ISSN: 3064-9846
Research Article
Journal of Agricultural, Earth and Environmental Sciences

LSTCN Approach For Self-Adaptive Real-Time Weather Forecasting

MBA TENE Salomon[1]* and KAMLA Vivient Corneille[2]

[1]Department of Mathematics and Computer Science, Faculty of Sciences, University of Ngaoundéré, Cameroon
[2]Department of Mathematics and Computer Science, ENSAI, University of Ngaoundéré, Cameroon

*Corresponding author: MBA TENE Salomon, Department of Mathematics and Computer Science, Faculty of Sciences, University of Ngaoundéré, Cameroon.

Submitted: 09 December 2025 **Accepted:** 15 December 2025 **Published:** 22 December 2025

https://doi.org/10.63620/MKJAEES.2025.1102

Citation: *Mba Tene, S., & Kamla, V. C. (2025). LSTCN Approach For Self-Adaptive Real-Time Weather Forecasting. J of Agri Earth & Environmental Sciences, 4(6), 01-11.*

Abstract
Weather observations are recorded using sensor networks in the form of data streams. This data is most often used by predictive models based on machine learning to make weather forecasts. However, weather conditions can vary (or change) gradually or abruptly due to the chaotic nature of the climate. These characteristics mean that conventional machine learning-based weather forecasting models deployed in a real-world environment are inadequate for accurately modeling the dynamics of weather conditions due to their static parameters. This highlights the need for other approaches capable of learning beyond the production phase, which will allow their parameters to be kept up to date in real time in response to changing weather conditions. The objective of our study is first to evaluate the performance of online learning approaches (LSTCN and ARIMA-Online) compared to conventional ML methods (LSTM, CNN, LSTM+CNN), and then to evaluate and compare the performance between these online learning approaches. This evaluation and comparison study will be carried out on univariate and multivariate time series with short- and long-term horizons. The metrics used are MAE, learning time, and testing time. According to the results of our study, the LSTCN model is the most effective compared to conventional machine learning methods in terms of training time, testing time, and accuracy on short- or long-term univariate or multivariate time series. Similarly, this model outperforms the ARIMA OGD (Online Gradient Descent) model in terms of accuracy and testing time. However, its training time is slightly longer than that of the ARIMA OGD model. So, LSTCN model is better suited for real-time weather forecasting

Keywords: Weather Forecasting, Time Series, Machine Learning, Online Learning Algorithm, LSTCN, Data Stream

Introduction

Predicting the future is one of humanity's age-old dreams [1]. Science therefore aims to predict, but also to understand, the phenomenon observed, using an explanatory model [1–3] . The phenomenon chosen for our study is meteorology. Meteorology is the interdisciplinary science of atmospheric physics, which studies weather conditions, the atmospheric environment, the phenomena produced, and the laws that govern it [4]. The process of meteorological measurement is directly linked to the establishment of a meteorological station [4–7]. In addition, a meteorological station is a set of sensors that record and provide physical measurements and meteorological parameters related to climate variations in a locality [4, 5, 7]. These collected meteorological data will be used for meteorological forecasting. Meteorological forecasting is a vital issue in the field of science around the globe [8, 9].Thus, meteorological forecasting is the application of science and technology to predict atmospheric conditions for a specific location and a given period in the future [10–15]. Its purpose is to provide information to relevant individuals and organizations that can be used to reduce losses and improve societal benefits such as property protection, public health and safety, economic prosperity, and quality of life [15]. This forecasting involves data collection, data processing, and data analysis [16, 22]. The data flow generated by the meteorological station constitutes time series. A time series is a sequence of observations recorded at regular intervals [17–19]. Depending on the frequency of observations, a time series can generally be hourly, daily, weekly, monthly, quarterly, or annual. But, the decline in the performance of meteorological forecasting models can have a severe impact on both the economy and the environment. As a result, several models have been proposed over the centuries to solve the problem of predictive model performance.

First, numerical weather prediction (NWP) physical models use complex mathematical equations to obtain forecasts based on current weather conditions [22]. These models have long been used to predict events, but they do not allow for short-term predictions [23] and lack accuracy due to the chaotic nature of the atmosphere [23,24]. In addition, the complexity of these models poses significant challenges in their implementation [24]. To reduce the computing power of NWP systems, statistical and machine learning models have been proposed. Several authors have proposed statistical models for meteorological forecasting. Among these authors, we have, among others, (SHIVAM, et al., 2019) [26], who used ARMA, SARIMA, and ARMAX models to predict temperature and rainfall parameters. (GARIMA and BHAWANA, 2017) [27] used the ARIMA model to predict temperature data time series. However, these methods work in univariate time series and with relatively small data sets [19,21,28]. Similarly, they do not work on nonlinear relationships [19]. Furthermore, they are more limited in predicting a long time horizon or in processing multivariate time series [19,28]. In this case, machine learning tools, in this instance deep learning, would be the ideal tool for time series forecasting according to several authors, such as [18] and [19]. To provide more accurate short- and/or long-term weather forecasts, machine learning techniques can be used to understand and analyze weather patterns. (Azencott, 2019) [29] states that, and I quote: "At the intersection of statistics and computer science, machine learning is concerned with data modeling." Thus, machine learning for meteorological forecasting uses quantitative meteorological data to build models and attempts to improve model performance by learning from the dataset. According to (Sebastian and Gabriele, 2016) [30], machine learning technologies can provide intelligent models that are much simpler than physical models. It is with this in mind that (Amir et al., 2018) [23] state that ML (machine learning) models are highly accurate compared to physical and statistical models.

