energies

MDPI

Article

Crop Yield Estimation Using Deep Learning Based on Climate Big Data and Irrigation Scheduling

Khadijeh Alibabaei [1,2,*], Pedro D. Gaspar [1,2] and Tânia M. Lima [1,2]

[1] C-MAST Center for Mechanical and Aerospace Science and Technologies, University of Beira Interior, 6201-001 Covilhã, Portugal; dinis@ubi.pt (P.D.G.); tmlima@ubi.pt (T.M.L.)
[2] Deparment of Electromechanical Engineering, University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
* Correspondence: k.alibabaei@ubi.pt

Abstract: Deep learning has already been successfully used in the development of decision support systems in various domains. Therefore, there is an incentive to apply it in other important domains such as agriculture. Fertilizers, electricity, chemicals, human labor, and water are the components of total energy consumption in agriculture. Yield estimates are critical for food security, crop management, irrigation scheduling, and estimating labor requirements for harvesting and storage. Therefore, estimating product yield can reduce energy consumption. Two deep learning models, Long Short-Term Memory and Gated Recurrent Units, have been developed for the analysis of time-series data such as agricultural datasets. In this paper, the capabilities of these models and their extensions, called Bidirectional Long Short-Term Memory and Bidirectional Gated Recurrent Units, to predict end-of-season yields are investigated. The models use historical data, including climate data, irrigation scheduling, and soil water content, to estimate end-of-season yield. The application of this technique was tested for tomato and potato yields at a site in Portugal. The Bidirectional Long Short-Term memory outperformed the Gated Recurrent Units network, the Long Short-Term Memory, and the Bidirectional Gated Recurrent Units network on the validation dataset. The model was able to capture the nonlinear relationship between irrigation amount, climate data, and soil water content and predict yield with an MSE of 0.017 to 0.039. The performance of the Bidirectional Long Short-Term Memory in the test was compared with the most commonly used deep learning method, the Convolutional Neural Network, and machine learning methods including a Multi-Layer Perceptrons model and Random Forest Regression. The Bidirectional Long Short-Term Memory outperformed the other models with an R^2 score between 0.97 and 0.99. The results show that analyzing agricultural data with the Long Short-Term Memory model improves the performance of the model in terms of accuracy. The Convolutional Neural Network model achieved the second-best performance. Therefore, the deep learning model has a remarkable ability to predict the yield at the end of the season.

Keywords: agriculture; deep learning; LSTM; support decision-making algorithms; yield estimation; irrigation management

Citation: Alibabaei, K.; Gaspar, P.D.; Lima, T.M. Crop Yield Estimation Using Deep Learning Based on Climate Big Data and Irrigation Scheduling. *Energies* **2021**, *14*, 3004. https://doi.org/10.3390/en14113004

Academic Editor: Amparo López Jiménez

Received: 20 April 2021
Accepted: 20 May 2021
Published: 22 May 2021

1. Introduction

Agriculture is in a state of flux, and obstacles are emerging, such as climate change, environmental impacts, and lack of labor, resources, and land. Annual population growth and increasing demands on agricultural society to produce more from the same amount of agricultural land while protecting the environment are the significant challenges of this century [1]. This scenario reinforces the constant need to seek alternatives in the face of challenges to ensure higher productivity and better quality. Sustainable production of sufficient, safe, and high-quality agricultural products will be achievable if new technologies and innovations are adopted. Smart farms rely on data and information generated

by agricultural technology, bringing the producer closer to digital technology [1]. This includes the use of sensors and drones and the collection of accurate data such as weather data, soil mapping, and others. Extracting knowledge from these data and creating decision support systems is becoming increasingly important to optimize farms and add value to meet the food needs of the population and ensure the sustainable use of natural resources [1].

Deep learning (DL) is a subfield of machine learning. DL algorithms can be used throughout the cultivation and harvesting cycle in agriculture and are receiving considerable attention in developing such decision-making systems. The idea is to feed large artificial neural networks with increasingly large amounts of data, extract features from them automatically, and make decisions based on these data [2]. Deep here refers to the number of hidden layers of the neural network. The performance of the model improves as the network becomes deeper [2].

Crop yields and crop yield forecasts directly affect the annual national and international economy and play a major role in the food economy. Crop yields are highly dependent on irrigation and climate data. More irrigation does not necessarily increase yield [3], and therefore, optimization of irrigation and more efficient irrigation systems are critical. Predicting yield based on different types of irrigation is one way to optimize the process.

Machine learning algorithms are already used to estimate yield from images. Bargoti and Underwood [4] used Multi-Layer Perceptrons (MLP) and Convolutional Neural Network (CNN) models to extract features from input images, and then two image processing algorithms, Watershed Segmentation (WS) and Circular Hough Transform (CHT), were used to detect and count individual fruits in these features. They added metadata such as pixel position, row number, and sun azimuth to their algorithms and improved the detection performance. The best performance was obtained for fruit detection by CNN and WS with $R^2 = 0.826$. Habaragamuwa et al. [5] developed a Region-based CNN (R-CNN) model with AlexNet as the backbone and detected ripe and unripe strawberries in greenhouse images. The model achieved an average precision of 82.61%. Kang and Chen [6] implemented a clustering CNN model (C-RCNN) and a deep learning model, LedNet, to detect apples on trees. The C-RCNN module was used to generate a label for the training dataset, and LedNet was trained to detect apples on trees. A lightweight network (LW-Net), ResNet110, ResNet50, and Darknet-53 were used as the backbone. LedNet with the ResNet110 backbone, with 86% accuracy, and LedNet with LW-Net, with weight size and computation time of 7.4 M and 28 ms, respectively, outperformed the other models in terms of detection performance and computational efficiency. Koirala et al. [7] developed a DL model, named Mango-YOLO, based on YOLO-v3 and YOLO-v2 (tiny) for counting mangoes on trees. Mango-YOLO achieved the best performance in terms of memory consumption, speed, and accuracy compared to the Faster R-CNN, Single Shot multi-box Detector (SSD), and You Only Look Once (YOLO). Liang et al. [8] applied the SSD network to detect mango and almond on tree fruits. The SSD model with the data augmentation techniques and the smaller standard box was more accurate than the original SSD network in detecting mango on trees. Stein et al. [9] developed an FR-RCNN using VGG16 as a backbone for fruit detection and localization in a mango orchard. They used three datasets for training. The first contained the image of apple trees from one side of the trees, the second contained the image from both sides of the trees, and the third contained images from multiple views of the trees. Training the model with images from two and multiple views showed excellent performance ($R^2 \geq 0.90$). Tian et al. [10] developed YOLO-V3 with DenseNet as the backbone to detect apples on trees. They used two datasets for training. The first contained images of apples at one growth stage, and the second contained images taken at different growth stages. Their results showed that the F_1 score of the model trained with the first dataset was higher than that of the model trained with the second dataset. Apolo-Apolo et al. [11] used a Faster R-CNN model and a Long Short-Term Memory (LSTM) model to estimate fruit number and fruit size. An average

