agronomy

MDPI

Article

Deep Learning Model Effectiveness in Forecasting Limited-Size Aboveground Vegetation Biomass Time Series: Kenyan Grasslands Case Study

Efrain Noa-Yarasca [1,*], Javier M. Osorio Leyton [1] and Jay P. Angerer [2]

[1] Texas A&M AgriLife Research, Blackland Research and Extension Center, Temple, TX 76502, USA; javier.osorio@ag.tamu.edu
[2] USDA Agricultural Research Service, Fort Keogh Livestock and Range Research Laboratory, Miles City, MT 59301, USA; jay.angerer@usda.gov
* Correspondence: efrain.noa-yarasca@ag.tamu.edu

Abstract: Timely forecasting of aboveground vegetation biomass is crucial for effective management and ensuring food security. However, research on predicting aboveground biomass remains scarce. Artificial intelligence (AI) methods could bridge this research gap and provide early warning to planners and stakeholders. This study evaluates the effectiveness of deep learning (DL) algorithms in predicting aboveground vegetation biomass with limited-size data. It employs an iterative forecasting procedure for four target horizons, comparing the performance of DL models—multi-layer perceptron (MLP), long short-term memory (LSTM), gated recurrent unit (GRU), convolutional neural network (CNN), and CNN-LSTM—against the traditional seasonal autoregressive integrated moving average (SARIMA) model, serving as a benchmark. Five limited-size vegetation biomass time series from Kenyan grasslands with values at 15-day intervals over a 20-year period were chosen for this purpose. Comparing the outcomes of these models revealed significant differences ($p < 0.05$); however, none of the models proved superior among the five time series and the four horizons evaluated. The SARIMA, CNN, and CNN-LSTM models performed best, with the statistical model slightly outperforming the other two. Additionally, the accuracy of all five models varied significantly according to the prediction horizon ($p < 0.05$). As expected, the accuracy of the models decreased as the prediction horizon increased, although this relationship was not strictly monotonic. Finally, this study indicated that, in limited-size aboveground vegetation biomass time series, there is no guarantee that deep learning methods will outperform traditional statistical methods.

Keywords: aboveground vegetation biomass; time series modeling; deep learning; convolutional neural network; long short-term memory; seasonal autoregressive integrated moving average; Kenyan grassland

check for updates

Citation: Noa-Yarasca, E.; Osorio Leyton, J.M.; Angerer, J.P. Deep Learning Model Effectiveness in Forecasting Limited-Size Aboveground Vegetation Biomass Time Series: Kenyan Grasslands Case Study. *Agronomy* **2024**, *14*, 349. https://doi.org/10.3390/agronomy14020349

Academic Editors: Theodoros Mavromatis, Thomas Alexandridis and Vassilis Aschonitis

Received: 4 January 2024
Revised: 1 February 2024
Accepted: 6 February 2024
Published: 8 February 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Aboveground vegetation biomass is a key indicator of an ecosystem's structure and function. Early forecasting of aboveground vegetation biomass is crucial for vegetation management and food security, as it could provide stakeholders and planners with an important tool to make more informed decisions and strategically address potential issues that could arise from shortages [1–3]. Furthermore, since aboveground vegetation biomass provides vital ecosystem services, early forecasting of vegetation biomass is essential for sustainable vegetation management [4,5]. Moreover, vegetation biomass management is essential for climate change mitigation since it plays an important role in the carbon cycle and the collection and storage of greenhouse gas emissions from a variety of human activities and industrial processes [2]. While aboveground vegetation biomass alone doesn't store all emissions, it does significantly contribute to carbon sequestration, helping to offset the impact of certain emissions. This underscores the significance of effective vegetation

biomass management as a part of comprehensive climate change strategies. For these reasons, aboveground vegetation biomass prediction has been the topic of multiple studies and publications [6,7]. These studies have developed multiple statistical and physical-based predictive models that achieved moderate to high levels of accuracy [2,8–10]. Despite endeavors to determine vegetation biomass indirectly via methods such as proxies or remote sensing information as opposed to direct measurements, few studies have concentrated on forecasting vegetation biomass in the near future (defined as 1–12 months) [11,12]. Forecasting approaches that have been utilized successfully in other disciplines (e.g., finance, hydrology) could aid in addressing the existing gap in vegetation biomass forecasting and provide early warning to pastoralists, farmers, planners, and other stakeholders on vegetation deficits that could affect livelihoods and environmental conditions.

Artificial intelligence (AI) algorithms are being increasingly employed to estimate future scenarios based on time series data in various fields such as finance [13,14], hydrology [15,16], and economics, among others [17]. AI algorithms can address complex nonlinear problems by integrating several nonlinear transformations. Thus, AI techniques have grown in popularity among academics in recent years as they have proven effective in time series modeling by identifying patterns that humans may not see or perceive immediately [18,19]. Furthermore, when new data becomes available, AI models can be updated, allowing them to adapt and improve over time [19]. The debate about the superiority of certain AI models over others is currently ongoing [20]. Prior studies have revealed that the performance of a model is related to the time series characteristics [19,21]. In this regard, previous studies agree that it is necessary to conduct additional research using time series with distinct characteristics from diverse disciplines, evaluating various models, and controlling the parameters and hyperparameters of these models in order to better understand and evaluate the merits various AI models [19,22]. In this regard, AI models for time series forecasting of aboveground vegetation biomass have not yet been explored.

Neural networks, integral to artificial intelligence, have evolved in time series analysis with increasing efficiency. While early studies, like White's on stock data [23], showed modest accuracy, subsequent works [14,24] demonstrated significant improvements. Despite not being agriculture-focused, these studies highlighted the increasing effectiveness of neural networks across diverse domains. As vast amounts of data become available, breakthroughs in deep learning techniques and increased processing capacity of computers have enabled the development of more sophisticated neural network architectures, resulting in improved accuracy and generalization of new data. The multilayer perceptron (MLP) model, for example, which includes several hidden layers in its architecture, has been used successfully in finance, economics, hydrology, and energy-demand forecasting [25]. The long-term memory model (LSTM) that is characterized by capturing long-term dependencies in time series has also performed well against other models in various studies, such as those by [13] in economics and finance, [16] in hydrology and others [26]. The Convolutional Neural Network (CNN) model, which was initially applied to image and video recognition, segmentation, and classification, has also proven to be useful in time series analysis by capturing more relevant features without requiring manual feature engineering [27]. CNN models outperform other DL models in studies undertaken, for example, by [27] in finance, ref. [28] in crop price prediction, and [29] in health care. More developed neural network approaches have involved hybrid models that combined two or more different types of neural networks—and in some cases also statistical models—to take advantage of the strengths of each model to produce a more powerful and flexible hybrid [30,31]. While beyond the primary scope of this agriculture-focused research, the hybrid CNN-LSTM model exhibits superior performance across diverse disciplines, including finance, environmental engineering, atmospheric sciences [32], and soil sciences [30]. Similarly, across various domains—economics [33], production planning, finance, and climatology [34,35]—LSTM-ARIMA consistently demonstrates superior predictive performance.

The above-mentioned studies show that debates on best time series modeling continue in diverse fields and are likely to continue for years to come. However, notwithstanding the abundance of time series modeling research, there are limited studies where these techniques have been applied to ecology, agriculture, and environmental sciences [36], and to our knowledge, no studies using vegetation biomass time series. This study evaluates established deep learning models (MLP, GRU, LSTM, CNN, CNN-LSTM), emphasizing the hybrid CNN-LSTM for forecasting aboveground vegetation biomass with limited size data and comparing them to the classic SARIMA model. Our analysis, based on diverse neural network architectures, provides insights into handling dataset complexities. Widely used as benchmarks [14,31], these models enable meaningful comparisons. Focused on a specific problem domain with proven effectiveness [22,37,38], the selected DL models, including MLP for simplicity, LSTM and GRU for sequential modeling, and CNN for spatial dependencies, offer a comprehensive evaluation. The hybrid CNN-LSTM model integrates spatial and temporal characteristics for a robust assessment [31].

2. Methodology

2.1. Aboveground Biomass Database

Aboveground biomass assessment employs techniques like clip harvesting and non-destructive methods such as remote sensing and allometric equations. Monitoring stations, strategically placed based on research goals and utilizing grid patterns or clustering, operate at various scales, ranging from small plots to global observations through satellites [1,2,6]. For this study, the Food and Agriculture Organization (FAO) and Texas A&M AgriLife Research (TAMU), in collaboration with the Kenya National Drought Monitoring Authority (KNDMA), have implemented a significant number of monitoring sites for aboveground biomass in grasslands and rangelands of Kenya. These monitoring sites are part of a network of sites that comprise the Predictive Livestock Early Warning System in East Africa [4,39]. Data from these sites were used to calibrate the Phytomass Growth Model (PHYGROW) [39,40] and generate near-real-time aboveground vegetation biomass time series in several strategic locations throughout Kenya [3,4]. The time series database employed for this study is accessible at https://github.com/noayarae/forecasting_biomass_using_DL_models.git, accessed on 18 August 2023.

Five aboveground biomass time series were chosen for this study and will be referred to as TS1, TS2, TS3, TS4, and TS5, respectively. These included values at 15-day intervals (two values per month) from 14 January 2002 to 31 August 2022. Therefore, each time series included 496 aboveground biomass values in kg/m^2. The chosen time series were visually scrutinized to confirm the absence of any irregularities or abnormal patterns, both in terms of isolated events and chronological sequences. Figure 1 shows the evaluated aboveground vegetation biomass time series. Significantly, these meticulously selected time series aptly capture a broad spectrum of regional aboveground biomass production, with an average low of 104.5 kg/m^2 and an average high of 3171.5 kg/m^2, effectively representing the range of production for the studied area.