These machine learning models, in this case offline machine learning models (traditional or conventional machine learning), can numerically formulate the non-linearity of events, based entirely on historical data without the need for knowledge of physical processes [23]. everal offline machine learning methods have been used to develop weather forecasting models, namely LSTM [10,31], CNN [32], and hybrid models (CNN + LSTM) [33]. However, once these meteorological forecasting models are designed using offline machine learning algorithms (LSTM, CNN, LSTM+CNN) and historical data, and then deployed in a real world environment, their parameters remain static. As a result, these deployed models will encounter new environmental data dynamics that may differ from those used in the historical data. This new dynamic is due to the chaotic nature of the atmosphere, which causes gradual or sudden variations (or changes) in weather conditions. In the context of real-time meteorological forecasting, these characteristics render prediction models based on conventional ML methods (LSTM, CNN, LSTM+CNN) inadequate for accurately modeling environmental dynamics from online data streams, resulting in inaccuracy in predictive models. Similarly, existing conventional ML algorithms require model selection, which is time-consuming and not suited to the context of online learning [34]. Also, in conventional machine learning, these models are trained using the entire dataset at once. This process often requires a lot of computing time and cannot reflect changes in real time. Furthermore, these models based on offline machine learning are neither adaptive nor scalable in real time in the context of real-time meteorological forecasting, because once the models have been trained or formed, their parameters do not change.

In addition, some authors have used new approaches to improve prediction, namely algorithms capable of learning beyond the production phase, which will also enable them to be kept up to date at all times. These approaches are online machine learning methods such as ARIMA Online and LSTCN. (CHENGHAO et al., 2016) [21] proposed online learning algorithms (ARIMA Online Newton Step and ARIMA Online Gradient Descent) for the efficient estimation of ARIMA model parameters using its recursive formulation in an online learning framework. This online learning approach processes data observations arriving sequentially and updates the models simultaneously. This online ARIMA learning is an online optimization task without noise terms. Finally, in terms of memory cost, their algorithm is independent of the sample size, making it more scalable for processing real-time time series forecasting tasks. Their algorithms have been empirically compared to two online ARMA algorithms. According to their results obtained on synthetic and real data, their proposed algorithms are effective for time series prediction. However, this approach only uses univariate time eries. Furthermore, to our knowledge, this technique has not yet been used in the context of meteorological forecasting. There is another online learning approach that, in additionto using univariate time series, also processes multivariate series : the LSTCN approach. (ALEJANDRO et al., 2022) [20] used long-term and short-term cognitive networks LSTCN) in their article to predict wind turbine time series in online contexts. These recently introduced neural systems consist of a chain of short-term cognitive network blocks, each processing a block of time data.

An LSTCN model is defined as a collection of STCN blocks, each processing a specific time slice (or range) and transferring knowledge to subsequent STCN blocks in the form of weighted matrices. An incoming data block triggers new learning on the last STCN block using the stored knowledge that the network has learned during previous iterations. Then, the prior knowledge matrices are recalculated using an aggregation operator and stored for use as prior knowledge during reasoning. According to their results, the solution of their simulations outperforms traditional neural networks and reports significantly shorter training and testing times. However, to our knowledge, his technique has not yet been used in the context of meteorological forecasting. The forecasting methods used for the environment, particularly for weather, are based on physical, statistical, or machine learning models [20]. According to [20], although conventional machine learning models often achieve the highest performance compared to other models, their deployment in real-world applications shows inappropriate results in terms of accuracy. This drop in performance is due to the static parameters of these real-time forecasting models because once these conventional ML models are generated, their parameters no longer change, making it impossible to self-adapt or update to the non-linearity of the environment, i.e., to gradual and/or sudden changes in the climate. These variations or changes in meteorology or climate led to changes in the internal structure of meteorological data, i.e., new distributions of meteorological data. Conventional machine learning is also known as offline machine learning.

One of the main challenges of these models is to improve the accuracy of weather forecasting models. Real-time updates could ensure the accuracy of predictive models in the context of real-time meteorological forecasting. As indicated in [21], enabling forecasting systems to adapt to climate variability and change is a vital necessity. To overcome these limitations, online learning approaches have been proposed, such as the LSTCN (Long Short-Term Cognitive Network) method [20] and ARIMA-OGD [21]. To our knowledge, these approaches have not been applied in the context of real-time meteorological forecasting. The objective of our study is first to evaluate the performance of these online learning approaches against conventional ML methods (LSTM,CNN, LSTM+CNN), and then to evaluate and compare the performance between these online learning approaches (LSTCN and ARIMA-Online). This evaluation and comparison study will be carried out on univariate and multivariate time series and with short- and long-term horizons.

The remainder of this paper is organized as follows. Section 2 presents the materials and methods. Section 3 illustrates the results and analysis. Section 4 discusses the findings. Finally, Section 5 concludes the paper.

Materials and Methods

Time Series

A time series is a sequence of meteorological observations recorded at regular time intervals [17–19]. Depending on the frequency of meteorological observations, a time series can generally be hourly, daily, weekly, monthly, quarterly, and annual [19]. Let x ϵ R be a meteorological variable observed on a discrete time scale over a period t ϵ {1, 2, . . . , T} where T ϵ N is the number of meteorological observations. Therefore, univariate time series is defined as a sequence of meteorological observations $\{x^{(t)}\}^{T}{}_{t=1}=\{x^{(1)}, x^{(2)}, ..., x^{(T)}\}$. Similarly, we can define a multivariate series as a sequence $\{X^{(t)}\}^{T}{}_{t=1}=\{X^{(1)}, X^{(2)}, ..., X^{(T)}\}$ of vectors of M weather variables, such as $X^t = \{x_1^{(t)}, x_2^{(t)}, ..., x_M^{(t)}\}$. A model F is used to predict the next time steps in advance, denoted as L, such that L < T

Meteorological Forecasting Models

We will use two sets of models for our meteorological forecasting simulations of our meteorological data. These are the models based on conventional Machine Learning methods (LSTM, CNN, CNN+LSTM) and the models based on online learning methods (ARIMA-OGD and LSTCN). First of all, we will conduct a performance comparison between these two sets of methods, and then conduct a performance comparison between the online learning models.

Conventional Machine Learning

LSTM

The input gate, output gate, and forgetting gate are the three gates used by the LSTM model to modify previously stored data. Which data will be removed from the cell state is determined by the forget gate. The amount of new data that is added to the cell state is controlled by the input gate. The value that is invited to be output by the output gate is determined state of the cell. To safeguard and regulate data, the LSTM integrates the architecture of three gates. The cell's condition is shown by the upper horizontal line in Figure 1. Allow communications to pass through three gates selectively. Selecting which messages make it through the cell, entering the input gate, determining how many more messages to add to the cell state, and selecting the output message through the output gate are all done using the forget gate.

