Neural Computing and Applications (2024) 36:6119–6132
https://doi.org/10.1007/s00521-023-09398-9

ORIGINAL ARTICLE

A predictive analytics framework for sensor data using time series and deep learning techniques

Hend A. Selmy[1] · Hoda K. Mohamed[1] · Walaa Medhat[2,3]

Received: 5 July 2023 / Accepted: 13 December 2023 / Published online: 18 January 2024

Abstract

IoT devices convert billions of objects into data-generating entities, enabling them to report status and interact with their surroundings. This data comes in various formats, like structured, semi-structured, or unstructured. In addition, it can be collected in batches or in real time. The problem now is how to benefit from all of this data gathered by sensing and monitoring changes like temperature, light, and position. In this paper, we propose a predictive analytics framework constructed on top of open-source technologies such as Apache Spark and Kafka. The framework focuses on forecasting temperature time series data using traditional and deep learning predictive analytics methods. The analysis and prediction tasks were performed using Autoregressive Integrated Moving Average (ARIMA), Seasonal Autoregressive Integrated Moving Average (SARIMA), Long Short-Term Memory (LSTM), and a novel hybrid model based on Convolution Neural Network (CNN) and LSTM. The purpose of this paper is to determine whether and how recently developed deep learning-based models outperform traditional algorithms in the prediction of time series data. The empirical studies conducted and reported in this paper demonstrate that deep learning-based models, specifically LSTM and CNN-LSTM, exhibit superior performance compared to traditional-based algorithms, ARIMA and SARIMA. More specifically, the average reduction in error rates obtained by LSTM and CNN-LSTM models were substantial when compared to other models indicating the superiority of deep learning. Moreover, the CNN-LSTM-based deep learning model exhibits a higher degree of closeness to the actual values when compared to the LSTM-based model.

Keywords Deep learning · Time series · Predictive analytics · Internet of things · IoT · LSTM · CNN-LSTM · ARIMA · SARIMA

1 Introduction

The Internet of Things (IoT) is a global network that facilitates communication among individuals, objects, and objects themselves by allocating a distinct identification to each entity. Big Data and the IoT are interconnected concepts that are closely related. In recent years, there has been a significant emphasis and focus on big data analytics in the context of IoT, mostly because of the substantial volume, velocity, variety, and veracity of the data involved. As IoT technology advances [1], an increasing number of sensors are being implemented in a variety of sectors, including transportation, industrial manufacturing, healthcare, and many more. This progression leads to the continuous generation of a substantial volume of time series data.

✉ Hend A. Selmy
1902141@eng.asu.edu.eg

Hoda K. Mohamed
hoda.korashy@eng.asu.edu.eg

Walaa Medhat
WMedhat@nu.edu.eg

1 Department of Computer and Systems Engineering, Faculty of Engineering, Ain Shams University, Cairo, Egypt

2 Information Technology and Computer Science, Nile University, Giza, Egypt

3 Faculty of Computer and Artificial Intelligence, Benha University, Cairo, Egypt

The sources of the time series data may include environmental pollution, architecture, meteorology, finance, transportation, medical treatment, and more, so this data mirrors a variety of evolving phenomena and events. Concurrently with this, the complexity of time series data is growing. The substantial quantity and complexity of the data causes further difficulties to the process of analysis to tackle these difficulties, a multitude of technologies are utilized to manage [2], store [3], process, and analyze [4] time series data with the objective of extracting valuable insights, such as providing forecasts.

Time series forecasting can be defined as a technique for predicting future values by understanding past values; it involves the use of certain predictive analytics models to provide forecasts based on historical data in order to create fitting models that characterize the underlying data structure and properties [5, 6]. Numerous studies have been conducted on the topic of time series prediction, employing a variety of methodologies. Commonly used techniques for time series forecasting encompass ARIMA [7] and SARIMA models. However, it should be noted that these models possess inherent constraints and are not proficient in effectively extracting nonlinear features [8].

ARIMA model [9] has gained widespread attention because of its statistical flexibility and its use of a number of exponential smoothing approaches. It takes into account the past values (autoregressive, moving average) and predicts future values based on that. However, ARIMA models work efficiently when the time series is stationary with no missing values that can be filled using advanced interpolation techniques [10, 11]. SARIMA works in a similar way like ARIMA in addition to looking at seasonal patterns. Since seasonality is a parameter in SARIMA, it is better than ARIMA at making predictions in complex data spaces with cycles where seasonality is a factor [12, 13].

The utilization of machine learning has facilitated the application of many deep learning models in the field of time series prediction. These models include multilayer perceptron (MLP), recurrent neural network (RNN), long short-term memory (LSTM), convolution neural network (CNN), and gated recurrent units (GRU).

Deep learning techniques are the most popular methods for developing the current state-of-the-art solutions for time series forecasting. It outperforms machine learning models in accuracy and is effective at feature engineering. LSTM is a model improved by RNN architecture, proposed by Hochreiter and Schmidhuber in 1997 [14]. LSTMs preserve memory of the input internally. Therefore, they are ideal for situations involving sequential data because they can capture nonlinear trends and dependencies [15, 16]. It was first introduced to resolves the vanishing gradient RNN issue because it has a longer memory and can learn from inputs with substantial time lags [17]. In recent years, LSTM has shown promising results in time series analysis as it can figure out what would happen in the future based on different sequences of different lengths. Furthermore, unlike most other machine learning models, a multi-step LSTM network is highly efficient at automatically learning features from time series and sequence data.

To enhance the efficacy of LSTM in processing time series data, it is proposed to incorporate a CNN to extract the features from the input dataset. The application of a composite structural model that combines CNN and LSTM models for time series prediction demonstrates significant performance [18].

In this paper, the primary objective is to propose a comprehensive framework for big data management, specifically designed to handle IoT sensor data in either batch or real-time processing scenarios. This approach integrates both traditional statistical and deep-learning predictive models. The study continues by conducting a comparative analysis of the results acquired, with the aim of selecting the predictive models that have the highest level of accuracy in making predictions. The case utilized to assess this framework concerns the temperature of Delhi's climate. The work demonstrates innovation through the demonstration, and the prediction results are subjected to comparison, evaluation, and visualization.

The main contributions in this paper include the following: (1) the utilization of advanced open-source tools such as Apache Spark and Kafka for data processing and analysis, as well as Hadoop Distributed File System (HDFS) for storage, in the implementation of ARIMA, SARIMA, LSTM, and hybrid CNN-LSTM models. (2) design of the CNN–LSTM model with forgetting input, and output gates to capture the long-term memory and reduce the dimensionality of weather temperature data; (3) comparison of the performances of different deep learning models by using RMSE, MAE, MAPE, and goodness of fit. Its novelty is that the integration of CNN and LSTM with forgetting, input, and output gates to predict Delhi temperature time series data.

The rest of the paper are organized as follows: Sect. 2 presents related work and explains how we extended prior research. Section 3 explains our proposed framework along with descriptions of the various components involved in its implementation and explains the proposed techniques for time series modeling and forecasting. In Sect. 4, the experiments and results are presented and discussed. In Sect. 5, the conclusions and directions for future works are provided.

2 Literature review

This study presents the utilization of predictive analytics models, particularly ARIMA, SARIMA, LSTM, and hybrid CNN-LSTM, for the purpose of forecasting the temperature of Delhi's weather. The models are implemented using a big data framework to use the advantages of batch and real-time processing for the time series input data. The present study aims to enhance the proficiency and distinctiveness by discussing the following relevant studies.

In [19], the proposed work analyzed rainfall data over a time period of 11 years (2010–2020) and focused on the ARIMA method for predicting monthly rainfall in the Bengaluru area. Regarding future recommendations, this research suggests acquiring additional data to improve the performance of the ARIMA model.