2.2. Data Preprocessing

The time series $\{x_1, x_2, \ldots, x_{496}\}$ were preprocessed by normalizing them to help improve the model training process and to avoid issues such as vanishing gradients that can speed up model convergence. Normalization also helps to reduce the sensitivity of the model to the scale of the input features, improving the model's generalization ability [41]. Normalization was accomplished by scaling between 0 and 1 using the following equation:

$$y = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (1)$$

where x is the time series value, y is the normalized value, x_{max} is the highest value of the time series, and x_{min} the lowest value of the time series. For subsequent applications, if new data aligns closely with the training set, the models can use the same normalization

values. However, significant differences may warrant a re-evaluation and normalization using relevant values. As the initial models were trained with limited data, incorporating new data presents an opportunity for enhanced performance through retraining, ensuring continued optimal accuracy.

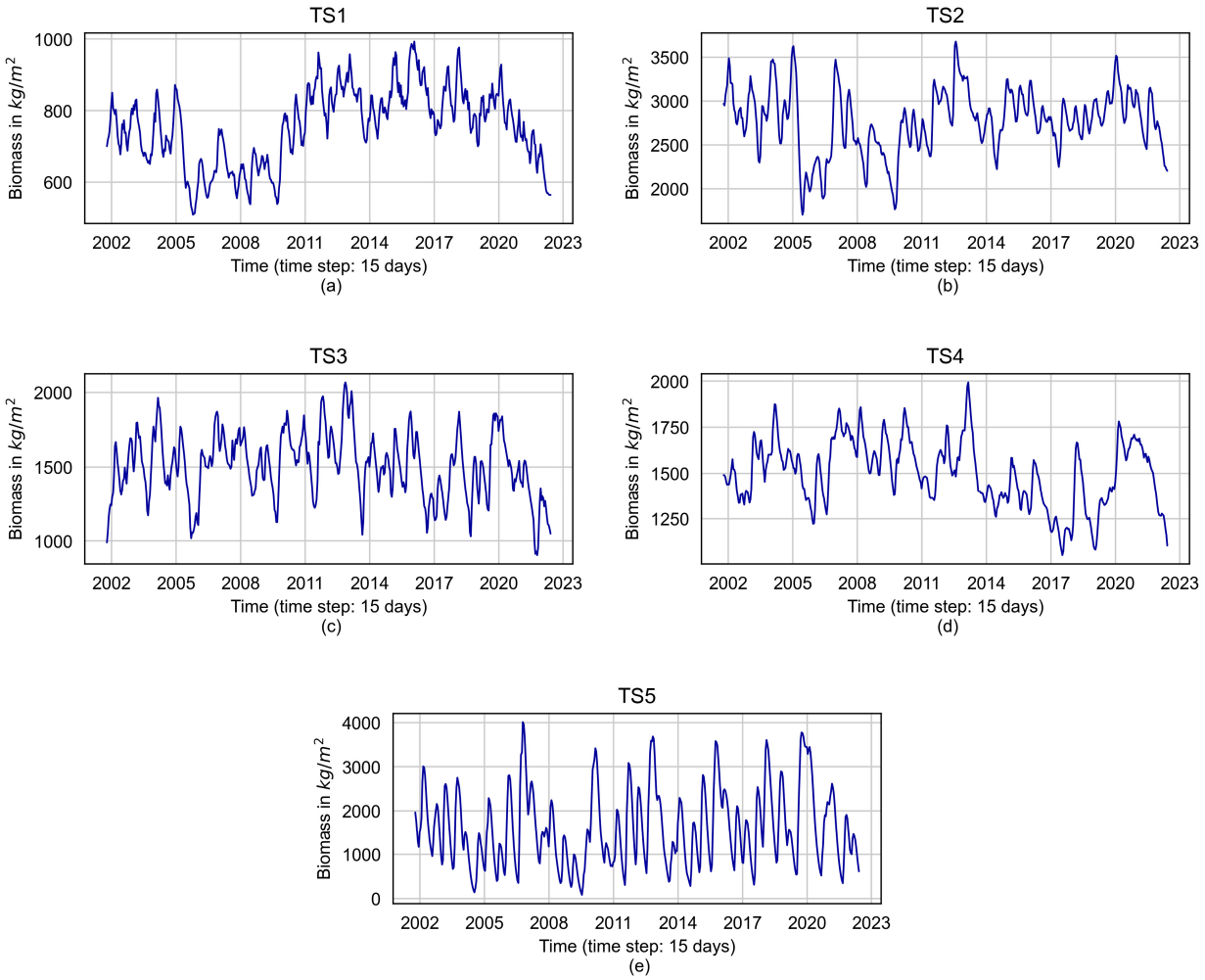

Figure 1. Aboveground vegetation biomass time series generated from calibrated PHYGROW model simulations across five representative rangeland sites in Kenya. (**a**) Time series 1 (TS1), (**b**) Time series 2 (TS2), (**c**) Time series 3 (TS3), (**d**) Time series 4 (TS4), and (**e**) Time series 5 (TS5).

Transforming Time Series Data into Supervised Dataset

After normalizing, the time series were arranged to form a supervised database, maintaining the time dependency. The last year of the time series (24 values in total) was held out for the purpose of testing the model. The rest of the series (472 values) were used to train the learning model. The first subset was formed by taking the first 25 values $\{y_1, y_2, \ldots, y_{25}\}$, of which the first 24 $\{y_1, y_2, \ldots, y_{24}\}$ were taken as predictors and the 25th $\{y_{25}\}$ as targets. The second subseries was obtained by sliding forward one time step in the full-time series and again taking 25 values ($\{y_2, y_3, \ldots, y_{25}\}$ as predictors and the 25th $\{y_{26}\}$ as targets. This process was repeated until the last value of the training series was reached (Figure 2). Following this process, the number of subseries obtained was: $N_{sd} = N_{train} - w - out + 1$.

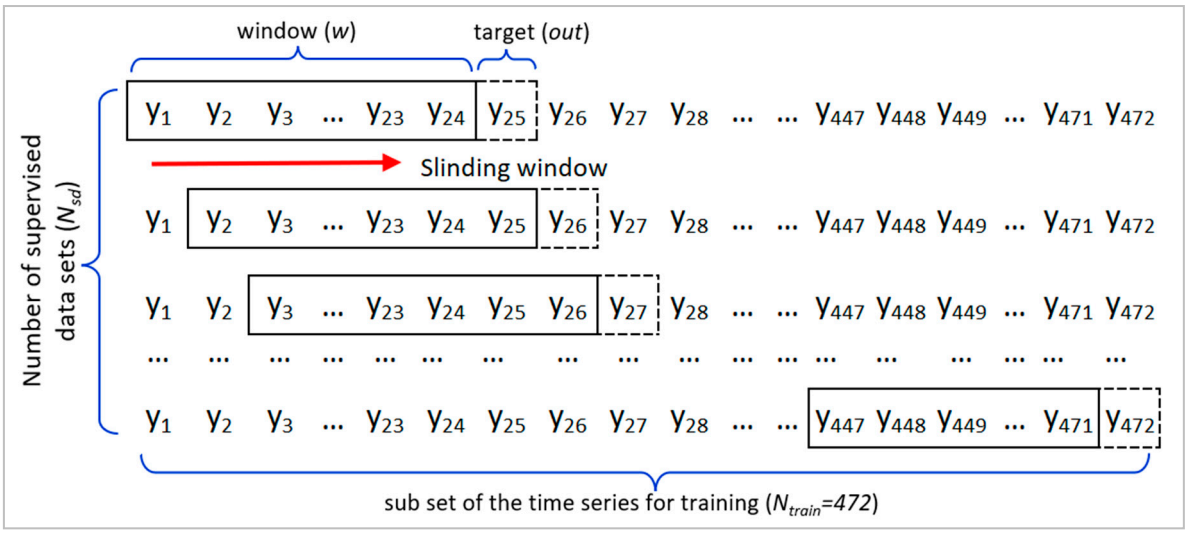

Figure 2. Using a sliding window to configure time series as a supervised data set.

2.3. Modeling, Prediction, and Tool Setup

The modeling process involved a sliding window of 24 data points representing the historical aboveground biomass data over one year. The forecasting horizons included 6, 12, 18, and 24 values, equivalent to 3, 6, 9, and 12 months, respectively. These forecasting intervals were strategically chosen to align with the practical needs of stakeholders. Farmers, ranchers, and decision-makers often require long-term forecasts (one year) for planning purposes, while the 3–6-month forecasting intervals are critical for rapid decision-making to mitigate issues such as forage scarcity impacting animal health. The horizons were achieved using one-step ahead iterative prediction. Taking into account the range of forecasting horizons and the number of time series, this study encompassed a total of twenty cases, with each case involving twenty-five replications of forecasting to ensure a reliable outcome. This comprehensive approach effectively covers a spectrum of decision points, providing valuable insights applicable to various scenarios where timely actions are paramount.

