Hindawi
Journal of Control Science and Engineering
Volume 2021, Article ID 8785636, 23 pages
https://doi.org/10.1155/2021/8785636

Research Article

COVID-19 Pandemic Forecasting Using CNN-LSTM: A Hybrid Approach

Zuhaira M. Zain and Nazik M. Alturki

Information Systems Department, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia

Correspondence should be addressed to Zuhaira M. Zain; zuhaira.muhdzain@gmail.com

Received 6 April 2021; Revised 6 July 2021; Accepted 15 July 2021; Published 30 July 2021

Academic Editor: Mohamed Hamdy

COVID-19 has sparked a worldwide pandemic, with the number of infected cases and deaths rising on a regular basis. Along with recent advances in soft computing technology, researchers are now actively developing and enhancing different mathematical and machine-learning algorithms to forecast the future trend of this pandemic. Thus, if we can accurately forecast the trend of cases globally, the spread of the pandemic can be controlled. In this study, a hybrid CNN-LSTM model was developed on a time-series dataset to forecast the number of confirmed cases of COVID-19. The proposed model was evaluated and compared with 17 baseline models on test and forecast data. The primary finding of this research is that the proposed CNN-LSTM model outperformed them all, with the lowest average MAPE, RMSE, and RRMSE values on both test and forecast data. Conclusively, our experimental results show that, while standalone CNN and LSTM models provide acceptable and efficient forecasting performance for the confirmed COVID-19 cases time series, combining both models in the proposed CNN-LSTM encoder-decoder structure provides a significant boost in forecasting performance. Furthermore, we demonstrated that the suggested model produced satisfactory predicting results even with a small amount of data.

1. Introduction

The year 2020 witnessed the global spread of the coronavirus disease (COVID-19) pandemic [1]. As of September 29, 2020, the virus had infected over 33.2 million people and had killed over 1 million in more than 216 countries [2]. COVID-19 was first discovered by Chinese authorities in Wuhan City, China, on January 7, 2020, as the cause of a new type of pneumonia [1]. It was then identified as a member of the zoonotic coronavirus family [3]. COVID-19, which is highly infectious, quickly transmitted through close human-to-human contact. Therefore, to minimize the number of infected cases, many countries followed procedures that included quarantine, online schools and businesses, and bans on travel [4, 5].

Given the severity of the disease, identifying the COVID-19 spread rate is vital for governments. Hence, by knowing the spread rate at a given time, governments can act accordingly by planning public health and forming policies and strategies to minimize COVID-19 consequences [6, 7]. This can be achieved by performing COVID-19 tests on a large scale. However, as of April 23, 2020, no country was able to test more than 13.4% of their population [8]. Another method that identifies the COVID-19 spread rate is accurately predicting the figure of active cases at a specified time. However, COVID-19 cases are exponentially increasing and the data are nonlinear and nonstationary. Therefore, predicting the epidemic's future is challenging. As a result, the situational demand is to introduce an efficient model with the highest accuracy [9].

Several attempts have been made by various studies that applied mathematical and machine-learning predictive techniques to approximate the disease's spread and effects globally [10] or for specific countries, such as the United States of America (USA) [11–15], Italy [12–17], Spain [12–16], France [12, 13, 15, 16], Canada [13, 18], India

[11, 14, 19], Greece [20], Brazil [11, 13, 14, 21, 22], Mexico [13, 23], Hungary [24], Russia [12–14, 25], Saudi Arabia [26], United Kingdom (UK) [11–15], Germany [13, 14], Iran [13, 25], China [13, 14, 17], Peru [16], and Israel [14]. These studies applied various traditional time series predictive techniques, including statistical, linear, and machine-learning methods. For example, the application of AutoRegressive Integrated Moving Average (ARIMA) [12, 16, 19, 26], Holt-Winters Additive Model (HWAAS) [12], TBATS [12], Neural Basis Expansion Analysis for Interpretable Time Series Forecasting (N-Beats) [12], multilayer perceptron (MLP) neural network (NN) [10], and evolutionary modeling [13].

Unfortunately, statistical techniques generally make assumptions about the stationarity and linear correlation of historical data, whilst machine-learning methods seem incapable of detecting and recording the nonlinear and complex behavior of COVID-19 time series. Hence, none of the previous studies could ensure the creation of an accurate and robust COVID-19 forecasting model, as they concluded moderately deprived outcomes [27] or their predictions did not align with real data [28, 29]. In addition, according to [18], the current models have several defects, including being linear, nontemporal, and based on several assumptions. Moreover, it seems that they are unable to deal with noisy and chaotic time series data.

On the other hand, deep learning techniques have lately been effectively utilized to a range of difficult prediction issues encountered in the real world, including time-series forecasting [30–32]. Deep learning is often regarded as the most effective technique for dealing with the noisy and chaotic character of time series predicting issues, since it produces more accurate forecasts. One of the most efficient and extensively utilized deep learning approaches is long short-term memory (LSTM). LSTM has been applied by several studies to forecast COVID-19 cases [14, 17, 18, 21, 25]. LSTM models can easily capture sequence pattern information, but they are tailored to deal with temporal correlations and only use the features specified in the training set. Another popular deep learning method is convolutional neural networks (CNNs). CNN models are capable of filtering out noise in the input data and extracting more valuable knowledge for the final forecasting model. While vanilla CNNs are compatible for handling spatial autocorrelation data, they are seldom modified to handle complicated and lengthy temporal dependencies [33]. As a result, a time-series model that takes advantage of both deep learning techniques, i.e., LSTM and CNN could enhance forecast accuracy.

The primary goal of this study was to aid in the accurate forecasting of COVID-19 confirmed cases. We therefore proposed a hybrid CNN-LSTM forecasting model. Seventeen baseline predictive machine-learning models were also built in this study for comparison with our proposed model. The key contributions of this study are

(i) A hybrid CNN-LSTM model was proposed to combine the advantages of the CNN model, which is effective at filtering out noise in the input data, obtaining valuable information, and learning the time series internal representation, with the benefits of the LSTM model, which is effective at identifying and modeling short- and long-term temporal dependencies embedded in the data sequence.

(ii) On test and prediction data, the proposed model was assessed and compared to 17 baseline models. The findings indicate that our proposed CNN-LSTM model beats the other 17 models in predicting new confirmed cases with the lowest error value. In terms of RRMSE, the proposed hybrid model outperforms the standalone CNN-1D and LSTM models by 1.15 percent and 3 percent, respectively. Even with a small amount of data, this result demonstrates the efficacy of merging CNN-1D and LSTM models.

The remaining sections of this manuscript are as follows: Section 2 studies the related literature, Section 3 explains the materials and methods used in this research, Section 4 reports the main results and compares the forecasted trend to the actual trend, Section 5 reports the threats to validity, and Section 6 summarizes the study and suggests future works.

2. Related Works

Time-series prediction is a forecasting method that analyses historical data to capture the relationship and trends of a random variable. It will then be applied to forecast the value of that random variable in the future [34]. This method is particularly useful if the underlying distribution/process data generation is unknown or if there is no explanatory model capable of precisely linking the prediction variable with other explanatory variables. A great deal of effort and production of research has gone into the construction and advancement of time series forecasting techniques over the last several decades. The next paragraph summarizes many fruitful researches that demonstrate several models for forecasting COVID-19 cases.

Many researchers have employed the standard forecasting method with statistical modeling to predict COVID-19 outbreak [9]. For example, Ceylan [16] used ARIMA techniques to forecast the pattern of COVID-19 prevalence in France, Spain, and Italy from 21/2/2020 to 15/4/2020, using data from the World Health Organization (WHO) website. Several ARIMA regressors were built using various ARIMA parameters. They chose three different ARIMA regressors to predict the spread of COVID-19 for the three selected countries based on the lowest MAPE values. The ARIMA (0, 2, 1) was found to be the best model for Italy with MAPE = 4.7520, ARIMA (1, 2, 0) for Spain with MAPE = 5.8486, and ARIMA (0, 2, 1) for France with MAPE = 5.6335. Another study performed by Roy et al. [19] also used ARIMA to forecast the epidemiologic trend in the prevalence and incidence of COVID-19 using an Indian dataset from 30/1/2020 to 26/4/2020. ARIMA (2, 2, 2) was discovered to be the most reliable model for predicting COVID-19 events, with RMSE = 95.322 and MAE = 50.109. However, because COVID-19 data are nonlinear and nonstationary, the ARIMA model is not optimal with such

cases [35]. This applies to other statistical approaches. Statistical analysis can be used to perform modeling on known data. However, it struggles to grasp the complexities of the analyzed data when dealing with extremely complex models [10]. As a result, statistical tools are deemed inadequate for analyzing pandemic unpredictability, and generated models are difficult to generalize [9].

Alternatively, algorithms based on artificial intelligence (AI) learn from historical data to forecast future results. Machine-learning and deep-learning algorithms are two types of AI algorithms. It is a field that is focused on computer algorithms learning and developing on their own. Machine-learning-based forecasting regressors change their parameters to match their forecasts to the actual data. Some related studies that used machine-learning algorithms in forecasting the dispersion of COVID-19 disease are discussed in the following paragraphs.

Car et al. [10] implemented an MLP-ANN to forecast the number of COVID-19 deceased, recovered, and infected cases worldwide using a dataset taken from the Johns Hopkins University Center for Systems Science and Engineering (JHU CSSE) from 22/1/2020 to 12/3/2020. Their best models used the ReLU activation feature and have four hidden layers, each with four neurons, with coefficient of determination (R^2) = 0.98599 for infected patients, 0.99429 for deceased patients, and 0.97941 for recovered patients. Salgotra et al. [13] used genetic programming (GP) to model the possible effects of COVID-19 on confirmed and death cases in 15 of the world's most affected countries between January 2020 and May 2020. They discovered that the GP efficiency was superior, with RMSE and R^2 values close to 1. However, most studies that relied primarily on machine-learning models experienced underfitting or overfitting issues [36], limiting to retrospective analysis, or only projecting short-term trends due to noisy time series data or a lack of training data and appropriate features [37–39].

To address the aforementioned problems, time series forecasting has lately included deep learning algorithms [30–32], resulting in more accurate predictions. As one of the most successful deep learning methods, LSTM has been utilized to predict COVID-19 cases in many researches [14, 17, 18, 21, 25]. These studies revealed that LSTM models can easily capture sequence pattern information, but they are tailored to deal with temporal correlations and only use the features provided in the training set. Convolutional neural network (CNN) is another well-known deep learning and has also been applied in forecasting COVID-19 cases [40–42]. Results from these studies showed that CNN is excellent for filtering out noise in input data and extracting more beneficial features for the final forecasting model. Although standard CNNs are compatible to dealing with spatial autocorrelation data, they are seldom modified to cope with complicated and lengthy temporal dependencies [33]. Consequently, a time-series forecasting model that employs both deep learning methods, namely, LSTM and CNN, may improve prediction accuracy. A research performed by [43] used a hybrid CNN-LSTM model to determine whether individuals had COVID-19 disease based on lung ultrasound. The hybrid method provided the highest levels of accuracy, recall, and AUC. To the best of our knowledge, however, the CNN-LSTM technique has not been tested on COVID-19 time series data. As a result, in this study, we proposed using the hybrid CNN-LSTM model to predict the number of COVID-19 infected patients across the globe.

3. Materials and Methods

This section provides information on the study's materials and procedures.

3.1. Materials. The data for this analysis came from the WHO COVID-19 dashboard [2]. It contains information on coronavirus cases in each specific country, such as the number of confirmed, dead, cumulative confirmed, and cumulative deaths (defined by the name of the country, country code, and WHO region) every day from the beginning of the COVID-19 infections (4/1/2020) to 24/9/2020. The dataset contained 62,510 records for 216 different countries and 265 days, totaling 31,798,308 new cases and 973,653 death cases at the time this analysis was conducted. Figures 1 and 2 show the spatial distribution of cases that have accumulated at three different time stamps. The global scope of COVID-19 confirmed and death cases is depicted in Figure 3.

3.2. Methods. The proposed research method consists of three phases: Preparing time series data, building the predictive models, and applying the predictive model. We used the time series "New COVID-19 cases." The first phase "Preparing time series data" consists of three steps: Convert dataset into time series, normalize time series data, and split time series data. The second phase "Building the predictive model" consists of three steps: Optimize the models, train the models, and evaluate the models. The models were optimized to get the best hyperparameter. The models were then trained using the best hyperparameter on a train set, the time series of which starts on January 4, 2020, when the first case of COVID-19 started and ends on July 17, 2020. The trained models were then evaluated on a test set, the time series of which starts on July 18, 2020 and ends on August 14, 2020. The forecasting between July 18, 2020 and August 14, 2020 was estimated and compared with the real values. The third stage "Applying the predictive model" entails the information of COVID-19 starting on August 15, 2020 and ends on September 18, 2020. Then, the forecasting between September 12, 2020 and September 18, 2020 was estimated and compared with the real values. The second and third phases were applied 18 times, one time per predictive model (1 proposed model and 17 baseline models). Figure 4 depicts a high-level description of the entire procedure.