The paper [20] analyzes the monthly mean temperature in Nanjing, China, from 1951 to 2017 using Seasonal Autoregressive Integrated Moving Average (SARIMA) techniques. The paper explains the steps involved in time series forecasting and model selection. It presents the results of the analysis. The study shows that the proposed research approach obtains good forecasting accuracy. However, it would have been helpful if the paper had discussed the limitations of the study and the potential sources of error in the forecasting models.

The author in [21] used LSTM for weather forecasting. The paper discusses the limitations of traditional weather prediction methods and how LSTM can overcome these limitations by retaining relevant information for the long term and forgetting irrelevant information. The paper also explains the use of gates in LSTM to regulate the addition, keeping, and removal of information. The paper concludes that LSTM has shown substantial performance in a variety of real-world applications, including weather forecasting.

The research [22] focuses on weather forecasting, specifically visibility, in the context of flight navigation. The authors have made significant contributions to the project, including coordinating development, conducting experiments, analyzing data, and writing the manuscript. The research methodology involves using ARIMA and LSTM models to forecast visibility at Hang Nadim Airport in Batam, Indonesia, using weather data as predictors and moderating variables. The experimental findings indicate that the LSTM model exhibits superior performance in comparison to the ARIMA model, both with and without intermediate variables. However, further evaluation and validation of the models may be necessary to determine their accuracy and effectiveness in real-world scenarios.

CNN and LSTM networks were used to create a powerful solar energy forecasting algorithm [23]. The CNN extracted spatial elements from historical time records. Then, the LSTM extracted temporal information and predicted solar energy production. The CNN-LSTM hybrid was created for forward and backward time series. The suggested forecasting method was trained and evaluated using Moroccan photovoltaic plant data. Production, weather, and produced power consumption data are included. Experimental findings showed that the CNN-LSTM algorithm can forecast properly.

The article [24] discusses the development of a hybrid CNN-LSTM based time series data prediction model for analyzing and predicting air pollution information. The article discusses the limitations of traditional prediction models through statistical approaches and the potential of deep learning in improving predictive power in various fields dealing with time series data. However, the article does not provide a thorough analysis of the model's performance or compare it to other existing prediction models.

The paper [25] introduces big data framework that focuses on predicting weather conditions using machine learning models. The study evaluates the performance of different classifiers, including Classification and Regression Trees (CART), K-nearest neighbor, Random Forest, and Support Vector Machine. The Random Forest classifier achieved the highest classification accuracy of around 89%.

In [26], the authors utilized data science techniques such as data mining, machine learning, and data visualization**.** They created a platform for big data that enabled the integration and combination of unstructured data from multiple sources in order to address problems in the production of distribution networks. The results indicated that data mining techniques can considerably reduce the operational costs of distribution networks and increase the reliability of the power supply.

The Lambda architecture is introduced in [27]; it is a scalable and fault-tolerant framework for handling both real-time and historical data concurrently. Lambda architecture was developed to quickly, accurately, and reliably process massive amounts of incoming data. Its primary goal was to accelerate the processing of Online Analytical Processing (OLAP), such as page view and click stream analysis. It was not aimed to make decisions or react to occurrences on an event-by-event basis. It is complex and challenging to deploy and manage, as it consists of batch, speed, and serving layers that need to coordinate closely with one another.

3 The proposed framework

The proposed predictive analytics framework depicted in Fig. 1 is used for weather temperature forecasting. It can process and analyze the dataset by applying traditional time

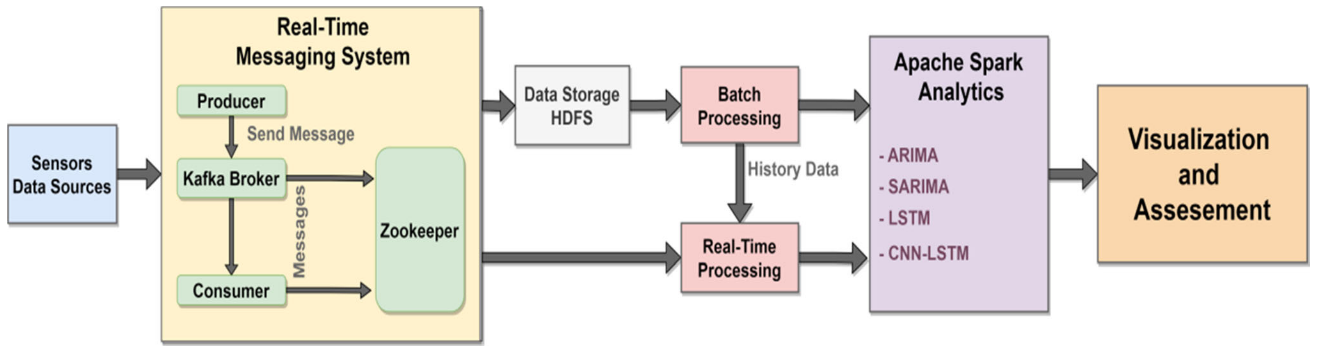


Fig.1 Proposed framework architecture

series forecasting methods, including ARIMA and SARIMA. Although predictions of temperature trends using these models have been made frequently, However, they might not work well with data that has nonlinear relationships. To address this limitation, the LSTM model and a novel hybrid prediction model that combines CNN and LSTM are presented. The framework can handle and analyze batch and stream data from IoT devices; the accuracy has been computed; and the findings have been evaluated and visualized. The framework meets the requirements of scalable historical data analytics as well as efficient real-time processing for IoT sensor datasets.

Compared to more conventional data mining tools, the use of Spark in this system significantly accelerates the processing of data. The primary distinction between the proposed framework and conventional techniques for data analytics is that conventional methods evaluate one instance at a time and depend on the volume of imported data. This proposed framework, however, can handle thousands of instances in real time using Spark Discretized Stream (Dstream), a collection of Resilient Distributed Datasets (RDDs), where each RDD represents one or more instances [28].

Kafka is chosen because it can handle massive backlogs of messages to accommodate periodic ingestion from numerous systems, and it also allows consumers to re-read messages if necessary. Consumers need to connect to Zookeeper to find out the location of the Kafka brokers that they need to connect to in order to consume the demanded data [29, 30]. Furthermore, Kafka supports both online and offline batch consumers who require low latency, and the input weather temperature dataset is employed as a data producer by Apache Kafka.

The Real-Time Messaging System component is responsible for publishing the input data to Apache Kafka Message Brokers for data ingestion because they allow publishing messages to specific topics and then subscribe to those topics. Each sensor node (or group of nodes) sends training and testing events per second in a predefined format to the specified topic. To execute either real-time or batch processing, these topics are either transmitted to the Spark streaming application or stored in the Hadoop Distributed File System (HDFS) [31]. At this stage, the amount and complexity of the data can be determined, affecting the architecture and all future decisions. The data will be retrieved, manipulated, and stored as quickly as is feasible.

In fact, IoT stream processing can be used independently in situations where the IoT stream data can be analyzed without the need for historical data. Historical data can usually provide additional insight to make intelligent recommendations based on real-time data. Moreover, the batch flows can operate separately from the real-time flows, either to provide long-term insight or to train predictive models using historical datasets.

Batch data processing begins with importing the data, ends with scaling the data, and transforms it into a data frame. Afterward, data processing using Apache Spark begins, while the historic data is stored in HDFS. The batch processing may operate on very huge datasets, for which the computation is time-consuming. It works well either when real-time analytics results are not required or when it is more critical to process a huge volume of data simultaneously.

In-stream data processing Kafka streaming and Spark streaming are integrated to analyze and store data for in-process analytics in real time. Apache Kafka picks up the message and saves it by writing it to its broker. In addition, it sends those messages to Apache Spark, then to Apache Spark Structured Streaming to be processed. Algorithm 1 indicates the steps.