2.3.1. Iterative Forecasting Approach

The iterative method is a long-established technique for performing multi-step ahead predictions. It takes the output of a one-step ahead prediction as input for the next step prediction [18,42]. The equation to compute the first step ahead of prediction as a function of the w preceding values is as follows:

$$y'_{t+1} = f(y_t, y_{t-1}, y_{t-2}, \ldots, y_{t-w+1}) \quad (2)$$

where w is the sliding window length. The next step y'_{t+2} would also be based on the previous w values, including the newly predicted value y'_{t+1}:

$$y'_{t+2} = f(y'_{t+1}, y_t, y_{t-1}, \ldots, y_{t-w+2}) \quad (3)$$

In general, the value of p-step ahead would be.

$$y'_{t+p} = f\left(y'_{t+p-1}, y'_{t+p-2}, \ldots, y_{t+p-w}\right) \quad (4)$$

2.3.2. Software and Computer Tools

This study utilized the NVIDIA RTX 4090 GPU, renowned for advanced architecture and parallel processing, to accelerate computational tasks. Both the DL and SARIMA models were implemented using the Python platform v3.9.13 [43]. The DL models were

built using various open-access libraries, including scipy, keras, and tensorflow, while the SARIMA model mainly relied on statsmodels. The codes developed for this study are available at https://github.com/noayarae/forecasting_biomass_using_DL_models.git, accessed on 18 August 2023.

2.4. Learning Models—Model Implementation

2.4.1. Multilayer Perceptron (MLP)

The MLP is a type of artificial neural network composed of at least three layers: an input layer, one or more intermediate hidden layers, and an output layer [44]. The output at each hidden layer and in the output layer is computed as follows:

$$y_i = f\left(\sum_{k=1}^{n_N} w_{k,j}^N \; h_N^k\right) \quad (5)$$

where f is the activation function, n_N is the number of neurons, $w_{k,j}^N$ is the matrix of weights between the i^{th} hidden layer and the $(i+1)^{th}$ hidden layer, h_N^k is the output of the hidden neuron, and y_i is the vector of the output layer.

The learning process of the MLP algorithm consists of finding, using backpropagation, the weight values at each network connection (matrix of weights) to minimize the error between the output layer values and the expected values. Before the MLP learning process, the model underwent two-step manual tuning using GridSearchCV for optimal hyperparameter selection. In the initial hyperparameter optimization step, crucial learning parameters, including number of neurons, activation function, optimizer, learning rate, batch size, and epoch, were finely tuned, with details in Table 1. Values were chosen within literature-aligned ranges, ensuring relevance and comparability. For instance, the number of neurons ranged from 50 to 400, activation functions included Relu, Sigmoid, and Tanh, and learning rates varied from 0.001 to 0.01. Batch sizes of 16, 32, and 64 were considered, with epoch numbers ranging from 100 to 500 and an early stopping of 30. In the second step, focusing on model architecture refinement, 1- and 2-layer networks were explored alongside dropout rates of 0, 0.1, 0.15, and 0.20. This comprehensive exploration, rooted in existing research, ensures an optimal model configuration balancing complexity and generalization, adhering to established practices in deep learning.

Table 1. Hyperparameters of the MLP, LSTM, and GRU models.

Hyperparameter.	Model		
	MLP	**LSTM**	**GRU**
Number of layers	1	1	1
Number of hidden units (nodes)	200	100	100
Activation function	Sigmoid	Relu	Relu
Optimizer	Adam	Adamax	Adam
Learning rate	0.001	0.01	0.001
Batch size	64	64	64
Epochs	200	200	200
Early stopping (patient)	30	30	30
Dropout rate	0.15	0.15	0.15
Loss function	mean_squared_error		

2.4.2. Long Short-Term Memory (LSTM) Network

LSTM is an enhanced type of Recurrent Neural Network (RNN) designed to model sequences and their dependencies accurately over a longer period, maintaining a single-cell structure with several modifications to the standard RNN architecture. The LSTM architecture addresses the inability of the standard RNN to remember long-term dependencies [45]. The LSTM cell features three gates, known as Forget, Input, and Output, that control the flow of information. The Forget gate decides what information from the previous time step to discard; the Input gate determines what new information to add/learn from the current

input, and the Output gate regulates what information to pass to the next time step [18,46]. Figure 3 shows the LSTM model architecture.

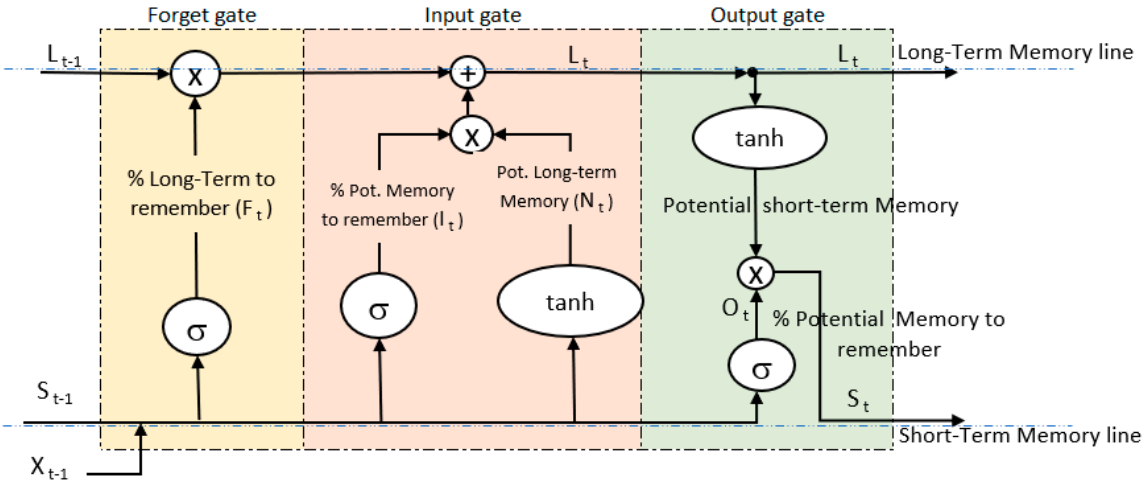

Figure 3. Long short-term memory network architecture.

As Figure 3 illustrates, in the first gate, the Forget gate, the LSTM decides what information to discard from the cell state using the sigmoid function. This layer considers the previous hidden state (S_{t-1}) and the current input (x_t) and, through the sigmoid function, outputs a value between 0 and 1 for each cell state in the long-term memory (L_{t-1}). A value of 1 means to keep the information in its entirety, while a value of 0 means to discard it completely.

In the second gate, the Input gate, the LSTM decides what new information to store in the cell state through two sub-steps: first, the input gate layer uses a sigmoid function to determine what percentage of the potential memory to add to the long-term memory. Second, the LSTM uses a *tanh* function to combine the short-term memory and the input to create a potential long-term memory. The product of the two sub-step outputs is added to long-term memory to update it (L_t).

In the third gate, the Output gate, the short-term memory is updated. Here, the new long-term memory is processed through a *tanh* function to find the potential short-term memory. The LSTM then decides how much of this potential short-term memory has to pass on by multiplying a percentage computed using a sigmoid function (O_t).

It is important to note that the LSTM memory cells utilize both addition and multiplication in transforming and transferring information. The use of addition is crucial in helping to maintain a consistent error during backpropagation. Instead of affecting the next cell state through the multiplication of the current state with new input, the two are combined through addition in the Input gate. Meanwhile, the Forget gate continues to rely on multiplication [46].

During the training process, the initial LSTM parameter values (weights and bias) are generated arbitrarily. These parameters are updated via the standard gradient descent method using the backpropagation algorithm. The performance of this algorithm is highly dependent on the selection of optimal hyperparameters, which improves the accuracy of time-series problems [18,45–47]. The LSTM hyperparameters were fine-tuned manually, following a two-step process using GridSearchCV, mirroring the approach taken for MLP. The optimization encompassed the following ranges for hyperparameters: number of neurons (50 to 400), activation functions (ReLU, Tanh, and Sigmoid), optimizers (Adam and Adamax), learning rate (0.001 to 0.01), batch sizes (16, 32, 64), the number of epochs (100 to 500) with early stopping. Additionally, one-layer and two-hidden-layer networks were assessed, considering dropout rates ranging from 0 to 0.2. Table 1 presents the optimized values.

2.4.3. Gated Recurrent Unit (GRU)

The GRU algorithm is an enhanced type of RNN that addresses the issue of vanishing gradients in standard RNNs [18]. It accomplishes this by utilizing two types of gates—an update gate and a reset gate—which are essentially two vectors that determine the relevant information to be transmitted to the output. These gates can be trained to selectively retain relevant information from previous time steps without losing value over time, as well as to eliminate irrelevant information that hinders accurate prediction [48]. Figure 4 illustrates this process:

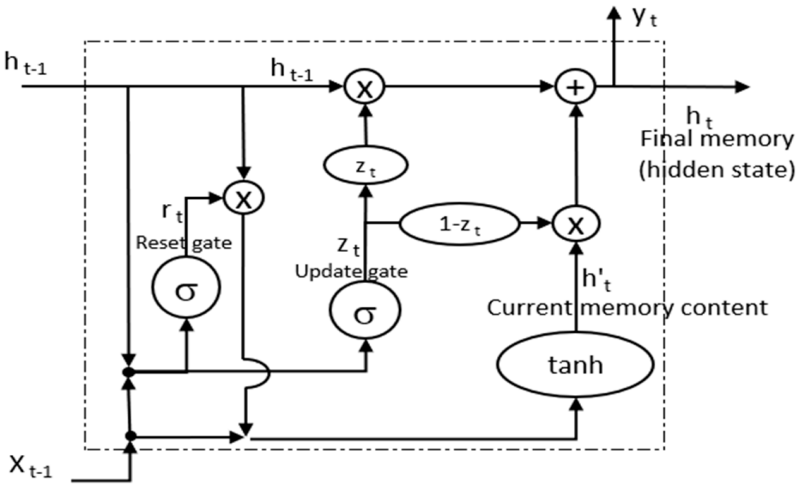

Figure 4. Gated Recurrent Unit network architecture.