3.2.1. Phase 1: Preparing Time Series Data

Step 1.1: Convert dataset into time series

In this study, we used the "New_cases" (COVID-19 confirmed cases) dataset. To make the dataset a time

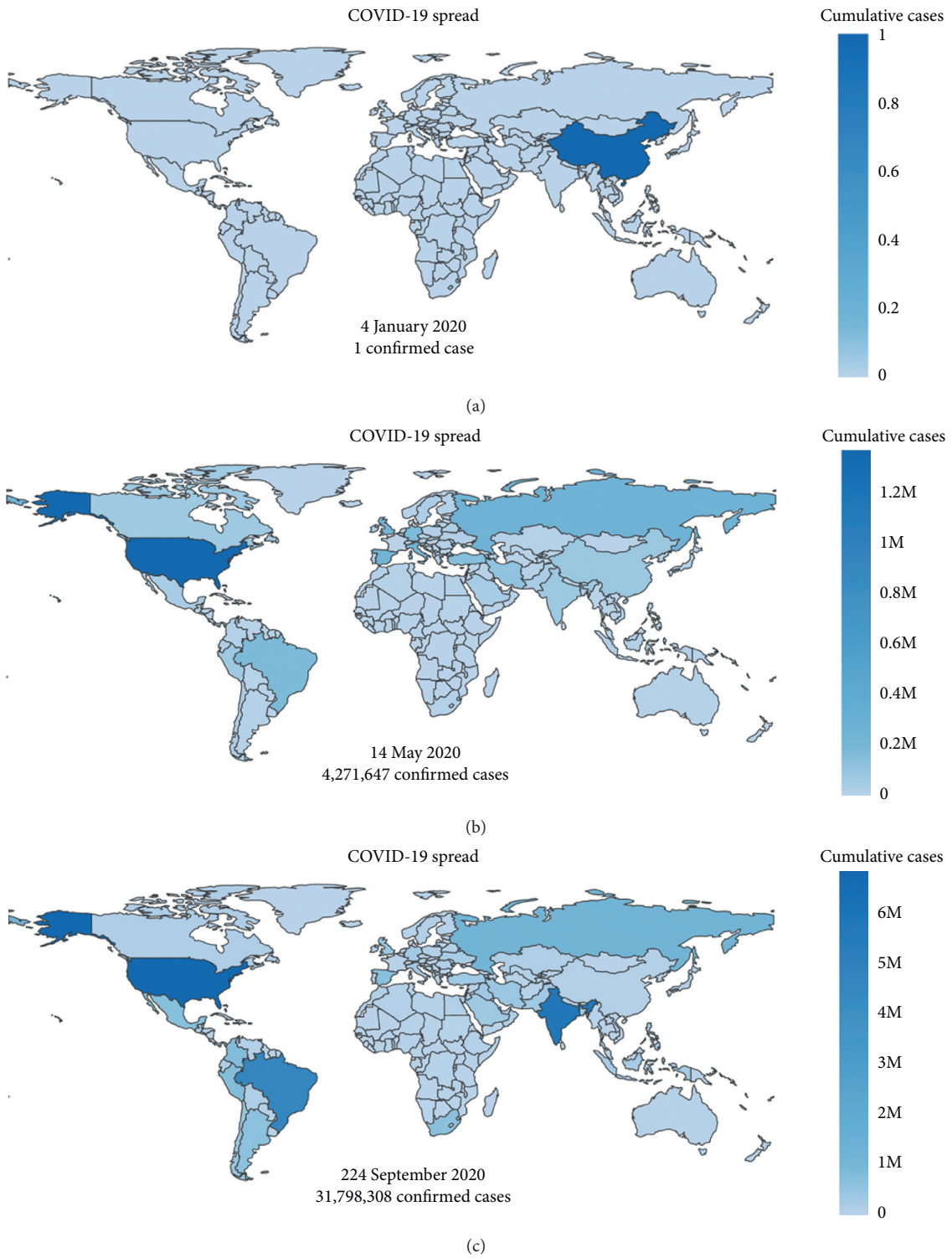


Figure 1: Geographical spread of the number of COVID-19 cumulative cases on January 4, 2020 (a), May 14, 2020 (b), and September 24, 2020 (c).

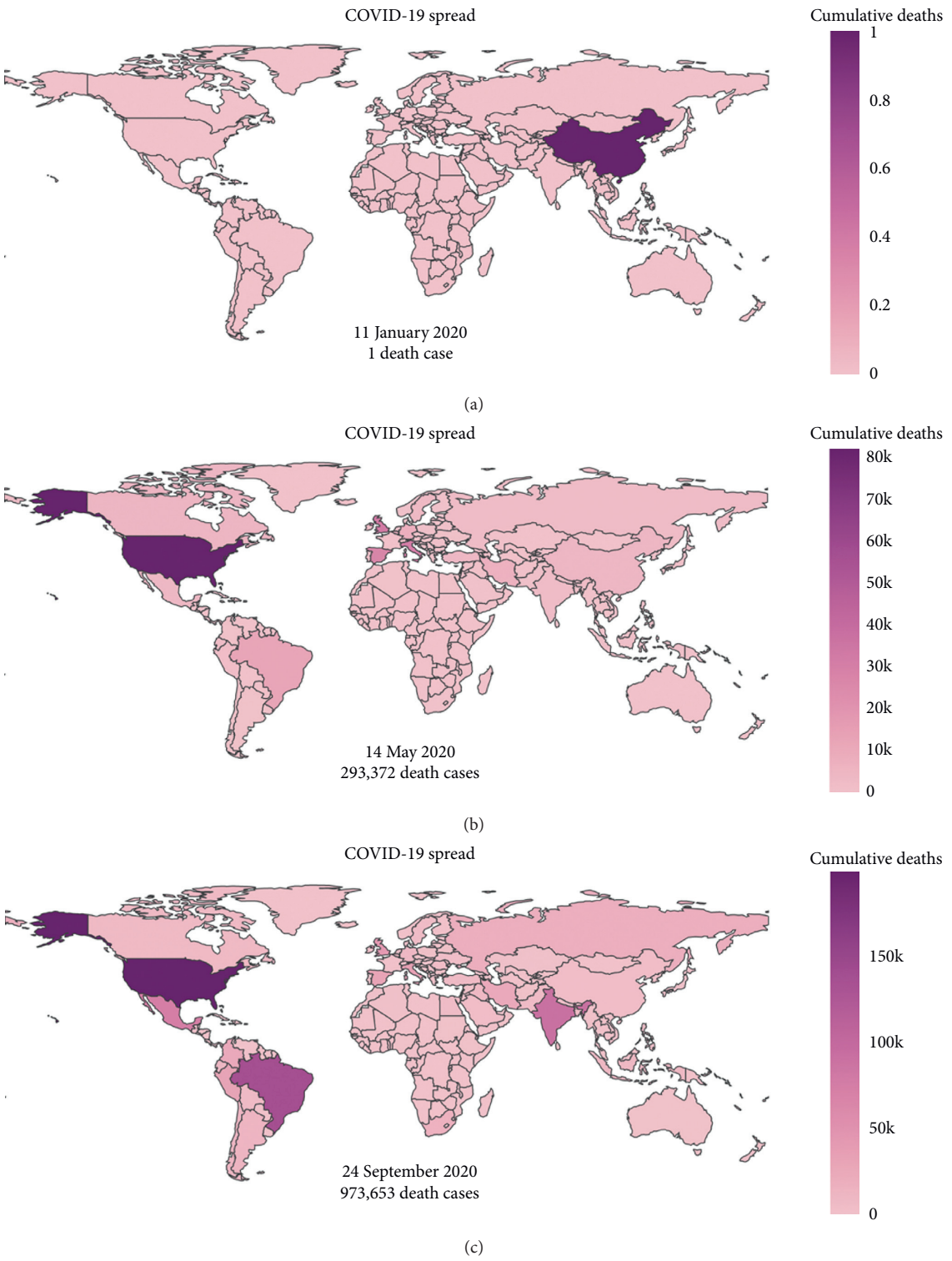


Figure 2: Geographical spread for the number of COVID-19 cumulative death cases on January 11, 2020 (a), May 14, 2020 (b), and September 24, 2020 (c).

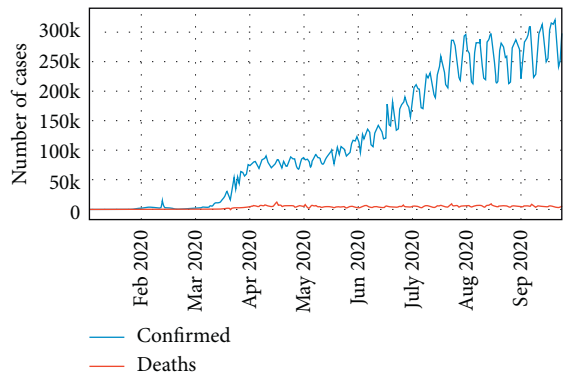


Figure 3: The spread of COVID-19 cases across the globe.

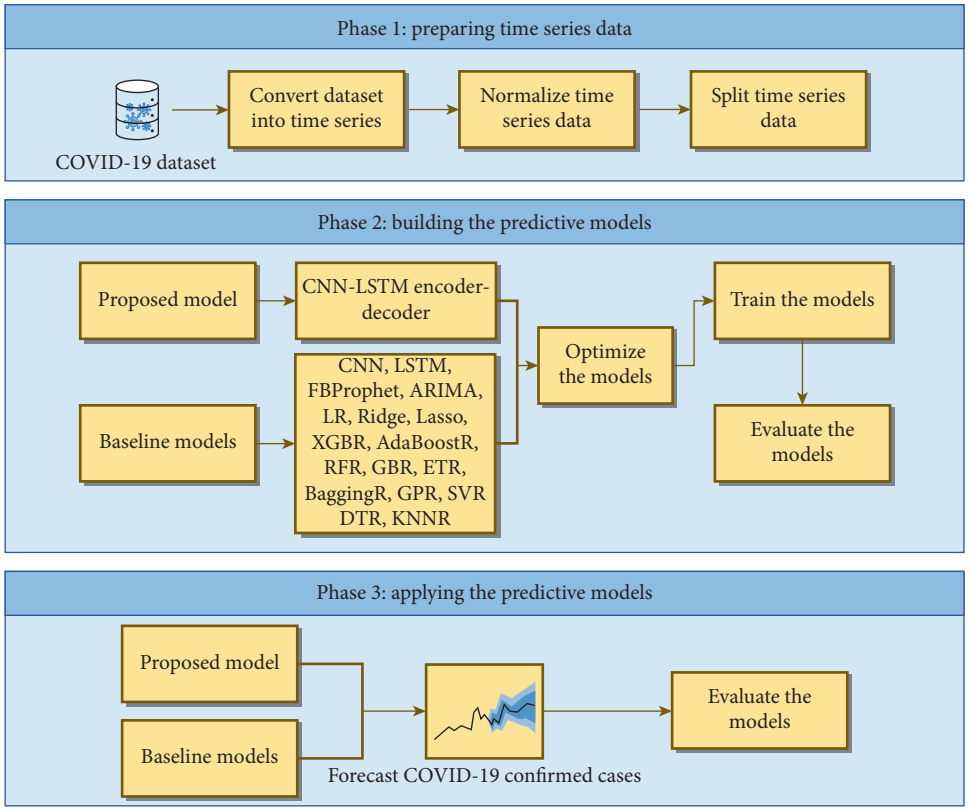


Figure 4: Research methods.

series, we set the date as *parse_dates* when opening the csv file. The loaded dataset was resampled and grouped by day using *resample()* function with the argument "D." The total of 265 data was divided into weeks. This gives 37 weeks of data.

Step 1.2: Normalize time series data

The time-series data from COVID-19 confirmed cases were normalized using min–max normalization within the range [0, 1]. This normalization must be reversed after predicting COVID-19 time series on testing data, such that the predicted data are similar to the original testing time-series data.

Step 1.3: Split time series data

Sequentially, to prepare the time series for model development, 37 weeks of the normalized time series data were divided into training, test, and forecast sets using the NumPy *split()* function. Table 1 shows the details of the output from the splitting procedure while Figure 5 visualizes them in graph.

3.2.2. Phase 2: Building the Predictive Models. In this study, we proposed a CNN-LSTM encoder-decoder model. The proposed model together with 17 baseline models were built

Table 1: Output of the splitting procedure.