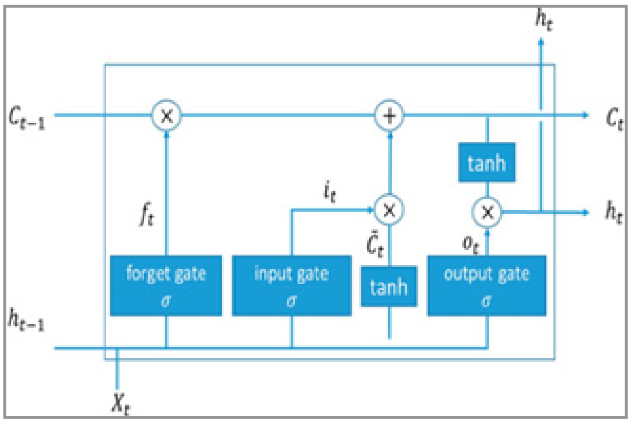


Figure 1: Basic Structure of a LSTM Model [31].

In Figure 1, the cell state and output value for the current moment are denoted by Ct and ht, while the cell state and output value for the prior moment are denoted by Ct−1 and ht−1. The activation vectors for forget, input/update, output, and cell input gates are denoted by the letters ft, it, ot, and C˜t, respectively. Let b be a bias vector parameter and W be a weight matrix that must be learned during training. Let σg and σc represent the hyperbolic tangent (Tanh) function and the sigmoid function, respectively. Using a sigmoid layer known as the forget gate layer, the initial step is to determine which data from the cell state should be deleted. For every number in the cell state Ct−1, it produces a number between 0 and 1 after examining ht−1 and the input vector xt. Keep in mind that a 1 means "completely keep this," while a 0 means "completely get rid of this." Consequently, the activation vector of the forget gate is provided by

$$f_t = \sigma_g(W_f \cdot [h_{t-1}, x_t] + b_i) \quad (1)$$

Selecting the new data to be stored in the cell state is the next stage. An update to the state is created by applying the Tanh layer and the input gate layer.

$$i_t = \sigma_g(W_i \cdot [h_{t-1}, x_t] + b_f) \quad (2)$$

$$C^{\sim} = \sigma_c(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3)$$

Then, the new cell state Ct is updated by

$$C_t = f_t * C_{t-1} + i_t * Ct \quad (4)$$

Lastly, we must determine what to output based on the cell condition. We start by running a sigmoid layer, which determines which cell states are used for the output. Next, we multiply the cell state by the sigmoid gate's output, which produces

$$o_t = \sigma_g(W_O \cdot [h_{t-1}, x_t] + b_O) \quad (5)$$

$$h_t = o_t * \sigma_c(C_t) \quad (6)$$

we used the same LSTM model configuration parameters as (Huang et al., 2020)[6]. For model training, TensorFlow and Keras are combined with the LSTM model architecture. For model training, TensorFlow and Keras are combined with the LSTM model architecture. The lookback LSTM parameter is set at 5. The network is trained using the Adam optimization technique. Regarding the last layer of our forecasting model, to predict one meteorological parameter, the unit parameter takes the value 1 while to predict five meteorological parameters, we have assigned the value 5 to the unit parameter. The Mean Absolute Error (MAE) is used to calculate the loss value. Tanh and scaled exponential linear units (Selu) functions are used in the activation functions. Table 1 summarizes each layer's units and activation functions.

Table 1: LSTM: Units and Activation Functions of Each Layer.

Layer	Units	Activation Function
LSTM	50	tanh
Time Distributed Dense	30	
LSTM	30	tanh
Dense	15	selu
Dense	1 or 5	selu

CNN

A deep learning architecture that utilizes the convolution process is called a convolutional neural network. The network can produce a smaller set of features thanks to the convolution procedure. This process takes place in a CNN between the inputs and a kernel. Putting a matrix on top of the feature matrix is all that constitutes the kernel. The dot product between the kernel and the features is obtained by sliding the kernel along the time axis in figure 2. As a result, a smaller collection of features is obtained, regularization is achieved, and aberrant values are filtered

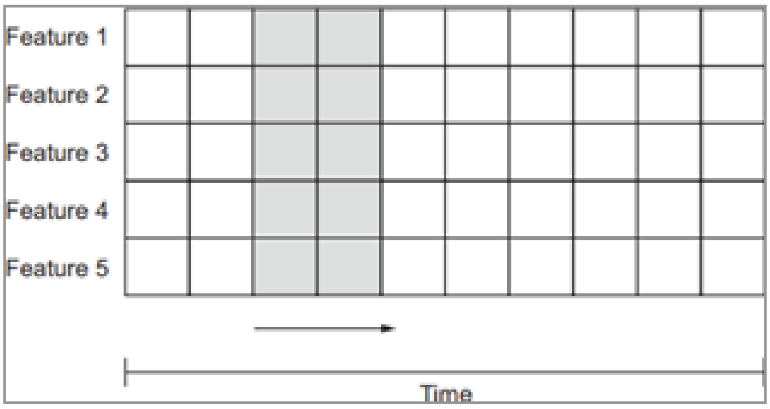


Figure 2: Visualizing the Kernel and the Feature Map [19].

The light gray matrix that is put over the feature map is called the kernel. The length is the time axis, and each row is a feature of the dataset. A kernel, which is also trained during model fitting, is used for the convolution. The number of steps the kernel shifts at each convolutional step is determined by its stride. The only convolution utilized in time series forecasting is 1D.We stacked many layers using Keras' sequential model. Since we are dealing with time series and the kernel only travels in the time dimension, we then used the Conv1D layer. The "filters" parameter simply indicates the number of neurons in the convolutional layer and is comparable to the "units" parameter of the dense layer. "kernel_size" is set to our kernel's width. The remaining dimensions don't need to be specified because Keras will adapt the form to the inputs on its own. We sent the CNN's output to a dense layer. As a result, the convolutional stage filtered a smaller collection of features before the model learned on them. Regarding the last layer of our forecasting model, to predict one meteorological parameter, the unit parameter takes the value 1 while to predict five meteorological parameters, we have assigned the value 5 to the unit parameter. Table 2 illustrates the parameters of the CNN model

```
Algorithm 1 ARIMA-OGD(k,d,q)
Input: parameter k, d, q; learning rate $\eta$
Set $m = \log_{\lambda_{max}}((TLM_{maxq})^{-1})$
for T=1 to T-1 do
    predict $X^{\tilde{}}t(\gamma^t) = \sum^{k+m} \gamma_i \nabla^d X_{t-i} + \sum^{d-1} \nabla^i X_{t-1}$;
receive $X_t$ and incur loss $\ell^m(\gamma^t)$;
Let $\nabla_t = \nabla \ell^m(\gamma^t)$;
Set $\gamma^{t+1} \leftarrow \prod_K(\gamma^t - \frac{1}{}) \nabla_t$;
end for
```

Table 2: The Parameters of the CNN Model.

