Expert Systems With Applications 202 (2022) 117123

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

BiCuDNNLSTM-1dCNN — A hybrid deep learning-based predictive model for stock price prediction

Anika Kanwal [a], Man Fai Lau [a,*], Sebastian P.H. Ng [a], Kwan Yong Sim [b], Siva Chandrasekaran [a]

[a] Department of Computing Technologies, School of Science, Computing and Engineering Technologies, Swinburne University of Technology, Hawthorn, VIC 3122, Australia

[b] School of Engineering, Swinburne University of Technology, Sarawak 93350, Malaysia

ARTICLE INFO

Keywords:
BiCuDNNLSTM-1dCNN
BiCuDNNLSTM
CNN
LSTM
Stock price prediction
Time series data

ABSTRACT

Within last decade, the investing habits of people is rapidly increasing towards stock market. The nonlinearity and high volatility of stock prices have made it challenging to predict stock prices. Since stock price data contains incomplete, complex and fuzzy information, it is very difficult to capture any nonlinear characteristics of stock price data, which usually may be unknown to the investors. There is a dire need of an accurate stock price prediction model that could offer insights to the investors on stock prices, which ultimately could deliver positive investment returns. This research is focused on proposing a hybrid deep learning (DL) based predictive model, that combines a Bidirectional Cuda Deep Neural Network Long Short-Term Memory (BiCuDNNLSTM) and a one-dimensional Convolutional Neural Network (CNN), for timely and efficient prediction of stock prices. Our proposed model (BiCuDNNLSTM-1dCNN) is compared with other hybrid DL-based models and state of the art models for verification using five stock price datasets. The predicted results show that the proposed hybrid model is efficient for accurate prediction of stock price and reliable for supporting investors to make their informed investment decisions.

1. Introduction

Due to rapid development of world's economy, the financial industry progresses rapidly. Many financial sectors are growing day by day, and fluctuation of their stock prices are also increasing radically. The financial stock markets produce huge amount of stock price data on a daily basis. The scholars and stakeholders are focused on learning and mastering the rules of predicting stock prices using available datasets. An effective financial prediction can set the foundation for financial development and decision-making to increase the interests of investors and traders (Heaton et al., 2017). A huge amount of stock price data is required for making the right investment choice within a short time. The stock price data is very significant for investors and traders due to stock market's instability which can lead to a significant loss of investment. The investigation of future stock prices is useful for traders, and beneficial for investigating the route of stock market indexes (Kim & Kang, 2019). The stock market prediction is extremely valued in the educational, financial and individuals' day-to-day lives. However, the prediction activities are very complicated and can be affected by many uncertain factors; thus, it is very challenging to understand the forthcoming stock prices. The prediction in financial time series is a most discussed topic in financial industry and the great profits in stock market investment motivates the scholars and investors to focus on stock market forecasting for decades (Dargan et al., 2020).

The stock price prediction has become more and more difficult due to the reason of significant number of records must be handled in a reasonably short time. The financial gain is the most fundamental motivation for any stock price prediction because many people are interested in the financial stock market where they need expert assistance for truthful predictions to invest wisely. Investors are constantly looking for the accuracy in predicting the future returns in investing stocks. The research scholars are continuously trying to find a best stock price prediction model that would generate highly accurate prediction results (Fawaz et al., 2020). From the literature it has been proven that neural network performs better than the traditional analysis (e.g. linear regression) for stock price prediction (Kandel & Castelli, 2020).

Recent research studies had discovered that Deep Learning (DL) models produce more accurate prediction results as compared to traditional Machine Learning (ML) models in financial market. There are several commonly used DL models in forecasting the stock prices such as Deep Neural Network (DNN) (Yong et al., 2017), Convolutional

* Correspondence to: Swinburne University of Technology, H39, John Street, Hawthorn, VIC., 3122, Australia.
E-mail addresses: akanwal@swin.edu.au (A. Kanwal), elau@swin.edu.au (M.F. Lau), sng@swin.edu.au (S.P.H. Ng), ksim@swinburne.edu.my (K.Y. Sim), schandrasekaran@swin.edu.au (S. Chandrasekaran).

https://doi.org/10.1016/j.eswa.2022.117123
Received 3 August 2021; Received in revised form 9 December 2021; Accepted 28 March 2022
Available online 12 April 2022

Neural Network (CNN) (Lecun et al., 1998), Recurrent Neural Network (RNN) (Hiransha et al., 2018) and Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997). In addition, hybrid models, such as CNN-SVM model (Cao & Wang, 2019), CNN-GRU (Sajjad et al., 2020) and CNN-LSTM model (Livieris et al., 2020; Lu et al., 2020; Yang et al., 2020) and many more, have good prediction accuracy.

By analyzing the huge success of DL-based hybrid models in different fields for stock price prediction, this research is focused on developing a hybrid model that can predict stock price accurately for the next trading day within a short period of time. Our proposed model, BiCuDNNLSTM-1dCNN, is a combination of the Bidirectional Cuda Deep Neural Network Long Short-Term Memory (BiCuDNNLSTM) model and a one-dimensional Convolutional Neural Network (CNN). This is a CUDA-enabled GPU based framework to predict the opening price of a stock item for the next trading day. Moreover, our proposed model is highly scalable, adaptive, cost effective and efficient.

The main contribution of this research is:

1. We propose a novel hybrid DL-based model (BiCuDNNLSTM-1dCNN) that is a highly efficient and scalable for stock price prediction.
2. Our proposed model is applicable to individual stock items, e.g. crude oil, as well as stock market's performance index, e.g. DAX Performance Index for Frankfurt Stock Exchange.
3. We perform experiments to compare our proposed model, BiCuDNNLSTM-1dCNN, with 4 other DL models, namely LSTM-CNN, LSTM-DNN, CuDNNLSTM and LSTM, using five stock price datasets (one from a subscription and four from public domain). Our experimental results show that our proposed model, BiCuDNNLSTM-1dCNN, has better prediction accuracy when compared with the other four DL models, and hence, demonstrating that our model is more appropriate for predicting stock price.

The remainder of this paper is organized as follows. The literature review is provided in Section 2. Section 3 explains our proposed model, BiCuDNNLSTM-1dCNN. Section 4 explains the datasets used in our experiments and the relevant experimental setup. Section 5 presents the results of our experiments. Section 6 discusses the threats to validity. Section 7 concludes the paper.

2. Literature review

The development of stock price prediction model for useful decision-making (e.g. increasing investment returns and reducing investment risks) has been actively researched in financial sectors. In reality it is very challenging to predict stock prices with high accuracy because of nonlinearity and high volatility of stock price data. Stock price prediction is also influenced by many factors such as past data, exchange rates, news, national policies and economic situations of countries, which is itself a huge research area and still no theory exist that can identify and pinpoint the relevant parameters (Chang et al., 2019; Maas et al., 2013).

In the past few decades, the application of Deep Learning (DL) methods in the area of stock price prediction has gradually become a hot research because these models can learn from the patterns in the past data and use that knowledge to predict the stock prices. Due to the nonlinear nature of stock price data, DL based models are more appropriate when compared to traditional machine learning approaches.

In this section, we will review several commonly used DL techniques/models for stock price prediction, including: Deep Neural Network (DNN) (Yong et al., 2017), Recurrent Neural Network (RNN) (Hiransha et al., 2018), Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) and Convolutional Neural Network (CNN) (Lecun et al., 1998).

Yong et al. (2017) proposed DNN based model for predicting the Straits Time Index (STI) prices of Singapore stock market. By using market simulation data, they have demonstrated that DNN model can lead to a profit factor of 18.67 and a 70.83% chance of profitable trade. However, the DNN model has two issues. First, the major drawback of DNN model is that it cannot capture the underlying patterns from the features space due to its fully linked nature. Second, DNN faces vanishing and exploding gradient problem in its training phase and this prevents the DNN model from detecting long-term effects (Passricha & Aggarwal, 2010). In other words, past features get less signification during weight updating information in comparison to the portion relative to latest features, that leading to poor generalization performances.

The RNN model have several variants, including LSTM and GRU. The LSTM is preferred by most researchers due to its comparatively simple model training phase (Houdt et al., 2020). LSTM and its variations showed high performance in time-fluctuating data with feedback implanted interpretations for financial time series prediction. The LSTM performed good in making weekly predictions by identifying fundamental trends and making long term forecasting on unpredictable stock data (Shah et al., 2018). The LSTM is the most popular DL model for financial time series prediction (Sezer et al., 2020). Baeka and Kim (2018) proposed an LSTM based approach for stock price prediction. This model has decreased the mean square error (MSE) to 54.1%, the mean absolute percentage error (MAPE) to 35.5%, and the mean absolute error (MAE) to 32.7% for S&P500 dataset. The LSTM tackles vanishing and exploding gradient problem by controlling information's flow through memory blocks (Hochreiter & Schmidhuber, 1997). However, LSTM sometimes missed the abrupt changes in the patterns and this affects the accuracy of the prediction model (Selvin et al., 2017).

Since CNN model performs well in feature extraction and pattern recognition from the feature space in time series data, various researchers have proposed to use CNN model to predict stock prices (Hiransha et al., 2018; Selvin et al., 2017). Selvin et al. (2017) proposed to use CNN model for stocks price prediction. They compared three stock price prediction models based on CNN, LSTM and RNN using three different datasets. Their results showed that their CNN model achieved less error percentage in two datasets. Hiransha et al. (2018) compared CNN, RNN, LSTM and Multilayer Perceptron (MLP) for stock price prediction in two different stock markets, namely National Stock Exchange of India and New York Stock Exchange. They claimed that CNN has outperformed other three models by accurately predicting the closing price of stock for the next trading day. CNN model does not rely on any prior evidence for prediction, it only uses the current sliding window for prediction to capture abrupt changes in patterns. However, the long-term memory cannot be captured by CNN due to its working on current frame for forecasting and CNN model also affected by semi clean data (Lecun et al., 1998).

Bao et al. (2017) proposed a hybrid prediction model (WSAEs-LSTM) for stock price prediction that increases the predictive accuracy and profitability of stocks. According to Bao et al. (2017), this model is time consuming and can be improved by hyperparameter and some advance GPU based frameworks (Bakhoda et al., 2009). Yan and Ouyang (2018) presented a hybrid model based on wavelet analysis with LSTM for prediction of the daily closing price. The proposed stock price prediction model improved the prediction accuracy of long-term dynamic trend. The LSTM overcomes the deficiencies of shallow ML algorithms like time series dependencies and weak data feature learning ability. However, their experiments are only based on one dataset that is Shanghai Composite Index. Chen et al. (2018) proposed four hybrid models based on LSTM. They are (1) Market Vector and LSTM with Attention, (2) Market Vector and LSTM, (3) Technical and LSTM with Attention and (4) Technical and LSTM. The most successful model for prediction is the first one. It used LSTM based on Attention mechanism with Market Vector input to achieve an average improvement of 55.68% in MSE value when compared to other proposed models. Their experiments are based on only one dataset, namely HS300 index.