Algorithm 1 Kafka and spark stream processing

```
Inputs: Sensor Data Stream.
Output: Processed Data Stream.
    Step1: Create an instance of "SparkContext" and "StreamingContext", for complete utilization of Spark streaming.
    Step2: Get Kafka streams using the create "DirectStream" method of "KafkaUtils". It creates the direct stream with the Kafka parameters and topic.
    Step3: Data processing using the "ForeachRDD" method, extracts identifier and attributes from each stream and each topic.
    Step4: Apply time series and deep learning techniques to do predictive analytics.
    Step5: Start the computation streaming context using the " StartSpark " method.
```

For further details, once the SparkContext is created, Spark treats the input training dataset as an RDD via the textFile() function. Specifically, the cache() method is employed to store a previously calculated RDD for later usage. Spark uses the SparkSQL and SparkMLib libraries to manage the Kafka topics that are stored in the datasets.

Apache Spark analytics is the main component in the proposed framework, where the time series data is analyzed, visualized, and evaluated using ARIMA, SARIMA, LSTM, and CNN-LSTM predictive models.

In addition, being a general system, it can support batch, interactive, iterative, and streaming computations in the same runtime, which is advantageous for complicated applications with several computing techniques. In this stage, a comparison between time series and deep learning techniques is conducted. The accuracy of these models is verified using the Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Average Absolute Percentage Error (MAPE); the smaller the values, the better the model performance [32, 33].

3.1 ARIMA prediction model

The ARIMA model is one of the most used and well-known statistical forecasting models for time series. It is a statistical method that is used for predicting based on historical values that are lag values and lagged forecast value errors. The ARIMA model, often known as the Box-Jenkins model, has gained widespread popularity due to its impressive forecasting accuracy and efficiency in utilizing historical data to predict future data values [34].

This model can predict time series data that does not have seasonality or a trend [35]. The ARIMA model has three parts: autoregressive (AR), integrated (or differencing), and moving average (MA) terms. It is characterized by three parameters: p, q, and d, where "p" is the order of the AR term known as auto-correlation, "q" is the order of the MA term known as the partial autocorrelation, and "d" is the number of differencing required to make the time series stationary [36]. The ARIMA model can be expressed as the following Eq. (1):

$$y'_t = c + \delta_1 y'_{t-1} + \ldots + \delta_p y'_{t-p} + \phi_1 \varepsilon_{t-1} + \ldots + \phi_q \varepsilon_{t-q} + \varepsilon_t \tag{1}$$

where y'_t is the differencing series, c is a constant, $\delta_1 y'_{t-1} + \ldots + \delta_p y'_{t-p}$ is the AR term with δ_1 to δ_p as coefficients at p order, $\phi_1\varepsilon_{t-1} + \ldots + \phi_q\varepsilon_{t-q}$ is the MA term with ϕ_1 *to* ϕ_q as coefficients at q order, and ε_t is an error term for random background noise at time t. Point and interval forecasts can be created once the orders and coefficients for a certain time series have been estimated by fitting the model to historical data.

The ARIMA model is completed by adjusting the parameters p, d, and q based on the dataset at hand. In this study, the parameters are calculated using an "auto-ARIMA" function of the "pmdarima" library [37] that uses the Akaike Information Criterion (AIC) and the Bayesian Information Criterion (BIC) to determine which parameters to use in the final model [38]. The BIC indicator is preferred over the AIC indicator because the former is more consistent and penalizes free parameters.

3.2 SARIMA prediction model

SARIMA is an extension of ARIMA that explicitly supports seasonal components in univariate time series data. Unfortunately, the basic ARIMA model does not supporting seasonality. For this reason, the SARIMA model is suggested by Box and Jenkins as a highly successful version of the ARIMA model, called the Seasonal ARIMA (SARIMA), to account for the fact that the periodicity of periodic time series is typically due to seasonal variations or other natural reasons [39]. SARIMA is formed by adding seasonal parameters in the ARIMA model and written as shown in Eq. (2).

$$\mathrm{SARIMA}(p, d, q)(P, D, Q)m \quad (2)$$

In this equation, (p, d, q) and $(P, D, Q)_m$ are the non-seasonal and seasonal part of the model, respectively. The parameter "m" is the number of periods per season. The parameters P, D and Q are seasonal autoregressive parameter, seasonal integrated parameter and seasonal moving average parameter, respectively. Practically, the order of the SARIMA model is generally not too high [40].

In general, the model is non-stationary, Thus, it is necessary to check the original dataset for stationarity before using the method and this can be performed using the Augmented Dickey-Fuller (ADfuller) test [41]. This test is founded on a hypothesis, where, if the p-value is less than 0.05, then we can consider the time series stationary, and if the p-value is greater than 0.05, then the time series is non-stationary. The library "statmodels" is used to perform the ADfuller stationarity test and implement the SARIMA method [42]. The parameters $(p, d, q)(P, D, Q)$ selection can be performed automatically using the "auto_ARIMA" function of the "pmdarima" library.

3.3 LSTM prediction model

LSTM is a type of network that facilitates data persistence [43]. It is particularly well suited for handling large amounts of data due to its ability to use nonlinear activation functions in each layer and manage large parameter sizes [44]. Unlike the original RNN, LSTM solves the issue of vanishing gradient problem by processing single data points and entire data sequences, while also capturing nonlinear trends in the data and retaining past information for an extended period [45].

LSTMs update its cell state and various gates and has the ability of learning from long-term dependencies [46, 47]. The LSTM architecture comprised multiple hidden LSTM layers. When LSTM hidden layers are stacked, the model becomes more complex and properly classified as a deep learning method.

The four components of an LSTM memory cell are depicted in Fig. 2; these are the input gate, the output gate, the forget gate, and the self-circulation neuron. Memory cells can communicate with one another and their neighbours via an input gate. The input gate determines if the state of the memory cell can be changed by the incoming signal. On the other hand, the output gate could regulate the memory cell's state based on whether or not the other memory cell's state is mutable. The forgetting gate can also choose to remember or discard the previous state [48].

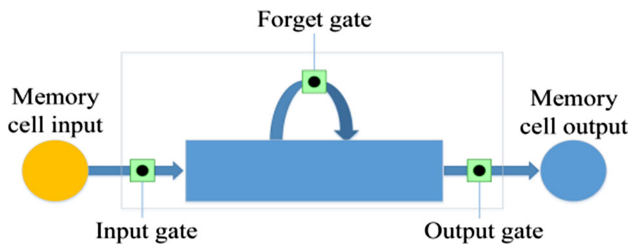


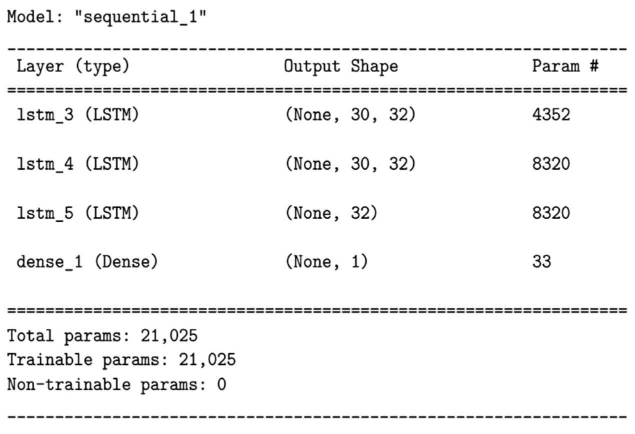
Fig. 2 LSTM unit structure [49]

In this study, we used a three layers stacked LSTM architecture. The first and second hidden layers contained 32 cells with the activation function (relu), which overcomes the problem of vanishing gradients, where the accuracy and efficiency of output prediction are at their maximum. The third layer contains 32 cells with the activation function (sigmoid) to predict the probability of the output. The output layer consists of a single fully connected neuron. The model description is shown in Fig. 3.

We used the "keras" library and a set of functions Sequential, Dense, and LSTM, which were used to configure the neural network [50, 51]. The model was constructed using the following hyperparameters: Units = 32—the output space dimensionality; batch size = 30—number of samples per gradient update; epochs = 500—number of epochs for the model training. The optimization algorithm, Adaptive Moment Estimation (Adam) [52] was used with a learning rate of 0.001, which has been demonstrated to be effective in practice. Early stopping was used as a type of regularization to prevent over fitting.