Layer	Units
Filters	32
kernel_size	3
Activation function	relu
Dense	32
Dense	1 or 5

LSTM-CNN

The hybrid CNN-LSTM model combines the advantages of Long Short-Term Memory (LSTM) networks for capturing temporal dependencies and Convolutional Neural Networks (CNNs) for extracting spatial features. This hybrid model allows us to filter our input sequence via the CNN before sending it to an LSTM to generate the forecast data. Regarding the last layer of our forecasting model, to predict one meteorological parameter, the unit parameter takes the value 1 while to predict five meteorological parameters, we have assigned the value 5 to the unit parameter. Table 3 illustrates the parameters used in our hybrid model

Table 3: The Parameters of the CNN-LSTM Model.

Layer	Units
filters	32
kernel_size	3
Activation function	relu
LSTM	32
LSTM	32
Dense	1 or 5

Online learning methods

ARIMA OGD (Online Gradient Descent)

Algorithm 1 presents the ARIMA-OGD algorithm proposed by (Chenghao et al., 2016) [21] to optimize the coefficient vector γ^t (with $\gamma \in R^{m+k}$) using the OGD algorithm. With X_t denoting the meteorological observation at time t. We used the method (Chenghao et al., 2016) which approximates the original ARIMA(k, d, q) model to the ARIMA(k + m, d, 0) model (without noise term), where $m \in N$ is an appropriately chosen constant such that the new ARIMA model with a coefficient vector $\gamma \in R^{m+k}$ in (m + k) dimensions is efficient enough to approximate the initial prediction:

$$\tilde{X}_t(\gamma^t) = \sum_{i=1}^{k+m} \gamma_i \nabla^d X_{t-i} + \sum_{i=0}^{d-1} \nabla^i X_{t-1} \tag{7}$$

Where k represents the number of previous meteorological observations to be considered in the model equation Consider an online ARIMA iteration at time t, the learner makes a prediction $X^{\tilde{}}$, anthen the true X_t is revealed to him. As a result, the learner incurs a loss $\ell_t(X_t, \tilde{X}_t)$. Morformally, we can define the loss function as follows:

$$\ell_t^m(\gamma^t) = \ell_t(X_t, \tilde{X}_t(\gamma^t)) = \ell_t(X_t, \sum_{i=1}^{k+m} \gamma_i \nabla^d X_{t-i} + \sum_{i=0}^{d-1} \nabla^i X_{t-1}) \tag{8}$$

The objective of online ARIMA learning is to minimize the sum of losses over a number of rounds T

The projection $\prod K(y)$ refers to the Euclidean projection on K i.e K i.e $\prod_K(y) = \text{argmin}_{x \in K} \|y - 259\ x\|_2$

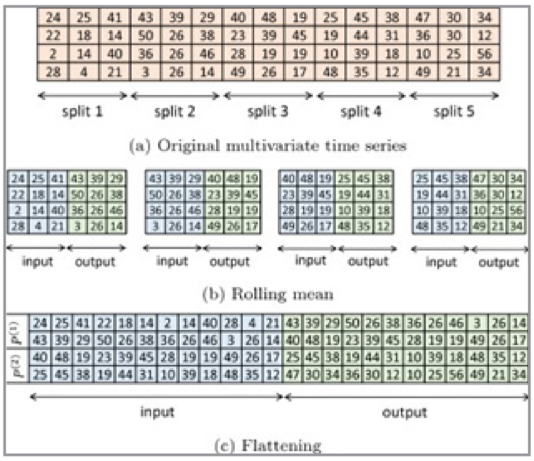
Long Short-Term Cognitive Network (LSTCN)

A model is used to predict the next steps L < T to come (or in advance). In this paper, we assume that the model is constructed as a sequence of neural blocks with local learning capabilities, each capable of capturing trends in the current time range (i.e., a piece of the time series) being processed regarding data preparation for online learning simulation, we suppose that $X \in R^{M \times T}$ is a set of meteorological data comprising a multivariate time series (Fig 3(a)) First, we must transform X into a set of Q tuples with the form (X^{t-R}, X^{t+L}), $t - R > 0$, $t + L \leq T$ where R represents the number of past steps that we will use to predict the next L steps forward (see Fig. 3(b)). In this manuscript, we assume that R = L for simplicity. Secondarily, each component in the tuple is flattened such that we obtain a Q(M(R + L)) matrix. Finally, we create a partition $P = P^{(1)}, P^{(2)}, ..., P^{(k)}, ..., P^{(K)}$ from the set of flattened tuples such as $P = (P_1^{\ k}, P_2^{\ k})$ is the kth temporal patch involving two pieces of data $P_1^{\ k}, P_1^{\ k} \in R^{C \times N}$ where

N = MR and C denotes the number of instances in this time patch.

Figure 3 shows an example of such a preprocessing method. First, the time series is divided into pieces of equal length, defined by the parameters L and R. Second, we use the resulting pieces to create a set of input-output pairs. Finally, we flatten these pairs to obtain tuples containing the network inputs and the corresponding expected outputs.

It should be noted that the forecasting model will have access to a time patch at each iteration, as is generally the case online. If the neural model is fed by several time steps, it will be able to make forecasts at several time steps for all variables describing the time series.



