdf .

[9] C.Szeged, W.Liu, Y.Jia, P.Sermanet, S.Reed, D.Anguelov, D.Erhan, V.Vanhoucke, A.Rabinovich, "Going deeper with convolutions", *Proc. of the IEEE conference on computer vision and pattern recognition,* Boston, MA, 2015, pp. 1-9.

[10] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation", 2014, arXiv: 1406.1078.

[11] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Comput*., vol. 31, no. 7, pp. 1235–1270, Jul. 2019, doi: 10.1162/NECO_A_01199.

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory. Neural computation", 9(8): 1997, pp.1735–1780.

[13] C. Olah, "Understanding LSTM Networks", 2015. https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[14] I. Valova, N. Gueorguieva, S. Smudidonga, " Short-Term Traffic Forecasting Using Deep Learning", Avestia Publ., in *Proceedings of the 7th World Congress on Electrical Engineering and Computer Systems and Sciences (EECSS'21)* Prague, Czech Republic, July, 2021 Paper No. MVML 102 DOI: 10.11159/mvml21.102, pp. MVML 102-1 – 102-8.

[15] J. Chung, C. Gulcehre, and K. Cho, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling", 2014. https://arxiv.org/pdf/1412.3555.pdf

[16] www.tushare.pro; https://github.com/waditu/tushare.

[17] X. Hu, T. Liu, X. Hao *et al., "* Attention-based Conv-LSTM and Bi-LSTM networks for large-scale traffic speed prediction", *J Supercomput* **78**, 2022, pp. 12686–12709. https://doi.org/10.1007/s11227-022-04386-7

[18] Z. Shi, Y. Hu, G. Mo, and J. Wu, "Attention-based CNN-LSTM and XGBoost hybrid model for stock prediction," Apr. 2022, doi: 10.48550/arxiv.2204.02623. https://arxiv.org/pdf/2204.02623.pdf

[19] V. Polepally, N. S. N. Reddy, M. Sindhuja, N. Anjali, and K. J. Reddy, "A Deep Learning Approach for prediction of Stock Price Based on Neural Network Models: LSTM and GRU," *12th Int. Conf. Comput. Commun. Netw. Technol.,* ICCCNT 2021, 2021, doi: 10.1109/ICCCNT51525.2021.9579782.