Hossain et al. (2018) proposed a hybrid DL model, LSTM-GRU, for stock prices prediction. The input dataset is passed to the LSTM layer for generation of first level prediction. The output of the LSTM layer is then passed to the GRU layer to get the final prediction. They used one dataset, the S&P 500 with a total of 66 years of historical data, in their experiment. Their model achieved a MSE value of 0.00098 in prediction accuracy. Lu et al. (2020) employed the CNN-LSTM to forecast stock prices. They used CNN model for features extraction from input data and LSTM for forecasting. In their experiments, they compared MLP, CNN, RNN, LSTM, and CNN-RNN models. The results showed that their proposed model CNN-LSTM achieved smallest MAE value (27.564) and root mean square error, RMSE, value (39.688) among other models and the coefficient of correlation in terms of R^2 is close to 1. Their prediction results are based on closing price of an individual stock and there is still room for improvement in MAE and RMSE values. Eapen et al. (2019) presented a hybrid model based on CNN and Bi-Directional LSTM for improved stock market forecasting. This model achieved the improved prediction to 9% upon single pipeline DL model by avoiding overfitting on challenging dataset (S&P500). Cao et al. (2019) proposed the LSTM model combined with EMD and CEEMDAN algorithm. LSTM with CEEMDAN model achieved a smaller prediction errors in MAE (3.9177), RMSE (4.8291) and MAPE (0.1617) as compared to other prediction models such as LSTM, SVM, CEEMDAN-SVM, and CEEMDAN-MLP, EMD-LSTM. According to Cao et al. (2019) the accuracy and robustness can be improved by adding more financial parameters. Lu et al. (2021) proposed a hybrid model CNN-BiLSTM-AM for stock price prediction. It is reliable for prediction and suitable for investors in their investment's decisions. This model achieved the smallest error values in terms of MAE (21.952) and RMSE (31.694). According to Lu et al. (2021), the results can be more accurate if some adjustments can be made in the parameters. Yang et al. (2020) proposed a prediction model based on three dimensional CNN and LSTM for financial time series data. According to Yang et al. (2020), designing a better prediction model is still a big challenge that can intelligently extract more valuable features from the dataset. Livieris et al. (2020) proposed the hybrid model based on CNN-LSTM for the gold price time-series prediction. The proposed model gives a substantial improvement in the gold price prediction accuracy. They achieved lowest MAE and RMSE for gold price prediction.

From previous research studies, some models performed well in having good prediction accuracy in financial time series data. However, there are still rooms for improvement. First, the major drawback of DNN model is that it cannot capture the underlying patterns, if any, from the feature space due to its fully linked nature. Second, DNN faces vanishing and exploding gradient problem in its training (Passricha & Aggarwal, 2010). In other words, past features get less signification during weights updating information in comparison to the portion relative to latest features, leading to poor generalization performances. The LSTM tackles vanishing and exploding gradient problem by controlling information flow through memory blocks (Hochreiter & Schmidhuber, 1997). CNN effectively captured the abrupt changes from the feature space. However, it cannot capture the long-term memory because it works on the current frame for forecasting and the CNN model is also effected by semi clean data (Lecun et al., 1998). The LSTM offers higher precision by obtaining long term memorization especially for noise vigorous dataset. The LSTM performance can be enhanced by a special architecture called Bidirectional LSTM that runs the input series in both directions to make judgments on pattern (Schuster & Paliwal, 1997). The above-mentioned models work on huge time series data that take a lot of time for training and there is still a dire need of an accurate prediction model for stock price prediction. The recent research studies state that CUDA enabled GPU model has been introduced for high computation on different data series which has a potential for high-performance co-processors (Bakhoda et al., 2009). The proposed hybrid model in this research will combine bidirectional CuDNNLSTM based on CUDA programming model and one dimensional CNN, aiming at having an accurate stock price prediction model.

3. Our proposed model - BiCuDNNLSTM-1dCNN

We propose a hybrid model BiCuDNNLSTM-1dCNN to enhance the accuracy of predicting stock prices and minimizing errors in dataset. Our proposed model involves two components: the Bidirectional CuDNNLSTM (BiCuDNNLSTM) component and the one-dimensional CNN (1dCNN) component. The BiCuDNNLSTM component has the ability of "identifying" the overall temporal information using long and short-term sequential patterns, if any, from the input data in both directions:- forward (from past to future) and reverse (from future back to past). The purpose of this is to preserve any valuable characteristics of historical information in sequences. The BiCuDNNLSTM component enhances the training capability, when compared with a unidirectional one, because this component receives information from the past and future instances simultaneously. This can help in extracting features that are "globally" existed in the data, if any. The output of the BiCuDNNLSTM component is then used as the input in the 1dCNN component. This CNN component has the capabilities to (1) extract the quality feature set and (2) capture any abrupt changes in the current window for prediction. This allows the model to recognize the dynamic variations and patterns happening in the current window, which is a localized behavior of the data. Hence, it is anticipated that such a combination of the BiCuDNNLSTM and 1dCNN components will enhance the training of the model and, hopefully, produce good prediction results.

Fig. 1 shows the schematic representation of our model with two components: (a) the BiCuDNNLSTM component and (b) the 1dCNN component. We aim to effectively combine the advantages of the BiCuDNNLSTM and CNN models to form a novel hybrid model. The first component is the BiCuDNNLSTM network, which is CUDA based. It is an effective technique for recognizing short-term and long-term dependencies in the input time series dataset, in both forward and reverse directions. The forward CuDNNLSTM layer accepts sequence of input time series data in forward direction (from past to present). The reverse CuDNNLSTM layer accepts sequence of input time series data in reverse direction (from present back to past). By doing so, when compared with a uni-directional approach, it can increase the amount of available information fed to the neural network for extracting valuable patterns from time dependencies and information in memory blocks. It also enables additional training by traversing the input time series data in two directions to enhance the accuracy of prediction model by extracting further important patterns from the same input dataset. As a result, it enhances the learning of long-term dependencies, leading to lower feature loss when it is applied before the CNN layer. The output of this BiCuDNNLSTM component is then fed to the 1dCNN component. The CNN model is chosen after the BiCuDNNLSTM model due to its ability to dig out abrupt useful patterns from the representation of stock price data. The 1dCNN component consists of the one-dimensional convolution layer, the max pooling layer, flatten layer, fully connected layer, and the output layer. In the following sub-sections, we briefly describe these DNN, LSTM and CNN layers which are the core parts of our proposed model.

3.1. Deep Neural Network (DNN)

Deep Neural Network (DNN) is a neural network that has one input layer, two or more hidden layers and one output layer (Glorot et al., 2011; Montenegro & Molina, 2019). In each layer, there are many nodes. The nodes between any two layers are fully connected. The input layer accepts input data from the data streams. The data are then passed through the hidden layers to the output layer without going back. In other words, the connections among these layers are unidirectional. The hidden layers are used to capture the complex features from the huge dataset. A hidden layer have multiple neurons, and each neuron has a function called *Activation Function(AF)*. Activation function decides that neuron should be active or not by calculating weighted sum and it also works as gateway of passing the signal to the next neuron. The output layer gives the values predicted by the DNN model.

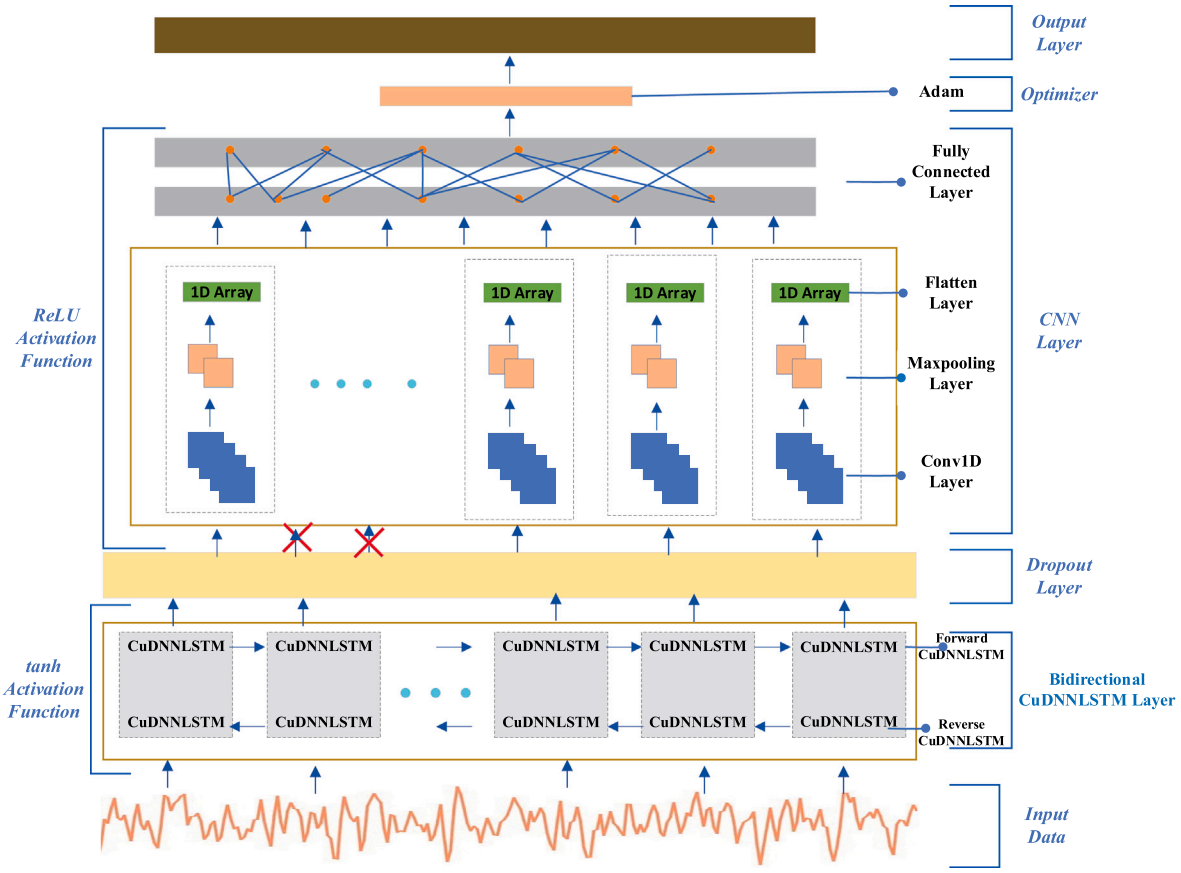

Fig. 1. Framework of Deep Learning based Hybrid Prediction Model.