standard error (SE) of 6.59% between visual fruit count and fruit detection by the model was determined. An LSTM model was trained for per-tree yield estimation and total yield estimation. Actual and estimated yields per tree were compared, yielding an approximate error of SE = 4.53% and a standard deviation of SD = 0.97 kg. Maimaitijiang et al. [12] used Partial Least Squares Regression (PLSR), Random Forest Regression (RFR), Support Vector Regression (SVR), DNN (DNN-F1) based on input-level feature fusion, and DNN (DNN-F2) based on mid-level feature fusion to estimate soybean yeast. The results showed that multimodal data fusion improved the accuracy of yield prediction. DNN-F2 achieved the highest accuracy with an R^2 score of 0.720 and a relative root mean square error (RMSE) of 15.9%. Yang et al. [13] proposed a CNN architecture for predicting rice grain yield from low-altitude remote sensing images at the maturity stage. The proposed model consisted of two separate branches for processing RGB and multispectral images. In a large rice-growing region of Southern China, a 160-hectare area with over 800 cultivation units was selected to investigate the ability of the model to estimate rice grain yield. The network was trained with different datasets and compared with the traditional vegetation index-based method. The results showed that the CNNs trained with RGB and multispectral datasets performed much better than the VI-based regression model in estimating rice grain yield at the maturity stage. Chen et al. [14] proposed a faster Region-based Convolutional Neural Network (R-CNN) for detecting and counting the number of flowers, mature strawberries, and immature strawberries. The model achieved a mean average accuracy of 0.83 for all detected objects at 2 m height and 0.72 for all detected objects at 3 m height. Zhou et al. [15] implemented an SSD model with two lightweight backbones, MobileNetV2 and InceptionV3, to develop an Android app called KiwiDetector to detect kiwis in the field. The results showed that MobileNetV2, quantized MobileNetV2, InceptionV3, and quantized InceptionV3 achieved true detection rates of 90.8%, 89.7%, 87.6%, and 72.8%, respectively.

The disadvantages of estimating the yield from images are:

- Pictures of the entire field must be collected each year to identify the crop in the pictures and then estimate the yield.
- To train the model, a large number of labeled images is needed, which is very time-consuming.
- Illumination variance, foliage cover, overlapping fruits, shaded fruits, and scale variations affect the images [16].

Ma et al. [17] used climate, remote sensing data, and rice information to estimate rice yield. A Stacked Sparse Auto-Encoder (SSAE) was trained and achieved a percent mean square error of 33.09 kg $(10a)^{-1}$. Han et al. [18] implicated machine learning methods including Support Vector Machine (SVM), Gaussian Process Regression (GPR), Neural Network (NN), K-Nearest Neighbor Regression, Decision Tree (DT), and Random Forest (RF) to integrate climate data, remote sensing data, and soil data to predict winter wheat yield based on the Google Earth Engine platform (GEE). SVM, RF, and GPR with an $R^2 > 0.75$ were the three best yield prediction methods, among others. They also found that different agricultural zones and temporal training settings affected the prediction accuracy. Kim et al. [19] developed an optimized deep neural network for crop yield prediction using optimized input variables from satellite products and meteorological datasets. The input data were extracted from satellite-based vegetation indices and meteorological and hydrological data, and a matchup database was created on the Cropland Data Layer (CDL), a high-resolution map for classifying plant types. Using the optimized input dataset, they implemented six major machine learning models, including multivariate adaptive regression splines (MARS), SVM, RF, highly randomized trees (ERT), ANN, and DNN. The DNN model outperformed the other models in predicting corn and soybean yields, with a mean absolute error of 21–33% and 17–22%, respectively. Abbas et al. [20] used four machine learning algorithms, namely Linear Regression (LR), Elastic Net (EN), K-Nearest Neighbor (k-NN), and Support Vector Regression (SVR), to predict tuber yield of potato (Solanum tuberosum) from soil and plant trait data acquired by proximal sensing. Four

datasets were used to train the models. The SVR models outperformed all other models in each dataset with RMSE ranging from 4.62 to 6.60 t/ha. The performance of k-NN remained poor in three out of four datasets.

In these papers, however, the amount of irrigation was not considered as an input to the model. Yield is highly dependent on the amount of irrigation, and a change in the amount of water can make a big difference in the yield. Considering irrigation scheduling as an input to the model can help to create an intelligent system that selects the best irrigation schedule to save water consumption without affecting production. To optimize irrigation based on productivity, the irrigation amount must be considered as an input to the model.