24	25	41	43	39	29	40	48	19	25	45	38	47	30	34
22	18	14	50	26	38	23	39	45	19	44	31	36	30	12
2	14	40	36	26	46	28	19	19	10	39	18	10	25	56
28	4	21	3	26	14	49	26	17	48	35	12	49	21	34

$P^{(1)}$	24	25	41	22	18	14	2	14	40	28	4	21	43	39	29	50	26	38	36	26	46	3	26	14
$P^{(1)}$	43	39	29	50	26	38	36	26	46	3	26	14	40	48	19	23	39	45	28	19	19	49	26	17
$P^{(2)}$	40	48	19	23	39	45	28	19	19	49	26	17	25	45	38	19	44	31	10	39	18	48	35	12
$P^{(2)}$	25	45	38	19	44	31	10	39	18	48	35	12	47	30	34	36	30	12	10	25	56	49	21	34

Figure 3: Data pre-processing using R = L = 3. (a) The original multivariate time series $X \in R^{M \times T}$, with rows as variables and columns as timestamps. (b) Selection of subsequences of the time series according to parameters R and L. (c) Each sub-sequence is flattened to obtain the temporal instances. In this example, the flattened dataset is divided into two time patches [21] .

In the configuration (parameterization) of machine learning, we consider a time series (regardless of the number of observed variables) as a sequence of time ranges of a certain length. Such a sequence refers to the partition $P = P^{(1)}, ..., P^{(k)}, ..., P^{(K)}$, with the data preparation steps described above.

An LSTCN model can be defined as a collection of STCN blocks, each processing aspecific time slice and transferring knowledge to the following STCN blocks in the form of weighted matrices. Figure 4 shows the recurrent pipeline of an LSTCN comprising three STCN blocks to model a multivariate time series decomposed into three time ranges. It should be noted that learning takes place within each block to prevent the flow of information from disappearing as the network processes more time ranges.

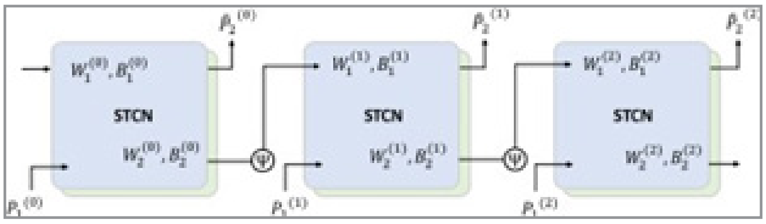


Figure 4: LSTCN architecture of three STCN blocks. The weights learned in the current block are transferred to the following STCN block as a prior knowledge matrix [21].

In addition, the estimated weights in the current STCN block are transferred to the next STCN block to perform the next reasoning process (see Figure 5). These weights will no longer be modified in subsequent learning processes, which preserves the knowledge acquired up to the current time range. This approach is therefore well-suited to online learning.

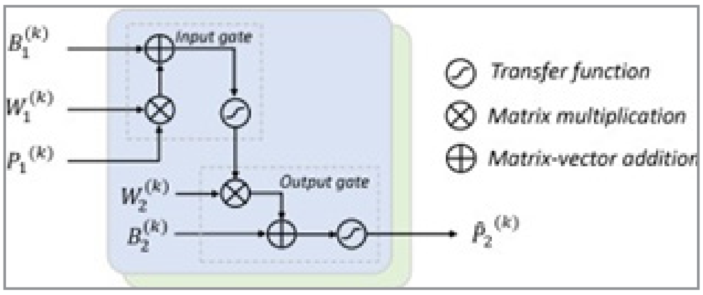


Figure 5: Reasoning Within an STCN Block.

Firstly, the current time patch is mixed with the prior knowledge matrices $W_1^{(k)}$ and $B_1^{(k)}$. This operation produces a temporal state matrix $H^{(k)}$. Secondly, we operate the $H^{(k)}$ matrix with the matrices $W_2^{(k)}$ and $B_2^{(k)}$. The result of such an operation will be an approximation of the expected output $P_2^{(k)}$ [21]. Reasoning within an STCN block involves two gates: the input gate and the output gate. The input gate uses prior matrix knowledge $W_t^{(k)} \in R^{N \times N}$ with input data $P_t^{(k)} \in$

$R^{C\times N}$ and the prior bias matrix $_2B_1^{(k)} \in R^{1\times N}$ representing the bias weights. The two matrices $W_1^{(k)}$ and $B_1^{(k)}$ are transferred from the previous block and remain locked for the duration of the learning phase to be executed in this STCN block. The result of the front door is a temporary state $H^{(k)} \in R^{C\times N}$ which represents the output that the block would have produced $P^{(k)}$ given, if the block had not been adjusted to the expected output of block $P^{(k)}$. Such an adaptation is made in the exit gate where the temporal state operates ith the matrices $W_2^{(k)} \in R^{N\times N}$ and $B_2^{(k)} \in R^{1\times N}$ which contain learnable weights.

Figure 5 illustrates the reasoning process within the kth block. Reasoning within an STCN block. Firstly, the current time patch is mixed with the prior knowledge matrices $W_1^{(k)}$ and $B_1^{(k)}$. This operation produces a temporal state matrix $H^{(k)}$. Secondly, we operate the $H^{(k)}$ matrix with the matrices $W_2^{(k)}$ and $B_2^{(k)}$. The result of such an operation will be an approximation of the expected output $P_2^{(k)}$.

Equations (9) and (10) show the short-term reasoning process of this model in the kth iteration.

$$\hat{P}_2^{(k)} = f(H^{(k)}W_2^{(k)} \bigoplus B_2^{(k)}) \tag{9}$$

and

$$H^{(k)} = f(P_1^{(k)}W_1^{(k)} \bigoplus B_1^{(k)}) \tag{10}$$

Where $f(x) = \frac{1}{1+e^{(-x)}}$ whereas $\hat{P}_2^{(k)}$ is an approximation of the expected block output.