Split	Data in days	Data in weeks	Date
Train	196	28	January 4, 2020–July 17, 2020
Test	28	4	July 18, 2020–August 14, 2020
Forecast	Input: 28 Forecast: 7	Input: 4 Forecast: 1	Input: August 15, 2020–September 11, 2020 Forecast: September 12, 2020–September 18, 2020

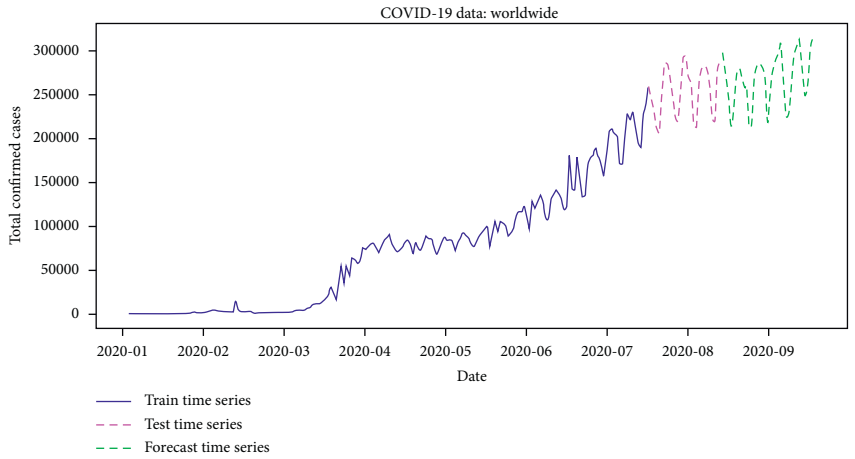


Figure 5: Train, test, and forecast time series.

by training and evaluating them on some standard performance metrics.

(1) Proposed Model. A hybrid deep learning architecture called CNN-LSTM was proposed to maximize the utilization of the CNN model for obtaining valuable knowledge and learning the time series internal representation with the efficacy of the LSTM model for detecting and modeling short- and long-term temporal dependencies embedded in the data sequence.

To do this, our suggested model, CNN-LSTM, is composed of two primary components: The first component, a one-dimensional CNN, is composed of convolutional and pooling layers that execute complex mathematical procedures on the input data to create features, while the second component makes use of the generated features through LSTM and dense layers.

Convolutional and pooling layers [44] function as purpose-built data preprocessing layers, filtering incoming data, and extracting important information to be used as an input to a fully connected network layer. Convolution is performed between the raw input data and convolution kernels by the convolutional layers, resulting in the creation of new feature values. Because this method was initially designed to extract features from picture datasets, the input data must be in organized matrix form [45]. Consider the convolution kernel to be a narrow window containing the coefficient values in the matrix form. This window moves across the input matrix, executing convolution on each subregion it intersects. All these processes result in a convolved matrix that signifies a feature value defined by the coefficient values and filter dimension size. By utilizing various convolution kernels to the input data, numerous convolved features may be produced, which are often more valuable than the input data's original starting features, thus improving the performance of the model.

Following the convolutional layers, a nonlinear activation function (e.g., a rectified linear unit) is typically used, trailed by a pooling layer. This layer is a technique for subsampling that removes certain values from the convolved features and creates a matrix with a smaller dimension. Similarly, for the operations done on the convolutional layer, the pooling layer employs a tiny sliding window that accepts the values of each patch of the convolved features as input and outputs a single new value determined by an operation provided for the pooling layer. For instance, max pooling and average pooling are used to determine the maximum and average values of each patch. Consequently, the pooling layer generates new matrices that may be thought of as summarized versions of the convolutional layer's convolved features. The pooling process may contribute to the system's robustness by ensuring that minor changes in the input do not affect the pooled output values.

LSTM [46] is a subclass of recurrent neural networks (RNNs) that may learn long-term dependencies through feedback connections. Traditional RNNs seek to resolve the issue associated with feedforward neural networks, referred to as "loss of memory," which results in low performance on sequence and time-series issues. These models make use of cyclic connections in their hidden layer to develop short-term memory and have the ability to extract knowledge from time series and sequence data. Nonetheless, RNNs are constrained by the well-known loss gradient issue, which prevents the model from learning long-range dependencies. Thus, LSTMs address this issue by keeping valuable

information in memory cells and erasing irrelevant data, resulting in a performance that is generally superior to that of a conventional RNN.

Every LSTM unit has a memory cell and 3 primary gates: input, output, and forget. This structure enables the LSTM to maintain a regulated information flow by determining which information must be "forgotten" and which must be "remembered," thereby learning long-term dependencies. More precisely, the input gate i_t, in conjunction with a second gate c_i^*, regulates the volume of new knowledge saved in the memory state c_t at time t. The forget gate f_t determines whether knowledge from the past must be deleted or retained on the memory cell at time $t-1$, while the output gate o_t determines which knowledge may be used for the memory cell's output. Equations (2)–(5) summarize the activities of an LSTM unit.

$$i_t = \sigma\left(U_i x_t + W_i h_{t-1} + b_i\right), \tag{1}$$

$$f_t = \sigma\left(U_g x_t + W_g h_{t-1} + b_g\right), \tag{2}$$

$$c_t^* = \tanh\left(U_c x_t + W_c h_{t-1} + b_c\right), \tag{3}$$

$$c_t = g_t \odot c_{t-1} + i_t \odot c_t^*, \tag{4}$$

$$o_t = \sigma\left(U_o x_t + W_o h_{t-1} + b_o\right), \tag{5}$$

where x_t represents the input, W_* and U_* signify the weight matrices, b_* signifies the bias term vectors, σ signifies the sigmoid function, and the operator n represents component-wise multiplication. Conclusively, the hidden state h_t which corresponds to the memory cell's output is computed by

$$h_t = o_t \odot \tanh\left(c_t\right). \tag{6}$$

In general, the CNN-LSTM method uses the CNN as an encoder to learn features from subsequence of input data that are fed into an LSTM as time steps. The LSTM will function as a decoder, identifying and modeling both short- and long-term temporal relationships inherent in the data stream. Figure 6 illustrates the structure of our proposed CNN-LSTM architecture, which combines CNN and LSTM to create a deep hybrid architecture.

A brief description for each layer is presented in the following:

Input layer: Receive input of 7 days' COVID-19 confirmed cases.

First Conv1D layer: The first convolutional layer scans through the input sequence, acquires new information, and deals with noise in the input data, before projecting the findings onto feature maps.

Second Conv1D layer: The second layer repeats the process on the feature maps generated by the first, trying to enhance any noteworthy features. We utilized 64 feature maps per convolutional layer and a kernel size of 3 time steps to read the input sequences.

Max pooling layer: By removing specific values from the convolved features, the max pooling layer simplifies the feature maps and produces a matrix with a smaller dimension.

Dropout layer: This layer was added to the network to prevent the model from being overfit. Due to the random subsampling of the outputs of a layer under dropout, this has the effect of decreasing the capacity or thinning the network during training.

Flatten layer: Following the dropout layer, the distilled feature maps are flattened into a single long vector that may be utilized as input to the decoding process.

Repeat Vector layer: Several times, once for each time step in the output sequence, the internal representation of the input sequence is repeated. The LSTM decoder will be shown this vector sequence.

LSTM layer: The decoder is then defined as a 200-unit hidden layer. Notably, the decoder will output the whole sequence, with each of the 200 units supplying a value for each of the seven days, serving as the foundation for predicting what would happen on each day in the output sequence.

Fully connected layer: Before the final output layer, a fully connected layer is utilized to understand each time step in the output sequence. Notably, the output layer makes a single prediction about the output sequence. This indicates that every step in the output sequence will have the similar layers applied. This implies that the decoder will operate every time step using a similar fully connected layer and output layer. This was done by wrapping the interpretation and output layers in a Time-Distributed wrapper, which was utilized for each time step from the decoder. This allows the LSTM decoder to define the context needed for every step in the output sequence, while the wrapped dense layers interpret every time step individually, while still reusing the similar weights.

Output layer: The number of new COVID-19 cases for the 8^{th} day was predicted.

(2) Baseline Models. Seventeen baseline predictive models were also built in this study for comparison with our proposed model: 2 deep learning models (CNN and LSTM), 2 statistical models (ARIMA and Fbprophet), 3 linear models (LR, Ridge, and Lasso), 5 ensemble models (AdaBoost Regressor, Random Forest Regressor, Gradient Boosting Regressor, Extra Trees Regressor, and Bagging Regressor), 5 other machine-learning models (XGBoost Regressor, Gaussian Process Regressor, Support Vector Regressor, Decision Trees Regressor, and K-Nearest Neighbor Regressor) using Python (Jupyter Notebook) on the Anaconda platform. A brief description of each model is presented in the following subsections:

(i) *Deep Learning Models. CNN:* LeCun et al. [47] pioneered the use of convolutional networks in their current form for zip code recognition. CNNs are typically composed of convolutional, pooling, and

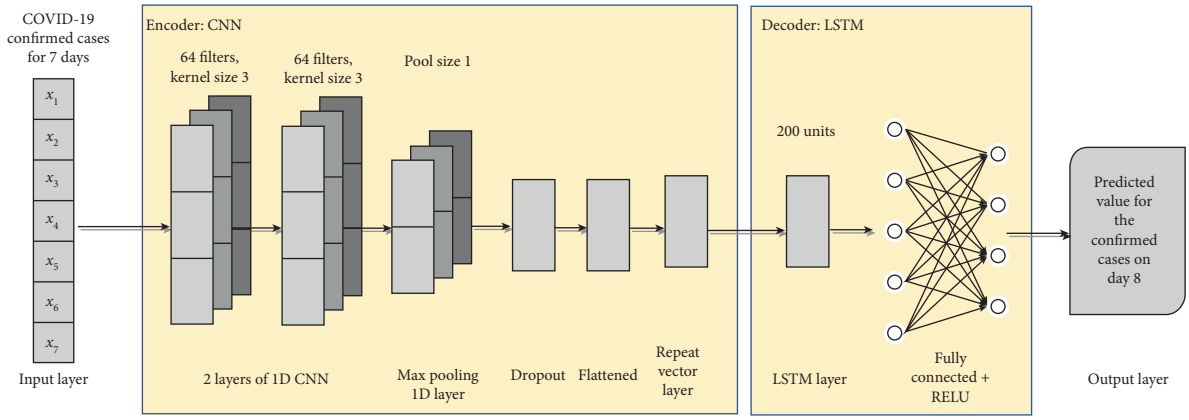


Figure 6: CNN-LSTM structure.

fully linked layers. A collection of feature maps, also known as activation maps, is generated in the convolutional layers. Each neuron in the feature map is linked to a subset of neurons in the input layer that corresponds to it. The feature map's neurons all have the same weights, substantially decreasing the number of parameters compared to a fully connected neural network. Pooling layers alternate with convolutional layers in the most popular CNN designs. The pooling layer lowers the spatial dimension of the feature maps in preparation for the subsequent computational stages, thus reducing computational burden and avoiding overfitting. After an arbitrary number of preceding layers, fully connected layers aggregate the resultant feature maps and provide a classification measure at the network's conclusion.

LSTM: Hochreiter and Schmidhuber [46] suggested the LSTM model, which Graves and Schmidhuber [48] improved and promoted. The memory-based RNN cell is at the heart of the LSTM's basic structure. It is useful for storing and retrieving information from the past. It also facilitates the transmission of previous information to the next level. LSTM chooses previous data based on its training requirements. Remembering beneficial information over time is an ordinary practice, but it is also an essential behavior of the LSTM network [49]. When the data pass in the model, the cells in the LSTM determine what they are, and the information that follows the rules is retained, while the information that does not is lost. This notion will resolve the long sequence dependence problem in neural networks by enhancing the hidden layer structure.

(ii) *Statistical Models.* ARIMA*:* Presented by Box et al. in 1970 [50], ARIMA can be modified to approximate reality and is flexible in describing the behavior of many actual nonstationary and seasonal time series. It is assumed that a time series' future values have a consistent functional relationship with its present, past, and white noise values. ARIMA's main advantage is its ability to forecast accurately in a short amount of time. ARIMA is capable of handling wide-ranging data types, including patterns, seasonality, and cyclicity. It can also be utilized to model the temporal dependency structure of a time series. It does, however, require a significant amount of historical data, ideally 100 or more. The ARIMA model is also known as an ARIMA (p, d, q), where p stands for autoregression order (AR), d for degree of variance (I), and q for moving average order (MA) [51]. These three aspects are relevant to classify time series, where p is in charge of storing and retrieving the process's past information, d is in charge of converting nonstationary time series to stationary time series, and q is in charge of regulating the process's noise-related past information.