The Reset gate (r_t) determines how much of the previous hidden state should be forgotten and how much of the current input should be considered. The gate considers the previous hidden state (h_{t-1}) and the current input (x_t) and, through the sigmoid function, outputs a value between 0 and 1. The Update gate (z_t) for long-term memory is computed similarly but with different weights. This gate enables the model to decide the amount of past information that should be propagated to the future time steps. The model can opt to replicate all past information, thus eliminating the vanishing gradient problem [48,49]. In the next step, Current memory content, the content of a new memory (h'_t) (candidate hidden state), is computed considering the reset gate output (r_t), the prior memory (h_{t-1}), and the input (X_t) through the *tanh* function that outputs a value between -1 and 1. The Final memory (h_t; hidden state) is computed using a single equation (Equation (6), where the operator $\odot$ denotes the Hadamard product) to control both the historical information (h_{t-1}) and the new information coming from the candidate hidden state (h'_t) affected by a factor formed by the update gate output (z_t) as follows [48,50]:

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \quad (6)$$

For z_t close to 0, the new hidden state relies mostly on the candidate state, while for z_t close to 1, it relies on the previous hidden state. Therefore, the value of z_t is critical and ranges from 0 to 1. GRU hyperparameters were tuned in two steps, like MLP and LSTM. Hyperparameters' ranges included neurons (50–400 units), activations (Relu, Tanh, Sigmoid), optimizers (Adam, Adamax), learning rates (0.001–0.01), batch sizes (16, 32, 64), and epochs (100–500) with an early stop (20). Step two explored one/two hidden layers and dropout rates (0–0.2). Optimization aimed at reducing errors and simplifying the model. Table 1 contains optimized values.

2.4.4. Convolutional Neural Network (CNN)

The CNN algorithm is a neural network specialized in processing complex data. Although widely used in image processing, CNN's algorithms have also shown excellent

results in time series simulation [51]. In addition to the conventional MLP algorithm, the CNN algorithm includes a convolution and pooling layer, which makes it a powerful algorithm [52].

In contrast to the LSTM, which considers past information in a sequence-based context window, the CNN considers past information through a context window that represents the spatial relationships between values in the feature map. As a result, the convolutional layers of the CNN can scan the input feature map and identify local patterns, using this past information to construct a representation of the input.

CNN differs from the LSTM by considering past information via a spatial context window instead of a sequential one. Thus, the convolution process in the CNN scans the input feature map and identifies local patterns, forming a representation of the input by utilizing this past information.

In the context of 1D CNNs, time series are processed as one-dimensional data ($1xN$) with a dedicated 1D kernel ($1xK$) and multiple channels (n_1) for diverse kernel training. Post-convolution, pooling shrinks the series size, optimizing computational efficiency. Subsequently, the resulting series is flattened to extract a single node for each value within the generated series. Preserving both time and channel dimensions, this methodology transforms time series into a format conducive to comprehensive feature extraction and analysis within the 1D CNN framework, preserving both the time and channel dimensions. Thus, the flattening process determines the neural network's input nodes, computed as the product of the length of the grouped series and the number of channels, expressed as $n_1.\left(\frac{N-K+1}{S}\right)$. The fully connected MLP layer utilizes the flattened data to produce the final Output (Figure 5).

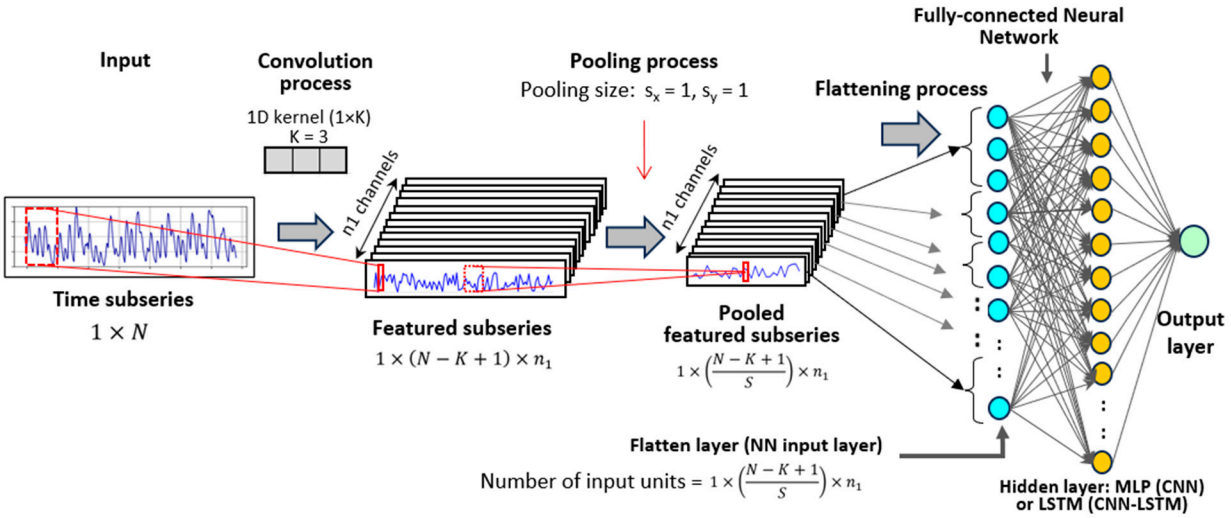

Figure 5. Model architecture overview: CNN with MLP hidden layer, CNN-LSTM with LSTM hidden layer.

The gradient descent optimization algorithm is used to update the parameters (weights and biases) of the CNN during training to minimize the loss between the predicted and actual output for the training samples [53]. The CNN hyperparameters underwent a two-step manual tuning process using GridSearchCV. In the initial step, various hyperparameters were explored, including the number of neurons (50 to 400), activation functions (Relu, Tanh, and Sigmoid), optimizers (Adam and Adamax), learning rates (0.001 to 0.01), batch sizes (16, 32, 64), epoch numbers from 100 to 500 with early stopping set at 20. Regarding convolution, considerations encompassed kernel sizes of 3 and 5, number of filters of 8 and 16, "same" padding for maintaining subsequent length, strides of 1 and 2, and pool sizes of 1 and 2. In the second step, the evaluation extended to one and two hidden layers, with dropout rates ranging from 0 to 0.2. Hyperparameter optimization prioritized minimizing error and model complexity. The optimized values are detailed in Table 2.

Table 2. Hyperparameters of the CNN and CNN-LSTM models.

Hyperparameter	Model	
	CNN	**CNN-LSTM**
Convolution layer filters	16	16
Convolution layer kernel size	3	3
Convolution layer activation function	Relu	Relu
Convolution layer padding	same	same
Strides	1	1
Pooling layer pool_size	1	1
Pooling layer padding	same	same
Number of hidden layers	1	1
Number of hidden units MLP/LSTM layer	50	50
LSTM layer activation function	Relu	Relu
Batch size	64	64
Learning rate	0.01	0.001
Optimizer	Adamax	Adam
Loss function	mean_squared_error	
Epochs	100	100
Early stopping (patient)	20	20

2.4.5. The Hybrid CNN-LSTM Algorithm

The hybrid CNN-LSTM algorithm, useful for tasks involving sequential data such as time series forecasting, combines the strengths of both the CNN and LSTM networks. Here, the CNN layer is typically used to extract high-level features from the input sequence. These features are then passed to the LSTM layer, which is responsible for modeling the temporal dependencies between the features and producing the final output [54].

The key advantage of the hybrid CNN-LSTM algorithm is that it can effectively capture both local and global dependencies in the input sequence [31]. The CNN layer is able to capture local patterns and features within the sequence, while the LSTM layer is able to model longer-term dependencies between these features. In addition, the hybrid CNN-LSTM algorithm that typically trains end-to-end using backpropagation allows the model to learn both the feature extraction and temporal modeling stages together, resulting in improved performance as compared to traditional methods that use separate feature extraction and modeling stages [55]. Overall, the hybrid CNN-LSTM algorithm is a powerful tool for modeling complex sequential data and has been successfully applied in a wide range of applications.

The convolution process by which the CNN-LSTM algorithm architecture incorporates a LSTM network layer into the CNN algorithm architecture is illustrated in Figure 5. CNN-LSTM hyperparameters were tuned in two steps with GridSearchCV, following a similar approach to CNN. They were optimized within the same ranges as CNN, seeking to reduce error and model complexity. Optimized hyperparameters are in Table 2.

2.4.6. Seasonal Autoregressive Integrated Moving Average (SARIMA)

The SARIMA model is an extended statistical technique of the ARIMA model that is widely used to analyze and forecast time series data that exhibit seasonal behavior. The SARIMA model consists of six parameters $(p, d, q)(P, D, Q)_s$ that are used to estimate the future value of a variable as a linear function of past observations and random errors. The first three parameters (p, d, q) represent the non-seasonal components of the model and are identical to those in the ARIMA model. The last three parameters $(P, D, Q)_s$ represent the seasonal components of the model and are used to model the seasonal behavior in the data. As proposed in past studies [56], the method for creating the time series is as follows:

$$\Phi_P(B^s)\,\phi(B)\,\nabla_s^D\,\nabla^d\,(X_t-\mu) = \Theta_Q(B^s)\,\theta(B)\,w_t \tag{7}$$

where X_t is the actual value (predicted value) and w_t is the random error (non-stationary time series), both at time t. $\phi(B) = 1 - \phi_1 B - \ldots - \phi_p B^p$ and $\theta(B) = 1 + \theta_1 B + \ldots + \theta_q B^q$ are the nonseasonal autoregressive and moving average components of order p and q, respectively. $\Phi_P(B^s) = 1 - \Phi_1 B^s - \ldots - \Phi_P B^{Ps}$ and $\Theta_Q(B^s) = 1 + \Theta_1 B^s + \ldots + \Theta_Q B^{Qs}$ are the seasonal autoregressive and moving average components of order P and Q, respectively. $\nabla^d = (1 - B)$ and $\nabla_s^D = (1 - B^s)$ are the non-seasonal and seasonal difference components. $B^k X_t = X_{t-k}$ is the backshift operator. The aboveground biomass time series in this study is a 15-day time step, hence the seasonal period is $s = 24$. The SARIMA model is estimated using the Box-Jenkins methodology, which involves identifying the optimal values of the model parameters based on Akaike Information Criterion (AIC) statistical test. The SARIMA model parameters are shown in Table 3.