In this work, Recurrent Neural Networks (RNN), including the LSTM model and Gated Recurrent Units (GRU) model and their extensions, Bidirectional LSTM (BLSTM) and Bidirectional GRU (BGRU), were implemented to estimate tomato yield based on climate data, irrigation amount, and water content in the soil profile. Agricultural datasets are time-series, and agricultural forecasting relies heavily on historical datasets. The advantage of RNN is its ability to process time-series data and make decisions for the future based on historical data. The proposed models predict the yield at the end of the season given historical data from the field such as temperature, wind speed, solar radiation, ETo, the water content in the soil profile, and irrigation scheduling during a season (the codes are available at the following links: https://github.com/falibabaei/tomato-yieldestimation/blob/main/main (accessed date 22 May 2021), https://github.com/falibabaei/potato-yield-estimation/tree/main (accessed date 22 May 2021)). The performance of the model was evaluated using the mean square error and R^2 score. In addition, the performance of these models was compared with a CNN, an MLP model, and a Random Forest Regression (RF). The advantages of the yield estimation model are:

- Using RNN models to extract features from past observations in the field and predict yield at the end of the season.
- Using climatic data collected in the field as input to the model, which is easier than using collected images from the field.
- Irrigation amount was used as input in the model, and it is shown that the model can capture the relationship between irrigation amount and yield at the end of the season.
- It is shown that the model can be used as part of an end-to-end irrigation decision-making system. This system can be trained to decide when and how much water to irrigate and maximize net return without wasting water.

2. Materials and Methods

2.1. Deep Learning Algorithms

Machine learning (ML) is a subfield of artificial intelligence that uses computer algorithms to transform raw data from the real world into valuable models. ML techniques include Support Vector Machines (SVM), Decision Trees, Bayesian learning, K-Means clustering, association rule learning, regression, neural networks, and many others [2].

Deep learning is a subfield of ML. The word "deep" refers to the number of hidden layers in DL algorithms, making them more complex than ML algorithms. Deep neural networks can learn the features from data with multiple hidden layers and solve more complex problems. Unlike ML methods, DL models automatically extract useful features from the raw data through training and do not require feature engineering. The training time of DL models is longer than that of ML methods, and they require a large amount of data to train, but when trained, they are more accurate and faster [21]. For these reasons, they have been widely used in recent years.

The most widely used algorithm of DL consists of the CNN model and Recurrent Neural Network. The CNN models are already used for classification, recognition, and localization. In 2012, AlexNet [22] won the LSVRC competition for classification. Sermanet et al. [23] showed that DL algorithms could be used for classification, recognition,

and localization and achieved excellent results. However, CNN models make predictions based on current input data and do not use past observations to make future decisions.

Unlike CNN, information in recurrent neural networks goes through a loop that allows the network to remember the previous outputs [24]. It enables the analysis of sequences and time series. RNN is commonly used for natural language processing and other sequences. A recurrent network can be thought of as multiple copies of the same network, each passing information to the next (see Figure 1).

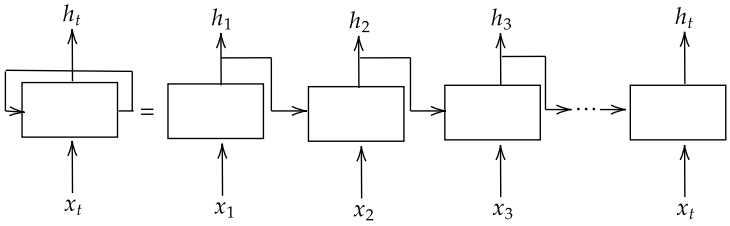

Figure 1. RNN looped and unfolded sequentially.

The output of an RNN unit is calculated by Equation (1).

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_t), \tag{1}$$

where h_{t-1} is the recurrent output from the previous step, x_t is the input at the current time, and W_x, W_h, b_t are the weights and bias of the network to be trained when training.

The problem with RNN is that if the sequential input is large, the gradient of the loss function can end at zero, effectively preventing the weights of the model from being updated [24].

2.1.1. LSTM and GRU Structures

The LSTM [25] and Gated Recurrent Units (GRU) [26] were developed to address the problems of the RNN.

The LSTM contains a forget gate that can be used to train individual neurons on what is important and how long it remains important. An ordinary LSTM unit consists of a block input z_t, an input gate i_t, a forget gate f_t, an output gate o_t, and a memory cell c_t (see Figure 2). The forget gate f_t is used to remove information that is no longer useful in the cell state using Equation (2). The input at a given time x_t and the previous cell output h_{t-1} are fed to the gate and multiplied by weight matrices, followed by the addition of the bias. The result is passed through a sigmoid function that returns a number between 0 and 1. If the output is 0, the information is forgotten for a given cell state; if the output is 1, the information is retained for future use. Adding useful information to the cell state is performed by the input gate i_t using Equation (3). First, the information is controlled by the sigmoid function, which filters the values to be stored, similar to the forget gate. Then, a vector of new candidate values of h_{t-1} and x_t is generated with the block gate z_t using Equation (4), which outputs from -1 to +1. The vector values and the controlled values are multiplied to obtain useful information using Equation (5). The output gate o_t decides which information in the cell is used to calculate the output of the LSTM unit using Equation (6).

$$f_t = \sigma(W_{fx} x_t + W_{fh} h_{t-1} + b_f), \tag{2}$$
$$i_t = \sigma(W_{ix} x_t + W_{ih} h_{t-1} + b_i), \tag{3}$$
$$z_t = \tanh(W_{zx} x_t + W_{zh} h_{t-1} + b_z), \tag{4}$$
$$c_t = f_t * c_{t-1} + i_t * z_t. \tag{5}$$
$$o_t = \sigma(W_{ox} x_t + W_{oh} h_{t-1} + b_o) \tag{6}$$

First, a vector is created by applying the tanh function to the cell. Then, the information is regularized using the sigmoid function, which filters the values to be stored based

on the inputs h_{t-1} and x_t. The vector values and the regulated values are multiplied by Equation (7) to be sent as output and input to the next cell.