3.4 CNN-LSTM prediction model

Convolution Neural Networks (CNNs) are a class of deep learning models specifically developed to effectively process data that exhibits grid-like structures, such as photographic data. CNNs typically consist of three main components: convolution, pooling, and a fully connected layer, as depicted in Fig. 4.


```
Model: "sequential_1"

_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 lstm_3 (LSTM)                (None, 30, 32)            4352

 lstm_4 (LSTM)                (None, 30, 32)            8320

 lstm_5 (LSTM)                (None, 32)                8320

 dense_1 (Dense)              (None, 1)                 33

=================================================================
Total params: 21,025
Trainable params: 21,025
Non-trainable params: 0
_________________________________________________________________
```

Fig. 3 Proposed LSTM predictive model

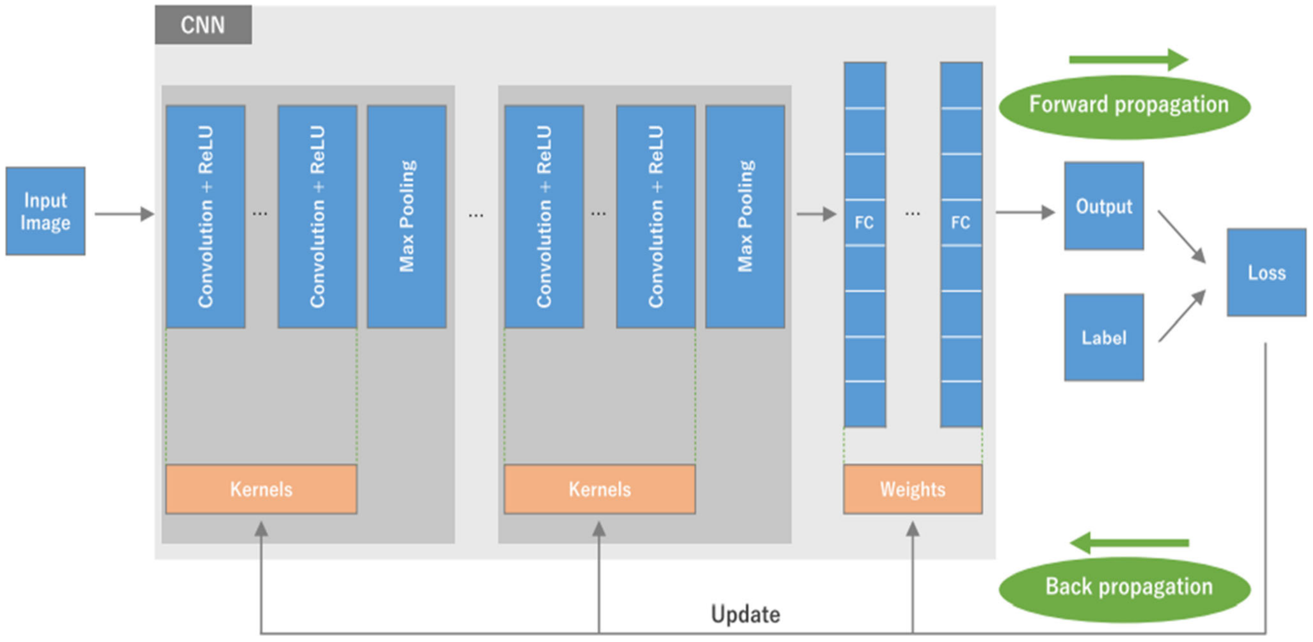


Fig. 4 CNN structure and learning process [54]

In the convolution and pooling layers, feature extraction is carried out. Subsequently, the extracted features are mapped by the fully connected layer into the final output, which may involve classification. Pixel values in digital images can be represented as two-dimensional grids that are regarded as matrices. Kernels, an additional small grid parameter, are employed to extract image features. The images features that are extracted by the "kernel" are what make the whole picture learning process efficient [53].

This research presents a novel approach for temperature time series data forecasting by proposing a hybrid CNN-LSTM deep learning model. The forecasting process functions as a sequence-to-sequence learning paradigm, where a series of historical data points is utilized to forecast a subsequent series of future data points. The preceding 12-month data is utilized to forecast the values for the subsequent 6-month period. The model acquires knowledge of the patterns that are anticipated on the test dataset that has not been previously observed. The model effectively captures both seasonality and trends.

The initial stage involves executing matrix operations using a three-dimensional array structure for CNN operations on one-dimensional time series data. Additionally, the time series data, which has been processed as a three-dimensional array, undergoes a convolution operation at the CNN layer.

The feature map is obtained by performing computations on the CNN layer using Conv1D. Subsequently, the LSTM cells of the recurrent layer are connected to learn the time series data. The architectural design possesses the capability to autonomously acquire knowledge of the series' seasonality and trend, eliminating the need for thorough fine-tuning. The schematic representation of the suggested model is depicted in Fig. 5.

The data is first reshaped and rescaled to fit the three-dimensional input requirements of Keras sequential model. The input shape would be with one feature for a simple univariate model. And we chose the kernel size to be 5. The dense layer produces six output numbers. Below, in Fig. 6, is a detailed model summary.

Batch input size is 16. In the end, the model is fitted for 200 epochs and output the loss. Mean squared error loss function and Adam optimization were used.

The performance methodology for forecasting temperature in this model is accomplished by obtaining a high-quality foundational training dataset through the procedures of data selection, collection, and preparation. The next step involves performing a transfer test and resolving any missing values within the dataset that has been adopted. The model is acquired through the process of tracking and determining the ideal hyperparameters for the proposed model.

4 Experiment and results

The efficacy of the suggested framework was thoroughly examined through extensive testing. The evaluation of the forecasting models was conducted using a dataset that is publicly accessible. Various evaluation indicators were employed to assess the performance of the predictive models.

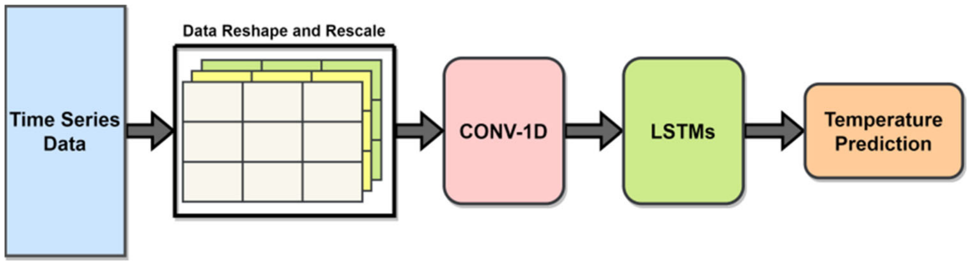
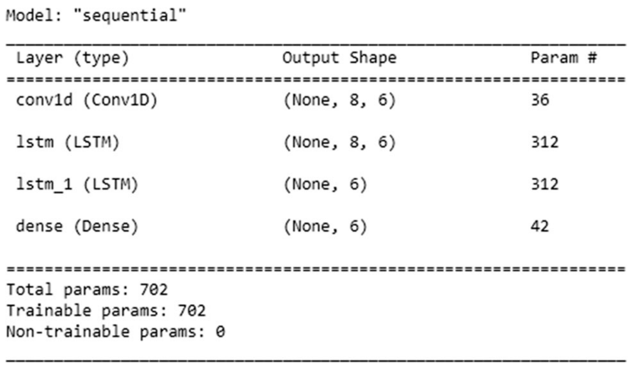