3.2. Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) (Lecun et al., 1998) is traditionally devised for image datasets to obtain all local relations that are invariant among different dimensions. The CNN model performs good on single dimensional data such as time series data and multi-dimensional data such as images. CNN has four major layers: (1) convolutional layer, (2) max pooling layer, (3) flatten layer and (4) fully connected layer (Sezer et al., 2020). First, the convolution layer accepts input data from data streams and has a filtering function to extract the features from the input data. Second, the max pooling layer is used to shape the obtained features and transform them into more conceptual forms among its hidden layers and memory cells. Third, the flatten layer then converts all these multidimensional data into one dimensional array for the final fully connected layer. Finally, the fully connected layer, in the form of an artificial neural network, determines the output of the model. This fully connected layer includes weights, biases and neurons in many different layers. The neurons in one particular layer are fully connected to neurons in the subsequent layer, that is responsible for learning all features from the previous layer. The optimizer is used at the end of the neural network, to achieve the output of the CNN model. The look back procedure is used for the learning of CNN model. Similar to other DL models, the performance of the CNN model is affected by many factors such as the number of hidden layers, the number of neurons in each individual layer, network weight commencement, activation functions, learning rate, the number of epochs, the batch size, the optimization algorithm adopted in the optimizer, the dropout rate, the kernel size and the filter size (Bergstra & Bengio, 2012). These factors are usually referred to as the hyperparameters of the CNN model.

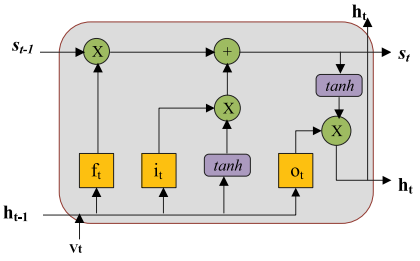

Fig. 2. LSTM Architecture.

3.3. Long Short-Term Memory (LSTM)

Hochreiter and Schmidhuber (1997) proposed the Long Short-Term Memory, LSTM, model. It has been used in language modeling, language translation, speech recognition, opinion analysis, analytical analysis and stock data analysis (Wu et al., 2016). Many DL model's developers preferred to use LSTM model when they are dealing with challenges like automated speech recognition and handwriting recognition. The schematic presentation of LSTM is shown in Fig. 2. The following equations establish the relationships among the relevant quantities depicted in Fig. 2.

$$f_t = Sigmoid(W_f[h_{t-1}, v_t] + b_f) \quad (1)$$

$$i_t = Sigmoid(W_i[h_{t-1}, v_t] + b_i) \quad (2)$$

$$\tilde{s}_t = \tanh(W_{\tilde{s}}[v_t + h_{t-1}] + b_{\tilde{s}}) \quad (3)$$

$$o_t = Sigmoid(W_o[h_{t-1}, v_t] + b_o) \quad (4)$$

$$h_t = o_t \odot \tanh(s_t) \quad (5)$$

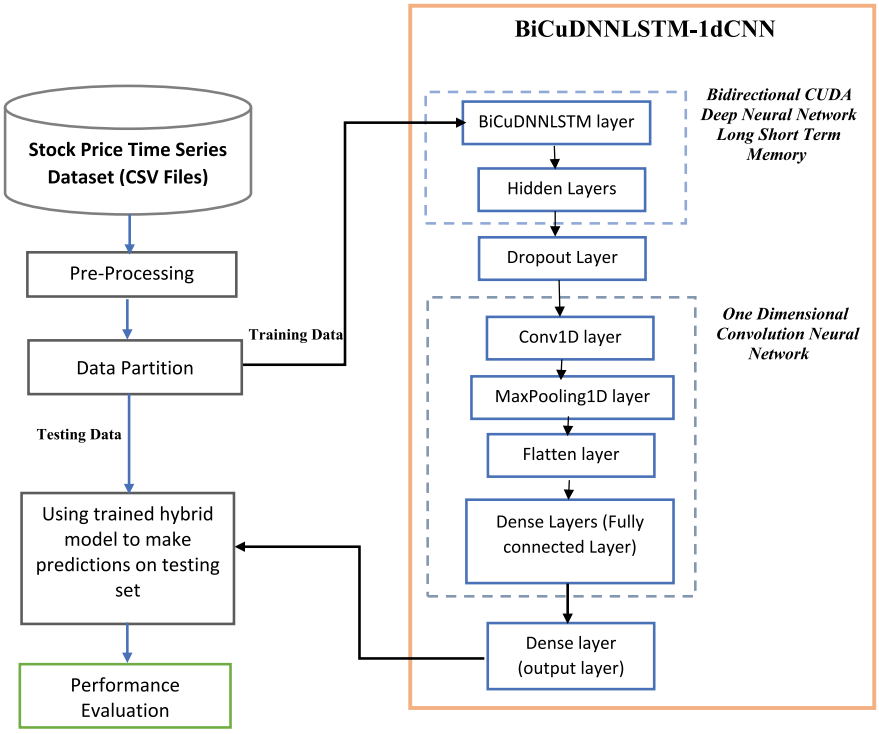

Fig. 3. Flowchart of Deep Learning based Hybrid Prediction Model.

$$s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t \tag{6}$$

where $W_{\cdot}$ represents the weight of inputs to the gates, v_t is the input vector, f_t is the forget gate, i_t is the input gate, o_t is the output gates, t is the time, $\tilde{s}_t$ and s_t denote the distorted input and content to the memory cell, h_t denotes the value of the hidden node and $\odot$ denotes the element-wise product operation. These gates control the information stream that can remember the desired values over random time intervals.

LSTM model performance can be optimized by using hyperparameters such as number of epochs, learning rates, neurons, activation functions, hidden layers, batch size (Yadav et al., 2019). To achieve the finest hyperparameters of LSTM, the hyperparameter optimization techniques used for RNN are also applicable to LSTM (Bergstra et al., 2011). The LSTM performance can be enhanced by a special architecture called Bidirectional LSTM, that runs the input series in both directions to make judgments for extracting meaningful features (Schuster & Paliwal, 1997). The bidirectional architecture is used to increase performance of the relevant prediction model (Althelaya et al., 2018). LSTM can learn data pattern for a longer period that overcomes the long-term dependency issues. It has memory units that supports the neural network to ignore the prior circumstances and reinstate it with new evidence (Hochreiter & Schmidhuber, 1997).

The CuDNNLSTM model is an efficient LSTM implementation of cuDNN that is a GPU-accelerated library of primitives for deep neural networks (Aveleira-Mata et al., 2019). The cuDNN model facilitates robust and simple multi-threading for LSTM networks with high progression modeling power because of its GPU accelerated library. In addition, the overall performance of the CuDNNLSTM model is enhanced by fast matrix multiplication. The CuDNNLSTM model learns progressively to its extreme capacity. Basic LSTM's cell chronological dependency issue is resolved by CuDNNLSTM (Appleyard et al., 2016).

3.4. BiCuDNNLSTM-1dCNN prediction process

In this section, we discuss the processes of training and verifying our hybrid BiCuDNNLSTM-1dCNN model. Fig. 3 depicts the flow of these processes. The five major steps are discussed as follows:

1. Data Preprocessing: The purpose of this step is to preprocess the data (e.g. dealing with NaN) in the dataset, if necessary. Since we obtained the datasets through subscription and some publicly available web sites, we do not have control over how the data were collected and entered into the datasets. We discovered that there is a large gap in the input data. It can also bring the data into a state that DL models can easily extract the required patterns for training purposes. There are many techniques that can be used to pre-process the data such as outlier removal, data clustering and data normalization. In our experiments, we adopt the following two data preprocessing techniques:

   (a) Data Transformation: We deal with the NaN values, if any, in the dataset. Since we are dealing with numerical values in stock prices, a particular NaN entry will be replaced by the mean of its relevant predecessor and successor values from the dataset because NaN values can significantly influence the importance of records and intended evaluation model.

   (b) Data Normalization: To boost the efficiency of the model, we use the MIN–MAX scalar function for the normalization of feature vectors of stock prices dataset This process is necessary to alter all values to a scaled shape as it minimizes the impact of gross effect on dataset (Narayanadoss et al., 2019). The MIN–MAX scalar function avoids values that do not belong to the definition interval of the log function (0,1]. Its formula is given by Eq. (7),

   $$z_i = \frac{x_i - Min(X)}{Max(X) - Min(X)} \tag{7}$$

   where z_i denotes the normalized value, x_i denotes the individual raw data, $X = \{x_1, \ldots, x_n\}$ denotes the set of all x_i's, $Min(X)$ and $Max(X)$ are the minimum and maximum values of X respectively.

2. Data Partition: After the original dataset has been preprocessed, it is then divided into the two disjoint sets:- (1) *training* dataset used for training the model and (2) *testing* dataset used for

verifying the model. In our experiments, we partitioned a dataset with a ratio of 90% (training dataset) to 10% (testing dataset).

3. Model Training: The training dataset is then passed through to our proposed model, BiCuDNNLSTM-1dCNN, for training. This involves the initialization of the various hyperparameters of the model. These hyperparameters are selected on the basis of experiments and literature studies. The Bidirectional CuDNNLSTM layer is responsible for feature extraction. The Dropout layer is used to deal with the issues of overfitting with a proper choice of the dropout rate. The one-dimensional CNN components has the Conv1D layer that takes the remaining features from the previous Dropout layer to capture the abrupt changes (local patterns). Then, the fully connected layer with rectified linear unit (ReLU) activation function is responsible to produce the prediction result.

Theoretical speaking, after the data has passed through the model once, a trained model is obtained. However, due to non-deterministic nature of machine learning models, it is better to perform the training by fitting the data into the model multiple times, each time resulting in a trained model.

Furthermore, during different training attempts, it is better to use different number of parameters (e.g. the number of hidden layers used in the neural network, the number of neurons in each layer) for training so as to achieve better training results. These parameters are collectively referred to as the *hyperparameters* of our hybrid model. According to Tran et al. (2020), neural networks, especially in DL models, have various hyperparameters that are selected to optimize validation errors for training purposes. These hyperparameters may include the number of hidden layers, the number of neurons used in each layer, the dropout rate, the number of epochs, and the batch size of the input data. Optimal selection of hyperparameters for obtaining best trained models is still a big research area. The training process is very time-consuming and resource intensive. Hence, the determination of hyperparameter values is subjective and strongly relied on the researchers (Jain et al., 2018; Jin et al., 2020; Livieris et al., 2020; Yadav et al., 2019). Bergstra and Bengio (2012) proposed to use a random search technique in "optimizing" the hyperparameter values. They have performed an experimental study to compare the performance of neural networks via the random search technique with the grid search technique proposed by Larochelle et al. (2007). Their results show that, within the same boundary conditions (for example, if the hyperparameter is about a range of values, both the minimum and maximum values of the hyperparameter need to be the same), the random search technique can find models that are as good or better using a small fraction of the computation time when compared with the grid search technique. In this paper, we adopt the random search technique in training our BiCuDNNLSTM-1dCNN model.