Fbprophet*:* Fbprophet is a Facebook-developed time series forecasting model that was created to solve business time series problems. While there are numerous approaches to forecasting market outcomes, many of them share common characteristics, such as seasonal effects [51]. It employs a decomposable time-series model [52] with 3 main elements: pattern, seasonality, and holidays. It is a regression model with interpretable parameters that fit with the default values, as well as allows the user to automatically choose the elements that are pertinent to their forecasting predicament and use the appropriate modifications with ease [51]. The Fbprophet predicts trends using two methods: a saturation model of growth and a linear model. In the case of growth forecasting, a model such as the population growth model in natural ecosystems is utilized, in which nonlinear growth reaches a saturation point at a carrying capacity [53]. If the saturating point is never attained, a piecewise model of a constant growth rate can be an alternative solution. Fbprophet employs Fourier series to give periodic effects to a flexible model of [52], whereas holidays must be accounted for using a

predetermined list of past and future events. It is simple to incorporate holiday effects into the model because they are considered independent [51].

(iii) *Linear Models*. Linear Regression*:* LR is a technique for modeling the correlation between a dependent variable and an independent variable using a linear approach.

Ridge*:* Ridge regression (RR) is a kind of linear regression that uses a sum-of-squares error function and a regularization method to manage the bias-variance trade-off, with the goal of revealing the linear patterns hidden in the data. [54].

Lasso*:* Lasso [55] is a technique for shrinking and selecting variables in linear regression. It considers the total of l_1-norms of the regression coefficients as a penalty in addition to the total of squares error reduction. This bias–variance trade-off almost always results in improved forecasts.

(iv) *Ensemble Models*. XGBoost Regression*:* XGBoost is Tianqi Chen's gradient boosting equipment from 2016 [56]. The algorithm's implementation has been optimized for consistency based on processing time and memory space. XGBoost is usually very fast when compared to other gradient boosting applications. For classification and regression predictive modeling problems, XGBoost dominates standardized or tabular datasets. As a supervised learning algorithm, XGBoost predicts a target using train data with several features. Although XGBoost follows the similar processes as gradient boosting, it has its own tree. The predictive initial value set distinguishes regular and extreme gradient boosting in regression. The initial prediction value in gradient boosting is the result of the average real value of one feature that will be predicted. In XGBoost, the initial prediction value is chosen at random, but the most used value is 0.5.

AdaBoost Regressor*:* The AdaBoost Regressor converted the delicate regression model into a robust regression learning model, from which the prediction model was built [57].

Random Forest Regressor (RFR)*:* The RFR method is similar to Breiman's pioneering Regression Tree Analysis [58]. RFR isolates the predictand (desired parameter) iteratively using a series of binary splits. Each of these divides corresponds to a value on an individual predictor grid that maximizes the disparities between the branches of the "tree." A split, together with its associated branches, is regarded to constitute a single decision tree. Each branch is constructed using a random selection of nodes representing individual predictors. Each predictor node has a large number of potential predictands, and it is at these nodes that a random choice is taken to divide the branch further, thus adding two additional predictors. This is repeated repeatedly until no further splits occur, resulting in terminal nodes, or "leaves." Typically, the RFR will do binary splits until a single predictor on a leaf is discovered.

Gradient Boosting Regressor*:* Friedman [59] developed the gradient boosting regressor (GBR) in 2000. The GBR combines a huge amount of ineffective learning methods to create a more effective learning algorithm. It learns from earlier learning algorithms' errors.

Extra Trees Regressor*:* Like the random forest tree, the extra trees regressor (ETR) is built using many decision trees [60]. On ETR, all decision trees are trained in their entirety utilizing all training sets. To get the bifurcation value, the ETR model randomly bifurcates the decision tree.

Bagging Regressor*:* Bagging is a kind of parallel ensemble model, and the bagging model is constructed using bootstrap sampling [61]. That is, given an initial dataset of m samples, a sample is chosen for the sampling set using the replacement technique for it to be chosen again for the subsequent sampling round. Then, by iterating bootstrap sampling n times, n sampling sets are produced. During the sampling process, some of the samples in the original dataset may be selected many times, while others may never be chosen. In addition, the DT model is trained on n weak learners, and the final judgment is reached through majority vote for the classification problem or average meaning for the regression problem.

(v) *Machine-Learning Models*. Gaussian Process Regressor: Gaussian process regression (GPR) is a very effective method. Indeed, in addition to their simple structure and computationally acceptable predictions, GPRs have the major advantage of being nonparametric and able to account for projected value uncertainty.

Support Vector Regressor (SVR): SVR is a regression algorithm that employs a technique similar to support vector machines (SVMs) for regression analysis [62]. SVR offers the freedom to define the acceptability of an error in a model and finds an appropriate line to adapt the data. SVR's goal is to reduce coefficients, in particular, the $l2$-norm of the coefficient vector. The error term is managed in the constraints, where the absolute error is found to be less than or equal to a denoted margin, known as the epsilon (maximum error).

Decision Trees Regressor (DT)*:* A DT divides the dataset into two nodes and repeatedly builds a tree-like structural model using the information gain (IG). The gain parameter specifies the anticipated decrease in entropy associated with the chosen features.

K-Nearest Neighbor Regressor: The K-nearest neighbor regressor (KNNR) is a nonparametric model that seizes a sample of K-nearest neighbors and predicts the sample value using the nearest neighbor response value (y).

Step 2.1: Optimize models

Since selecting the best and most accurate forecasting model for predicting the COVID-19 pandemic is a very complicated process, the 18 forecasting models constructed in this study were

Table 2: Hyperparameter tuning.

Model type	Forecasting model	Hyperparameter	Range	Best hyperparameter
Proposed model	CNN-LSTM (2 convolutional layers with 64 filters, kernel size 3, 1 max pooling layer with size 1, 1 dropout layer, 1 LSTM layer with 200 units, 1 fully connected layer)	Epochs	(32, 1000)	472
		Batch_size	(2, 30)	22
		Verbose	(0, 1)	1
Deep learning model	CNN (2 convolutional layers with 64 filters, kernel size 3, 1 max pooling layer with size 1, 1 fully connected layer)	Epochs	(32, 1000)	472
		batch_size	(2, 30)	22
		Verbose	(0, 1)	1
	LSTM (1 LSTM layer with 200 units)	Epochs	(32, 1000)	472
		Batch_size	(2, 30)	22
		Verbose	(0, 1)	1
Statistical model	ARIMA	p	(0, 10)	9
		d	(0, 3)	2
		q	(0, 3)	2
	FBProphet	Changepoint_prior_scale	(0.0001, 0.5)	0.5
		Seasonality_prior_scale	(0.01, 10)	0.25
		Seasonality_mode	(0, 1)	1
Linear model	LR	Fit_intercept	[True, false]	True
		n_jobs	(−1, 1)	−1
	Ridge	Alpha	(1, 5)	5
	Lasso	Alpha	(1, 5)	5
Ensemble model	XGBoostR	n_estimators	(0, 1000)	545
		Max_depth	(0, 25)	6
		Reg_alpha	(0, 5)	1
		Reg_lambda	(0, 5)	3
		Gamma	(0, 5)	1
		Learning_rate	(0.005, 0.5)	0.1225
	AdaBoostR	n_estimators	(0, 1000)	545
	RFR	n_estimators	(0, 1000)	545
	GBR	n_estimators	(0, 1000)	545
	ETR	n_estimators	(0, 1000)	545
	BaggingR	n_estimators	(0, 1000)	545
Machine-learning model	GPR	Kernel	DotProduct, Matern, RBF, WhiteKernel	DotProduct
		Alpha	(0, 1)	0.16000000000000003
	SVR	Kernel	rbf, poly	poly
		C	(0, 10)	1.5
		Gamma	(0, 5)	3
		Epsilon	(0, 1)	0.1
	DTR	Max_depth	(0, 25)	5
	KNNR	n_neighbors	(0, 10)	3

fine-tuned based on the selected hyperparameters under various prediction criteria using the Optuna framework to optimize the hyperparameters for each model. The science of tuning or selecting the best set of hyperparameters for a learning algorithm is known as hyperparameter optimization. Any machine-learning algorithm's output is heavily influenced by a collection of optimal hyperparameters. It is one of the most time-consuming steps in the machine-learning training pipeline, but it is also one of the most important. In forecasting COVID-19 confirmed events, Table 2 presents the best hyperparameter values for each chosen hyperparameter and its range for each forecasting model.

Step 2.2: Train the models

The models were trained using the best hyperparameter on a train set which is time series that starts on January 4, 2020 and ends on July 17, 2020.

Step 2.3: Evaluate the models

The trained models were then evaluated on a test set, the time series of which starts on July 18, 2020 and ends on August 14, 2020. From a forecasting perspective, the forecasting horizon is critical for an intelligent model's prediction accuracy. The forecasting horizon refers to the number of daily confirmed cases that a forecasting model considers while projecting the next daily verified case. More precisely, when the forecasting horizon is equal to 9,

the algorithm uses verified instances from the previous nine days to estimate the price on the tenth day. In this research, the predicting horizon was set to seven days.

Taking into consideration the occurrence of randomness, the train and prediction processes were run 10 times. The average forecasting between July 18, 2020 and August 14, 2020 was computed and compared with the real values. The performance of the models was then assessed using testing data and statistical error measures such as MAPE, RMSE, and relative root mean square error (RRMSE).

MAPE: This metric is calculated as the average absolute difference between the estimated and measured values.

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{f(t)-\widehat{f}(t)}{f(t)}\right| \times 100. \quad (7)$$

RMSE: This metric is normally employed to evaluate forecasting errors of different models. In terms of absolute variance, the lower the RMSE value, the better a model's predictive capability. Nevertheless, the existence of a few major errors will lead to a higher RMSE value.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}[f(t)-\widehat{f}(t)]^2}. \quad (8)$$

RRMSE: This metric is computed by dividing RMSE by the average value of the measured data. The accuracy of the model is deemed excellent when the RRMSE is less than 10 percent, good when the RRMSE is between 10 and 20 percent, fair when the RRMSE is between 20 and 30 percent, and poor when the RRMSE is greater than 30 percent [63].

$$\text{RRMSE} = \frac{\sqrt{(1/n)\sum_{t=1}^{n}[f(t)-\widehat{f}(t)]^2}}{\sum_{t=1}^{n}[f(t)]} \times 100. \quad (9)$$

Since most of the models are in stochastic nature, we evaluated all 18 predictive models 10 times and reported the mean performance on a test time series data.

3.2.3. Phase 3: Applying the Predictive Models

Step 3.1: Forecast COVID-19 new cases

This step requires the information of COVID-19 that starts on August 15, 2020 and ends on September 18, 2020. The new COVID-19 cases from August 15, 2020 until September 11, 2020 (4 weeks) were used as input to forecast the new cases for the subsequent week (September 12, 2020–September 18, 2020).

Step 3.2: Evaluate the models

Then, the forecasting between September 12, 2020 and September 18, 2020 was computed and compared with the real values. The performance of the models was calculated based on MAPE, RMSE, and RRMSE.

To guide us in evaluating the results, the following research questions were constructed:

RQ1: How good is the performance of the proposed model in this study compared to the 17 baseline models in forecasting the new cases of the COVID-19 pandemic around the world based on the test data?

RQ2: How good is the performance of the proposed model in this study compared to the 17 baseline models in forecasting the new cases of the COVID-19 pandemic around the world based on the forecast data?

RQ3: How good is the performance of the proposed model in this study compared to the state-of-the-art?

4. Results and Discussion

The empirical results of this study are presented in this section to reply to RQ1–RQ3.

4.1. RQ1 Answer: The Performance of the Proposed Model Compared to the Baseline Models on the Test Data. The predicted and actual values between July 18, 2020 and August 14, 2020 were plotted in 5 different graphs according to the type of the models. The values predicted by the proposed model were plotted in each graph to compare the trend obtained with other baseline models. Figures 7–11 show the comparison between actual and predicted data for each type of model compared to the proposed model in forecasting confirmed cases. From these figures, we observe that most of the models are following the trend of confirmed COVID-19 cases on the test data except for SVR model.

To evaluate the performance of each model quantitatively, the MAPE, RMSE, and RRMSE for the proposed model and 17 baseline models in predicting the confirmed cases of COVID-19 were calculated using equations (7)–(9), respectively, based on the predicted and actual values between July 18, 2020 and August 14, 2020. The mean for each performance metric was then computed. The example of the computed RMSE for the proposed model compared to the CNN-1D and LSTM models is illustrated in Figure 12. Based on the mean values of MAPE, RMSE, and RRMSE shown in Table 3, the proposed model surpassed the 17 baseline models in forecasting the new confirmed cases with the minimum error value: MAPE = 0.19, RMSE = 13275.00, and RRMSE = 5.30. In terms of the RRMSE value that is less than 10%, the proposed model performed excellently in predicting the confirmed cases of COVID-19. The RRMSE values also show that compared to CNN-1D and LSTM vanilla models, the combination of both models in the structure of CNN-LSTM increase the performance by decreasing the RRMSE value by 1.04% and 1.9%, respectively.