$$h_t = o_t * \tanh(c_t). \tag{7}$$

GRUs discard the cell state and use the hidden state to transmit information. This architecture contains only two gates: the update gate z_t and the reset gate r_t. Like LSTM gates, GRU gates are trained to selectively filter out all irrelevant information while preserving the useful information and can be calibrated using Equations (8)–(11):

$$z_t = \sigma(W_{ux}x_t + W_{uh}h_{t-1} + b_u), \tag{8}$$
$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r), \tag{9}$$
$$o_t = \tanh(W_{ox}x_t + W_{oh}(r_t * h_{t-1}) + b_o). \tag{10}$$
$$h_t = (1 - z_t) * o_t + z_t * h_{t-1}. \tag{11}$$

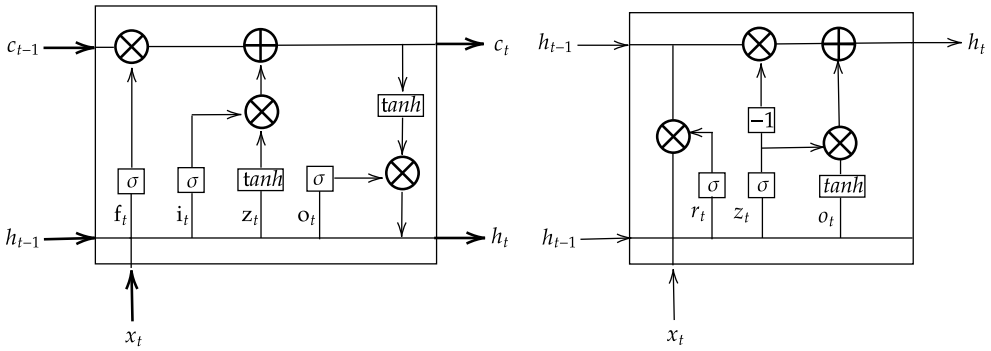

Figure 2. LSTM and GRU structures. Left side shows LSTM cell, right side shows GRU cell.

The functions tanh and σ add nonlinearity to the network. These functions allow the model to capture the nonlinear relationships between the inputs and outputs of the model. At the beginning of training, the weights and biases in Equations (2)–(4), (6) and (8)–(10) are set randomly. During training, the model tries to set the weights and biases in such a way that the loss function is minimized. Therefore, the algorithm of an RNN model is an optimization problem.

The LSTM and GRU models have similar units, but they also differ. For example, in the LSTM unit, the amount of memory content seen or used by other units in the network is controlled by the output gate. In contrast, the GRU releases all of its content without any control [27]. From these similarities and differences alone, it is difficult to conclude which model performs better on one problem than another. In this paper, both models were implemented to see which model performs better on the yield estimation problem.

2.1.2. Bidirectional LSTM Structure

Bidirectional LSTM (BLSTM) is an extension of the LSTM model that can improve the results [28]. Its architecture consists of a stack of two separate intermediate LSTM layers that send a sequence forward and backward to the same output layer, using contextual information from both sides of the sequence (see Figure 3).

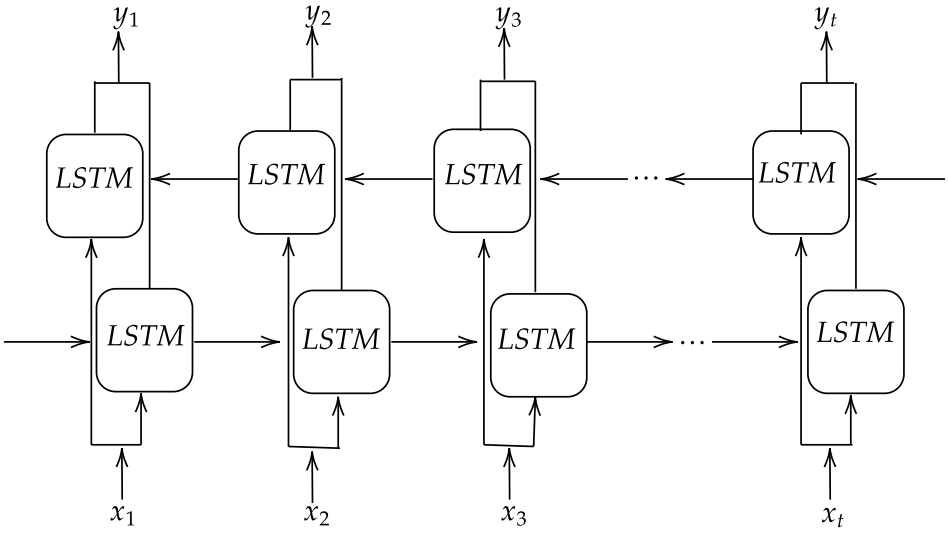

Figure 3. BLSTM layer.

2.1.3. Dropout and Early Stopping

Overfitting occurs when the model learns noise in the training data, and the generalization of the model to the unseen data is unreliable and cannot make accurate estimates. It is proposed to use regularization techniques to prevent overfitting. One of them is "dropout", which consists of randomly selecting some neurons of the hidden layer and blocking their output so that they are not evaluated in the learning algorithms, and then, after a while, releasing the outputs of the blocked neurons and blocking other neurons [29]. This leads to the neural network becoming more general and not depending only on one group of neurons to make certain decisions. In LSTM models, the dropout can be added to the inputs of the layers, the outputs of the layers, and the recurrent outputs [29]. The dropout size is a hyperparameter that should be set during training (see Figure 4). Table 1 shows the effect of the dropout on the validation loss.