Algorithm 1 shows our proposed algorithm for training our model using the random search technique. BiCuDNNLSTM layer minimizes the vanishing and exploding gradient problem effectively (line 7). After we apply the single dimensional CNN layer, the model can capture any sudden spatial characteristics proficiently from current frame (line 15). Finally, obtained features are used as input into the Dense layer (a fully connected neural network) with some learning rate and Adam optimizer (line 25). Below is a discussion on the hyperparameters and their potential values set in Algorithm 1 with the relevant line number indicated in the discussion:

(a) Maximum number of trials (*MaxTrial*): This is about how many times we train our models so as to obtain a list of trained models. Once we have these trained models, we will then evaluate and select the one with the smallest MAE value. We anticipated that the more trials we train our model, the better prediction model can be obtained. In Algorithm 1, we set this value to 5 (line 3).

(b) Number of epochs (*epochs*): A single epoch means one round of the full training set. In our experiments, we set the number of epochs to 32 (line 4). According to (Prechelt, 2012), the choice of epochs in DNN can be monitored by the use of "early stopping call back function". When the call back function indicates that no further improvements could be made from the training process, the training of the model could be stopped. We set the maximum number of epochs to 32 and performed our training. After the training process, we found that all

Algorithm 1 Hybrid BiCuDNNLSTM-1dCNN Prediction Model

```
1: Input: Open Price of Stocks
2: Bidirectional CuDNNLSTM Layer=B; CNN layer=C; DropOut Layer=D; Dense Layers=De; epochs=e; batch-size=b; weights=w; bias=bi;
3: for trial =1 : 5 do
4:     for epochs =1 : e do
5:         for batch-size =1 : b do
6:             for i in range(hp.Int(num-layers, 2, 20)) do
7:                 if select.Layer[B] == Bidirectional CuDNNLSTM then
8:                     Randomly generate w and bi of BiCuDNNLSTM;
9:                     Extract Features;
10:                    Compute Hidden Layers of BiCuDNNLSTM;
11:                end if
12:                if select.Layer[D] == DropOut Layer then
13:                    Decrease Interdependent learning between nodes;
14:                end if
15:                if select.Layer[C] == Conv1D layer then
16:                    Randomly generate w and bi of CNN;
17:                    Activation function = ReLU
18:                end if
19:                if select.Layer[C] == MaxPooling1D Layer then
20:                    Shape the obtained features;
21:                end if
22:                if select.Layer[C] == Flatten Layer then
23:                    Convert the Two Dimensional features into one dimensional features;
24:                end if
25:                if select.Layer[De] == Dense Layer (Fully connected) then
26:                    units=hp.Int(units, min-value=32, max-value=512)
27:                    Optimizer = Adam
28:                    Activation function = ReLU
29:                    Choose Learning rate(1e-2, 1e-3, 1e-4)
30:                end if
31:                if select.Layer[De] == Dense Layer (Output) then
32:                    Activation function = ReLU
33:                    Optimizer = Adam
34:                end if
35:            end for i
36:        end for batch-size
37:    end for epoch
38:    TrainedModel[trial]=the trained hybrid BiCuDNNLSTM-1dCNN model
39: end for trial
40: Return the trained model from TrainedModel[1..5] such that it has the least MAE value
```

training are stopped at a maximum of 25 to 30 epochs. We feel that the use of a maximum of 32 epochs is justified.

(c) Batch size (*batch-size*): It defines the total input samples to work through before updating any parameters in the internal model. We set this value to 64 in our experiments (line 5). In fact, we have tried to train our models with a batch size of 32 and 128. However, the relevant trained models perform worse than that of 64. Furthermore, this is consistent with the work done by (Fawaz et al., 2020; Kandel & Castelli, 2020; Lu et al., 2021; Yang et al., 2020).

(d) Number of hidden layers (*num-layers*): This is the number of hidden layers between the input and output layers. For each trial, our algorithm randomly generates a number between 2 and 20 (line 6), both inclusive, and uses that number as the number of hidden layers in that particular trial. There are two reasons for this. First, according to (Yadav et al., 2019), if the number of hidden layers is greater than one, the trained model is more stable. Second, previous researchers set this number to 2 in (Yang et al., 2020), 5 (Bao et al., 2017), a random number between 1 and 7 (Yadav et al., 2019) and 10 (Singh & Srivastava, 2017) in their experiments, and they reported that their trained models are performing well for prediction.

(e) Dropout Layer: The dropout layer is a technique for regularization in DL model which facilitates in decreasing interdependent learning between the nodes. It is used to prevent model from overfitting and this layer is determined by a number between 0 and 1. This number is referred to as the *dropout rate*. In our experiments, we used a dropout rate of 0.2 or 20% (line 12). This is similar to the work in (Jain et al., 2018; Rizwan et al., 2019; Srivastava et al., 2014).

(f) Optimizer: This is the optimization function used to obtain best results. In our experiments, we used the Adam optimizer (line 27) because it is computationally efficient and is well suited for large datasets (Kingma & Ba, 2015). Moreover, Adam optimizer performs well with LSTM-based prediction problems (Chang et al., 2019).

(g) Learning rate: This hyperparameter is applied for accurate model convergence on a prediction. In our experiments, we allow the algorithm to randomly choose one of 0.01, 0.001 and 0.0001 as the learning rate (line 29). This is similar to those in (Houdt et al., 2020; Lu et al., 2021; Singh & Srivastava, 2017). Furthermore, Hastie et al. (2009) suggested to choose a learning rate smaller than 0.1. Our values are definitely smaller than 0.1.

After obtaining several trained models, we can then select the one that has the best prediction accuracy in terms of some evaluation criteria. In our experiments, the trained model having the smallest mean absolute error, MAE, will be returned. The returned trained model is referred to as *the "best-trained-model-so-far"*. The formula for MAE is given by Eq. (8)

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{8}$$

where y_i denotes the actual values from the dataset and $\hat{y}_i$ denotes the predicted values from the trained model.

4. Model Evaluation: Once the "best-trained-model-so-far" is obtained, the testing dataset is then fit into the trained model to calculate the predicted values. These predicted values and their corresponding actual values in the testing dataset will then be used to evaluate the model's performance using some evaluation metrics. There are many evaluation metrics. For our experiments, we use root mean square error, RMSE, and mean absolute error, MAE. RMSE measures the differences between original and predicted values of the trained model. A small RMSE value signifies that the model is a good fit. The smaller the RMSE value, the better the model fits. RMSE has been used as a standard statistical metric to evaluate model performance in weather forecasting, air quality, and temperature research studies. The formula for RMSE is given by

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{9}$$

where y_i denotes the actual values and $\hat{y}_i$ denotes the predicted values from the trained model.

Our second evaluation metric is MAE. It is mainly used to balance and boost the confidence on the achieved values. The formula for MAE is mentioned above in Eq. (8).

5. Model Prediction: Once the model has been trained and the evaluation metric is small enough such that the users accept the trained model to be of good use, the trained model can be used for future prediction.

Table 1
Sample data from CL=F dataset.

Date	Open	High	Low	Close	Volume
09/07/2021	73.26	74.76	72.72	74.55	422 154
12/07/2021	74.73	74.93	73.16	74.09	397 931
13/07/2021	74.18	75.51	73.68	75.25	444 078
14/07/2021	75.16	75.44	72.20	73.12	528 360
15/07/2021	72.95	72.95	71.40	71.65	368 679
16/07/2021	71.48	72.30	70.41	71.80	368 679

4. Experiments

In order to demonstrate the effectiveness of our proposed hybrid model BiCuDNNLSTM-1dCNN, we performed experiments to compare our model with 4 other existing models, namely LSTM-CNN, LSTM-DNN, CuDNNLSTM and LSTM, using 5 different datasets. Details of the 5 datasets are presented in Section 4.1.

4.1. Datasets

In this research, five datasets have been used for experiments. They fall into two categories:- the individual stock item and the stock market's performance index. For individual stock item, we have three datasets:- (1) Crude Oil (30/03/1983 to 15/08/2018), (2) Crude Oil (CL=F[1]) (23/08/2000 to 15/01/2021) and (3) Global X DAX Germany ETF (DAX) (23/10/2014 to 31/12/2020). For stock market's performance index, we have two datasets:- (4) DAX Performance-Index (^GDAXI) (01/01/ 2000 to 16/06/2021) of Frankfurt Stock Exchange and (5) Hang Seng Index (^HSI) (01/01/2000 to 25/06/2021) of Hong Kong Stock Exchange. The first Crude Oil price dataset is obtained from Metastock.com by paid subscription. The remaining four datasets (that is, CL=F, DAX, ^GDAXI and ^HSI) are publicly available on YAHOO finance.

Table 1 shows a sample dataset of the Crude Oil (CL=F) from YAHOO finance. The "Open" price is the first transaction price after the opening of the stock exchange for the trading day. The "High", and "Low", prices are respectively the highest and lowest prices of a stock in the trading day. The "Close" price represents the last transaction price before closing of stock exchange at the end of the trading day. The "Volume" indicates the overall number of stocks traded on a

[1] This is the code of the stock in YAHOO Finance, https://finance.yahoo.com. Same as the other datasets, DAX, ^GDAXI and ^HSI.

day. We use only the raw time series data (Date, Open, High, Low, Close, Volume; or simply, OHLCV) for stock price prediction in our experiments.

4.2. Experimental processes and settings

Below is an outline of our experimental processes from pre-processing the data, fitting data into the DL model, and finally evaluating the trained model. As a reminder, we use the term "model" to mean the relevant DL model (e.g. BiCuDNNLSTM-1dCNN) and the term "trained model" to mean the results of the training process performed by the relevant DL model.

First, for each dataset, we pre-process the data to (1) remove any empty or noisy values and (2) normalize the raw data, as discussed in Section 3.4, in order to obtain better prediction results. Second, the pre-processed dataset is then split into the training and testing datasets with a ratio of 90% to 10% for model training. Third, we prepare each of the five DL models (that is, BiCuDNNLSTM-1dCNN, LSTM-CNN, LSTM-DNN, CuDNNLSTM and LSTM) for training with their individual hyperparamaters or parameters. Details on these values are discussed in Section 4.3. Moreover, for our BiCuDNNLSTM-1dCNN model, we set the lookback window using a time step of 50. Fourth, the training dataset will then be fit into each DL model for training. Finally, after obtaining a best trained model, the testing dataset will then be used to evaluate the performance of the trained model using two evaluation metrics, namely RMSE and MAE.