Table 3. Parameters of the SARIMA model.

Time Series	SARIMA Model Parameters (p,d,q) × (P,D,Q)$_{24}$.
TS1	(1,1,3) (1,1,1)$_{24}$
TS2	(2,1,1) (2,1,1)$_{24}$
TS3	(2,1,2) (1,1,1)$_{24}$
TS4	(2,1,1) (1,1,1)$_{24}$
TS5	(3,1,1) (1,1,1)$_{24}$

2.5. Model Performance Evaluation

The model's efficiency was assessed using the Root Mean Square Error (*RMSE*) as follows:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}} \tag{8}$$

where x_i is the observed time series value at time i, $\hat{x}_i$ is the estimated/forecast time series value at time i, and N is the number of time series data. RMSE values vary from 0 to ∞, using the same units as the variable being measured, with 0 indicating a perfect model with zero prediction error.

In addition, the model's performance in percentage terms was assessed using the Mean Absolute Percentage Error (*MAPE*):

$$MAPE = 100\frac{1}{N}\sum_{i=1}^{N}\left|\frac{x_i - \hat{x}_i}{x_i}\right| \tag{9}$$

MAPE values vary from 0% to ∞, with lower values indicating more accurate predictions and 0% being a perfect model with zero prediction error. MAPE values less than 5% suggest an acceptable level of forecast accuracy, while values between 10% and 25% indicate low but acceptable accuracy, and values greater than 25% indicate an unreliable model. On the Relationship among Values of the Same Summary Measure of Error).

Moreover, model error was assessed using the Symmetric Mean Absolute Percentage (*SMAPE*):

$$SMAPE = 100\frac{1}{N}\sum_{i=1}^{N}\frac{2\,|x_i - \hat{x}_i|}{|x_i| + |\hat{x}_i|} \tag{10}$$

Unlike the mean absolute percentage error, SMAPE has defined limits. The formula mentioned yields outcomes ranging from 0% to 200%. SMAPE's bounded nature improves forecast accuracy assessment by constraining extreme values for clearer error interpretation.

Furthermore, to rigorously compare RMSE values across distinct DL algorithms and cases, we conducted the Analysis of Variance (ANOVA) statistical test. This method revealed significant variations in RMSE outcomes, enhancing our understanding of relative efficacy. Within each case, we compared average RMSEs from twenty-five replications

using the CNN-LSTM algorithm with those from other DL algorithms, providing insights into its performance against counterparts.

3. Results

Time Series Modeling

Figure 6 depicts a summary of the mean RMSE values reflecting the performance of the models utilized for the various horizons assessed. As previously stated, this study conducted twenty-five forecasts for each case, which involved five time series and four horizons. Since the learning models have random components, the predicted series varied among the twenty-five iterations. To address this, the RMSE values, as presented in Figure 6, were averaged.

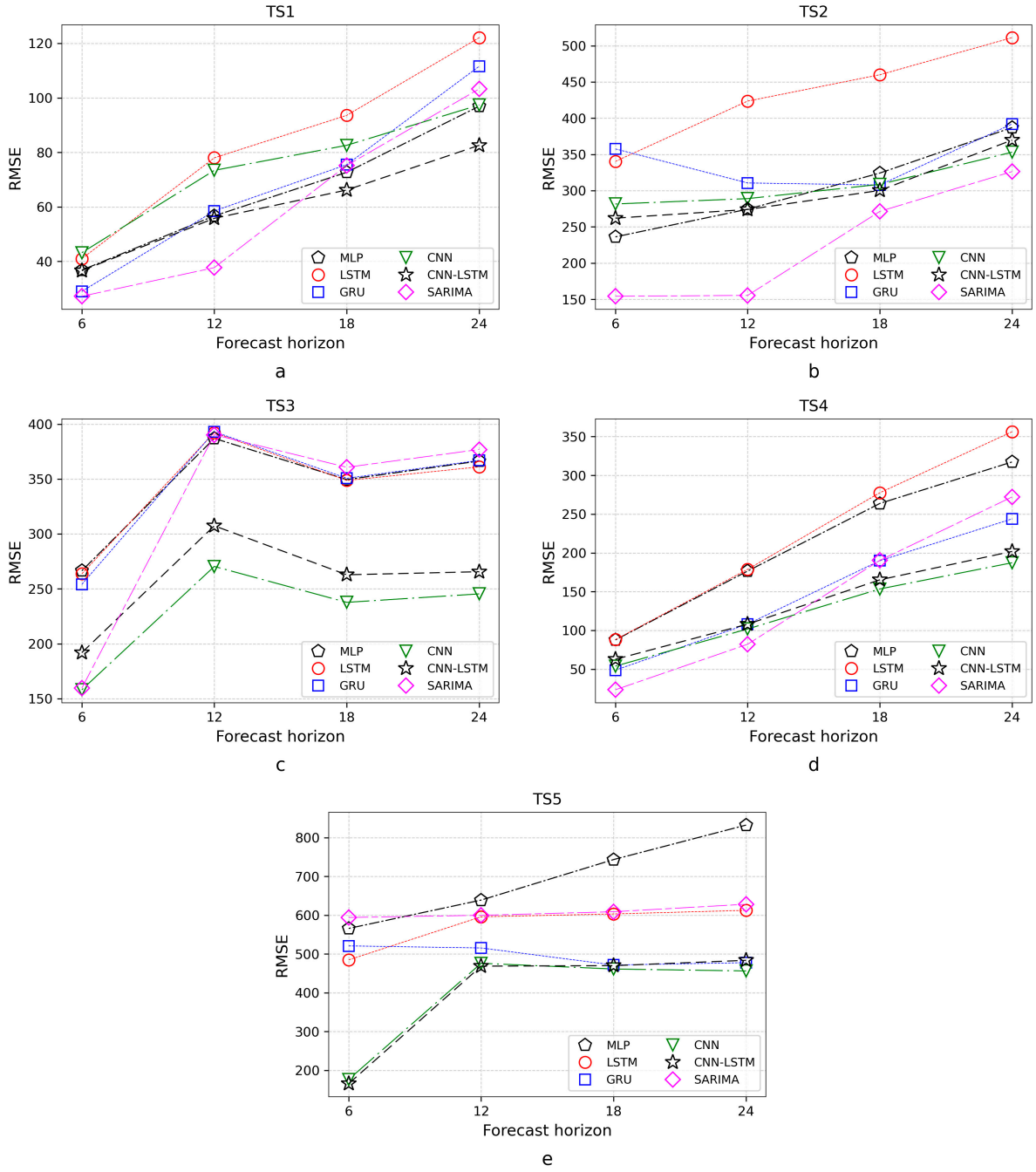

Figure 6. Performance of DL and statistical models in aboveground vegetation biomass time series forecasting over four horizons. (**a**–**e**) correspond to time series TS1 to TS5, respectively.

In order to evaluate whether there were differences in the performance of the models, an ANOVA test was conducted on the RMSE values. The one-way ANOVA revealed that, overall, significant differences in RMSE existed among all the models at the $p < 0.05$. Nonetheless, some pairs of models demonstrated no significant difference in their performance, as was the case with the MLP and CNN models when predicting the TS1 time series for a one-year horizon. ($t_{(48)} = 0.487$, $p = 0.97$) (Figure 6a).

In terms of accuracy measured by RMSE, MAPE, and SMAPE, none of the six evaluated models consistently demonstrated a significantly better performance across the five time series and the four horizons assessed, totaling 20 cases. The three top-performing models were the SARIMA, CNN, and CNN-LSTM models, with one of these three models outperforming the others in most cases. The SARIMA model, for example, performed better than all other models in eight cases. The CNN model outperformed the other models in eight cases as well, and the CNN-LSTM model in four cases. Upon reviewing the top runners-up, the CNN-LSTM model emerged as the leading contender, securing second place nine times. This is evident in Table 4, where the top two models for each forecast horizon within each time series are highlighted in bold. Notably, most of these highlighted values correspond to the CNN, CNN-LSTM, and SARIMA models.

Table 4. Performance (RMSE, MAPE, and SMAPE) of all methods on the five aboveground vegetation biomass time series. For each case, the two best-performing approaches were highlighted.