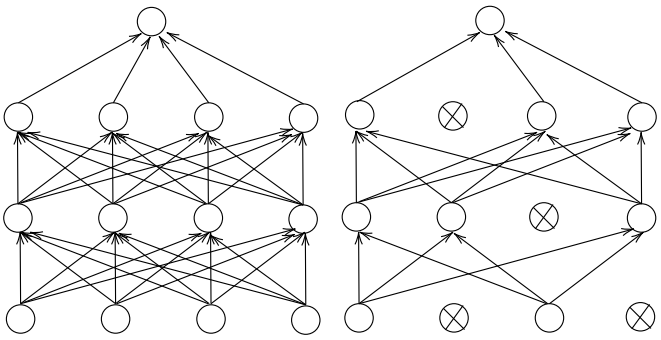

Figure 4. Left side shows the normal neural network and the right side shows the network with dropout.

Table 1. Validation lost under different dropout sizes.

Dropout Size	**Dropout Size**				
	0	**0.1**	**0.2**	**0.3**	**0.4**
Tomato	0.00021	0.00019	0.00033	0.00050	0.00068
Potato	0.00140	0.00109	0.00101	0.00092	0.00185

Another solution is early stopping, which consists of splitting the dataset into three sets, one for training, one for validation, and one test set [30]. In this method, the validation

loss is constantly evaluated in each episode, and if the validation loss does not improve for a certain number of episodes, the training is stopped. This technique does not allow the network to be very specific about the training set.

In this work, dropout and early stopping were used to prevent overfitting.

2.2. Data Collection

The climate big data were collected by an agricultural weather station for a site in Portugal. They were retrieved from the government agency of the Ministries of Agriculture and the Sea, the Direção Regional de Agricultura e Pescas do Centro, Portugal (www.drapc.gov.pt (accessed date 22 May 2021)). The soil type in Fadagosa is either sandy or sandy loam, permeable, with low to medium organic matter content, with low acid to neutral reaction, rich in phosphorus and potassium, and without salt effects. The climate type of Fadagosa is the Mediterranean hot summer climate (Csa). These are areas with mild winters, with precipitation falling mainly in autumn and winter and occasionally in spring. Summers are hot and dry, with maximum temperatures above 40 °C. Figure 5 shows the Fadagosa region from Google Earth and Table 2 shows location details.

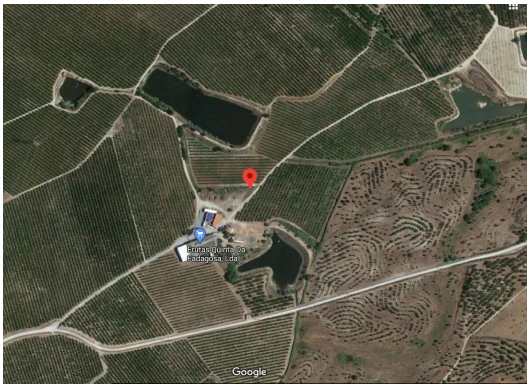

Figure 5. Map of Fadagosa region.

Table 2. Details of the location.

Location	Longitude	Latitude	Start Date	End Date	Temporal Resolution
Fadagosa	40°1′46.55″ N	7°26′36.27″ W	1 January 2010	23 March 2020	15-min

The big data included minimum, maximum, and average temperature; minimum, maximum, and average relative humidity; average solar radiation; minimum and average wind speed; and precipitation. They were recorded every 15 mins and converted to the daily variable in the data preprocessing section. Table 3 shows the details of these variables. Figure 6 shows the daily variables from 2010 to 2019 (10 years of data).

Table 3. Dataset details.

Variables	Unit	Data Source	Max	Min	Mean	SD
T_{Min}	°C	DRAP-Centro	27	−4.7	9.76	5.63
T_{Max}	°C	DRAP-Centro	42.7	1.8	21.84	8.42
T_{Avg}	°C	DRAP-Centro	34.84	−0.12	15.68	6.90
HR_{Min}	%	DRAP-Centro	95	0	38.72	20.20
HR_{Max}	%	DRAP-Centro	97	24	81.29	15.05
HR_{Avg}	%	DRAP-Centro	95.89	27.75	60.38	18.78
SR_{Avg}	wm^{-2}	DRAP-Centro	346.66	6.35	172.02	89.25
WS_{Max}	ms^{-1}	DRAP-Centro	86.5	3.5	24.67	10.61
WS_{Avg}	ms^{-1}	DRAP-Centro	28.85	0.031	4.62	3.80
$Prec$	mm	DRAP-Centro	101.6	0	2.28	7.20
ETo	mmd^{-1}	Penman-Monteith equation (AquaCrop)	9.8	0.2	3.68	2.088

Table 3. *Cont.*

Variables	Unit	Data Source	Max	Min	Mean	SD
WCTot (Tomato)	mm	Aquacrop	365.2	145.9	247.96	36.18
WCTot (Potato)	mm	Aquacrop	432.4	165.6	311.62	51.95
Potato Yield	ton $(ha)^{-1}$	Aquacrop	12.706	5.539	10.8	2.053
Tomato Yield	ton $(ha)^{-1}$	Aquacrop	8.482	3.434	7	1.39

The abbreviations stand for the following: Max: maximum, Min: minimum, SD: standard deviation, Avg: average, T: temperature, HR: relative humidity, SR: Solar Radiation, WS: Wind Speed, Prec: precipitation, ETo: Reference Evaporation, WCTot: water content in the total soil profile.