In this study, all five hybrid DL models under investigation are trained and tested using 'Python 3.8.5' in Keras. Moreover, we use cuDNN library GPU based Tensor flow to facilitate parallel processing and high-speed matrix calculation on Google Collaboratory. It is worth revealing that the computer hardware was made available by Google's Collaboratory, a free python atmosphere that involves slight setup and runs completely in the cloud. The experimental work is conducted on a PC client with i7-8565U CPU, 1.99 GHz processor and eight GB RAM.

4.3. Models under experimentation

Table 2 shows the five models (i.e. BiCuDNNLSTM-1dCNN, LSTM-CNN, LSTM-DNN, CuDNNLSTM and LSTM) studied in our experiments and their related hyperparameters values. For example, for our BiCuDNNLSTM-1dCNN model, the settings of the relevant parameters are:- (1) there is one bidirectional cuDNNLSTM layer with 512 neurons in the layer, (2) the dropout rate is 0.2 in the dropout layer, (3) the convolution (Conv) layer has a filter size of 64 and kernel size of one, (4) the maximum pooling (MaxPool) layer has a pool size of one, (5) there is one flatten layer, (6) there is one dense layer, the number of neurons in the layer is randomly selected between given range 32 to 512 (both inclusive) and (7) there is one output layer. The learning rate of the Adam optimizer is randomly selected from 0.01, 0.001 and 0.0001. As a reminder, all five DL models uses 32 epochs, batch size of 64, ReLU as the activation function, MAE as the loss function, and Adam as the optimizer.

In order to have a fair comparison among these five DL models, we allow each DL model to be trained with same number of epochs, number of layers and batch sizes. After the training, the "best-trained-model-so-far" (in terms of having the smallest MAE) of a particular DL model will be used for comparison with each other.

Table 2
Description of the relevant parameters of the five models under investigation.

Algorithm	Layers	Neurons/Filter/ Kernal/Pool size
BicuDNNLSTM-1dCNN	Bidirectional cuDNNLSTM	(512)
	Dropout rate	(0.2)
	Conv layer	(64, 1)
	MaxPool layer	(1)
	Flatten layer	Default
	Dense layer	(32–512)
	Output layer	(1)
LSTM-DNN	LSTM layer	(32)
	Dropout rate	(0.2)
	Dense layer(2)[a]	(32, 8)[b]
	Output layer	(1)
LSTM–CNN	LSTM layer	(64)
	Dropout rate	(0.2)
	Conv layer	(64, 1)
	MaxPool layer	(1)
	Flatten layer	Default
	Dense layer(2)	(50, 50)
	Output layer	(1)
CuDNNLSTM	Bidirectional CuDNNLSTM	(512)
	Dropout rate	(0.2)
	Dense layer	(32–512)
	Output layer	(1)
LSTM	LSTM layer(2)	(32, 32)
	Dropout rate	(0.2)
	Output layer	(1)

All models use 32 Epochs, 64 as the Batch Size, ReLU as the Activation Function, Mean Absolute Error (MAE) as the Loss Function, and Adam as the Optimizer.
[a]This dense layer has two hidden layers.
[b]The first hidden layer has 32 neurons whereas the second layer has 8 neurons.

5. Results and observations

As mentioned in Section 4.1, we use two different categories of stock price datasets in our experiment, namely individual stock price dataset and stock market's performance index dataset. We report our findings in the following two subsections, first about individual stock price and then the stock market's performance index. To the best of our knowledge, this is the first time to evaluating performance of hybrid models using both categories of datasets. Last, but not least, we used RMSE and MAE to compare the performance of all the trained models involved in our experiments. That is, after the training process, the performance of these models is judged by their individual RMSE and MAE values. As a reminder, the smaller the RMSE (MAE) value is, the better performance the model has.

5.1. Performance on individual stock item dataset (Crude Oil, CL=F, DAX)

Figs. 4 and 5 present the RMSE and MAE, respectively, values of the relevant training datasets and testing datasets on the five DL trained models based on the Crude Oil, CL=F and DAX datasets. From these two figures, we have the following seven observations altogether:

1. From Fig. 4, we have the following three observations
   (a) BiCuDNNLSTM-1dCNN trained model has the least RMSE values for both of the training and testing datasets of the Crude Oil, CL=F and DAX, among all five trained models. Hence, in terms of prediction accuracy with respect to RMSE, BiCuDNNLSTM-1dCNN trained model performs the best among the five DL models under investigation with respect to our datasets.
   (b) For the remaining 4 DL trained models other than BiCuDNNLSTM-1dCNN, LSTM-CNN trained model performs better than the remaining three DL trained models with respect to RMSE for both training and testing datasets of Crude Oil, CL=F and DAX.
   (c) Results for the performance of LSTM-DNN, LSTM and CuDNNLSTM are not clear cut. For example, the LSTM-DNN trained model has smaller RMSE values when compared with the relevant LSTM and CuDNNLSTM trained models with training datasets of the Crude Oil and CL=F. On the other hand, it has a higher RMSE value when compared with LSTM trained model and a smaller RMSE value than the CuDNNLSTM trained model.

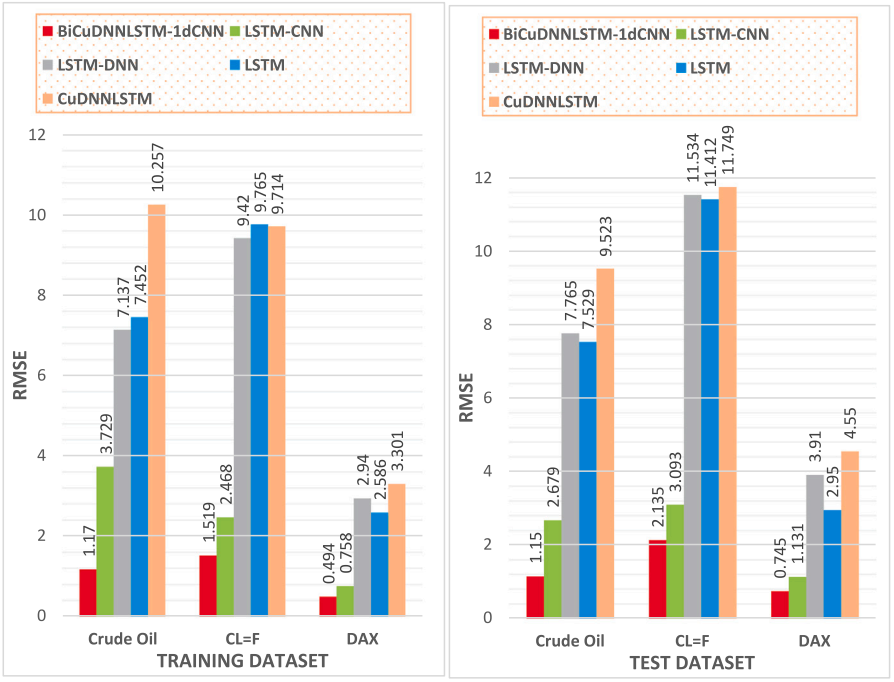

Fig. 4. RMSE values of five DL trained models, individual stock item.

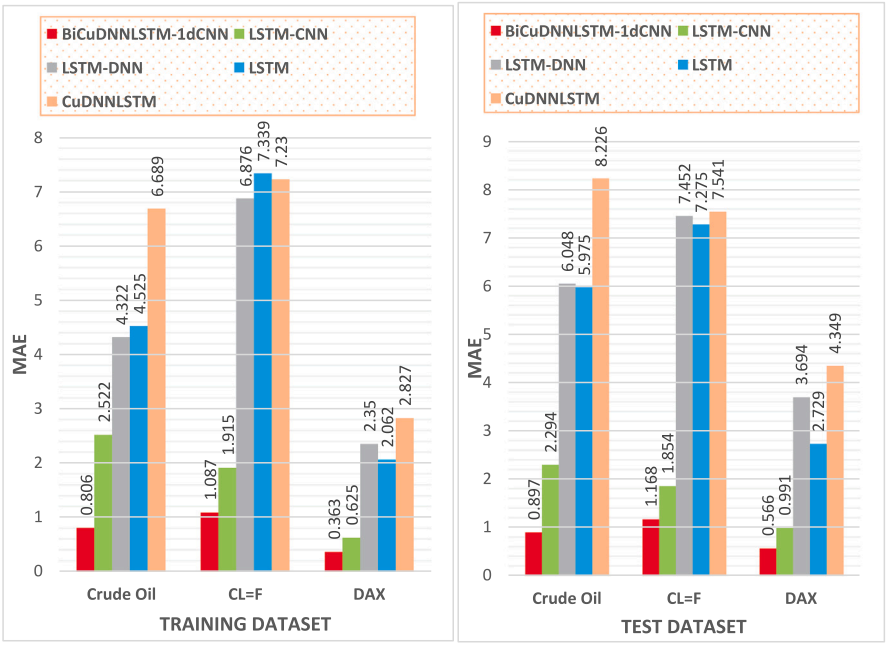

Fig. 5. MAE value of five DL trained models, individual stock item.

2. From Fig. 5, we have the following three observations, which are similar to the case of RMSE

   (a) BiCuDNNLSTM-1dCNN trained model has the least MAE values for both of the training and testing datasets of the Crude Oil, CL=F and DAX, among all five trained models. Hence, in terms of prediction accuracy with respect to MAE, BiCuDNNLSTM-1dCNN trained model performs the best among the five DL models under investigation with respect to our datasets.

   (b) For the remaining 4 DL trained models other than BiCuDNNLSTM-1dCNN, LSTM-CNN trained model performs better than the remaining three DL trained models with respect to MAE for both training and testing datasets of Crude Oil, CL=F and DAX.

   (c) The performance of LSTM-DNN, LSTM and CuDNNLSTM, the results are not clear cut. For example, the LSTM-DNN trained model has smaller MAE values when compared with the relevant LSTM and CuDNNLSTM trained models in case of training datasets of the Crude Oil and CL=F. On

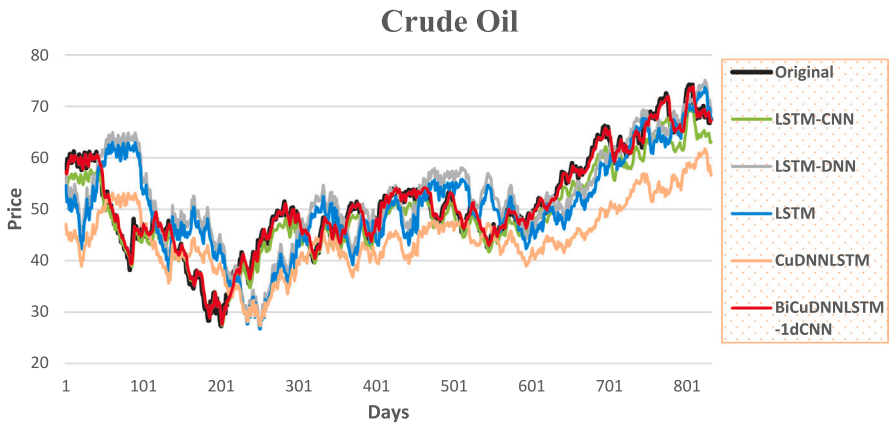

Fig. 6. Comparison of original and predicted values of 5 DL trained models, Crude Oil.