Fig. 5 Proposed CNN-LSTM structure



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv1d (Conv1D)             (None, 8, 6)              36

 lstm (LSTM)                 (None, 8, 6)              312

 lstm_1 (LSTM)               (None, 6)                 312

 dense (Dense)               (None, 6)                 42

=================================================================
Total params: 702
Trainable params: 702
Non-trainable params: 0
_________________________________________________________________
```

Fig. 6 Proposed CNN-LSTM predictive model

4.1 Experiment details

The framework is flexible enough to be used for many different IoT applications. Here, we show how it has been used in the real world and how it can be used to optimize, automate, and provide context-aware solutions for large-scale IoT deployments. In the subsequent sections, we will provide an overview of the dataset utilized for both training and evaluating the suggested forecasting models. Furthermore, we will present the assessment measures employed for the purpose of evaluation. Moreover, the obtained outcomes are provided and analyzed to demonstrate the effectiveness of the predictive models and their influence on temperature prediction.

4.2 Dataset

The DelhiDaily climate open dataset [55] was used to obtain the training data where the average daily temperatures are computed from 24 hourly temperature sensors readings since the air temperature is the most common and accurate way to tell what the weather will be like. The collected data was regularly updated on a daily basis. The data was gathered across a period of four years, specifically from 2013 to 2017, hence offering a historical time series and information on weather conditions.

Data are divided into train and test sets for model implementation and accuracy. The training set is the period from 01 to 01-2013 to 31-12-2015, and the test set is from 01 to 01-2016 to the beginning of 2017. This information is confirmed in the Exploratory Data Analysis (EDA) time series plot of the Delhi mean temperature in Fig. 7.

The time series data has seasonality pattern, for instance, temperatures are always low at the beginning of the year and high at the middle of the year.

4.3 Experimental details

In numerical weather prediction, two approaches exist. First is using weather data approach where machine learning techniques are used. Second is using time series data prediction approach where Artificial Neural network (ANN) and ARIMA model-based techniques are all part of this approach [56]. The following predictive models aim to forecast using ARIMA, SARIMA, LSTM, and CNN-LSTM deep learning and comparatively analyze prediction results of each of these models based on performance.

4.3.1 ARIMA

The ARIMA model is employed for the purpose of forecasting time series temperature data. The determination of the precise values for p, d, and q is subsequently conducted through the utilization of the least information criterion (AIC) and the Bayesian criteria (BIC). Models with smaller values of AIC and BIC tend to exhibit higher performance. Based on the computation, it has been determined that the ideal parameters for the ARIMA model are (1, 1, 1), with $p = 1$, $d = 1$, and $q = 1$. The AIC and BIC values for this model are 4161.59 and 4176.59, respectively, representing the minimal values achieved.

The residuals of the ARIMA (1,1,1) model were examined for white noise using the Python programming language. The obtained P-value from the statistical test was 0.8975, indicating a lack of statistical significance since it exceeds the commonly accepted threshold of 0.05. This finding suggests that the residuals did not meet the criteria for significance testing, indicating a lack of independence in the residuals. Based on the results depicted in Fig. 8, it can be deduced from the graphical representation that the model exhibited a lack of precision in forecasting the daily mean temperature, as it deviated from the anticipated trend.

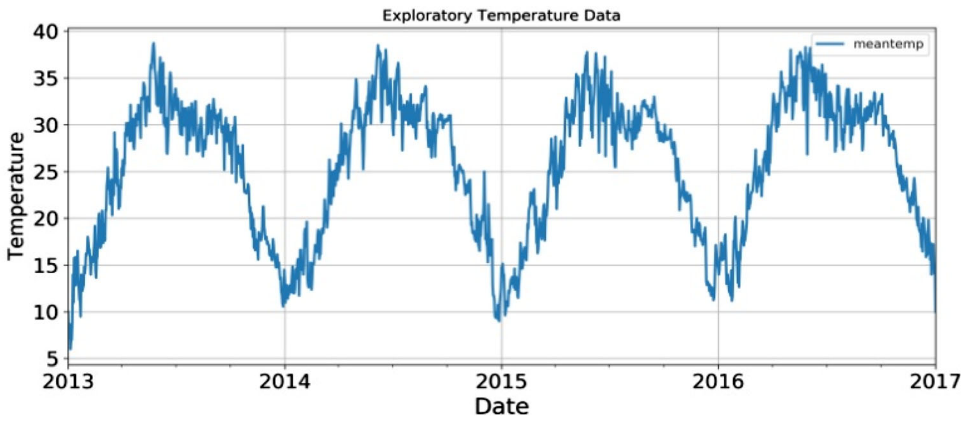


Fig. 7 Delhi temperature exploratory temperature data

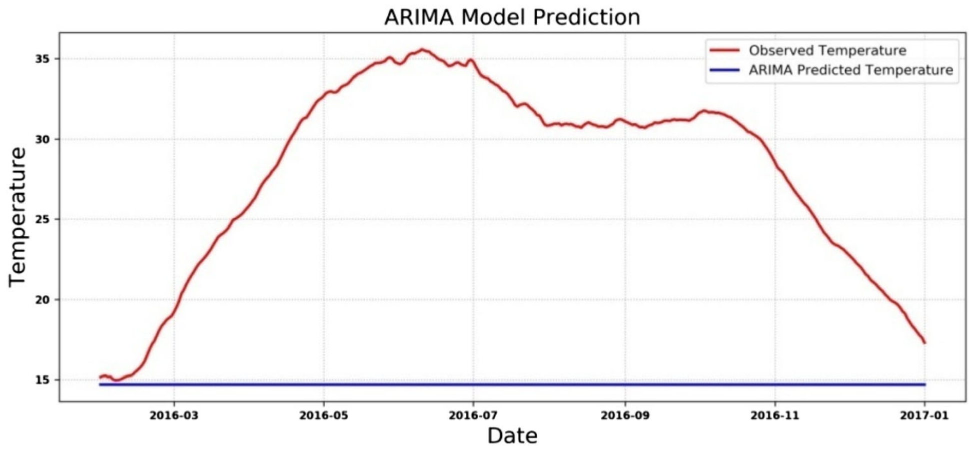


Fig. 8 ARIMA model visualization for temperature prediction

4.3.2 SARIMA

The SARIMA model is utilized to make linear predictions on the trend and seasonal components of a given time series, specifically in the context of temperature data. The utilization of a Seasonal Trend Decomposition Procedure based on Loess (STL) decomposition is employed for the preprocessing of sequences. The process of conducting local weighted regression involves use STL decomposition [57] to break down a sequence into three distinct components: the trend term, the seasonal term, and the residual term. These components can be mathematically represented as shown in Eq. (3).

$$X_t = T_t + S_t + R_t \tag{3}$$

The variable T_t represents the trend component of the data at time t, whereas S_t represents the seasonal component and R_t represents the residual component. The STL is comprised of two distinct loops, namely the inner loop and the outer loop. The primary function of the inner loop is to conduct trend fitting and compute periodic components. The inner loop can be subdivided into six distinct steps, which are as follows:

The first step is detrending, which entails subtracting the trend component from the previous round's outcome. The second step involves the application of cycle-subseries smoothing. The third step involves the implementation of a low-pass filtering technique. The fourth step involves the detrending of the smoothed cycle subseries. the fifth step is deseasonalization. The final step in the analysis process involves the application of trend smoothing techniques. The primary purpose of the outer loop is to modify the robustness weight, as depicted in Fig. 9.

There are a total of four distinct subgraphs, arranged in a vertical sequence from the highest to the lowest. The initial subgraph represents the original sequence graph, while the subsequent subgraphs, namely the second, third, and fourth subgraphs, depict the trend, seasonal, and residual sequences, respectively, which were obtained by STL decomposition.

The data exhibits both seasonal and nonseasonal components, with the temperature series displaying an annual trend. Hence, the SARIMA $(p, d, q) \times (P, D, Q)_m$ model is employed for forecasting the linear component. The optimal parameters of the model are determined using the Akaike information criteria, and the residual term is