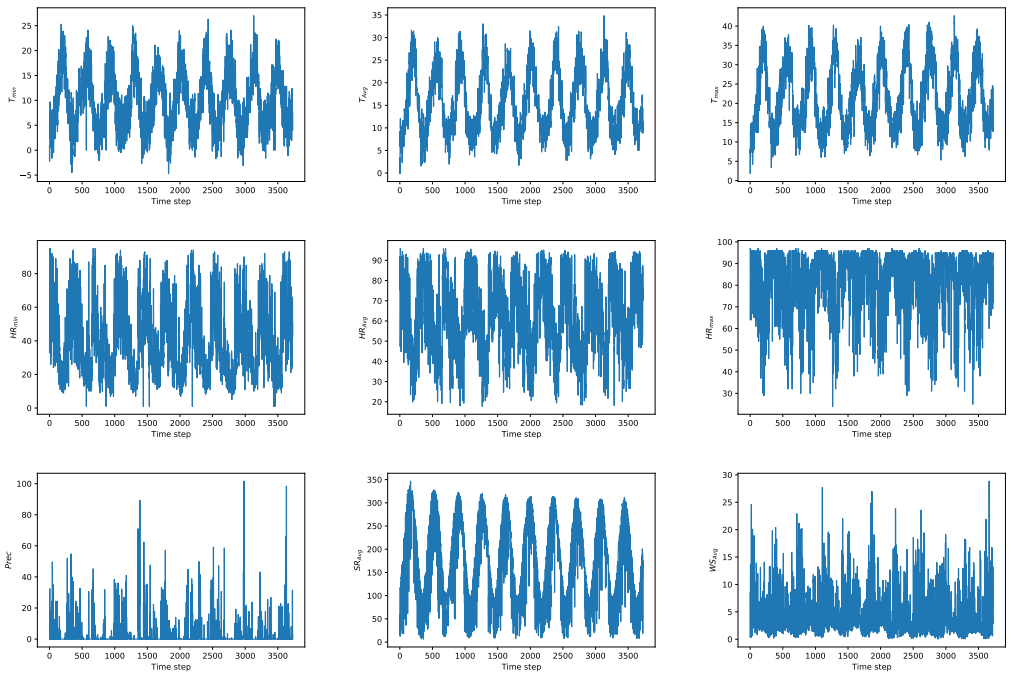

Figure 6. Fadagosa dataset.

To train a deep learning model on a dataset, a sufficient amount of data is needed. However, in reality, recording crop yields at the end of the season based on different irrigation schedules is extremely slow and sometimes impossible (e.g., for a season without irrigation). The simulation model package Aquacrop was used to overcome this problem. AquaCrop is a crop growth model developed by the Food and Agriculture Organization (FAO) to ensure food security and assess the impact of environmental and management influences on crop production [31]. The structure of the model was designed to be applicable across different locations, climates, and seasons. To achieve this goal, AquaCrop distinguishes between conservative (fixed) and non-conservative (case-specific) parameters. The conservative parameters do not change with geographic location, crop type, management practices, or time and are intended to be determined with data from favorable and non-stressful conditions but remain applicable under stressful conditions by modulating their stress response functions [31]. The Aquacrop model was calibrated with climate data collected over the past ten years at Fadagosa. The crops selected for model calibration were tomato and potato. In the crop menu of Aguacrop, a planting/sowing date is generated by automatically evaluating the rainfall or temperature data prior to seeding. The temperature criterion was selected to determine the planting/sowing date. April 7 was generated for tomato and February 17 for potato. The experiments were conducted on sandy loam soil typical of Fadagosa with no salinity. The irrigation method chosen was sprinkler irrigation, which is adjustable in Aquacrop. A fixed interval of four days and a fixed value between 0 and 60 mm were used as time and depth criteria for calculating the irrigation regimes. Different irrigation treatments were applied in each experimental

year, including no irrigation and a fixed irrigation depth of 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60 mm every four days or when the allowable depletion reached the thresholds of 90%, 80%, and 70%, respectively. Other parameters were kept unchanged. Figure 7 shows tomato yield under the fixed irrigation depth of 20 mm and water content throughout the soil profile from 2010 to 2018. As can be seen in the figure, the yield varies under different climatic conditions.

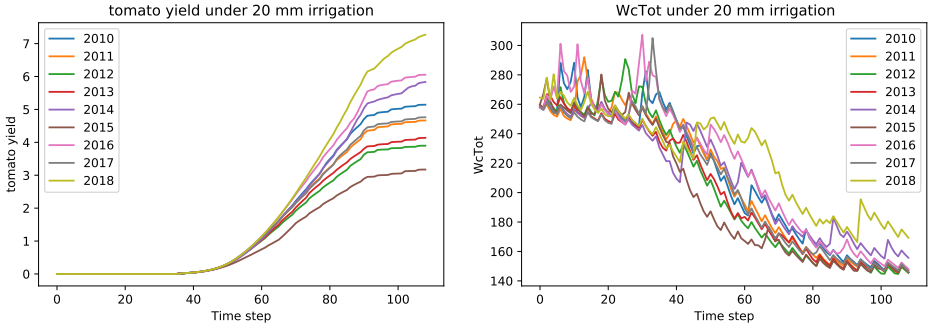

Figure 7. Evolution of tomato yield and total water content profile under fixed irrigation depth of 20 mm. Left side shows tomato yield and right side shows WCTot.

Evapotranspiration is the amount of water that evaporates from the Earth, and soil water content is the volume of water per unit volume of soil. Evapotranspiration (ETo) and water content in the total soil profile (WCTot) were also simulated during the simulation and used as inputs to the models. AguaCrop estimated ETo from meteorological data using the FAO Penman–Monteith Equation (12) [32].

$$ETo = \frac{\Delta(R_n - G) + \rho_a c_p \frac{(e_s - e_a)}{r_s}}{\Delta + \gamma(1 + \frac{r_s}{r_a})} \quad (12)$$

where R_n is the net radiation, G is the soil heat flux, $(e_s - e_a)$ represents the vapor pressure deficit of the air, ρ is the mean air density at constant pressure, c_p is the specific heat of the air, Δ represents the slope of the saturation vapor pressure temperature relationship, γ is the psychrometric constant, and r_s and r_a are the (bulk) surface and aerodynamic resistances. Figure 8 shows the ETo from 2010 to 2019.

The advantage of using deep learning models is that they do not require manual adjustments once the model is trained and can be used automatically. These models can be used to create an end-to-end decision support system for irrigation scheduling.