In these equations, the operator performs a matrix-vector addition by operating each row of a matrix with a vector, provided that the matrix and vector have the same number of columns. Note that we have assumed that the values to be predicted lie in the interval [0, 1]. As indicated, the LSTCN model consists of a sequential collection of STCN blocks. In this neural system, knowledge from one block is passed on to the next using an aggregation procedure (see Figure 4). This aggregation operates on the knowledge acquired (or learned) in the previous block (i.e., the matrix W2(k−1)). In this paper, we use the

following nonlinear operator in all our simulations:

$$W_1^{(k)} = \Psi(W_2^{(k-1)}), k-1 \geq 0 \tag{11}$$

$$B_1^{(k)} = \Psi(B_2^{(k-1)}), k-1 \geq 0 \tag{12}$$

Such as $\Psi(x) = \tanh(x)$. However, we can design operators that combine the knowledge $W_1^{(k-1)}$ $W_2^{(k-1)}$

Categories of Learning Models For Meteorological Forecasting

These two sets of machine learning models consist of two categories of learning models for meteorological forecasting, mainly SISO (Single-Input Single-Output) and MIMO (Multi Input Multi-Output) [24]. The SISO is used in univariate time series, while the MIMO is used in multivariate time series. Regarding the SISO model, only one variable is introduced into the learning model to generate only one output. And the MIMO model, on the other hand, allows for the introduction of N variables into the learning model to generate N outputs. Our simulations will be conducted on univariate and multivariate time series over short-term and long-term horizons. Following the same logic as [24], in our study, the short-term horizon is assigned to 1 hour and the long-term horizon is assigned from 2 hours onward.

Meteorological Data

Our historical meteorological dataset consists of meteorological data from the city of Ngaoundéré-Cameroon, spanning from 01 − 01 − 2011 to 31 − 12 − 2020, covering a period of 10 years. The frequency of observations is hourly in our dataset. Our dataset contains the following five meteorological parameters: temperature, precipitation, atmospheric pressure, air humidity, dew point. The total number of observations is 87, 671 meteorological observations (or records). These data were standardized using Zscore normalization to ensure a uniform scale across features. This process is defined as (13):

$$X_{scaled} = \frac{X - \mu}{\sigma} \tag{13}$$

For the simulation of SISO models, we will use only the Temperature parameter. Regarding the simulation of MIMO models, we will use five meteorological parameters. Subsequently, our dataset will be divided into 80% for training data and 20% for test data. The metrics used will be MAE (Mean Absolute Error), training time, and testing time. The mathematical equation of the MAE is as follows :

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|X_i - \tilde{X}_i| \tag{14}$$

where n is the number of meteorological observations, Xi denoting the meteorological observation and X˜ is a expected meteorlogical value.

We will conduct an evaluation and performance study between the models to determine the most performing model in the case of univariate series on one hand and in The case of multivariate series on the other hand. this study will be conducted based on short-term and long-term horizons.

Tools

The programming language used to code all these methods is Python, version 3.6. Our simulations will be performed on the Jupyter Notebook IDE. Our PC (Personal Computer) on which our simulations will be performed has the following characteristics: Brand HP, operating system (Windows 10 Professional 64-bit), memory (4GB), hard drive (500GB), processor type (AMD), and processor speed (1.80GHz).

Results and Analysis

The results and our analyzes will be done respectively on univariate time series (for SISO models) and multivariate time series (for MIMO models). Let's start with the results and our analyzes on univariate time series. It should be noted that our results on univariate time series are based solely on the temperature parameter. Table 4 presents the metrics of our different models in the case of univariate series for a prediction horizon of 1 hour.

Table 4: Metrics on Univariate Series With a Prediction Horizon of 1 hour (Short-Term Horizon).

SISO : 1 hour				
Methods	**Error(MAE)**	**Time**		**Parameter**
	Test	Training	Test	
LSTM	$8972 * 10^{-6}$	1960s	10.7s	Temperature
CNN	$9860 * 10^{-6}$	560s	6.5s	
LSTM+CNN	$6909 * 10^{-6}$	3196s	16.2s	
LSTCN	$5020 * 10^{-6}$	20.28s	0.053s	
ARIMA OGD	$7828 * 10^{-6}$	5.85s	1.60s	

Table 4 shows that the LSTCN model performs best in terms of mean absolute error ($5020 * 10^{-6}$). It also has the best execution time for test data (0.053 seconds). In addition, it is considered the second-best model in terms of training time (20.28 seconds). In contrast, the ARIMA OGD model performs the worst in terms of mean absolute error ($7828 * 10^{-6}$). However, it has the fastest training time (5.85 seconds). Table 5 and Table 6 present the metrics of our different models for long-term horizons, namely: a 2-hour horizon for Table 5 and a 12-hour horizon for Table 6.

Table 5: Metrics on Univariate Series With a 2-Hour Prediction Horizon (Long-Term Horizon).

SISO : 2 hour				
Methods	**Error(MAE)**	**Time**		**Parameter**
	Test	Training	Test	
LSTM	$19478 * 10^{-6}$	1534s	10.7s	Temperature
CNN	$20325 * 10^{-6}$	688s	5.8s	
LSTM+CNN	$12294 * 10^{-6}$	3878s	66s	
LSTCN	$11499 * 10^{-6}$	8.3s	0.06s	
ARIMA OGD	$78172 * 10^{-6}$	5.15s	1.15s	

Table 6: Metrics on Univariate Series With a 12-Hour Prediction Horizon (Long-Term Horizon).

SISO : 2 hour				
Methods	**Error(MAE)**	**Time**		**Parameter**
	Test	Training	Test	
LSTM	$46739 * 10^{-6}$	778s	10.9s	Temperature
CNN	$71043 * 10^{-6}$	253s	13.9s	
LSTM+CNN	$36029 * 10^{-6}$	2061s	47.6s	
LSTCN	$33464 * 10^{-6}$	111s	0.33s	
ARIMA OGD	1.19696	5.12s	1.10s	

Regarding tables 5 and 6, for the long-term horizon, the LSTCN model performs best in terms of mean absolute error, with $11499 * 10^{-6}$ for the 2-hour prediction horizon and $33464 * 10^{-6}$ for the 12-hour prediction horizon. It also has the shortest testing time for both prediction horizons, namely 2 hours (0.06 seconds) and 12 hours (0.33 seconds). In addition, this model is considered to be the second best model with a reduced training time of 8.3 seconds. Furthermore, the ARIMA OGD model is the least performant model in terms of mean absolute error in the long-term horizons. However, this model is considered to be the best model in terms of training time for the 2-hour (5.15 seconds) and 12-hour 385 (5.12 seconds) prediction horizons. After analyzing our various SISO model results, we will now analyze the various MIMO model results.