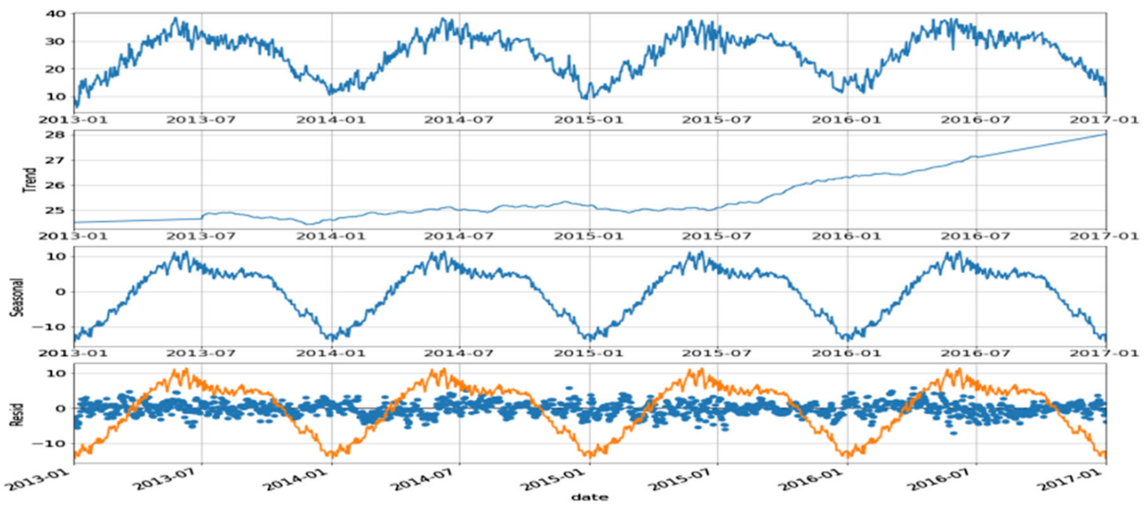


Fig. 9 Seasonal decomposition

excluded from the temperature sequence as an input variable. The model parameter results produced are (0, 0, 1) and (0, 1, 1). The value of "*m*" is set as 12 due to the monthly data being used, which has a duration of 12 months. Hence, the SARIMA (0, 0, 1) x (0, 1, 1) [12] model combination is determined to be the most effective parameter configuration.

The utilization of this model enables the generation of forecast outcomes according to both trend and seasonal components. The results of the forecast for the trend and seasonal variables in this section are depicted in Fig. 10.

SARIMA predictions based on the past up to the point in time of the planning to make forecasts have been the most accurate so far. This is particularly true for weather forecasting because the temperature today isn't much different from yesterday, and barring any additional factors, the temperature should indicate the start of a new season.

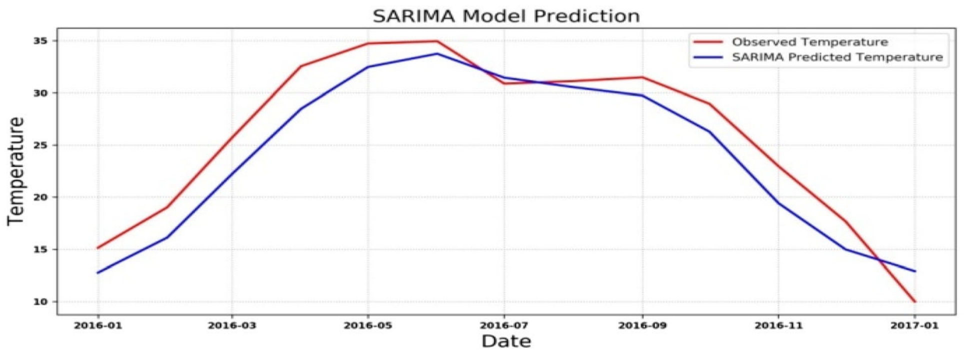
4.3.3 LSTM

The presented LSTM model consists one input layer, three LSTM layers, and a fully linked layer. The model's hyperparameters were evaluated using a series of experiments. Specifically, the model performed 500 iterations on the training set, with a training step value of 30. The model is applied to the divided dataset to train and test data, then neural network inputs are normalized, and then the network is trained for 500 epochs with a validation loss of MSE 0.0043 on the final epoch. The optimization algorithm is Adam with a learning rate of 0.001 and Early stopping was used for regularization.

The training loss exhibited a quick decline as the number of iterations increased, eventually reaching a state of smoothness after 150 iterations. This observation suggests that the model achieved a good match, as visually depicted in Fig. 11.

The results demonstrate that the comparison between predicted and actual values for the test dataset's arbitrary forecast points, one year in advance, can be considered satisfactory. The temperature forecast results for the test data that extend from January 2016 to January 2017 are depicted by the blue line. Conversely, the red line represents the actual temperature values for the corresponding data points. In this case, the model seems to accurately predict the temperature, with some small differences

Fig. 10 SARIMA Model visualization for weather



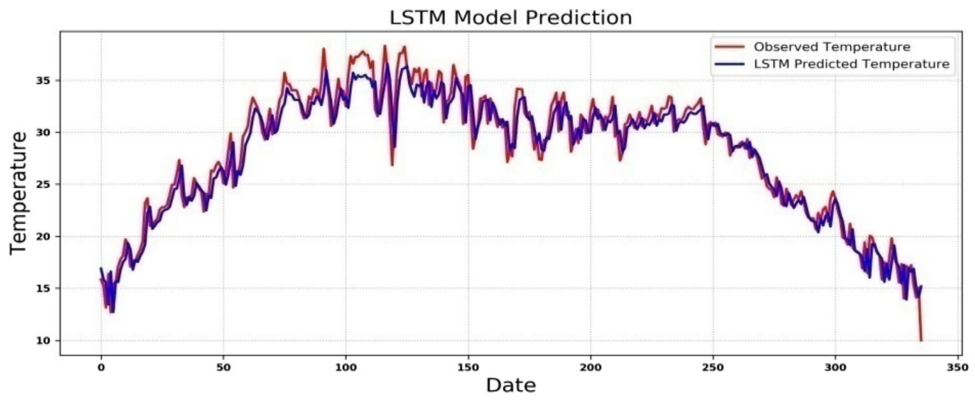


Fig. 11 LSTM Model prediction vs. actual values

between the actual temperature and the predicted temperature.

4.3.4 CNN-LSTM

CNNs provide the capability to effectively extract features, namely in the temporal dimension, hence enabling the identification of short-term patterns. Furthermore, it is possible for them to represent the interrelationships among variables at a local level. In contrast, LSTM models have the capability to capture temporal dependencies within time series data. CNNs combined with LSTM models has yielded good outcomes in the domain of time series data forecasting. The model has a high level of accuracy in its temperature predictions, as illustrated in Fig. 12.

The analysis of the pre-processed data is conducted by the CNN, while the prediction task is handled by the hybrid CNN-LSTM model. The utilization of LSTM in this context is motivated by its inherent ability to learn and make predictions within a sequential framework. In this context, the utilization of the first-order algorithm is employed for optimization as an alternative to the gradient descent method.

After the completion of pre-processing, the data analysis is conducted utilizing the CNN algorithm. Subsequently, the forecast is conducted with the hybrid CNN-LSTM model, which incorporates both CNN and LSTM components, as outlined in the steps that follow.

In the first step, the CNN model is provided with the cleaned data as its input. The CNN algorithm is employed to ascertain several properties of the provided information. The next one is pooling, wherein the significant features are extracted from the input data. Pooling operations are employed to decrease the dimensions of the input while retaining its fundamental characteristics. In the next step of the flattening process, the one-dimensional dimension of the data is further transformed into a one-dimensional format, with the purpose of facilitating analysis.

In the second step, the output of the CNN layer is subsequently inputted into the LSTM networks. Within the LSTM, the cell state plays a crucial role in providing memory to the network, enabling it to retain information from previous time steps. The gates inside the LSTM architecture are responsible for implementing the sigmoid function, which restricts the output values to a range between 0 and 1.

In the third step, the output of the hybrid model is passed on to the optimization model, which performs optimization using the first-order optimization process, namely gradient descent. In order to improve the model, it is necessary to optimize the weight values that are associated with it. In Step 4, the outcomes and visual representations are graphed in relation to the weight optimization technique, and the temperature data is determined for the purpose of analysis.

Compared to the predictive performance exhibited by the LSTM-based deep learning model, the CNN-LSTM-

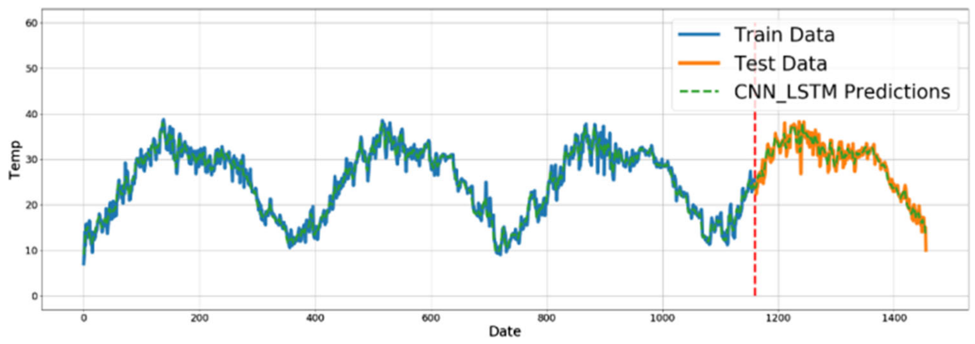


Fig. 12 CNN-LSTM model prediction vs. actual values

based deep learning model demonstrates a higher degree of proximity to the real value.

4.4 Evaluation metrics

The suggested prediction models were evaluated using three statistical metrics: Root Mean Square Error (RMSE), Average Absolute Percentage Error (MAPE), and Mean Absolute Error (MAE). Equations (4), (5), and (6) are used computation.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=0}^{N-1} (y_i - \hat{y}_i)^2}{N}} \tag{4}$$

$$\mathrm{MAE} = \frac{\sum_{i=0}^{N-1} |y_i - \hat{y}_i|}{N} \tag{5}$$

$$\mathrm{MAPE} = \left(\frac{1}{N}\sum_{i=0}^{N-1} \left|\frac{y_i - \hat{y}_i}{y_i}\right|\right) \times 100\% \tag{6}$$

where y_i is the actual value, $\hat{y_i}$ is the predicted value, and N is the number of observations or dataset rows. The evaluation statistical tests were employed to evaluate the effectiveness of the forecasting method. The diversity of assessment metrics offers valuable insights into the robustness of the proposed model. When the RMSE is substantially higher than the MAE, the forecasting system exhibits significant deviation from the desired prediction, while the forecasting system exhibits a moderate level of departure from the desired prediction value when the RMSE is about equal to the MAE.