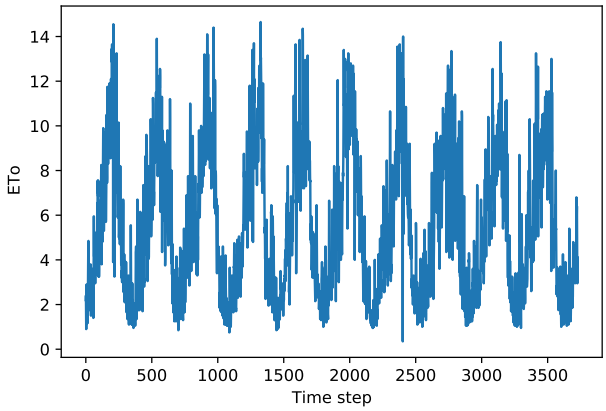

Figure 8. ETo calculated by Aquacrop.

2.3. Data Preprocessing

The quality of the features extracted from the data by a deep learning model is determined by the quality of the dataset provided as input. The actions performed in the preprocessing phase aim to prepare the data so that the feature extraction phase is more effective. In this work, the missing big data were filled using the moving average method [33].

The other preprocessing in this paper involves the normalization and removal of multicollinear parameters. The multicollinear parameters contain the same information about the variables and can lead to redundancy in the calculations. One way to measure multicollinearity is the Variance Inflation Factor (VIF), which assesses how much the variance of an estimated regression coefficient increases when its predictors are correlated. If the VIF is equal to 1, there is no multicollinearity between factors, but the predictors may be moderately correlated if the VIF is greater than 1. A VIF between 5 and 10 indicates a high correlation, which may be problematic [34]. If the VIF is greater than 10, it can be assumed that the regression coefficients are underestimated due to multicollinearity. The VIF was used to determine the multicollinear variables, and the variables with a VIF greater than five were removed. Finally, average temperature (T_{Avg}), average humidity (HR_{Avg}), average wind speed (WS_{Avg}), reference evapotranspiration (ETo), water content in total soil profile (WCTot), irrigation, and precipitation ($Prec$) were used as inputs.

The normalization aims to bring the values of the dataset into a normal form without distorting the differences in the ranges of values. In this work, min–max normalization was used. For each feature in the dataset, the minimum value of that feature is converted to 0, the maximum value is converted to 1, and every other value is converted to a decimal number between 0 and 1 using Equation (13).

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}} \tag{13}$$

where x_{min} and x_{max} are the minimum and maximum of each variable in the dataset.

2.4. Metric Evaluation

In regression problems, the evaluation of the model can be calculated by the distance between the actual value and the value predicted by the model. In this work, the mean square error (MSE) and R^2 score were used to evaluate the model's performance. The MSE calculates the variance explained by the model, and the R^2 score is a statistical measure that calculates the variance explained by the model over the total variance [35]. The higher the R^2 score, the smaller the differences between the observed data and the fitted values [36]. MSE and R^2 score can be calculated using Equations (14) and (15):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{14}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}, \tag{15}$$

where y_i, $\hat{y}_i$, and $\bar{y}_i$ are, respectively, the true value, the value determined by the model, and the mean of the true values.

3. Results and Discussion

The predictions were made using a computing system consisting of an Intel Core i7-8550 CPU, an NVIDIA GEFORCE MX130 graphics card, and 8.0 GB of RAM. Anaconda is a distribution of the Python and R programming languages for scientific computing that aims to simplify package management and deployment. TensorFlow is an open-source software library for numerical applications with computational graphs. It was developed directly by Google Brain Team for machine learning and deep neural networks. Keras is a deep neural network library written in Python and running as a front-end

in TensorFlow or Theano. It was developed to enable rapid experimentation [37,38]. The Anaconda environment was used to implement the model using Python version 3.7.9 and the Tensorflow GPU 2.1.0 framework, and Keras library version 2.2.4-tf.

The model obtained daily historical data for a season, including average temperature (T_{Avg}), average humidity (HR_{Avg}), average wind speed (WS_{Avg}), reference evapotranspiration (ETo), total soil profile (WCTot), irrigation amount, and precipitation ($Prec$), and estimated the yield at the end of the season. The tomato season was a string of length 110 days and the potato season was a string of length 121 days.

The data were divided into a training set, a validation set, and a test set. Training and test datasets are used to train the model and evaluate the performance of the trained network, respectively. The test set is ignored during training and hyperparameter selection, and its errors determine how well the model generalizes to the unseen data. Validation data are used to compare different models, select hyperparameters, and prevent overfitting during the training phase. Here, 30% of the data were selected as a test set, and the rest were split into 60–10% as training and validation sets, respectively. The tomato prediction model was trained on 77 samples (season) and tested on 34 samples, and the potato crop prediction model was trained on 45 samples and tested on 20 samples.

The error of a model is defined as the difference between the actual value and the value predicted by the model and must be minimized in the training step. The function used to calculate this error is called the loss function. Training of DL models is based on backpropagation [39]. Backpropagation uses the chain rule and computes the gradient of the loss function with respect to the weights of the network for a single input–output example, the adjusted weights of the network, and the biases. The optimization algorithm must be chosen to minimize the loss function. The choice of optimization algorithm and loss function is crucial. In this paper, the mean square error and Adam optimizer were used [2].

When training a neural network, some specific values must be initialized, which are called hyperparameters. Hyperparameters such as the number of hidden layers, activation function, etc., determine the structure of a model, and hyperparameters such as learning rate, decay time, etc., determine how the model is trained [24]. Results from multiple simulations of the same algorithm with different hyperparameters will vary. The hyperparameters considered in this work are the number of layers, the number of hidden units, batch size, the dropout size, the learning rate, and the learning rate decay. The learning rate is a hyperparameter that controls how much the weights of the model are adjusted with respect to the gradient of the loss function. It has a significant impact on the training process of the deep learning models. A very low learning rate makes the learning of the network very slow, while a very high learning rate causes fluctuations in training and prevents the convergence of the learning process [24]. Learning rate decay is another hyperparameter where the learning rate is calculated by decay at each update.