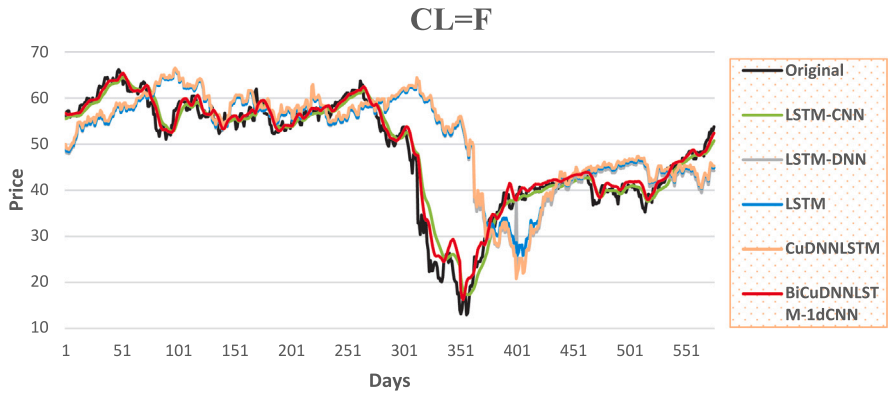

Fig. 7. Comparison of original and predicted values of 5 DL trained models, CL=F.

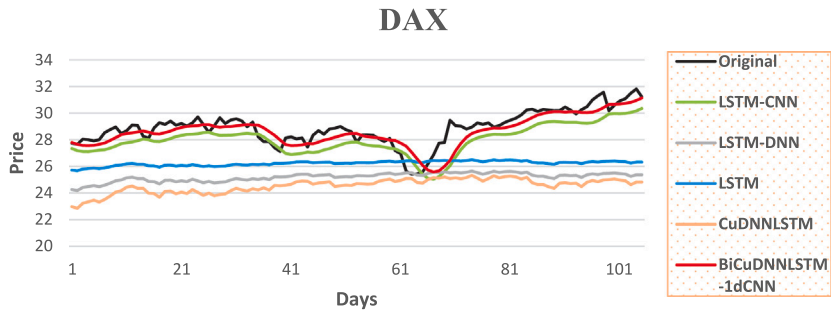

Fig. 8. Comparison of original and predicted values of 5 DL trained models, DAX.

the other hand, it has a higher MAE value when compared with LSTM trained model and a smaller MAE value than the CuDNNLSTM trained model.

3. The CuDNNLSTM model has the largest RMSE and MAE values on 10 out of 12 situations and the second largest RMSE values on the remaining two situations. As a result, CuDNNLSTM model performs the worst in the prediction of individual stock prices.

It will be interesting to know how the original stock price compare to the predicted values of these five DL trained models under investigation on a daily basis. Figs. 6, 7 and 8 plot the original values and the predicted values of the five DL trained models against time for Crude Oil, CL=F and DAX datasets, respectively. From these three figures, we have the following five observations:

1. From Fig. 6, we have the following three observations with respect to the Crude Oil dataset

(a) The line for the predicted values of the proposed BiCuDNNLSTM-1dCNN model (in short, the BiCuDNNLSTM-1dCNN line) follows the line for the original values (in short, the original line) very closely. The two lines nearly overlap with each other. This indicates that the predicted value is very close to original prices. Hence, the proposed BiCuDNNLSTM-1dCNN model has a very good prediction accuracy.

(b) The line for the LSTM-CNN trained model (in short, the LSTM-CNN line) also follows the original line very closely but not as close as the BiCuDNNLSTM-1dCNN line. Hence, this explains why the performance of LSTM-CNN is still good but not as good as the proposed BiCuDNNLSTM-1dCNN model.

(c) The lines for the remaining three models deviate significantly from the original line a lot. As a result, the performance of the remaining three models are not as

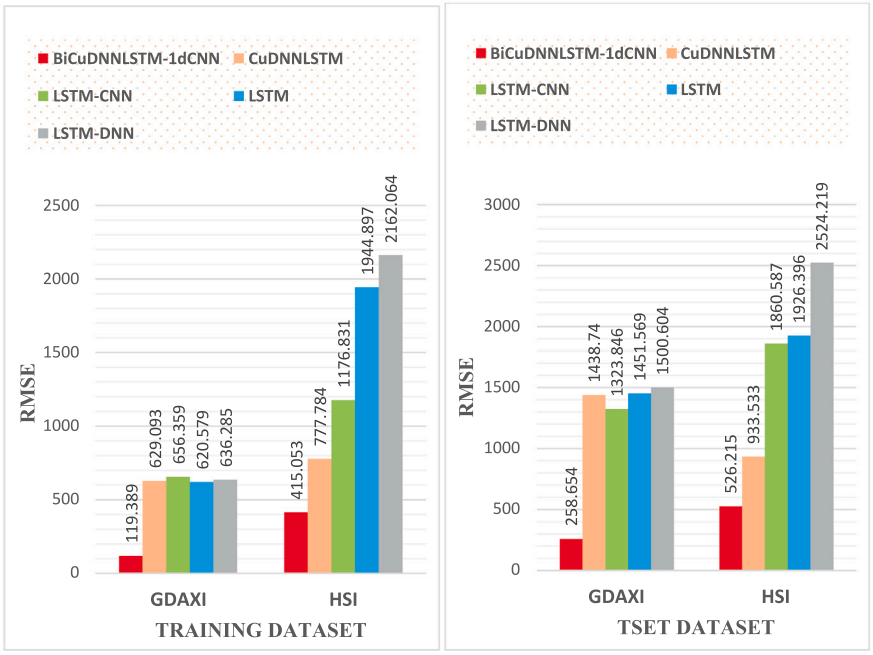

Fig. 9. RMSE values of five DL trained models, stock market performance index.

good as the proposed BiCuDNNLSTM-1dCNN model and the LSTM-CNN model.

2. From Figs. 7 and 8, we have similar observations as the above three for the CL=F and DAX datasets.
3. From Fig. 8, all models' predicted lines deviate from the original line. However, the BiCuDNNLSTM-1dCNN line deviates the least. Hence, it is still considered to be good but not as good as those in the Crude Oil and CL=F datasets. This may be because of the lack of sufficient data for DL model training purposes. In fact, this is one of the short coming of any DL related training models. In our experiments, we only have about 100 day's data for DAX whereas we have over 500+ day's data for both Crude Oil and CL=F datasets.

In summary, our proposed BiCuDNNLSTM-1dCNN model performs the best among the five DL models under investigation with respect to RMSE and MAE, based on the Crude Oil, CL=F and DAX (training and testing) datasets. Furthermore, the prediction accuracy of the BiCuDNNLSTM-1dCNN model is better than the other four DL models under investigation.

5.2. Performance on stock market's performance index (^GDAXI and ^HSI)

Similar to the situation for individual stock prices, we first present the results based on RMSE and MAE values of the five DL trained models.

Figs. 9 and 10 present the RMSE and MAE, respectively, values of the relevant training datasets and testing datasets on the five DL trained models for ^GDAXI and ^HSI datasets. From these two figures, we have the following seven observations:

1. From Fig. 9, we have the following two observations
   (a) BiCuDNNLSTM-1dCNN trained model has the least RMSE values for both of the training and testing datasets of ^GDAXI and ^HSI datasets, among all five DL trained models. Hence, in terms of prediction accuracy with respect to RMSE, BiCuDNNLSTM-1dCNN trained model performs the best among the five DL models under investigation with respect to our datasets.
   (b) For the remaining 4 DL trained models other than BiCuDNNLSTM-1dCNN, the results are not clear cut. For example, the LSTM trained model performs better than the remaining three DL trained models with respect to RMSE for the ^GDAXI training datasets but not for the ^HSI training datasets (actually in second last place). Furthermore, for the ^GDAXI testing datasets, LSTM-CNN trained model is the best among the four whereas for ^HSI testing dataset, the CuDNNLSTM trained model is better than the remaining three.
2. From Fig. 10, we have the following three observations
   (a) BiCuDNNLSTM-1dCNN trained model has the least MAE values for both of the training and testing datasets of ^GDAXI and ^HSI, among all five DL trained models. Hence, in terms of prediction accuracy with respect to MAE, BiCuDNNLSTM-1dCNN trained model performs the best among the five DL models under investigation with respect to our datasets.
   (b) For the remaining 4 DL trained models other than BiCuDNNLSTM-1dCNN, CuDNNLSTM trained model performs better than the remaining three DL trained models with respect to MAE for both of the relevant training and testing datasets except for the case of ^GDAXI testing dataset, in which LSTM-CNN has a smaller RMSE value.
   (c) The performance of LSTM-CNN, LSTM-DNN and LSTM, the results are not clear cut. For example, for the ^GDAXI training dataset, the LSTM trained model has smaller MAE values when compared with the relevant LSTM-CNN and LSTM-DNN trained models. On the other hand, for the ^HSI training dataset, its performance is better than the LSTM-DNN trained model but worse than the LSTM-CNN model.
3. For the case of ^GDAXI and ^HSI (training and testing) datasets, all RMSE and MAE values of our BiCuDNNLSTM-1dCNN model are significantly smaller than the other 4 DL trained models. For example, for the ^GDAXI testing dataset, the MAE value (approximately 188) of BiCuDNNLSTM-1dCNN mode is nearly

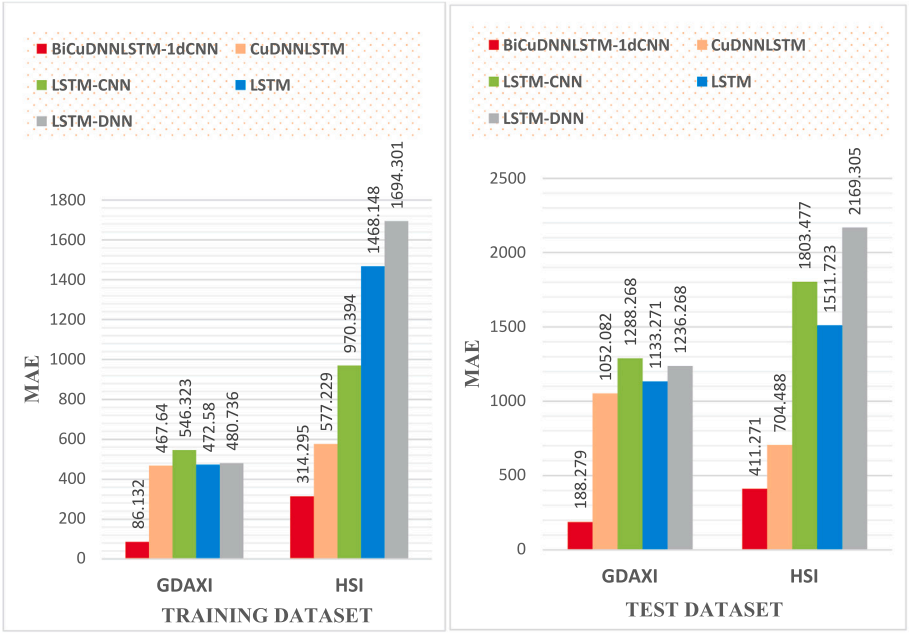

Fig. 10. MAE values of five DL trained models, stock market performance index.