With regard to the tables below, Table 7 presents the evaluation metrics for our different models for a short-term prediction horizon (1 hour), while Table 8 and Table 9 illustrate the evaluation metrics for our different models for long-term prediction horizons, namely a 2-hour prediction horizon for Table 8 and a 12-hour prediction horizon for Table 9. Furthermore, in the tables below, the ARIMA OGD model is excluded because this model does not take multivariate series into account.

Table 7: Metrics on Multivariate Time Series (MIMO) With a Prediction Horizon of 1 Hour (Short-Term Horizon).

MIMO : 1 hour			
Methods	**Error(MAE)**	**Time**	
	Test	Training	Test
LSTM	$7042 * 10^{-6}$	2038s	18s

CNN	7707 ∗ 10−⁶	856s	15.1s
LSTM+CNN	6503 ∗ 10−⁶	3143s	12.7s
LSTCN	4420 ∗ 10−⁶	12.8s	0.022s

Table 8: Metrics on Multivariate Time Series (MIMO) With a Prediction Horizon of 2 Hours (Long-Term Horizon).

MIMO : 1 hour			
Methods	**Error(MAE)**	**Time**	
	Test	Training	Test
LSTM	12292 ∗ 10−⁶	3434s	12.8s
CNN	13739 ∗ 10−⁶	481s	7.26s
LSTM+CNN	12297 ∗ 10−⁶	1987s	13.7s
LSTCN	1060 ∗ 10−⁶	17.7s	0.038s

Table 9: Metrics on Multivariate Time Series (MIMO) With a Prediction Horizon of 12 Hours (Long-Term Horizon).

MIMO : 1 hour			
Methods	**Error(MAE)**	**Time**	
	Test	Training	Test
LSTM	32370 ∗ 10−⁶	1521s	11.5s
CNN	36289 ∗ 10−⁶	945s	10.2s
LSTM+CNN	31233 ∗ 10−⁶	1696s	13s
LSTCN	30083 ∗ 10−⁶	98.7s	0.23s

Regarding Table 7 presenting the metrics for the short-term horizon, Table 8 and Table 9 for the long-term horizon for MIMO models, the LSTCN model is the most performing model in terms of MAE and in terms of training and testing time.

Discussion

Our simulations show that online learning models have a faster training and testing time than conventional Machine Learning models on univariate and multivariate time series. Although previous studies have not conducted a comparative analysis between the LSTCN and ARIMA OGD models, and performed simulations on meteorological data, we conducted our study in this direction. According to our results, the forecast data generated by the LSTCN model is more accurate than the ARIMA OGD model on univariate time series for both short-term and long-term forecasts. Moreover, the training time of the LSTCN model is slightly longer than that of the ARIMA OGD model. The ARIMA OGD model is only restricted to univariate time series, which limits the simulation on multivariate time series. And yet, the LSTCN extends the simulation to multivariate time series. The LSTCN model is also the most performant model compared to conventional Machine Learning models, whether in terms of MAE, training time, or testing on univariate and multivariate time series for short-term and long-term forecasts. Regarding long-term predictions for univariate and multivariate series, it turns out that the accuracy of forecast data can degrade as the prediction horizon increases. Based on the results of our simulations, we were able to observe that LSTCN model are better suited for real-time weather forecasting.

Conclusions

In summary, weather forecasting is the application of science and technology to predict weather conditions for a specific location and the given period in the future. Thus, several forecasting methods used for the environment, particularly for weather, are based on physical, statistical, or machine learning models. although conventional machine learning models often achieve the highest performance compared to other models, their deployment in a dynamic environment shows a certain inadequacy, namely the chaotic nature of the atmosphere in the context of real-time weather forecasting. Updating conventional models with new meteorological information is often very costly. This update could ensure the accuracy of the predictive model. To address these limitations, online learning approaches have been proposed, such as the LSTCN method and ARIMA-OGD. These approaches were compared against conventional Machine Learning methods, namely LSTM, CNN, and CNN+LSTM. According to the results, the LSTCN model is the most effective compared to conventional machine learning methods in terms of training time, testing time, and accuracy on short- or long-term univariate or multivariate time series. Similarly, this model outperforms the ARIMA OGD (Online Gradient Descent) model in terms of accuracy and testing time. However, its training time is slightly longer than that of the ARIMA OGD model. LSTCN model is better suited for real-time weather forecasting.

Patents

Author Contributions

For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used “Conceptualization, X.X. and Y.Y.; methodology, X.X.; software, X.X.; validation, X.X., Y.Y. and Z.Z.; formal analysis, X.X.; investigation, X.X.; resources, X.X.; data curation, X.X.; writing—original draft preparation, X.X.; writing—review and editing, X.X.; visualization, X.X.; supervision, X.X.; project administration, X.X.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.”, please turn to the Credit taxonomy for the term explanation. Authorship must be limited to those who have contributed substantially to the work reported.

Funding
This research received no external funding.

Data Availability Statement
The original data presented in the study are openly available in NASA POWER at https://power.larc.nasa.gov/data-access-viewer/.

Conflicts of Interest
The authors declare no conflicts of interest.