4.2. RQ2 Answer: The Performance of the Proposed Approach Compared to the Baseline Models on the Forecast Data. Similar to the processes that were conducted on the test data, the forecasted and actual values between September 12, 2020 and September 18, 2020 were plotted in 5 different graphs according to the type of the models. The values forecasted by

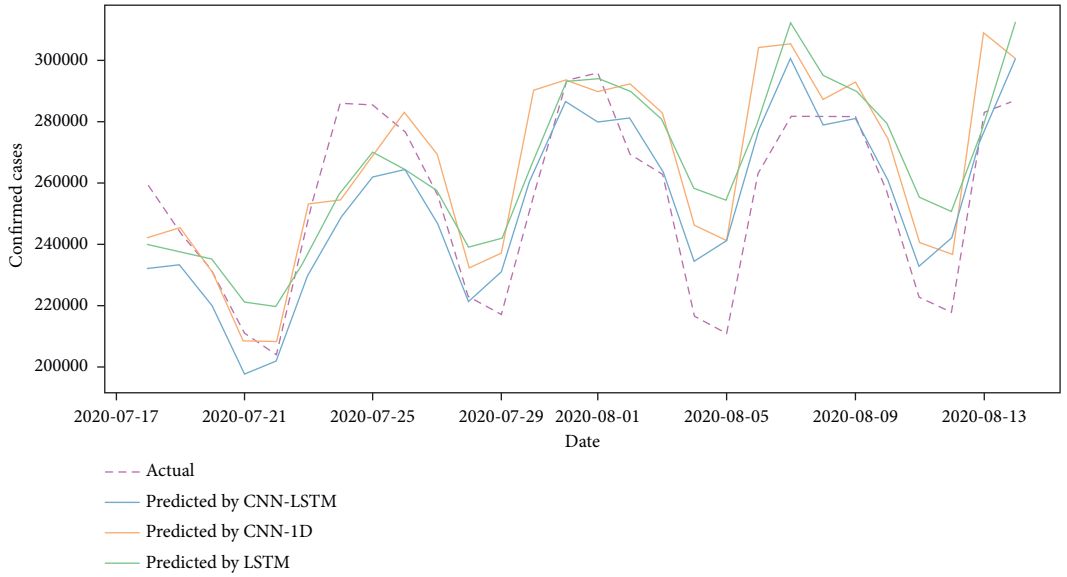


Figure 7: Actual and predicted data of the proposed approach compared to deep learning models.

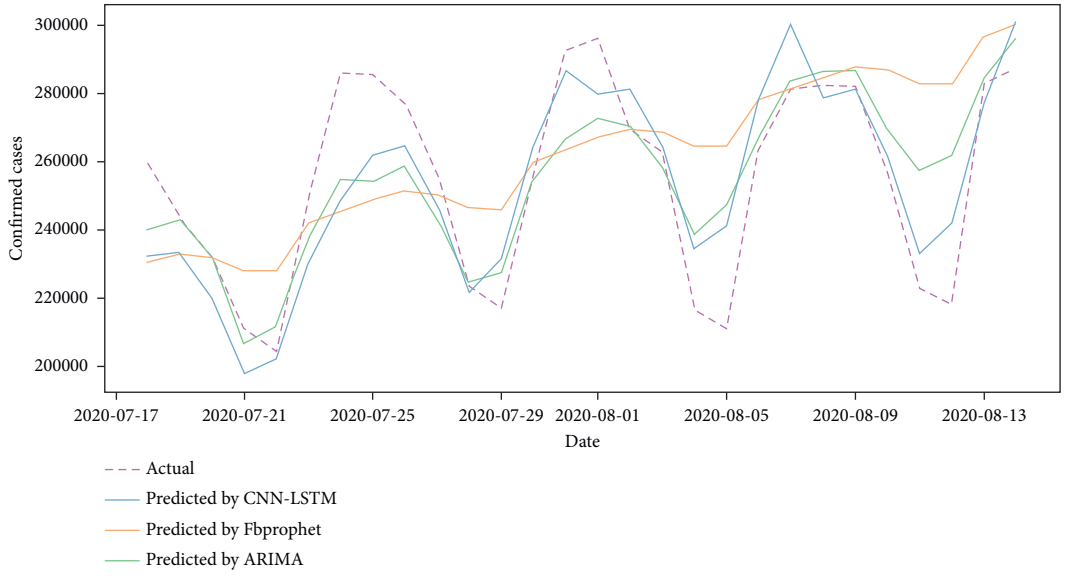


Figure 8: Actual and predicted data of the proposed approach compared to statistical models.

the proposed model were plotted in each graph to compare the trend obtained with other baseline models. Figures 13–17 show the comparison between actual and forecasted data for each type of model compared to the proposed model in forecasting confirmed cases. From these figures, again, we observe that most of the models are following the trend of confirmed COVID-19 cases on the forecast data except for the SVR model.

To evaluate the performance of each model, the MAPE, RMSE, and RRMSE for the proposed model and 17 baseline models in forecasting the confirmed cases of COVID-19 were calculated using equations (7)–(9), respectively, based on the forecasted and actual values between September 12, 2020, and September 18, 2020. The mean for each performance metric was then computed. The example of the computed RMSE for the proposed model compared to the standalone CNN-1D and LSTM models is illustrated in Figure 18. Based on the values of MAPE, RMSE, and RRMSE shown in Table 4, the proposed model again outperformed the 17 baseline models in forecasting the new confirmed cases with the minimum error value: MAPE = 0.43, RMSE = 8780.71, and RRMSE = 3.01. The RRMSE values show that the proposed hybrid model increases the performance of the standalone CNN-1D and LSTM models by

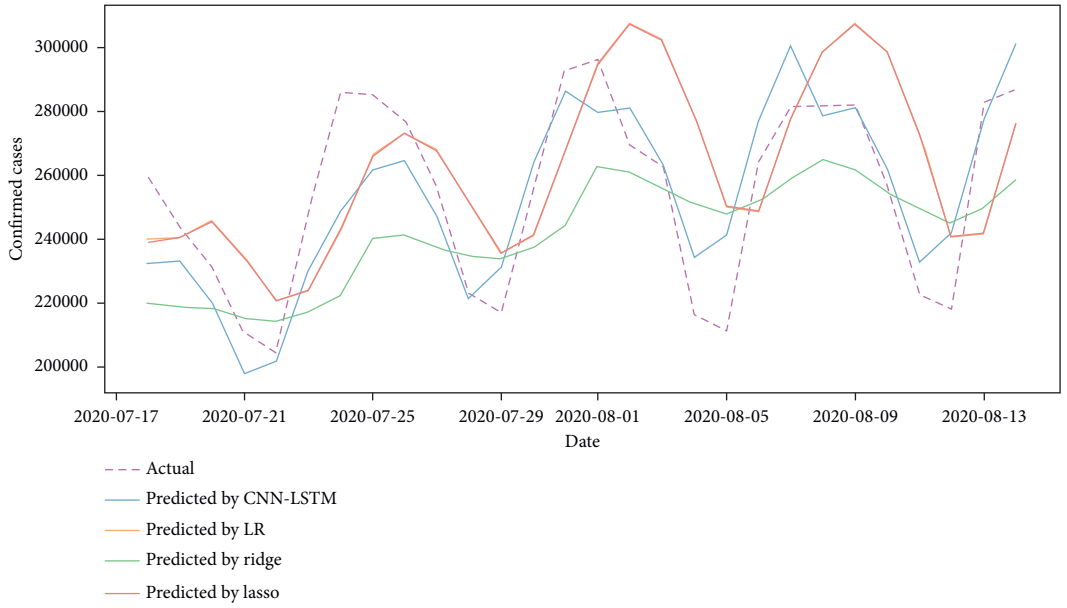


Figure 9: Actual and predicted data of the proposed approach compared to linear models.

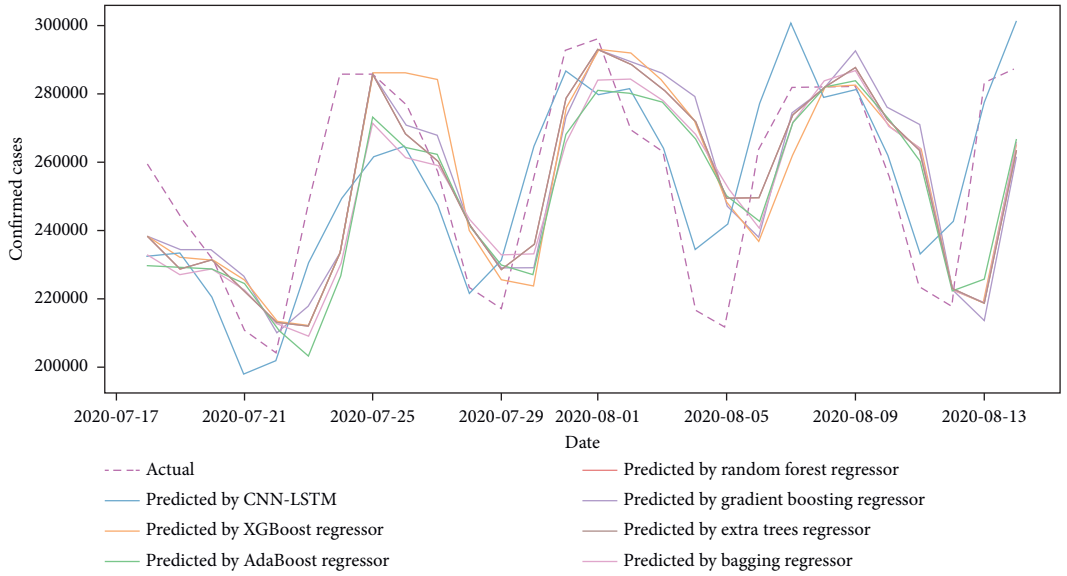


Figure 10: Actual and predicted data of the proposed approach compared to ensemble models.

reducing the RRMSE value by 1.15% and 3%, respectively. This result shows the effectiveness of combining both CNN-1D and LSTM models even on a small size of data.

4.3. RQ3 Answer: The Performance of the Proposed Approach Compared to the State-of-the-Art Approach. Table 5 compares the hybrid models developed in [64] and the current study. As demonstrated in Table 5, our method performs better in terms of MAPE and RMSE when forecasting confirmed viruses globally. As we can see, our approach has a lower MAPE and RMSE than theirs, although we examined 216 countries compared to theirs which was only seven.

5. Threats to Validity

5.1. Threats to Construct Validity. The performance metrics used in our analysis relate to threats to construct validity. In this study, 3 evaluation metrics based on statistical measures of errors were selected: RMSE, MAPE, and RRMSE. There are other measures, such as MAE, R^2, root mean squared