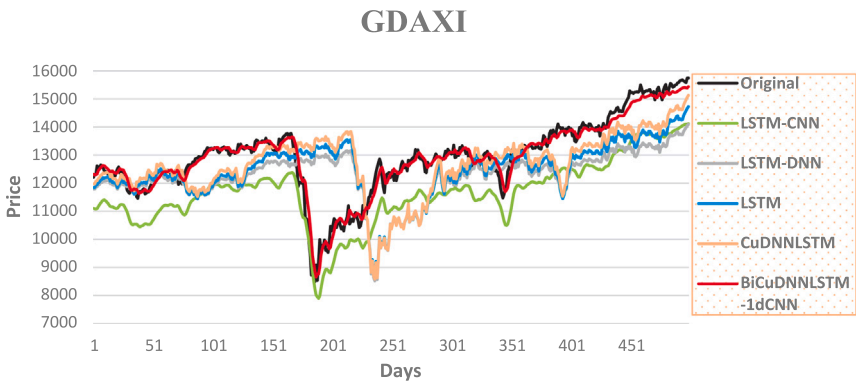

Fig. 11. Comparison of original and predicted values of 5 DL trained models, ^GDAXI.

18% of the second smallest MAE value (approximately 1052) obtained from the CuDNNLSTM trained model. For the ^HSI testing dataset, the MAE value (411) of the BiCuDNNLSTM-1dCNN model is nearly 58% of that of the second smallest MAE value (704) as obtained from the CuDNNLSTM model.

4. The LSTM-DNN model has the largest RMSE values on five out of eight situations and the second largest values on the remaining three situations. Hence, LSTM-DNN model performs the worst in predicting stock market's performance index.

Similar to the situation for individual stock item, Figs. 11 and 12 plot the original values and the predicted values of the five DL trained models against time for ^GDAXI and ^HSI datasets, respectively. We have the following five observations.

1. From Fig. 11, we have the following two observations with respect to the ^GDAXI dataset.

   (a) The BiCuDNNLSTM-1dCNN line follows the original line very closely. The two lines nearly overlap with each other almost everywhere except for the days after 401. This indicates that the predicted values from our BiCuDNNLSTM-1dCNN model is very close to the original values. Hence, the proposed BiCuDNNLSTM-1dCNN model has a very good prediction accuracy.

   (b) The lines for the remaining four models deviate significantly from the original line and these lines are further away than the BiCuDNNLSTM-1dCNN line. As a result, the performance of the remaining four models are not as good as the proposed BiCuDNNLSTM-1dCNN model.

2. From Figs. 12, we have the following three observations with respect to the ^HSI dataset.

   (a) The BiCuDNNLSTM-1dCNN line follows the original line very closely. In fact, the two lines almost overlap with each other. This indicates that the predicted value is very close to original prices. Hence, the proposed BiCuDNNLSTM-1dCNN model has a very good prediction accuracy.

   (b) The CuDNNLSTM line follows the original line as well but is further away from the BiCuDNNLSTM-1dCNN line. Hence, the performance of the CuDNNLSTM trained model is worse than that of the BiCuDNNLSTM-1dCNN model.

   (c) The lines for the remaining three models deviate further away from the BiCuDNNLSTM-1dCNN and the CuDNNLSTM lines. As a result, the performance of the remaining three models are not as good as the proposed BiCuDNNLSTM-1dCNN model.

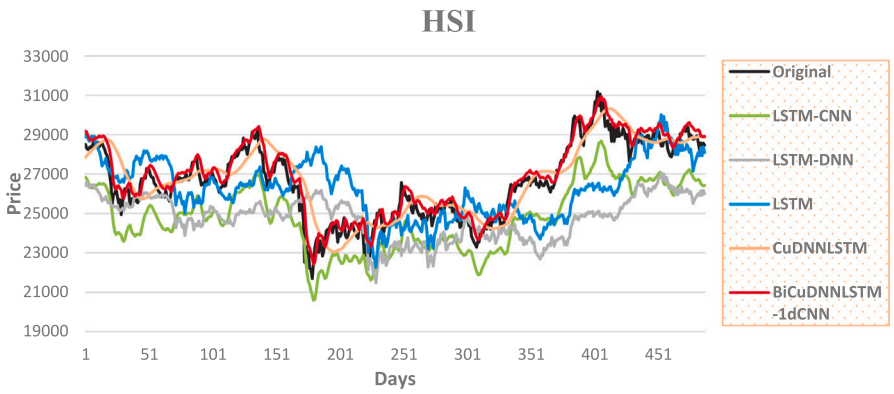

Fig. 12. Comparison of original and predicted values of 5 DL trained models, ^HSI.

Table 3
Training time in seconds per trial.

Date	BiCuDNNLSTM -1dCNN	LSTM -DNN	LSTM -CNN	CuDNN LSTM	LSTM
Crude Oil	106	67	96	40	48.2
CL=F	102	40.6	51.6	21	25.8
DAX	16	13.1	29.7	9	13
^GDAXI	78	42.9	62	23	45.7
^HSI	77	41.5	42.6	22	29.5

In summary, our proposed BiCuDNNLSTM-1dCNN model performs the best among the five DL models under investigation with respect to RMSE and MAE, based on the ^GDAXI and ^HSI (training and testing) datasets. Furthermore, the prediction accuracy of the BiCuDNNLSTM-1dCNN model is better than the other four DL models under investigation. It is evident from the above five figures that our proposed BiCuDNNLSTM-1dCNN model can predict stock prices accurately. This can serves as a guide for investors to make informed investment decisions. Last, but not least, our proposed BiCuDNNLSTM-1dCNN model always delivers the best results for predicting both the price of individual stock item and stock market's performance index. However, for other models, the performance varies. For example, for individual stock item's prediction, the second best model is LSTM-CNN and the worst model is CuDNNLSTM. However, for predicting stock market's performance index, the second best model is CuDNNLSTM and the worst model is LSTM-DNN.

5.3. Timing consideration

In general, hybrid DNN models require more time for training as compared with a single pure ML model. Interested readers may want to know whether the extra time is well spent on the training using a particular model hoping to obtain better prediction accuracy.

Table 3 shows the training time in seconds per trial for each of the five DL models involved in our experiments using each of the five individual datasets. From the table, we have the following three observations:

1. The training time of our BiCuDNNLSTM-1dCNN model ranges from 16 s (DAX) to 106 s (Crude Oil).
2. The training time of the LSTM-CNN model ranges from 29.7 s (DAX) to 96 s (Crude Oil). Related to the relative timing, our model is approximately 1 to 2 times of the LSTM-CNN model.
3. The training times of the LSTM-DNN, LSTM and CuDNNLSTM models range from 13.1 s (DAX) to 67 s (Crude Oil), from 13 s (DAX) to 48.2 s (Crude Oil), and from 9 s (DAX) to 40 s (Crude Oil), respectively. Related to the relative timing, our model is approximately 1 to 5 times that of these three models.

As a result, the training time of our BiCuDNNLSTM-1dCNN model per trail is approximately 106 s at most. With respect to the relative timing, our BiCuDNNLSTM-1dCNN model is approximately 1 to 5 times that of the other four models. However, judging from the relative prediction accuracy from Figs. 4, 5, 9 and 10, our prediction accuracy is approximately 1 to 9 times more accurate than the other four models.

6. Discussion: Threat to validity

In this section, we respond to internal validity and external validity of few threats to validity of our research. An external validity threat is that our stock price prediction required sufficient large amount of time series data for accurate prediction results. This has been observed in our results on the DAX dataset. In fact, this is a disadvantage of nearly all DL model training.

There are three internal validity threats. First, selection of dataset for training to mine historical patterns is important to our work. In order to overcome this, we use different values for partition of datasets and select the best option for our model training. Second, training the models with different values of the hyperparameters may yield different results. However, selecting hyperparameters is very time consuming and subjective because there is still no optimal selection strategy known to the research communities. In order to overcome this, we trailed our proposed model using different combinations of hyperparameters (e.g. the number of epochs, the batch size, the number of hidden layers, the dropout rate and the learning rate) and perform various trial experiments, aiming to find a better model. For example, in order to determine the batch size, we trained our model using 32, 64 and 128. We found that a batch size of 64 gives a better result than those using 32 and 128. We then decided to use 64 as the batch size in our experiment. After many such attempts, we then finalize the values of the hyperparameters of our model for our training, as explained earlier in Section 3.4. These values are reported in Table 2. As mentioned earlier, in each training, we use random search technique to find a better trained model, aiming to achieve the least error. After the training process, the returned model is considered as the "best-trained-model-so-far" for our experimental comparison. Third, as we only perform our experiments using five datasets, our results may be limited and should not be over generalized. However, as there are hundreds of thousands of stock price items in the current financial market and the financial stock data increases on a daily basis, it is practically infeasible for us to complete the analysis of all stock price items in the financial. Nonetheless, we have tried our best to cover two different categories of stock prices, namely individual stock item and stock market's performance index. Hopefully, this will provide the readers some ideas on the wide applicability of our prediction model in real life application.

7. Conclusion

In this paper, we propose a hybrid DL model to perform stock price prediction. Our proposed model, BiCuDNNLSTM-1dCNN, integrates two DL models, namely the bidirectional CuDNNLSTM and the CNN model, together. CuDNNLSTM is cuda enabled GPU accelerated model used to learn long short-term memories from stock's data and CNN is applied to obtain the abrupt features from the stock's dataset. The hyperparameter used for evaluation of the best prediction model have also been discussed. We have performed experiments to compare our proposed model with other common DL models used for stock price prediction using five different datasets. These five datasets fall into two categories, namely individual stock price and the stock market's performance index. The experimental outcomes show that the BiCuDNNLSTM-1dCNN has the best prediction accuracy and outstanding performance compared to LSTM-DNN, LSTM-CNN, CuDNNLSTM and LSTM. The same model can also be applied to predict individual stock prices as well as stock market's performance indexes. It is challenging to accomplish high prediction accuracy by using only a single deep learning model, but hybrid can achieve better prediction accuracy than any single DL model. BiCuDNNLSTM-1dCNN is a suitable hybrid model for predicting stock prices and can provide the right direction for stockholders to exploit investment returns.