References

1. Azencott, C.-A. (2019). Introduction au machine learning (2nd ed., pp. 1–240). Dunod.
2. Shi, X., Chen, Z., Wang, H., & Yeung, D.-Y. (2016). Convolutional LSTM network: A machine learning approach for precipitation nowcasting. Advances in Neural Information Processing Systems, 29, 802–810.
3. Mohamed, S. T. (2010). Application de l'apprentissage artificiel à la prévision des crues éclair (Doctoral dissertation). École Nationale Supérieure des Mines de Paris, Paris, France.
4. Pérez-Jiménez, A. P., Vargas-Fernández, C., Torres-Pérez, D., & Pérez-Mendoza, G. (2020). Diseño de una estación meteorológica automática para registrar las variables solar y eólica. Revista Arbitrada Interdisciplinaria KOINONIA, 5, 937–957. https://doi.org/10.35381/r.k.v5i2.107
5. Aris, M., Hanif, F., Muhammad, I. R., Rian, P. P., Jony, W. W., & Irfan, A. F. A. (2017). Design of real-time weather monitoring system based on mobile application using automatic weather station. In Proceedings of the ICACOMIT Conference (pp. 44–47). Jakarta, Indonesia. https://doi.org/978-1-5386-0510-3/17/
6. Zaid, K. H., Hadi, J. H., Moussa, R. A.-M., & Yaqeen, S. M. (2020). Low-cost smart weather station using Arduino and ZigBee. TELKOMNIKA, 18(1), 282–288. https://doi.org/10.12928/telkomnika.v18i1.12784
7. Chen, Y.-H., & Liu, S.-R. (2011). Design and realization of an automatic weather station at island. In Proceedings of SPIE – The International Society for Optical Engineering (Vol. 8205, pp. 1–6). https://doi.org/10.1117/12.906350
8. Garima, J., & Bhawna, M. (2016). A review on weather forecasting techniques. International Journal of Advanced Research in Computer and Communication Engineering, 5, 177–180. https://doi.org/10.17148/IJARCCE.2016.51237
9. Farhadi, M., Hosseini, M. K., Jafari, S. M., & Moradi, E. (2021). Big data analytics in weather forecasting: A systematic review. Archives of Computational Methods in Engineering, 28, 1–21. https://doi.org/10.1007/s11831-021-09616-4
10. Siddharth, S., Mayank, K., Ambuj, G., & Anil, K. M. (2019). Weather forecasting using machine learning techniques. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3350281
11. Ning, Y., Lewis, W., & Preeti, D. (2018). A weather prediction model with big data.
12. Rodriguez-Lopez, P. (2018). Application of machine learning techniques to weather forecasting (Doctoral dissertation). University of the Basque Country (UPV/EHU).
13. Wang, Z., & Ahmed, M. M. A. B. (2017). The weather forecast using data mining research based on cloud computing. Journal of Physics: Conference Series, 910, 012020. https://doi.org/10.1088/1742-6596/910/1/012020
14. Priya, S., & Radhamani, A. (2021). Weather prediction based on wireless sensor network and Internet of Things with analysis using hybrid SSOA with MA. Research Square. https://doi.org/10.21203/rs.3.rs-911875/v1
15. Deepak, D., & Chetan, W. (2022). A literature review on improvement of weather prediction by using machine learning. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.4140237
16. Shruti, D., Vibhakar, P., Rohit, M., & Ruchi, D. (2021). Machine learning for weather forecasting. In Machine learning for sustainable development (pp. 161–174). https://doi.org/10.1515/9783110702514-010
17. Selva, P. (2019). Analyse de séries chronologiques en Python: Un guide complet avec des exemples.
18. Pal, A., & PKS, P. (2017). Practical time series analysis. Packt Publishing.
19. Petrelli, M. (2022). Time series forecasting in Python. Manning Publications.
20. Martínez-Hernández, A., Nieto, G., Jasińska, A., Sánchez, Y., & Verleysen, K. (2022). Online learning of windmill time series using long short-term cognitive networks. Expert Systems with Applications, 205, 117721. https://doi.org/10.1016/j.eswa.2022.117721
21. Li, C., Hsu, S. C., Zhang, P., & Sun, J. (2016). Online ARIMA algorithms for time series prediction. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 30, pp. 1867–1873). https://doi.org/10.1609/aaai.v30i1.10257
22. Ismaila, O. (2022). Machine learning-based algorithms for weather forecasting. International Journal of Artificial Intelligence and Machine Learning, 2, 12–20. https://doi.org/10.51483/IJAIML.2.2.2022.12-20
23. Amir, M., Pinar, O., & Chau, K.-W. (2018). Flood prediction using machine learning models: Literature review. Water, 10, 1536. https://doi.org/10.3390/w10111536
24. Pradeep, H., Banerjee, A., Tofani, M., Pan, E., Ghaemi, M., Pisano, F., & Liu, Y. (2020). Temporal convolutional neural network for effective weather forecasting using time-series data from the local weather station. Soft Computing, 24, 1–12. https://doi.org/10.1007/s00500-020-04954-0
25. Bespalov, L., Katser, L., Kozitsin, V., & Makarov, E. (2019). Online ARIMA.
26. Shivam, B., Rajat, B., Ankit, S. C., Akshay, K. D., & Indu, C. (2019). Fuzzy logic-based crop yield prediction using temperature and rainfall parameters predicted through ARMA, SARIMA, and ARMAX models. In Proceedings of the 12th International Conference on Contemporary Computing (IC3). https://doi.org/10.1109/IC3.2019.8844901
27. Garima, J., & Bhawana, M. (2017). A study of time series models ARIMA and ETS. SSRN Electronic Journal, 4, 57–63. https://doi.org/10.2139/ssrn.2898968
28. Nieto, G., Gómez, I., Jasińska, A., & Sánchez, Y. (2021). Long short-term cognitive networks. Machine Learning with Applications. https://doi.org/10.48550/arXiv.2106.16233
29. Azencott, C.-A. (2019). Introduction au machine learning (2nd ed.). Dunod.
30. Scher, S., & Messori, G. (2016). Predicting weather forecast uncertainty with machine learning. Quarterly Journal of the Royal Meteorological Society, 144, 1–17. https://doi.org/10.1002/qj.3410

31. Huang, Z.-Q., Chen, Y.-C., & Wang, C.-Y. (2020). Real-time weather monitoring and prediction using city buses and machine learning. Sensors, 20, 5173. https://doi.org/10.3390/s20185173
32. Weyn, J. A., Durran, D. R., & Caruana, R. (2020). Improving data-driven global weather prediction using deep convolutional neural networks on a cubed sphere. Journal of Advances in Modeling Earth Systems, 12, 1–17. https://doi.org/10.1029/2020MS002109
33. Lim, S.-C., Huh, J.-H., Hong, S.-H., Park, C.-Y., & Kim, J.-C. (2022). Solar power forecasting CNN–LSTM hybrid model. Energies, 15, 8233. https://doi.org/10.3390/en15218233
34. Sun, W., Frossard, L. R., Sahin, F., & Aydin, S. (2021). Adaptive online learning for the autoregressive integrated moving average models. Mathematics, 9, 1523. https://doi.org/10.3390/math9131523