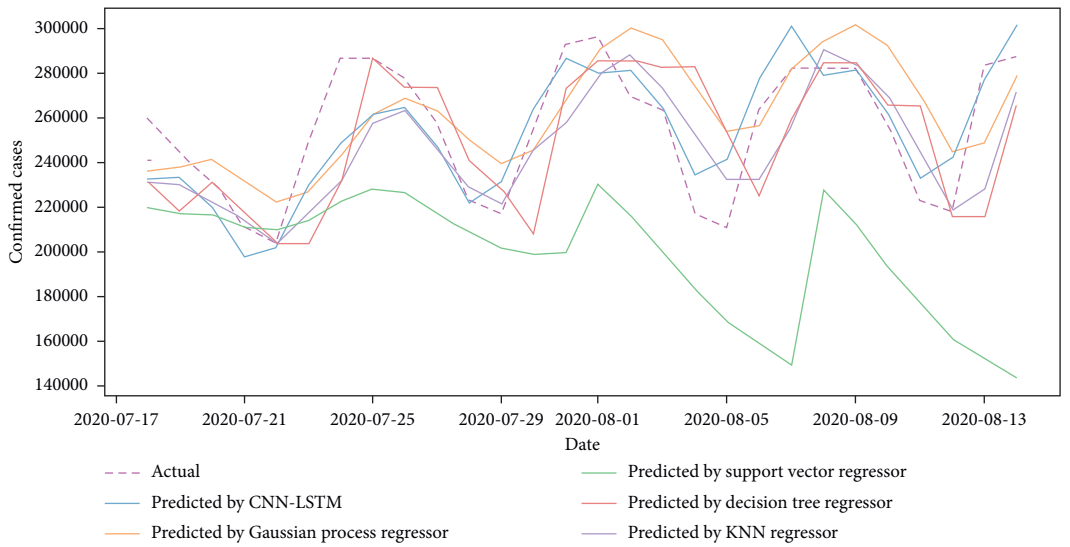
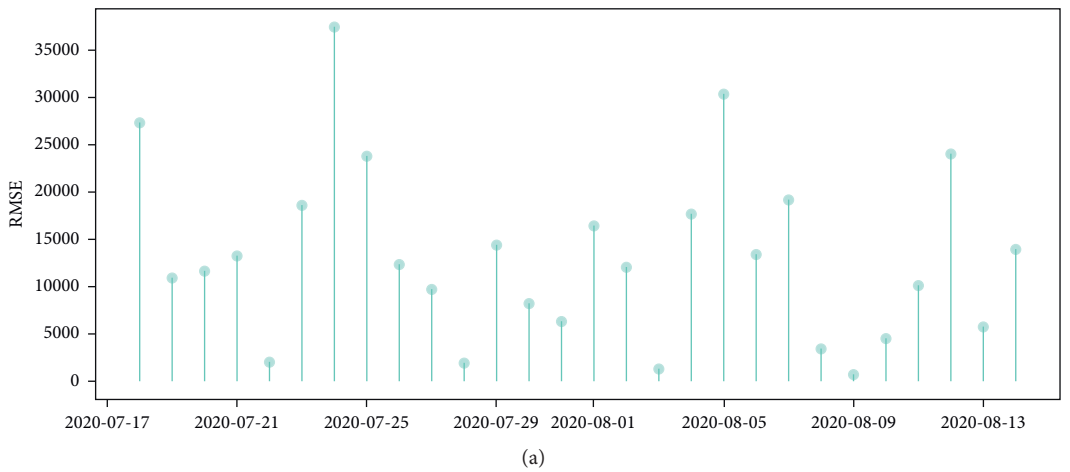
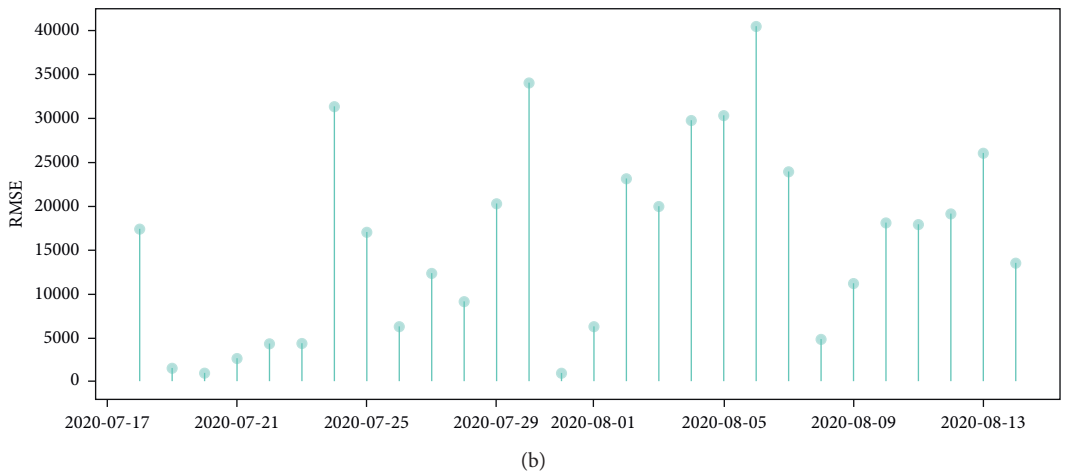


Figure 11: Actual and predicted data of the proposed approach compared to machine-learning models.



(a)



(b)

Figure 12: Continued.

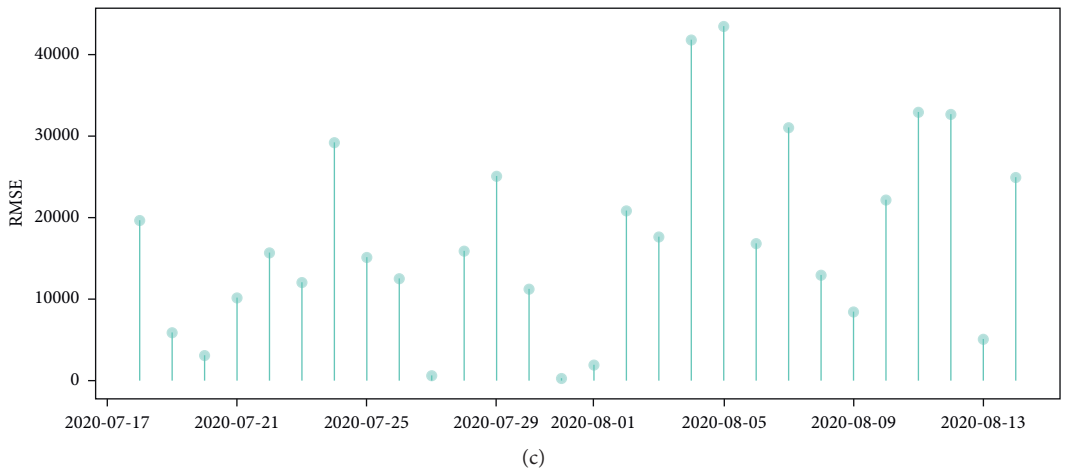


Figure 12: RMSE for the proposed model compared to CNN-1D and LSTM models on the test data. (a) CNN-LSTM: testing RMSEmean = 13275.0. (b) CNN1D: testing RMSEmean = 15955.0. (c) LSTM: testing RMSEmean = 17512.5.

Table 3: The average model performance evaluation for forecasting confirmed cases on test data.

Type	Models	MAPE	RMSE	RRMSE
Proposed approach	CNN-LSTM	**0.19**	**13275.00**	**5.30**
Deep learning	CNN-1D	0.23	15954.97	6.34
	LSTM	0.26	17512.45	7.20
Statistical method	ARIMA	0.20	13630.13	5.51
	FBProphet	0.33	22326.90	9.25
Linear model	LR	0.35	24150.13	9.82
	Ridge	0.34	24786.26	9.65
	Lasso	0.35	24143.65	9.82
Ensemble	XGBR	0.30	21267.73	8.51
	AdaBoostR	0.30	21065.30	8.39
	RFR	0.31	21433.81	8.56
	GBR	0.30	21140.69	8.49
	ETR	0.28	19210.06	7.73
	BaggingR	0.31	21435.07	8.55
Machine learning	GPR	0.33	22461.64	9.18
	SVR	0.76	56131.22	21.18
	DTR	0.33	22954.19	9.15
	KNNR	0.26	18954.48	7.28

The bold values present the lowest error values of MAPE, RMSE, and RRMSE. These values show that the proposed approach outperforms the baseline models based on the test data.

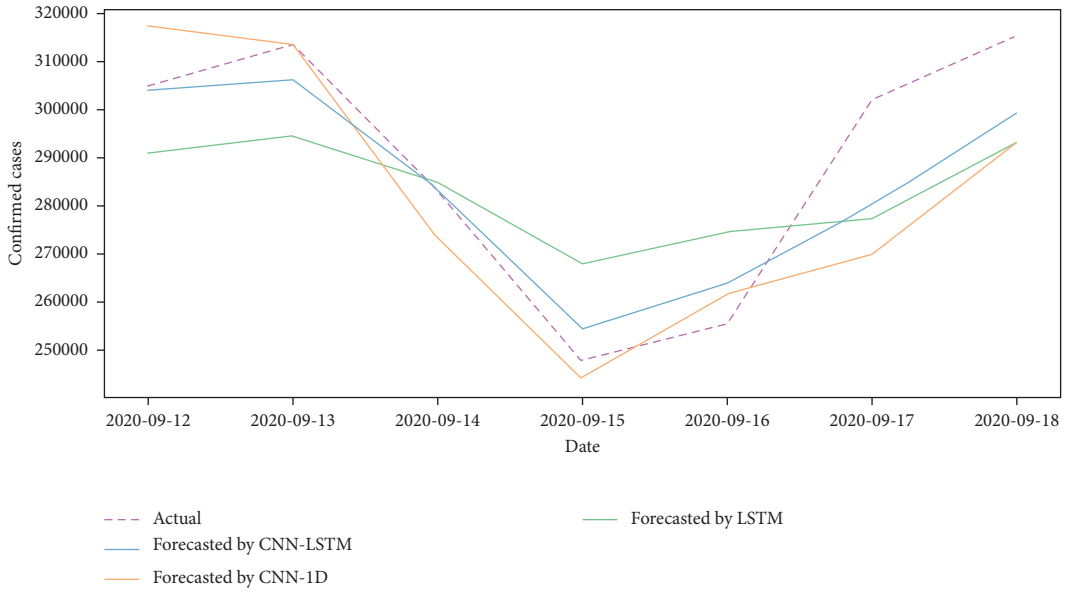


Figure 13: Actual and forecasted data of the proposed approach compared to deep-learning models.

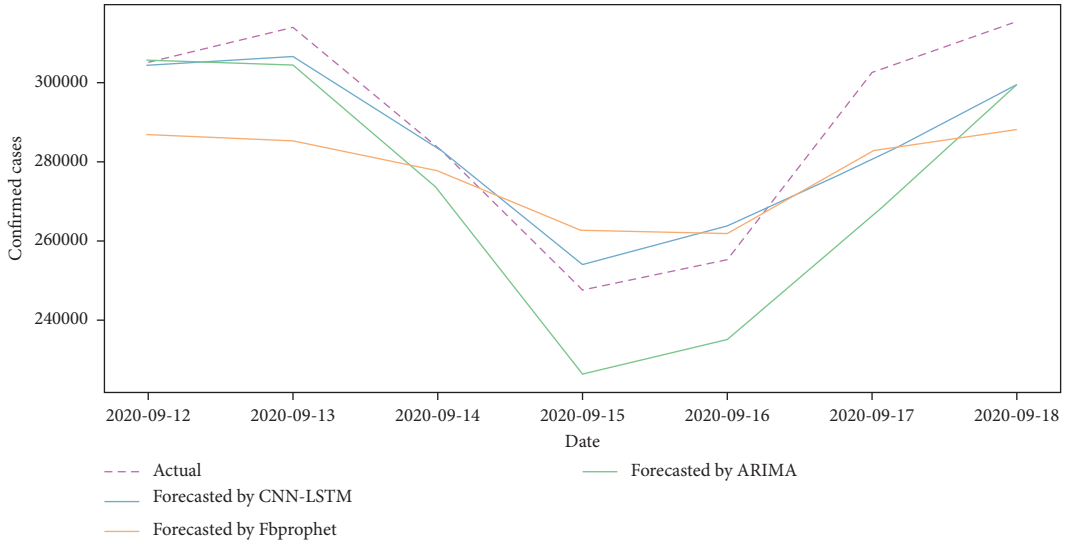


Figure 14: Actual and forecasted data of the proposed approach compared to statistical models.

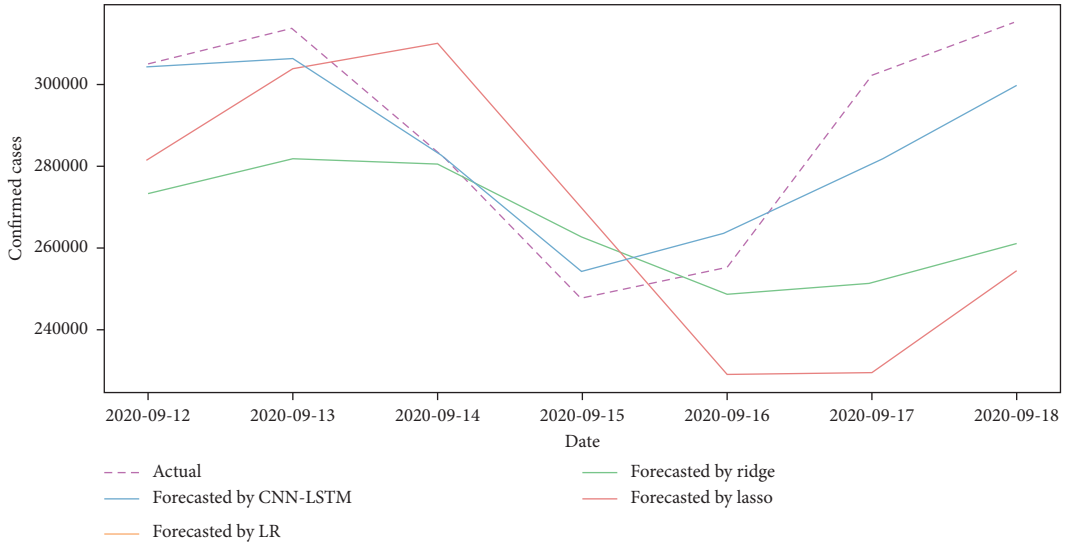


Figure 15: Actual and forecasted data of the proposed approach compared to linear models.