Model	TS1				TS2				TS3				TS4				TS5			
	6	12	18	24	6	12	18	24	6	12	18	24	6	12	18	24	6	12	18	24
Root Mean Square Error (RMSE)																				
MLP	36.8	56.7	**72.7**	**97.0**	**236.2**	274.4	324.3	387.1	266.5	387.0	349.5	366.4	87.8	176.6	263.8	317.3	566.0	639.2	743.7	832.6
LSTM	41.0	78.0	93.6	122.1	340.4	423.6	460.0	511.2	263.7	392.3	348.7	361.1	88.2	178.8	277.6	356.3	485.0	596.1	603.4	612.7
GRU	**29.0**	58.5	75.5	111.6	357.6	310.6	307.8	392.1	254.1	393.1	350.7	367.0	**48.8**	108.2	190.0	243.9	521.0	515.7	471.7	**477.8**
CNN	43.2	73.4	82.6	97.4	281.6	289.0	308.7	**353.1**	**158.1**	**270.4**	**237.7**	**245.4**	54.0	**101.9**	**153.5**	**187.3**	**176.6**	**476.2**	**461.4**	**456.4**
CNN-LSTM	36.6	**55.8**	**66.2**	**82.6**	262.0	**274.0**	**300.3**	369.8	191.9	**307.4**	**262.8**	**265.6**	62.8	108.1	**165.3**	**201.9**	**165.6**	**468.8**	**470.1**	483.8
SARIMA	**27.2**	**37.7**	75.2	103.3	**154.2**	**155.4**	**271.4**	**326.3**	**159.6**	390.3	360.7	376.9	**23.5**	**82.2**	190.4	272.2	594.3	599.9	609.1	628.4
Mean Absolute Percentage Error (MAPE)																				
MLP	4.6	7.2	9.4	**13.2**	**6.7**	8.2	9.9	12.6	20.7	34.3	29.2	31.1	5.1	10.4	16.8	21.7	108.9	77.2	75.9	84.0
LSTM	5.0	9.9	12.4	17.3	9.6	12.3	14.3	17.2	20.2	34.4	28.8	30.4	5.2	10.5	17.5	23.9	88.6	65.9	59.4	59.6
GRU	**3.8**	7.4	9.8	15.3	10.3	8.9	**9.2**	12.7	19.2	34.3	29.1	31.0	**2.8**	6.2	11.7	16.2	98.8	65.2	52.0	51.2
CNN	5.3	9.2	10.9	13.7	8.2	8.7	9.5	**11.6**	**11.2**	**22.6**	**18.5**	**19.4**	3.2	**5.9**	**9.6**	**12.5**	**30.4**	**32.5**	**30.4**	**31.6**
CNN-LSTM	4.5	**6.9**	**8.5**	**11.4**	7.7	8.1	9.4	12.3	14.1	**26.3**	**20.6**	**21.4**	3.8	6.5	**10.6**	**13.7**	**29.1**	**32.1**	**32.0**	**34.7**
SARIMA	**3.2**	**4.8**	**8.7**	13.5	**4.5**	**4.7**	**8.3**	**11.1**	**10.7**	31.7	28.5	30.7	**1.4**	**4.3**	10.6	16.9	99.8	70.9	63.5	64.6
Symmetric Mean Percentage Error (SMAPE)																				
MLP	4.6	7.0	9.0	**12.1**	**7.1**	**8.4**	9.7	11.8	17.9	27.6	24.0	25.6	4.9	9.6	14.8	18.6	56.2	48.0	48.9	52.4
LSTM	5.0	9.3	11.6	15.5	10.3	12.5	14.1	16.3	17.5	27.7	23.7	25.2	4.9	9.6	15.0	19.3	52.4	45.2	41.7	42.6
GRU	**3.8**	7.0	9.2	13.7	11.0	9.3	9.3	12.0	16.8	27.7	24.1	25.8	**2.8**	5.8	10.6	14.3	53.8	42.4	**35.8**	**35.5**
CNN	5.3	8.6	10.1	12.4	8.8	9.2	9.7	**11.3**	**10.0**	**19.0**	**15.9**	**16.7**	3.1	**5.7**	**8.8**	**11.2**	**34.2**	38.9	**34.7**	**33.8**
CNN-LSTM	4.5	**6.6**	**8.0**	**10.5**	8.1	8.4	**9.3**	11.7	12.6	**22.0**	**17.5**	**18.4**	3.7	6.3	9.9	**12.5**	**24.7**	**34.4**	35.9	36.7
SARIMA	**3.1**	**4.7**	**8.0**	12.2	**4.6**	**4.8**	**7.9**	**10.4**	**9.7**	25.4	23.5	25.5	**1.4**	**4.1**	**9.6**	14.8	46.8	**37.0**	37.3	40.6

The top two models for each forecast horizon within each time series are highlighted in bold.

In most instances, each model's accuracy varied considerably depending on the predicted horizon. In most cases, a positive relationship was found between the accuracy of the model and the forecast horizon with *p*-values less than 0.05 in most cases. Despite the upward trend in model performance on the horizon, this relationship was not monotonic. In the time series TS3 (Figure 6c), for example, the 18-month average RMSE was less than the 12-month average RMSE (Table 4). Likewise, in the time series TS2 and TS5 (Figure 6b,e), the slopes of RMSE values for certain models did not consistently rise with the increasing forecast horizon.

In evaluating algorithm performance, we calculated average computational speeds (training and testing) for MLP (17.355 s), LSTM (39.761 s), GRU (32.465 s), CNN (6.808 s), CNN-LSTM (17.149 s), and SARIMA (29.082 s). It's noteworthy that the reported times represent averages across multiple runs. The training phase consistently surpasses testing, in line with literature findings. Importantly, these times exclude hyperparameter tuning, a time-intensive process, particularly in deep learning, where extensive attention is required due to the numerous hyperparameters. These insights offer a concise perspective on algorithm efficiency for practical applications.

4. Discussion

The first finding of this study is that the CNN-LSTM hybrid model did not exhibit absolute superiority over other DL models or the statistical model in predicting vegetation biomass. This might suggest that the CNN-LSTM model did not have a significant advantage over the other models in accurately predicting vegetation biomass within the constraints of a limited dataset. Even though none of the models evaluated demonstrated complete dominance, the SARIMA, CNN, and CNN-LSTM models exhibited better performance compared to the other three models included in this study. Contrary to numerous studies that claim the supremacy of DL models over classical statistical approaches, this study shows that statistical models could still provide better or comparable predictions than sophisticated models.

Several recent studies have reached similar findings. For instance, [19] found that basic statistical models performed better than machine learning (ML) models and indicated that complex models are not necessarily superior to simple forecasting models [19]. Other research has demonstrated the superiority in predictive capabilities of statistical models over ML models with respect to energy production [38], financial market data [57], health, and cryptocurrency [58].

The disappointing performances of the DL models might be related to specific characteristics of time series data, such as periodicity, seasonality, and noise [19,21,38], along with factors such as the dataset size [19,37] and model parameters [58]. Since ML models involve multiple nonlinear transformations, it would seem that they should prove superior to statistical models; however, evidence from previous studies has indicated that ML models cannot be generalized from small datasets—a limitation relative to traditional statistics.

Although both statistical models and DL models rely on data to make predictions, ARIMA models concentrate on predicting future values based on past neighboring and seasonally lagged data, while DL models employ sliding window data that undergoes several optimized transformations during the model training process. When time series data is limited in size, it may be insufficient to determine the optimal transformations and identify the key predictor neighbors for accurate forecasting; consequently, model efficiency may be compromised. Furthermore, the presence of noise and chaos in the time series can add to the challenge of identifying the optimal transformations [57]. Likewise, the prevailing conditions of intense drought during the last two years of the analyzed biomass time series could be exerting a notable impact on the results of near-future forecasts through ARIMA models and neural networks. The ARIMA model's predictive accuracy, for example, is largely dependent on historical patterns and trends. Given the extended period of arid conditions, the model might overemphasize the significance of these dry years in its predictions. Consequently, it could be producing forecasts that underestimate the potential for variance, especially if the future is anticipated to be less drought-prone. The performance of the deep learning algorithms may also have been hampered by a lack of wet periods in the previous two years. Thus, the model might be struggling to generalize effectively from such a narrow range of variation, potentially resulting in overly optimistic or pessimistic predictions.

Cerqueira et al. similarly found that the ARIMA models are better than the ML models when time series data is limited in size; their experiments considered a time series of 1000 values [37]. Other studies, such as that of Makridakis et al., have obtained similar results for time series of between 1600 and 1800 values [19]. Similarly, Vabalas et al. showed limitations in the performance of ML models based on a limited sample size [59]. These studies are consistent with the findings of our study, which considered just 496 values with seasonality. This logic is consistent with the fundamental principles of DL algorithms, indicating that the size of the training dataset has a direct impact on their performance. A larger and more diverse dataset enables algorithms to generalize and produce more accurate predictions, whereas a small dataset may result in overfitting and poor performance on new data [19,42].

Additionally, this study found that, as with the other DL and statistical models, the forecast accuracy of the CNN-LSTM model typically diminished as the forecast horizon

increased. However, this relationship was not always monotonic, meaning that the decrease in forecast accuracy was not consistent throughout the forecast horizon. There could be instances where the accuracy of the forecast might increase, decrease, or stay the same as the forecast horizon changes. The reason behind this non-monotonic relationship could be due to several factors, including noise or underlying patterns in the data and the type of forecasting model used. This is consistent with the results obtained, for example, by Zhang, who employed SVR and found a non-monotonic relationship between MSE and the forecast horizon in iterative forecasting. The iterative prediction approach, despite being simple and practical, suffers from low performance over long periods due to the accumulation of errors. The errors found in each step are included in the forecast of the next step as we feed the model [19]. Although DL methods are sophisticated, the uncertainty characteristics of vegetation biomass will lead to forecast errors and the propagation of errors in an iterative forecast over a long horizon. In this regard, further exploration of DL models that include non-iterative processes is required. These results underscore the dynamic nature of vegetation biomass production across time scales. Climatic dynamics, including precipitation changes, temperature fluctuations, and other growth-influencing factors, might lead to varying predictability. DL models could be capturing these complexities, explaining the non-monotonic forecast accuracy with changing horizons. This highlights the need for further research to delve into these intricate relationships and their impact on vegetation biomass fluctuations.