4.5 Results and discussion

The efficacy of the ARIMA, SARIMA, LSTM, and CNN-LSTM models in predicting weather temperature is quantified in the numerical values presented in Table 1. Based on the results, it is clear that the CNN-LSTM model is most accurate predictive model because it was able to get the lowest RMSE, MAE, and MAPE values. Therefore, the best prediction results were given by the CNN-LSTM model.

Table 1 Temperature prediction models achieved results

Models	RMSE	MAE	MAPE[%]
ARIMA	14.189	12.575	42.413
SARIMA	2.607	2.383	11.167
LSTM	1.611	1.244	4.738
CNN-LSTM	1.429	1.076	3.886

CNN-LSTM, which requires a large amount of historical data, has been found to be the most effective model for future temperature forecasting. Moreover, the results show that the LSTM model has been able to make predictions that are almost in line with those given by the CNN-LSTM model for that time period.

This demonstrates the effectiveness of the deep learning model when compared to other approaches when making predictions. So, to accurately predict temperature, a lot of temperature measurements need to be made over a long period of time. Moreover, the results illustrate that the predictions of the LSTM model for the same time period are more accurate than ARIMA and SARIMA models. It highlights how the deep learning model's predictions are more precise than those obtained by utilizing alternative methods.

5 Conclusion and future work

This study introduces a comprehensive big data framework for predicting weather temperature time series data through the utilization of statistical and deep learning models. The framework incorporates sophisticated open-source tools, such as Apache Spark and Kafka, for the purpose of data processing and analysis. Additionally, it utilizes the Hadoop Distributed File System (HDFS) for storage. The analysis and prediction tasks were conducted using ARIMA, SARIMA, LSTM, and a novel model based on CNN-LSTM.

In this study, a real-time series temperature dataset was employed for the purposes of models identification and validation. The performance of these models was subsequently evaluated and compared based on their accuracy, as measured by MAE, RMSE, and MAPE metrics.

The results of the study indicated that the deep learning models had a higher level of performance compared to the ARIMA and SARIMA models. More specifically, The CNN-LSTM-based deep learning model exhibits a higher degree of proximity to the real value when compared to the LSTM-based model. The research outlined in this paper emphasizes the advantages of utilizing deep learning algorithms and methodologies for the prediction of time series data.

In the future, deep learning can be applied to several prediction issues beyond the scope of this discussion. Thus, the authors plan to investigate the advancements attained through deep learning by applying these techniques to some other problems and datasets with various numbers of features. Furthermore, the authors are planning to apply the framework to additional IoT datasets with complex structures and different formats.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Data availability Open-Source datasets have been used in this study.

Declarations

Conflict of interest The authors have no conflict of interest to declare.

References

1. Li S, Da XuL, Zhao S (2014) The internet of things: a survey. Inf Syst Front 17:243–259. https://doi.org/10.1007/s10796-014-9492-7
2. Jensen SK, Pedersen TB, Thomsen C (2017) Time series management systems: a survey. IEEE Trans Knowl Data Eng 29:2581–2600. https://doi.org/10.1109/tkde.2017.2740932
3. Wang C, Huang X, Qiao J, et al (2020) Apache IoTDB: time-series database for internet of things. Proc VLDB Endow 13:2901–2904. https://doi.org/10.14778/3415478.3415504
4. Ghaderpour E, Pagiatakis SD, Hassan QK (2021) A survey on change detection and time series analysis with applications. Appl Sci 11:6141. https://doi.org/10.3390/app11136141
5. Ninagawa C (2022) LSTM AI Modeling. In: AI Time Series Control System Modelling. Springer Nature Singapore, pp 67–90
6. Raicharoen T, Lursinsap C, Sanguanbhokai P (2003) Application of critical support vector machine to time series prediction. In: Proceedings of the 2003 International Symposium on Circuits and Systems, ISCAS '03. IEEE
7. Ghosh N, Anderson OD (1976) Time series analysis and forecasting (the box-Jenkins approach). Oper Res Q 27:644. https://doi.org/10.2307/3009061
8. Petropoulos F, Apiletti D, Assimakopoulos V et al (2022) Forecasting: theory and practice. Int J Forecast 38:705–871. https://doi.org/10.1016/j.ijforecast.2021.11.001
9. Suresh KK, Krishna Priya SR (2011) Forecasting sugarcane yield of tamilnadu using ARIMA models. Sugar Tech 13:23–26. https://doi.org/10.1007/s12355-011-0071-7
10. Chen T, Shang C, Yang J et al (2020) A new approach for transformation-based fuzzy rule interpolation. IEEE Trans Fuzzy Syst 28:3330–3344. https://doi.org/10.1109/tfuzz.2019.2949767
11. TOKSARI MD, (2022) A hybrid algorithm for forecasting transportation energy demand of turkey. SSRN Electron J. https://doi.org/10.2139/ssrn.4009459
12. ArunKumar KE, Kalaga DV, Sai Kumar CM et al (2021) Forecasting the dynamics of cumulative COVID-19 cases (confirmed, recovered and deaths) for top-16 countries using statistical machine learning models: Auto-Regressive Integrated Moving Average (ARIMA) and Seasonal Auto-Regressive Integrated Moving Average (SARIMA). Appl Soft Comput 103:107161. https://doi.org/10.1016/j.asoc.2021.107161
13. Qiu H, Zhao H, Xiang H et al (2021) Forecasting the incidence of mumps in Chongqing based on a SARIMA model. BMC Public Health. https://doi.org/10.1186/s12889-021-10383-x
14. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
15. Torres JF, Hadjout D, Sebaa A et al (2021) Deep learning for time series forecasting: a survey. Big Data 9:3–21. https://doi.org/10.1089/big.2020.0159
16. (2020) No Title. Mach. Learn. Time Ser. Forecast. With Python® 137–165
17. Lindemann B, Müller T, Vietz H et al (2021) A survey on long short-term memory networks for time series prediction. Procedia CIRP 99:650–655. https://doi.org/10.1016/j.procir.2021.03.088