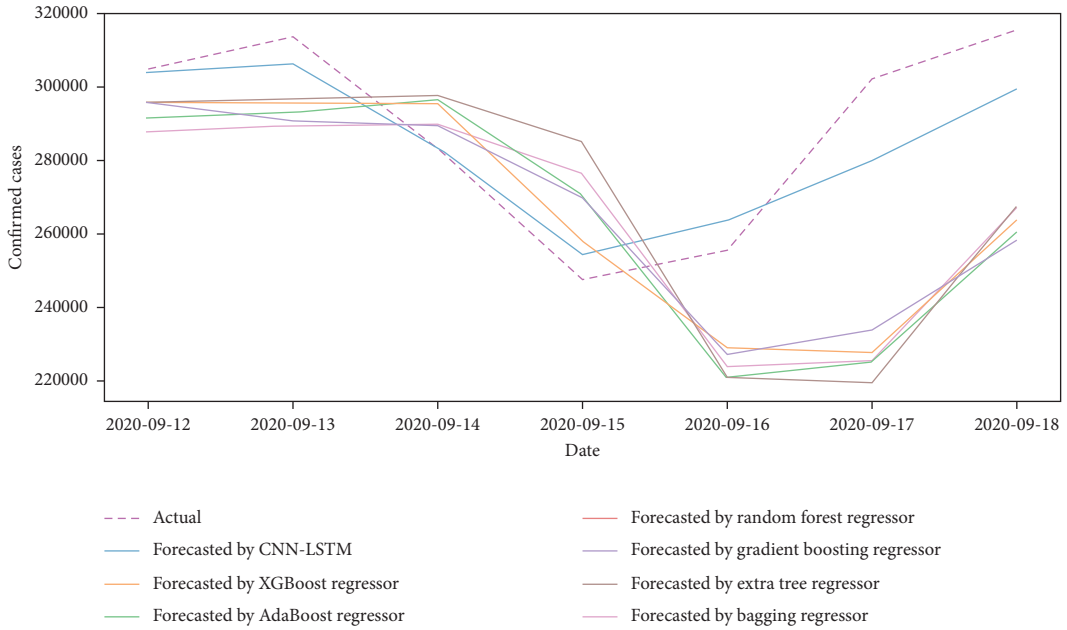


Figure 16: Actual and forecasted data of the proposed approach compared to ensemble models.

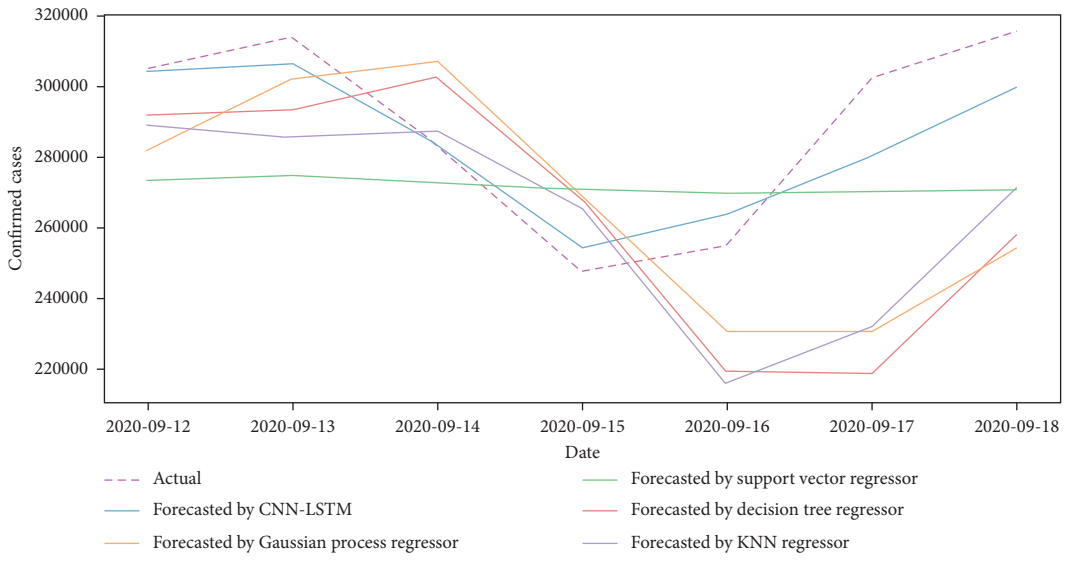


Figure 17: Actual and forecasted data of the proposed approach compared to machine-learning models.

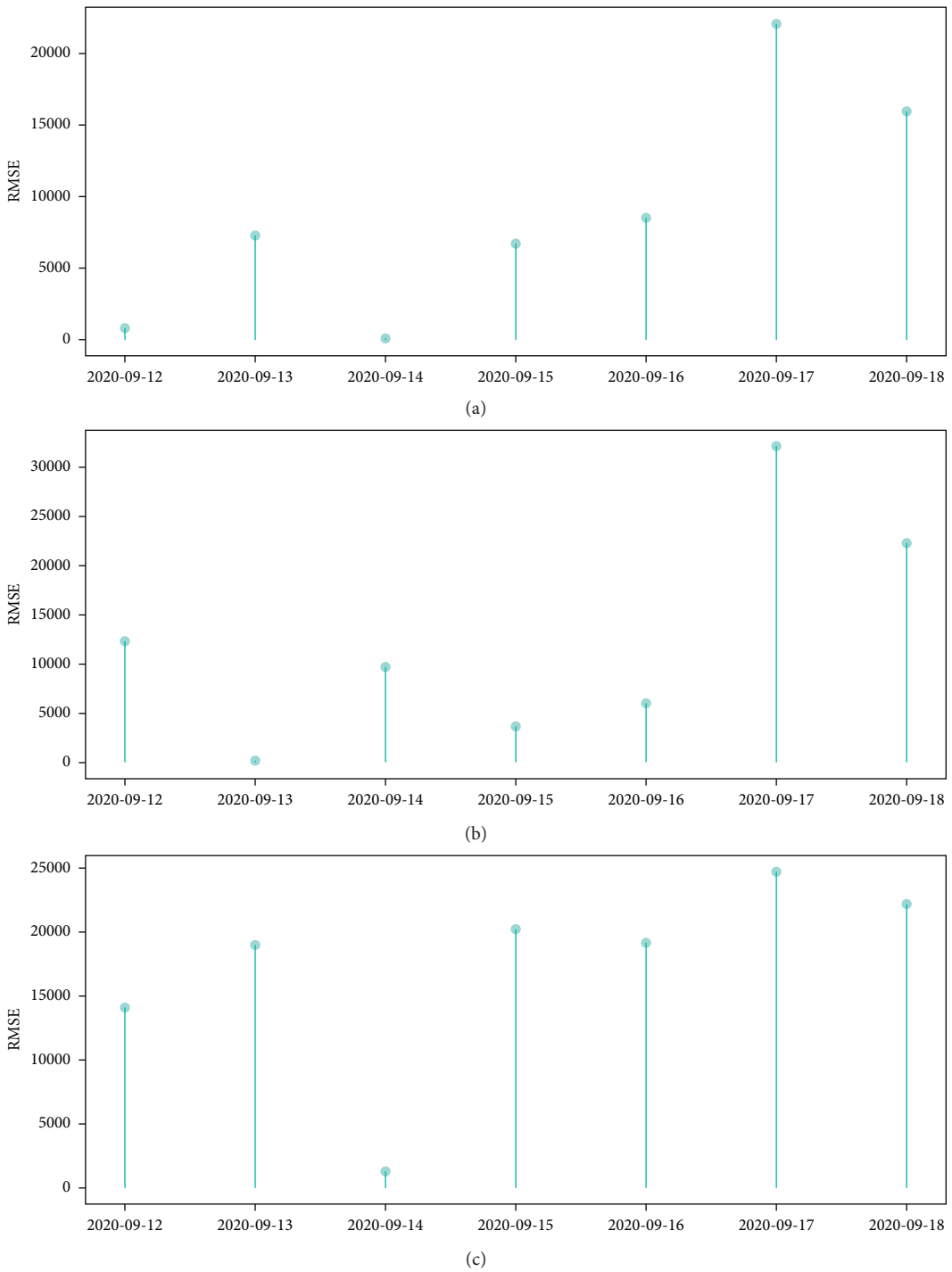


Figure 18: RMSE for the proposed model compared to CNN-1D and LSTM models on the forecast data. (a) CNN-LSTM: forecasting RMSEmean = 8780.7. (b) CNN1D: forecasting RMSEmean = 12349.5. (c) LSTM: forecasting RMSEmean = 17257.3.

Table 4: The average model performance evaluation for forecasting confirmed cases on forecast data.

Type	Models	MAPE	RMSE	RRMSE
Proposed approach	CNN-LSTM	**0.43**	**8780.71**	**3.01**
Deep learning	CNN-1D	0.59	12349.46	4.16
	LSTM	0.86	17257.32	6.01
Statistical method	ARIMA	0.82	16156.94	5.73
	FBProphet	0.83	17223.50	5.82
Linear model	LR	1.70	34688.47	11.88
	Ridge	1.31	27657.47	9.20
	Lasso	1.70	34669.07	11.88
Ensemble	XGBR	1.40	28970.67	9.83
	AdaBoostR	1.66	33822.49	11.63
	RFR	1.64	33395.34	11.51
	GBR	1.73	34770.86	12.12
	ETR	1.50	30593.47	10.47
	BaggingR	1.64	33337.85	11.48
Machine learning	GPR	1.65	33878.38	11.58
	SVR	1.35	27944.14	9.48
	DTR	1.75	35680.97	12.24
	KNNR	1.55	31464.57	10.82

The bold values present the lowest error values of MAPE, RMSE, and RRMSE. These values show that the proposed approach outperforms the baseline models based on the forecast data.

Table 5: Comparison between [64] and this work.

Study	Predictive model	Dataset	MAPE	RMSE
This study	CNN-LSTM	Confirmed cases for the whole world from January 4, 2020 to September 24, 2020	**0.19**	**13275.00**
Dairi et al. [64]	LSTM-CNN	Confirmed cases for 7 countries from January 22 to September 6, 2020.	2.36	28820.00

The bold values show the lower values of MAPE and RMSE, which is better than [64].

relative error (RMSRE), and mean bias error (MBE) that can be used to evaluate time series forecasters. However, the 3 metrics selected in this study are widely used measures to evaluate the time series forecasting of the spread of COVID-19 disease.

5.2. Threats to Internal Validity. The risks are primarily concerned with the unregulated internal variables that may affect the results of the experiment. The key internal threat is the possible faults during the implementation of our process. We used seven machine-learning techniques obtained from sci-kit-learn libraries, one from the *xgboost* library, five from ensemble library, one from the *fbprohet* library, two from *the Keras* library, and one from *the Statsmodels* library, to reduce this hazard. The best hyperparameter values for each module were set automatically using an established Optuna framework.

5.3. Threats to External Validity. External validity threats are related to the possibility of generalizing our findings. The experiments conducted in this study used the COVID-19 dataset from 4/1/2020 to 24/9/2020. The performance of the predictive models used in this study depends on how the dataset is split into train and test data. Different results can be generated by using different timelines of COVID-19 data.

6. Conclusions and Future Work

In this research, a novel hybrid forecasting model termed CNN–LSTM was presented for predicting the global number of COVID-19 infection cases. The proposed CNN–LSTM model was compared against 17 baseline models, including two deep-learning models, two statistical techniques, three linear models, five ensemble learning models, and five machine-learning models. Three performance measures were used to evaluate and compare forecasting performance: MAPE, RMSE, and RRMSE.

The primary finding of this research is that, when compared to 17 baseline time series forecasting models, our proposed CNN-LSTM model outperformed them all with the lowest average MAPE, RMSE, and RRMSE values on both test and forecast data. Finally, we note that although solo CNN and LSTM models perform well and efficiently for predicting verified COVID-19 instances time series, combining both models in the proposed CNN-LSTM encoder-decoder structure significantly improves forecasting performance. In addition, we demonstrated that the suggested model produced acceptable predicting results even when just a limited quantity of data was available.

Ultimately, the proposed CNN-LSTM model takes an efficient step in dealing with noise in the input data and using the internal representation of the time series through convolutional layers. The LSTM and dense layers are then used to

exploit the produced features to detect short- and long-term relationships in the time series and give a precise forecast.

However, there is still opportunity to improve the forecasting accuracy of the COVID-19 application. In the future, by adding more data and external factors to the COVID-19 datasets, such as changes in season, vaccination plan, and additional lockdowns, other resampling and restructuring forecasting methods will be used to further improve the accuracy of the COVID-19 forecasting system. In addition, an uncertainty management strategy should be developed in order to quantify uncertainty and provide users with more relevant information on the COVID-19 pandemic.