Finally, while sophisticated methods such as hybrid models have demonstrated outstanding effectiveness in a variety of fields, this study did not replicate such results, instead highlighting statistical models for forecasting vegetation biomass. The time series evaluated in this study exhibit specific characteristics, and thus, the findings of this study are limited to time series with these characteristics. In this sense, more studies on time series forecasting using modern forecasting techniques for limited data sizes are clearly required.

5. Conclusions

This study assessed the effectiveness of DL models, including the hybrid CNN-LSTM, for predicting aboveground vegetation biomass time series with limited size data, comparing their performance to the traditional SARIMA statistical model, which served as a benchmark for evaluation. To conduct this assessment, we utilized five aboveground vegetation biomass time series datasets, consisting of values collected at 15-day intervals over a 20-year period from fields in Kenya. The outcomes of our study demonstrated significant differences in the forecasting accuracy of the various models. The CNN-LSTM model was not significantly more accurate than the other models; however, it did rank alongside the SARIMA and CNN models as one of the three most reliable—with the statistical model, SARIMA, slightly outperforming the other two. The LSTM and MLP were the least accurate models. The moderate performance of the hybrid CNN-LSTM model could be attributed to the specific characteristics of the data, as well as the limited size of the vegetation biomass time series. These factors may have restricted the learning capacity of the model in spite of its sophisticated prediction architecture. It is possible that the performance of the proposed model could improve as larger quantities of aboveground vegetation biomass data are collected in the future.

Furthermore, this study revealed that the accuracy of the hybrid CNN-LSTM model—as well as the other models—differed significantly depending on the forecast horizon. As the prediction horizon increased, the accuracy of the models tended to decrease, resulting in larger forecast errors. Although there was a clear overall trend, the relationship between forecast horizon and model accuracy was not strictly monotonic. Consistent with prior research, the decline in model performance over longer forecast horizons could be attributed to the cumulative error that occurs in the iterative prediction process.

Findings from this study may not be generalizable to other vegetation biomass time series data with different characteristics and sample sizes. Meanwhile, based on our outcomes, it is evident that with the limited size and specific characteristics of aboveground

vegetation biomass time series, there is no guarantee that machine learning forecasting techniques will outperform statistical methods. Moving forward, it would be beneficial for future research to explore the application of more advanced machine learning models on larger vegetation biomass time series datasets, including but not limited to algorithms such as naive Bayes, gradient boosting algorithms, and Generative Adversarial Networks (GANs). The inclusion of these advanced models could shed further light on their efficacy in comparison to traditional statistical methods. For shorter time series, we recommend continued evaluation of various methods to achieve acceptable levels of accuracy. Moreover, future studies should explore diverse parameters unexplored here, such as the input window length, which could significantly impact forecasting accuracy.

Author Contributions: E.N.-Y. conceptualized and designed the methodology, while E.N.-Y. and J.M.O.L. gathered the data. E.N.-Y. conducted the modeling and analysis. All authors discussed the results and gave critical feedback on the paper. E.N.-Y. primarily authored the paper, incorporating inputs from all co-authors. All authors have read and agreed to the published version of the manuscript.

Funding: The support for this research has been provided by The Food and Agriculture Organization of the United Nations (FAO) under the project Establishing and Strengthening Predictive Livestock Early Warning System (PLEWs) in Kenya, Somalia, Sudan, and Uganda as part of the delivery of the project Food and Nutrition Security Resilience Programme (FNS-REPRO) (TAMU M-2002133—Account 408999-96610).

Data Availability Statement: The aboveground vegetation biomass time series data used in this study, as well as all code for training and testing the deep learning and seasonal autoregressive integrated moving average models are available at https://github.com/noayarae/forecasting_biomass_using_DL_models.git, accessed on 18 August 2023.

Acknowledgments: We express our gratitude to Erin Camody for her valuable contributions to the writing and proofreading of this document.

Conflicts of Interest: The authors declare that they have no conflicts of interest.

References

1. Saarela, S.; Wästlund, A.; Holmström, E.; Mensah, A.A.; Holm, S.; Nilsson, M.; Fridman, J.; Ståhl, G. Mapping aboveground biomass and its prediction uncertainty using LiDAR and field data, accounting for tree-level allometric and LiDAR model errors. *For. Ecosyst.* **2020**, *7*, 43. [CrossRef]
2. Das, B.; Patnaik, S.K.; Bordoloi, R.; Paul, A.; Tripathi, O.P. Prediction of forest aboveground biomass using an integrated approach of space-based parameters, and forest inventory data. *Geol. Ecol. Landscapes* **2022**, 1–13. [CrossRef]
3. Osorio Leyton, J.M. *Piloting of the Predictive Livestock Early Warning System (PLEWs)*; Final Report for FAO Letter of Agreement No. SS/085/20; FAO: Rome, Italy, 2021.
4. Matere, J.; Simpkin, P.; Angerer, J.; Olesambu, E.; Ramasamy, S.; Fasina, F. Predictive Livestock Early Warning System (PLEWS): Monitoring forage condition and implications for animal production in Kenya. *Weather. Clim. Extremes* **2020**, *27*, 100209. [CrossRef]
5. Braimoh, A.; Manyena, B.; Obuya, G.; Muraya, F. *Assessment of Food Security Early Warning Systems for East and Southern Africa*; World Bank: Washington, DC, USA, 2018; p. 20433. Available online: http://hdl.handle.net/10986/29269 (accessed on 5 February 2023).
6. Chen, S.; Feng, Z.; Chen, P.; Khan, T.U.; Lian, Y. Nondestructive Estimation of the Above-Ground Biomass of Multiple Tree Species in Boreal Forests of China Using Terrestrial Laser Scanning. *Forest* **2019**, *10*, 936. [CrossRef]
7. Yu, X.; Ge, H.; Lu, D.; Zhang, M.; Lai, Z.; Yao, R. Comparative Study on Variable Selection Approaches in Establishment of Remote Sensing Model for Forest Biomass Estimation. *Remote. Sens.* **2019**, *11*, 1437. [CrossRef]
8. Zarco-Tejada, P.J.; Hornero, A.; Hernández-Clemente, R.; Beck, P.S.A. Understanding the temporal dimension of the red-edge spectral region for forest decline detection using high-resolution hyperspectral and Sentinel-2a imagery. *ISPRS J. Photogramm. Remote Sens.* **2018**, *137*, 134–148. [CrossRef]
9. Jiang, F.; Kutia, M.; Ma, K.; Chen, S.; Long, J.; Sun, H. Estimating the aboveground biomass of coniferous forest in Northeast China using spectral variables, land surface temperature and soil moisture. *Sci. Total. Environ.* **2021**, *785*, 147335. [CrossRef]
10. Bordoloi, R.; Das, B.; Tripathi, O.; Sahoo, U.; Nath, A.; Deb, S.; Das, D.; Gupta, A.; Devi, N.; Charturvedi, S.; et al. Satellite based integrated approaches to modelling spatial carbon stock and carbon sequestration potential of different land uses of Northeast India. *Environ. Sustain. Indic.* **2021**, *13*, 100166. [CrossRef]
11. Huntington, T.; Cui, X.; Mishra, U.; Scown, C.D. Machine learning to predict biomass sorghum yields under future climate scenarios. *Biofuels Bioprod. Biorefining* **2020**, *14*, 566–577. [CrossRef]