18. Aksan F, Li Y, Suresh V, Janik P (2023) CNN-LSTM vs. LSTM-CNN to predict power flow direction: a case study of the high-voltage subnet of northeast Germany. Sensors 23:901. https://doi.org/10.3390/s23020901
19. V RT, Gouda KC, Kumar SS (2022) Novel approach for spatiotemporal weather data analysis. Int J Adv Comput Sci Appl https://doi.org/10.14569/ijacsa.2022.0130743
20. Chen P, Niu A, Liu D et al (2018) Time series forecasting of temperatures using SARIMA: an example from Nanjing. IOP Conf Ser Mater Sci Eng 394:52024. https://doi.org/10.1088/1757-899x/394/5/052024
21. Srivastava A, S A (2022) Weather prediction using LSTM neural networks. In: 2022 IEEE 7th International conference for Convergence in Technology (I2CT). IEEE
22. Salman AG, Heryadi Y, Abdurahman E, Suparta W (2018) Weather forecasting using merged long short-term memory model (LSTM) and autoregressive integrated moving average (ARIMA) model. J Comput Sci 14:930–938. https://doi.org/10.3844/jcssp.2018.930.938
23. Agga A, Abbou A, Labbadi M et al (2022) CNN-LSTM: an efficient hybrid deep learning architecture for predicting short-term photovoltaic power production. Electr Power Syst Res 208:107908. https://doi.org/10.1016/j.epsr.2022.107908
24. Han C, Park H, Kim Y, Gim G (2023) Hybrid CNN-LSTM based time series data prediction model study. In: Big Data, Cloud Computing, and Data Science Engineering. Springer International Publishing, pp. 43–54
25. Raksha S, Graceline JS, Anbarasi J, et al (2021) Weather forecasting framework for time series data using intelligent learning models. In: 2021 5th International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT). IEEE
26. Hao J, Jinming C, Yajuan G (2018) Data-driven lean management for distribution network. In: 2018 China International Conference on Electricity Distribution (CICED). IEEE
27. Warren J. & MN (2015) Big data: principles and best practices of scalable realtime data systems. Simon and Schuster
28. spark^{TM}—unified engine for large-scale data analytics. URL https://spark.apache.org/ A No Title
29. Vyas S, Tyagi RK, Jain C, Sahu S (2022) Performance evaluation of apache kafka—a modern platform for real time data streaming.

In: 2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM). IEEE
30. From https://kafka.apache.org/ AK (n. d.). R
31. From https://hadoop.apache.org/ AHR
32. Chai T, Draxler RR (2014) Root mean square error (RMSE) or mean absolute error (MAE)?—Arguments against avoiding RMSE in the literature. Geosci Model Dev 7:1247–1250. https://doi.org/10.5194/gmd-7-1247-2014
33. Gneiting T, Raftery AE (2007) Strictly proper scoring rules, prediction, and estimation. J Am Stat Assoc 102:359–378. https://doi.org/10.1198/016214506000001437
34. Jenkins BGGM (1976) TSAF, control, Rev. Holden-Day. No Title, San Francisco
35. Guerard JB (2012) An Introduction to Time Series Modeling and Forecasting. Introduction to financial forecasting in investment analysis. Springer, New York, pp 47–72
36. Castle JL, Doornik JA, Hendry DF (2021) Forecasting principles from experience with forecasting competitions. Forecasting 3:138–165. https://doi.org/10.3390/forecast3010010
37. pmdarima: ARIMA estimators for Python—pmdarima 2.0.3 documentation. (n.d.). Retrieved from https://alkaline-ml.com/pmdarima/index.html
38. Li G, Wang Y (2013) Automatic ARIMA modeling-based data aggregation scheme in wireless sensor networks. EURASIP J Wirel Commun Netw 2013 https://doi.org/10.1186/1687-1499-2013-85
39. Janacek G (2010) Time series analysis forecasting and control. J Time Ser Anal 31:303. https://doi.org/10.1111/j.1467-9892.2009.00643.x
40. Patowary AN (2017) monthly temperature prediction based on arima model: a case study in Dibrugarh station of Assam, India. Int J Adv Res Comput Sci 8:292–298. https://doi.org/10.26483/ijarcs.v8i8.4590
41. Dickey DA, Fuller WA (1979) Distribution of the estimators for autoregressive time series with a unit root. J Am Stat Assoc 74:427. https://doi.org/10.2307/2286348
42. statsmodels 0.14.0. (n.d.). Retrieved from https://www.statsmodels.org/stable/index.html No Title
43. Meenakshi D, Shanavas ARM (2022) Novel Shared Input Based LSTM for Semantic Similarity Prediction. J Adv Inf Technol https://doi.org/10.12720/jait.13.4.387-392
44. Albeladi K, Zafar B, Mueen A (2023) Time Series Forecasting using LSTM and ARIMA. Int J Adv Comput Sci Appl https://doi.org/10.14569/ijacsa.2023.0140133
45. Verma P, Chafe C (2021) A generative model for raw audio using transformer architectures. In: 2021 24th International Conference on Digital Audio Effects (DAFx). IEEE
46. Liu P (2022) Time Series Forecasting Based on ARIMA and LSTM. In: Advances in Economics, Business and Management Research. Atlantis Press
47. Palangi H, Deng L, Shen Y et al (2016) Deep sentence embedding using long short-term memory networks: analysis and application to information retrieval. IEEE/ACM Trans Audio, Speech, Lang Process 24:694–707. https://doi.org/10.1109/taslp.2016.2520371
48. Palangi H, Ward R, Deng L (2016) Distributed compressive sensing: a deep learning approach. IEEE Trans Signal Process 64:4504–4518. https://doi.org/10.1109/tsp.2016.2557301
49. 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:e0180944. https://doi.org/10.1371/journal.pone.0180944
50. Keras: Deep Learning for humans. (n.d.). Retrieved from https://keras.io/
51. Lee H, Song J (2019) Introduction to convolutional neural network using Keras; an understanding from a statistician. Commun Stat Appl Methods 26:591–610. https://doi.org/10.29220/csam.2019.26.6.591
52. Zhang Z (2018) Improved adam optimizer for deep neural networks. In: 2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS). IEEE
53. Douglass MJJ (2020) Book Review: Hands-on Machine Learning with Scikit-Learn, Keras, and Tensorflow, 2nd edition by Aurélien Géron: O' Reilly Media, Phys Eng Sci Med 43:1135–1136. https://doi.org/10.1007/s13246-020-00913-z
54. Yamashita R, Nishio M, Do RKG, Togashi K (2018) Convolutional neural networks: an overview and application in radiology. Insights Imaging 9:611–629. https://doi.org/10.1007/s13244-018-0639-9
55. Sumanthvrao. "Daily Climate Time Series Data." Kaggle 23 Aug. 2019 www. kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data
56. Chen L, Lai X (2011) Comparison between ARIMA and ANN models used in short-term wind speed forecasting. In: 2011 Asia-Pacific Power and Energy Engineering Conference. IEEE
57. Cleveland RB, Cleveland WS (1990) STL: a seasonal-trend decomposition procedure based on loess. J Offic Statist 6:3–33