Future research work will be extracting more valuable features from multivariate datasets to further improve prediction accuracy. An innovative hybrid prediction model based on various types of DL models can be proposed to improve prediction accuracy. Future research work will study whether the model can be applied in different fields of time series forecasting, such as gold price forecasting, weather forecasting, earthquake forecasting and so on.

CRediT authorship contribution statement

Anika Kanwal: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Writing – original draft, Writing – review & editing. **Man Fai Lau:** Conceptualization, Methodology, Validation, Data curation, Supervision, Writing – original draft, Writing – review & editing. **Sebastian P.H. Ng:** Conceptualization, Supervision, Writing – review & editing. **Kwan Yong Sim:** Data curation, Supervision, Writing – review & editing. **Siva Chandrasekaran:** Conceptualization, Supervision, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank various colleagues in the Department of Computing Technologies to review the work and improve the article during its preparation.

Funding source

This research is partly supported by a Malaysia Ministry of Higher Education Fundamental Research Grant Scheme (FRGS/1/2015/ICT04/SWIN/02/1).

References

Althelaya, K. A., El-Alfy, E.-S. M., & Mohammed, S. (2018). Evaluation of bidirectional LSTM for short-and long-term stock market prediction. In *2018 9th International conference on information and communication systems*.

Appleyard, J., Kocisky, T., & Blunsom, P. (2016). Optimizing performance of recurrent neural networks on GPUs. arXiv preprint. https://arxiv.org/abs/1604.01946.

Aveleira-Mata, J., Ondicol-Garcia, J., Munoz-Castaneda, A. L., Garcia, I., & Benavides, C. (2019). Multiclass classification procedure for detecting attacks on MQTT-iot protocol. *Advances in Complex Systems and their Applications to Cybersecurity*, *2019*, http://dx.doi.org/10.1155/2019/6516253.

Baeka, Y., & Kim, H. Y. (2018). Modaugnet: A new forecasting framework for stock market index value with an overfitting prevention LSTM module and a prediction LSTM module. *Expert Systems with Applications*, *113*(15), 457–480. http://dx.doi.org/10.1016/j.eswa.2018.07.019.

Bakhoda, A., Yuan, G. L., Fung, W. W., Wong, H., & Aamodt, T. M. (2009). Analyzing CUDA workloads using a detailed GPU simulator. In *2009 IEEE International symposium on performance analysis of systems and software*. Boston, MA, USA.

Bao, W., Yue, J., & Rao, Y. (2017). A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PLoS One*, *12*(7), Article e0180944. http://dx.doi.org/10.1371/journal.pone.0180944.

Bergstra, J., Bardenet, R., Bengio, Y., & Kegl, B. (2011). Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems, vol. 24*.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*(2), 281–305.

Cao, J., Li, Z., & Li, J. (2019). Financial time series forecasting model based on CEEMDAN and LSTM. *Physica A: Statistical Mechanics and its Applications*, *519*, 127–139. http://dx.doi.org/10.1016/j.physa.2018.11.061.

Cao, J., & Wang, J. (2019). Stock price forecasting model based on modified convolution neural network and financial time series analysis. *International Journal of Communication System*, *32*(12), Article e3987. http://dx.doi.org/10.1002/dac.3987.

Chang, Z., Zhang, Y., & Chen, W. (2019). Electricity price prediction based on hybrid model of adam optimized LSTM neural network and wavelet transform. *Energy*, *187*, Article 115804. http://dx.doi.org/10.1016/j.energy.2019.07.134.

Chen, Y., Wu, J., & Bu, H. (2018). Stock market embedding and prediction: A deep learning method. In *2018 15th International conference on services systems and services management*.

Dargan, S., Kumar, M., Ayyagari, R. M., & Kumar, G. (2020). A survey of deep learning and its applications: a new paradigm to machine learning. *Archives of Computational Methods in Engineering*, *27*, 1071–1092. http://dx.doi.org/10.1007/s11831-019-09344-w.

Eapen, J., Bein, D., & Verma, A. (2019). Novel deep learning model with CNN and bi-directional LSTM for improved stock market index prediction. In *2019 IEEE 9th Annual computing and communication workshop and conference*.

Fawaz, H. I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A., & Petitjean, F. (2020). Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery Volume*, *34*, 1936–1962. http://dx.doi.org/10.1007/s10618-020-00710-y.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Fourteenth international conference on artificial intelligence and statistics*.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Neural Networks (Chapter 11).

Heaton, J. B., Polson, N. G., & Witte, J. H. (2017). Deep learning for finance: deep portfolios. *Applied Stochastic Models Business Industry*, *33*(1), 3–12. http://dx.doi.org/10.1002/asmb.2209.

Hiransha, M., E.A., G., Menon, V. K., & K.P., S. (2018). NSE stock market prediction using deep-learning models. *Procedia Computer Science*, *132*, 1351–1362. http://dx.doi.org/10.1016/j.procs.2018.05.050.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hossain, M. A., Karim, R., Thulasiram, R., Bruce, N. D. B., & Wang, Y. (2018). Hybrid deep learning model for stock price prediction. In *2018 IEEE Symposium series on computational intelligence*.

Houdt, C. G. V., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, *53*, 5929–5955. http://dx.doi.org/10.1007/s10462-020-09838-1.

Jain, S., Gupta, R., & Moghe, A. A. (2018). Stock price prediction on daily stock data using deep neural networks. In *2018 International conference on advanced computation and telecommunication*.

Jin, Z., Yang, Y., & Liu, Y. (2020). Stock closing price prediction based on sentiment analysis and LSTM. *Neural Computing and Applications Volume*, *32*, 9713–9729. http://dx.doi.org/10.1007/s00521-019-04504-2.

Kandel, I., & Castelli, M. (2020). The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express*, *6*(4), 312–315. http://dx.doi.org/10.1016/j.icte.2020.04.010.

Kim, S., & Kang, M. (2019). Financial series prediction using attention LSTM. https://arxiv.org/abs/1902.10877.

Kingma, D. P., & Ba, J. (2015). Adam: a method for stochastic optimization. In Y. Bengio, & Y. LeCun (Eds.), *3rd International conference on learning representations*.

Larochelle, H., Erhan, D., Courville, A., Bergstra, J., & Bengio, Y. (2007). An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on machine learning, New York, USA*.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324. http://dx.doi.org/10.1109/5.726791.

Livieris, I. E., Pintelas, E., & Pintelas, P. (2020). A CNN-LSTM model for gold price time-series forecasting. *Neural Computing and Applications*, *32*, 17351–17360. http://dx.doi.org/10.1007/s00521-020-04867-x.

Lu, W., Li, J., Li, Y., Sun, A., & Wang, J. (2020). A CNN-LSTM-based model to forecast stock prices. *Complexity*, *2020*, http://dx.doi.org/10.1155/2020/6622927.

Lu, W., Li, J., Wang, J., & Qin, L. (2021). A CNN-bilstm-AM method for stock price prediction. *Neural Computing and Applications*, *33*, 4741–4753. http://dx.doi.org/10.1007/s00521-020-05532-z.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *30th International conference on machine learning, Atlanta, Georgia, US*.

Montenegro, C., & Molina, M. (2019). A DNN approach to improving the short-term investment criteria for S & P500 index stock market. In *Third international conference on e-commerce, e-business and e-government*.

Narayanadoss, A. R., Truong-Huu, T., Mohan, P. M., & Gurusamy, M. (2019). Crossfire attack detection using deep learning in software defined ITS networks. In *2019 IEEE 89th vehicular technology conference*.

Passricha, V., & Aggarwal, R. K. (2010). Understanding the difficulty of training deep feedforward neural networks. In *13th International conferenceon artificial intelligence and statistics*.

Prechelt, L. (2012). Early stopping–but when?. In G. B. Orr, & K.-R. Müller (Eds.), *Neural networks tricks of the trade* (pp. 53–67). Springer.

Rizwan, M., Narejo, S., & Javed, M. (2019). Bitcoin price prediction using deep learning algorithm. In *2019 13th International conference on mathematics, actuarial science, computer science and statistics*.

Sajjad, M., Khan, Z. A., Ullah, A., Tanveer, H., Ullah, W., Lee, M. Y., & Baik, S. W. (2020). A novel CNN-gru-based hybrid approach for short-term residential load forecasting. *IEEE Access*, *8*, 143759–143768. http://dx.doi.org/10.1109/ACCESS.2020.3009537.

Schuster, M., & Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681. http://dx.doi.org/10.1109/78.650093.

Selvin, S., Ravi, V., Gopalakrishnan, E. A., Menon, V. K., & Kp, S. (2017). Stock price prediction using LSTM, RNN and CNN-sliding window model. In *2017 International conference on advances in computing, communications and informatics*.

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2020). Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, *90*, Article 106181. http://dx.doi.org/10.1016/j.asoc.2020.106181.

Shah, D., Campbell, W., & Zulkernine, F. H. (2018). A comparative study of LSTM and DNN for stock market forecasting. In *2018 IEEE International conference on big data(big data), Seattle, Washington, USA*.

Singh, R., & Srivastava, S. (2017). Stock prediction using deep learning. *Multimedia Tools and Applications*, *76*(18), 18569–18584. http://dx.doi.org/10.1007/s11042-016-4159-7.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958.

Tran, T. T. K., Lee, T., Shin, J.-Y., Kim, J.-S., & Kamruzzaman, M. (2020). Deep learning-based maximum temperature forecasting assisted with meta-learning for hyperparameter optimization. *Atmosphere*, *11*(5), 487. http://dx.doi.org/10.3390/atmos11050487.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Łukasz, Gouws, S., Kato, Y., Kudo, T., Kazawa, H., ... Dean, J. (2016). Google's neural machine translation system: bridging the gap between human and machine translation. https://arxiv.org/abs/1609.08144.

Yadav, A., Jhaa, C. K., & Sharanb, A. (2019). Optimizing LSTM for time series prediction in indian stock market. *Procedia Computer Science*, *167*, 2091–2100. http://dx.doi.org/10.1016/j.procs.2020.03.257.

Yan, H., & Ouyang, H. (2018). Financial time series prediction based on deep learning. *Wireless Personal Communications*, *102*, 683–700. http://dx.doi.org/10.1007/s11277-017-5086-2.

Yang, C., Zhai, J., & Tao, G. (2020). Deep learning for price movement prediction using convolutional neural network and long short-term memory. *Mathematical Problems in Engineering*, *2020*(6), 1–13. http://dx.doi.org/10.1155/2020/2746845.

Yong, X. B., Rozaini, M., & bin Abdullah, A. S. (2017). A stock market trading system using deep neural network. In *Asian simulation conference, Singapore*.