12. Maddison, A.L.; Camargo-Rodriguez, A.; Scott, I.M.; Jones, C.M.; Elias, D.M.O.; Hawkins, S.; Massey, A.; Clifton-Brown, J.; McNamara, N.P.; Donnison, I.S.; et al. Predicting future biomass yield in *Miscanthus* using the carbohydrate metabolic profile as a biomarker. *GCB Bioenergy* **2017**, *9*, 1264–1278. [CrossRef]
13. Siami-Namini, S.; Tavakoli, N.; Namin, A.S. A Comparison of ARIMA and LSTM in Forecasting Time Series. In Proceedings of the 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), Orlando, FL, USA, 17–20 December 2018; IEEE: New York, NY, USA; pp. 1394–1401. [CrossRef]
14. NAhmed, N.K.; Atiya, A.F.; El Gayar, N.; El-Shishiny, H. An Empirical Comparison of Machine Learning Models for Time Series Forecasting. *Econ. Rev.* **2010**, *29*, 594–621. [CrossRef]
15. Hussain, D.; Hussain, T.; Khan, A.A.; Naqvi, S.A.A.; Jamil, A. A deep learning approach for hydrological time-series prediction: A case study of Gilgit river basin. *Earth Sci. Inform.* **2020**, *13*, 915–927. [CrossRef]
16. Atashi, V.; Gorji, H.T.; Shahabi, S.M.; Kardan, R.; Lim, Y.H. Water Level Forecasting Using Deep Learning Time-Series Analysis: A Case Study of Red River of the North. *Water* **2022**, *14*, 1971. [CrossRef]
17. Mahmoud, A.; Mohammed, A. A survey on deep learning for time-series forecasting. In *Machine Learning and Big Data Analytics Paradigms: Analysis, Applications and Challenges*; Hassanien, A.E., Darwish, A., Eds.; Springer International Publishing: Cham, Switzerland, 2021; Volume 77, pp. 365–392.
18. Lazzeri, F. *Machine Learning for Time Series Forecasting with Python*; John Wiley & Sons Inc.: Indianapolis, IN, USA, 2021.
19. Makridakis, S.; Spiliotis, E.; Assimakopoulos, V. Statistical and Machine Learning forecasting methods: Concerns and ways forward. *PLoS ONE* **2018**, *13*, e0194889. [CrossRef] [PubMed]
20. Cecaj, A.; Lippi, M.; Mamei, M.; Zambonelli, F. Comparing Deep Learning and Statistical Methods in Forecasting Crowd Distribution from Aggregated Mobile Phone Data. *Appl. Sci.* **2020**, *10*, 6580. [CrossRef]
21. Petropoulos, F.; Makridakis, S.; Assimakopoulos, V.; Nikolopoulos, K. 'Horses for Courses' in demand forecasting. *Eur. J. Oper. Res.* **2014**, *237*, 152–163. [CrossRef]
22. Liu, W.; Wang, Z.; Liu, X.; Zeng, N.; Liu, Y.; Alsaadi, F.E. A survey of deep neural network architectures and their applications. *Neurocomputing* **2017**, *234*, 11–26. [CrossRef]
23. White, H. Economic prediction using neural networks: The case of IBM daily stock returns. In Proceedings of the IEEE International Conference on Neural Networks, San Diego, CA, USA, 24–27 July 1988; IEEE: New York, NY, USA, 1988; Volume 2, pp. 451–458. [CrossRef]
24. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
25. Nakip, M.; Gül, B.C.; Rodoplu, V.; Güzeliş, C. Comparative Study of Forecasting Schemes for IoT Device Traffic in Machine-to-Machine Communication. In Proceedings of the 2019 4th International Conference on Cloud Computing and Internet of Things, Tokyo, Japan, 20–22 September 2019; ACM: New York, NY, USA; pp. 102–109. [CrossRef]
26. Zhou, K.; Wang, W.Y.; Hu, T.; Wu, C.H. Comparison of Time Series Forecasting Based on Statistical ARIMA Model and LSTM with Attention Mechanism. *J. Phys. Conf. Ser.* **2020**, *1631*, 012141. [CrossRef]
27. Hiransha, E.A.; Menon, G.; Krishna, V.; Soman, K.P. NSE Stock Market Prediction Using Deep-Learning Models. *Procedia Comput. Sci.* **2018**, *132*, 1351–1362. [CrossRef]
28. Cheung, L.; Wang, Y.; Lau, A.S.; Chan, R.M. Using a novel clustered 3D-CNN model for improving crop future price prediction. *Knowl. Based Syst.* **2023**, *260*, 110133. [CrossRef]
29. Istaiteh, O.; Owais, T.; Al-Madi, N.; Abu-Soud, S. Machine Learning Approaches for COVID-19 Forecasting. In Proceedings of the 2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA), Valencia, Spain, 19–22 October 2020; IEEE: New York, NY, USA; pp. 50–57. [CrossRef]
30. Zhang, L.; Cai, Y.; Huang, H.; Li, A.; Yang, L.; Zhou, C. A CNN-LSTM Model for Soil Organic Carbon Content Prediction with Long Time Series of MODIS-Based Phenological Variables. *Remote. Sens.* **2022**, *14*, 4441. [CrossRef]
31. Sina, L.B.; Secco, C.A.; Blazevic, M.; Nazemi, K. Hybrid Forecasting Methods—A Systematic Review. *Electronics* **2023**, *12*, 2019. [CrossRef]
32. Li, T.; Hua, M.; Wu, X. A Hybrid CNN-LSTM Model for Forecasting Particulate Matter (PM2.5). *IEEE Access* **2020**, *8*, 26933–26940. [CrossRef]
33. Dave, E.; Leonardo, A.; Jeanice, M.; Hanafiah, N. Forecasting Indonesia Exports using a Hybrid Model ARIMA-LSTM. *Procedia Comput. Sci.* **2021**, *179*, 480–487. [CrossRef]
34. Xu, D.; Zhang, Q.; Ding, Y.; Zhang, D. Application of a hybrid ARIMA-LSTM model based on the SPEI for drought forecasting. *Environ. Sci. Pollut. Res.* **2021**, *29*, 4128–4144. [CrossRef]
35. Wu, X.; Zhou, J.; Yu, H.; Liu, D.; Xie, K.; Chen, Y.; Hu, J.; Sun, H.; Xing, F. The Development of a Hybrid Wavelet-ARIMA-LSTM Model for Precipitation Amounts and Drought Analysis. *Atmosphere* **2021**, *12*, 74. [CrossRef]
36. Jayaprakash, A. *A Comparison of Deep Learning Methods for Time Series Forecasting with Limited Data*; Freie Universität Berlin: Berlin, Germany, 2022; Available online: https://www.wias-berlin.de/people/john/BETREUUNG/master_jayaprakash.pdf (accessed on 13 November 2023).
37. Cerqueira, V.; Torgo, L.; Soares, C. Machine Learning vs Statistical Methods for Time Series Forecasting: Size Matters. *arXiv* **2019**, arXiv:1909.13316.

38. Lu, Z.; Liu, N.; Xie, Y.; Xu, J. Time series analysis and forecasting of China's energy production during COVID-19: Statistical models vs machine learning models. *Res. Sq. Nov.* **2021**. [CrossRef]
39. Stuth, J.W.; Angerer, J. Livestock Early Warning System for Africa's Rangelands. In *Monitoring and Predicting Agricultural Drought*; Oxford University Press: Oxford, UK, 2005. [CrossRef]
40. Rhodes, E.C.; Tolleson, D.R.; Angerer, J.P. Modeling Herbaceous Biomass for Grazing and Fire Risk Management. *Land* **2022**, *11*, 1769. [CrossRef]
41. Bhanja, S.; Das, A. Impact of Data Normalization on Deep Neural Network for Time Series Forecasting. *arXiv* **2018**, arXiv:1812.05519.
42. Lu, Y.; Tian, Z.; Zhou, R.; Liu, W. Multi-step-ahead prediction of thermal load in regional energy system using deep learning method. *Energy Build.* **2020**, *233*, 110658. [CrossRef]
43. Van Rossum, G.; Drake, F.L. *Python 3 Reference Manual*; CreateSpace: Scotts Valley, CA, USA, 2009.
44. Ramchoun, H.; Idrissi, M.A.J.; Ghanou, Y.; Ettaouil, M. Multilayer Perceptron: Architecture Optimization and Training. *Int. J. Interact. Multimedia Artif. Intell.* **2016**, *4*, 26. [CrossRef]
45. Staudemeyer, R.C.; Morris, E.R. Understanding LSTM—A tutorial into Long Short-Term Memory Recurrent Neural Networks. *arXiv* **2019**, arXiv:1909.09586.
46. Olah, C. Understanding LSTM Networks. Available online: http://colah.github.io/posts/2015-08-Understanding-LSTMs (accessed on 29 January 2023).
47. Biswal, A. Recurrent Neural Network (RNN) Tutorial: Types, Examples, LSTM and More. Available online: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn (accessed on 29 January 2023).
48. Kostadinov, S. Understanding GRU Networks. Medium.com. Available online: https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be (accessed on 17 April 2023).
49. Saxena, S. Introduction to Gated Recurrent Unit (GRU). Analytics Vidhya. Available online: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-gated-recurrent-unit-gru/ (accessed on 17 April 2023).
50. Su, Y.; Kuo, C.-C.J. On extended long short-term memory and dependent bidirectional recurrent neural network. *Neurocomputing* **2019**, *356*, 151–161. [CrossRef]
51. Brownlee, J. Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python. Machine Learning Mastery, 2020. Available online: https://machinelearningmastery.com/deep-learning-for-time-series-forecasting/ (accessed on 20 July 2023).
52. Ghosh, A.; Sufian, F.; Sultana, A.; Chakrabarti, D. Fundamental concepts of convolutional neural network. In *Recent Trends and Advances in Artificial Intelligence and Internet of Things. Intelligent Systems Reference Library*; Springer: Cham, Switzerland, 2019; Volume 172. [CrossRef]
53. Fauvel, K.; Lin, T.; Masson, V.; Fromont, E.; Termier, A. XCM: An Explainable Convolutional Neural Network for Multivariate Time Series Classification. *Mathematics* **2021**, *9*, 3137. [CrossRef]
54. Elmaz, F.; Eyckerman, R.; Casteels, W.; Latré, S.; Hellinckx, P. CNN-LSTM architecture for predictive indoor temperature modeling. *J. Affect. Disord.* **2021**, *206*, 108327. [CrossRef]
55. Lu, W.; Li, J.; Li, Y.; Sun, A.; Wang, J. A CNN-LSTM-Based Model to Forecast Stock Prices. *Complexity* **2020**, *2020*, 6622927. [CrossRef]
56. Chang, X.; Gao, M.; Wang, Y.; Hou, X. Seasonal autoregressive integrated moving average model for precipitation time series. *J. Math. Stat.* **2012**, *8*, 500–505. [CrossRef]
57. Shah, V.; Shroff, G. Forecasting Market Prices Using DL with Data Augmentation and Meta-Learning: ARIMA Still Wins! Workshop at NeurIPS. 2021. Available online: https://openreview.net/pdf?id=udRAvWHIb2 (accessed on 25 April 2023).
58. Yamak, P.T.; Li, Y.; Gadosey, P.K. A Comparison between ARIMA, LSTM, and GRU for Time Series Forecasting. In Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence, Sanya, China, 20–22 December 2019; ACM: New York, NY, USA; pp. 49–55. [CrossRef]
59. Vabalas, A.; Gowen, E.; Poliakoff, E.; Casson, A.J. Machine learning algorithm validation with a limited sample size. *PLoS ONE* **2019**, *14*, e0224365. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.