Data Availability

The partial data used to support the findings of this study are included within the article, and other partial data are available from the first author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This research project was funded by the Deanship of Scientific Research, Princess Nourah bint Abdulrahman University, through the Program of Research Project Funding after Publication, Grant No. PRFA-P-42-3.

References

[1] World Health Organization (WHO), "Coronavirus disease (COVID-19) pandemic," 2020, https://www.euro.who.int/en/health-topics/health-emergencies/coronavirus-covid-19/novel-coronavirus-2019-ncov.

[2] World Health Organization (WHO), "WHO coronavirus disease (COVID-19) dashboard," 2020, https://covid19.who.int/.

[3] L. Wang, J. Li, S. Guo et al., "Real-time estimation and prediction of mortality caused by COVID-19 with patient information based algorithm," *The Science of the Total Environment*, vol. 727, Article ID 138394, 2020.

[4] M. Ozaslan, M. Safdar, I. H. Kilic, and R. A. Khailan, "Practical measures to prevent COVID-19: a mini-review," *Journal of Biological Sciences*, vol. 20, no. 2, 2020.

[5] M. U. G. Kraemer, C.-H. Yang, B. Gutierrez et al., "The effect of human mobility and control measures on the COVID-19 epidemic in China," *Science*, vol. 368, no. 6490, pp. 493–497, 2020.

[6] W. Preiser, G. van Zyl, and A. Dramowski, "COVID-19: getting ahead of the epidemic curve by early implementation of social distancing," *South African Medical Journal = Suid-Afrikaanse Tydskrif Vir Geneeskunde*, vol. 110, no. 4, p. 12876, 2020.

[7] M. Klompas, "Coronavirus disease 2019 (COVID-19): protecting hospitals from the invisible," *Annals of Internal Medicine*, vol. 172, no. 9, pp. 619-620, 2020.

[8] J. Pang, M. X. Wang, I. Y. H. Ang et al., "Potential rapid diagnostics, vaccine and therapeutics for 2019 novel coronavirus (2019-nCoV): a systematic review," *Journal of Clinical Medicine*, vol. 9, no. 3, 2020.

[9] N. Hasan, "A methodological approach for predicting COVID-19 epidemic using EEMD-ANN hybrid model," *Internet of Things*, vol. 11, 2020.

[10] Z. Car, S. Baressi Šegota, N. Anđelić, I. Lorencin, and V. Mrzljak, "Modeling the spread of COVID-19 infection using a multilayer perceptron," *Computational and Mathematical Methods in Medicine*, vol. 2020, Article ID 5714714, 10 pages, 2020.

[11] N. Feroze, "Forecasting the patterns of COVID-19 and causal impacts of lockdown in top five affected countries using Bayesian structural time series models," *Chaos, Solitons and Fractals*, vol. 140, 2020.

[12] V. Papastefanopoulos, P. Linardatos, and S. Kotsiantis, "COVID-19: a comparison of time series methods to forecast percentage of active cases per population," *Applied Sciences (Switzerland)*, vol. 10, no. 11, 2020.

[13] R. Salgotra, M. Gandomi, and A. H. Gandomi, "Evolutionary modelling of the COVID-19 pandemic in fifteen most affected countries," *Chaos, Solitons, and Fractals*, vol. 140, Article ID 110118, 2020.

[14] F. Shahid, A. Zameer, and M. Muneeb, "Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM," *Chaos, Solitons, and Fractals*, vol. 140, Article ID 110212, 2020.

[15] S. Singh, K. S. Parmar, J. Kumar, and S. J. S. Makkhan, "Development of new hybrid model of discrete wavelet decomposition and autoregressive integrated moving average (ARIMA) models in application to one month forecast the casualties cases of COVID-19," *Chaos, Solitons, and Fractals*, vol. 135, Article ID 109866, 2020.

[16] Z. Ceylan, "Estimation of COVID-19 prevalence in Italy, Spain, and France," *The Science of the Total Environment*, vol. 729, Article ID 138817, 2020.

[17] B. Yan, X. Tang, J. Zhou et al., "An improved method for the fitting and prediction of the number of COVID-19 confirmed cases based on LSTM," *Computers, Materials & Continua*, vol. 64, no. 3, pp. 1473–1490, 2020.

[18] V. K. R. Chimmula and L. Zhang, "Time series forecasting of COVID-19 transmission in Canada using LSTM networks," *Chaos, Solitons, and Fractals*, vol. 135, Article ID 109864, 2020.

[19] S. Roy, G. S. Bhunia, and P. K. Shit, "Spatial prediction of COVID-19 epidemic using ARIMA techniques in India," *Modeling Earth Systems and Environment*, vol. 7, no. 2, pp. 1385–1391, 2020.

[20] K. Demertzis, D. Tsiotas, and L. Magafas, "Modeling and forecasting the COVID-19 temporal spread in Greece: an exploratory approach based on complex network defined splines," *International Journal of Environmental Research and Public Health*, vol. 17, no. 13, pp. 1–18, 2020.

[21] I. G. Pereira, J. M. Guerin, A. G. S. Junior et al., "Forecasting covid-19 dynamics in Brazil: a data driven approach," *International Journal of Environmental Research and Public Health*, vol. 17, no. 14, pp. 1–26, 2020.

[22] M. Hawas, "Generated time-series prediction data of COVID-19's daily infections in Brazil by using recurrent neural networks," *Data in Brief*, vol. 32, pp. 1–18, Article ID 106175, 2020.

[23] P. Melin, J. C. Monica, D. Sanchez, and O. Castillo, "Multiple ensemble neural network models with fuzzy response aggregation for predicting COVID-19 time series: the case of Mexico," *Healthcare*, vol. 8, no. 2, p. 181, 2020.

[24] G. Pinter, I. Felde, A. Mosavi, P. Ghamisi, and R. Gloaguen, "COVID-19 pandemic prediction for Hungary; a hybrid machine learning approach," *Mathematics*, vol. 8, no. 6, 2020.

[25] P. Wang, X. Zheng, G. Ai, D. Liu, and B. Zhu, "Time series prediction for the epidemic trends of COVID-19 using the improved LSTM deep learning method: case studies in Russia, Peru and Iran," *Chaos, Solitons, and Fractals*, vol. 140, Article ID 110214, 2020.

[26] S. I. Alzahrani, I. A. Aljamaan, and E. A. Al-Fakih, "Forecasting the spread of the COVID-19 pandemic in Saudi Arabia using ARIMA prediction model under current public health interventions," *Journal of Infection and Public Health*, vol. 13, no. 7, pp. 914–919, 2020.

[27] Z. Hu, Q. Ge, S. Li, L. Jin, and M. Xiong, "Artificial intelligence forecasting of COVID-19 in China," 2020, https://arxiv.org/abs/2002.07112.

[28] T. Zeng, Y. Zhang, Z. Li, X. Liu, and B. Qiu, "Predictions of 2019-nCoV transmission ending via comprehensive methods," 2020, https://arxiv.org/abs/2002.04945.

[29] S. J. Fong, G. Li, N. Dey, R. G. Crespo, and E. Herrera-Viedma, "Finding an accurate early forecasting model from small dataset: a case of 2019-nCoV novel coronavirus outbreak," 2020, https://arxiv.org/abs/2003.10776.

[30] J. Li, Q. Dai, and R. Ye, "A novel double incremental learning algorithm for time series prediction," *Neural Computing and Applications*, vol. 31, no. 10, 2019.

[31] J. Zheng, X. Fu, and G. Zhang, "Research on exchange rate forecasting based on deep belief network," *Neural Computing and Applications*, vol. 31, 2019.

[32] W. Zou and Y. Xia, "Back propagation bidirectional extreme learning machine for traffic flow time series prediction," *Neural Computing and Applications*, vol. 31, no. 11, 2019.

[33] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: a review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–828, 2013.

[34] N. Tollenaar and P. G. M. van der Heijden, "Which method predicts recidivism best?: a comparison of statistical, machine learning and data mining predictive models," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 176, no. 2, pp. 565–584, 2013.

[35] Y. Du, Y. Cai, M. Chen, W. Xu, H. Yuan, and T. Li, "A novel divide-and-conquer model for CPI prediction using ARIMA, gray model and BPNN," *Procedia Computer Science*, vol. 31, 2014.

[36] J. Sun, X. Chen, Z. Zhang et al., "Forecasting the long-term trend of COVID-19 epidemic using a dynamic model," *Scientific Reports*, vol. 10, no. 1, p. 21122, 2020.

[37] K. Roosa, Y. Lee, R. Luo et al., "Real-time forecasts of the COVID-19 epidemic in China from February 5th to February 24th, 2020," *Infectious Disease Modelling*, vol. 5, pp. 256–263, 2020.

[38] L. Jia, K. Li, Y. Jiang, X. Guo, and T. Zhao, "Prediction and analysis of coronavirus disease 2019," 2019, https://arxiv.org/abs/2003.05447.

[39] Z. Yang, Z. Zeng, K. Wang et al., "Modified SEIR and AI prediction of the epidemics trend of COVID-19 in China under public health interventions," *Journal of Thoracic Disease*, vol. 12, no. 3, pp. 165–174, 2020.

[40] K. N. Nabi, M. T. Tahmid, A. Rafi, M. E. Kader, and M. A. Haider, "Forecasting COVID-19 cases: a comparative analysis between recurrent and convolutional neural networks," *Results in Physics*, vol. 24, 2021.

[41] M. Alazab, A. Awajan, A. Mesleh, A. Abraham, V. Jatana, and S. Alhyari, "COVID-19 prediction and detection using deep learning," *International Journal of Computer Information Systems and Industrial Management Applications*, vol. 12, 2020.

[42] L. Mohimont, A. Chemchem, F. Alin, M. Krajecki, and L. A. Steffenel, "Convolutional neural networks and temporal CNNs for COVID-19 forecasting in France," *Applied Intelligence*, 2021.

[43] A. G. Dastider, F. Sadik, and S. A. Fattah, "An integrated autoencoder-based hybrid CNN-LSTM model for COVID-19 severity prediction from lung ultrasound," *Computers in Biology and Medicine*, vol. 132, 2021.

[44] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: a comprehensive review," *Neural Computation*, vol. 29, no. 9, 2017.

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, 2017.

[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[47] Y. Lecun, B. Boser, J. S. Denker et al., "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, 1989.

[48] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks: The Official Journal of the International Neural Network Society*, vol. 18, no. 5-6, pp. 602–10, 2005.

[49] Z. Zhao, W. Chen, X. Wu, P. C. V. Chen, and J. Liu, "LSTM network: a deep learning approach for short-term traffic forecast," *IET Image Processing*, vol. 11, no. 1, 2017.

[50] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*, Wiley, New Jersey, NJ, USA, 5th edition, 2016.

[51] S. J. Taylor and B. Letham, "Forecasting at scale," *American Statistician*, vol. 72, no. 1, 2018.

[52] A. C. Harvey and S. Peters, "Estimation procedures for structural time series models," *Journal of Forecasting*, vol. 9, no. 2, 1990.

[53] S. C. H. Barrett and G. E. Hutchinson, "An introduction to population ecology," *Systematic Botany*, vol. 3, no. 3, 1978.

[54] S. Zhang, Q. Hu, Z. Xie, and J. Mi, "Kernel ridge regression for general noise model with its application," *Neurocomputing*, vol. 149, 2015.

[55] D. Yang, Z. Ye, L. H. I. Lim, and Z. Dong, "Very short term irradiance forecasting using the lasso," *Solar Energy*, vol. 114, 2015.

[56] T. Chen and C. Guestrin, "XGBoost: a scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13–17, pp. 785–794, August 2016.

[57] P. Tavallali, M. Yazdi, and M. R. Khosravi, "Robust cascaded skin detector based on AdaBoost," *Multimedia Tools and Applications*, vol. 78, no. 2, 2019.

[58] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, 2001.

[59] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics and Data Analysis*, vol. 38, no. 4, 2002.

[60] M. A. Hazar, N. Odabasioglu, T. Ensari, Y. Kavurucu, and O. F. Sayan, "Performance analysis and improvement of machine learning algorithms for automatic modulation recognition over Rayleigh fading channels," *Neural Computing and Applications*, vol. 29, no. 9, 2018.

[61] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
[62] H. Drucker, C. J. C. Surges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," 1997.
[63] M. F. Li, X. P. Tang, W. Wu, and H. B. Liu, "General models for estimating daily global solar radiation for different solar radiation zones in mainland China," *Energy Conversion and Management*, vol. 70, 2013.
[64] A. Dairi, F. Harrou, A. Zeroual, M. M. Hittawe, and Y. Sun, "Comparative study of machine learning methods for COVID-19 transmission forecasting," *Journal of Biomedical Informatics*, vol. 118, Article ID 103791, 2021.