In this paper, the hyperparameters were selected manually. Unlike neural networks, the recurrent network does not require a deep network to obtain accurate results [40]. Changing the number of nodes per layer has been shown to have a more significant impact on results than changing the number of layers [40]. Since choosing LSTM and GRU with four layers and BLTM with three layers significantly increases the number of trainable parameters, leading to overfitting of the models, the number of LSTM and GRU layers was kept below four and the number of BLSTM and BGRU layers below three. Moreover, the LSTM and GRU with one layer showed poor performance and were excluded from the experimental results. Finally, the network architectures with two and three LSTM and GRU layers and one and two BLSTM and BGRU layers were tested.

Due to the hardware-based reasoning, the number of nodes per layer is usually chosen as a power of two, which can speed up the training of the model [41]. Since the time step in the input of the model is more than 110 days, the number of nodes per layer was kept less than or equal to 512 to avoid overfitting due to the number of trainable parameters and to make the training faster and more efficient [24]. The models with 16 and 32 nodes

also performed very poorly and were excluded from the experimental results. In the end, the network architectures were tested with 64, 128, 256, and 512 nodes per layer.

The MSE was calculated for the validation dataset with all combinations of the number of layers and the number of nodes. The results for the different crops are shown in Table 4. As can be seen in Table 4, the BLSTM model improves the validation loss, but the GRU model has worse performance compared to the LSTM model. Therefore, removing the cell state from the LSTM reduces the performance of the model in the yield prediction problem.

Table 4. Validation MSE for different numbers of LSTM layers and LSTM nodes per layer.

Model Architecture	Crop	Number of Nodes per Layer			
		64	**128**	**256**	**512**
LSTM-2 layers	Potato	0.00447	0.00561	0.00627	0.00132
	Tomato	0.00389	0.00248	0.00119	0.00023
LSTM-3 layers	Potato	0.00497	0.00276	0.00140	0.00259
	Tomato	0.00275	0.00333	0.00280	0.00226
GRU-2 layers	Potato	0.00962	0.00213	0.00943	0.01067
	Tomato	0.00456	0.00150	0.00048	0.00022
GRU-3 layers	Potato	0.01170	0.01360	0.00968	0.00974
	Tomato	0.00258	0.00118	0.00075	0.00030
BLSTM-1 layer	Potato	0.00357	0.00461	0.00808	0.00110
	Tomato	0.00814	0.00046	0.00028	0.00017
BGRU-1 layer	Potato	0.01122	0.01058	0.0103	0.01250
	Tomato	0.00793	0.00084	0.00020	0.00029
BLSTM-2 layers	Potato	0.00914	0.00092	0.00167	0.00257
	Tomato	0.00329	0.00035	0.00012	0.00008
BGRU-2 layers	Potato	0.01610	0.00121	0.01183	0.01108
	Tomato	0.00249	0.00050	0.00029	0.00022

Finally, a two-layer BLSTM with 128 nodes and a two-layer BLSTM with 512 nodes were selected to predict potato and tomato yield, respectively. Tables 5 and 6 show the architecture of the models selected for crop yield prediction. Adding an additional dense layer after the BLSTM layers improved the results. The tanh function was used as an activation function after each BLSTM layer to capture the nonlinear relationship between input and output [24].

Table 5. BLSTM architecture used to predict tomato yield.

Layer (Type)	Output Shape	Param
Input	(Batch Size, 110, 7)	0
BLSTM	(Batch Size, 110, 1024)	2,134,016
Dropout	(Batch Size, 110, 1024)	0
BLSTM	(Batch Size, 1024)	6,295,552
Dropout	(Batch Size, 1024)	0
Dense	(Batch Size, 512)	524,800
Dense	(Batch Size, 1)	513

The batch size is also chosen as a power of two due to hardware reasons. The number of samples in the training datasets is less than 77, so the batch size is selected from $\{16, 32, 64\}$. For simplicity, the learning rate and decay time were chosen as negative powers of ten. With a learning rate and decay of 10^{-5}, the model trains very slowly, and even after 500 epochs, the validation loss is very high, and the learning rate and decay of 10^{-2} causes fluctuations in training. Therefore, the learning rate and decay are kept below 10^{-6} and above 10^{-2}. Table 7 shows the loss on the validation set for different crops when the hyperparameters are chosen differently. As can be seen, the same model with

different hyperparameters achieves different results. The models with a learning rate of 10^{-4} and 10^{-3} (potato and tomato, respectively), batch size of 64, and decay of 10^{-5} had the best performance.

Table 6. BLSTM architecture used to predict potato yield.

Layer (Type)	Output Shape	Param
Input	(Batch Size, 121, 7)	0
BLSTM	(Batch Size, 121, 256)	139,264
Dropout	(Batch Size, 121, 256)	0
BLSTM	(Batch Size, 512)	394,240
Dropout	(Batch Size, 512)	0
Dense	(Batch Size, 512)	131,584
Dense	(Batch Size, 1)	513

Dropout size is the percentage of nodes that randomly drop out during training. The dropout size of 0.4 did not improve the validation loss, so it was chosen to be less than or equal to 0.4 and from the set $\{0.1, 0.2, 0.3, 0.4\}$. In LSTM models, dropout can be added to the input layer, outputs, and recurrent outputs [29]. Adding dropout on the recurrent outputs with dropout size 0.1 for the potato yield estimation model and on the outputs of the layers with dropout size 0.3 for the tomato yield estimation model improved the validation loss and was therefore selected for each model (see Table 1).

Table 7. Validation lost under different hyperparameters.

		Batch Size	
Batch Size	16	32	64
Tomato	0.00012	0.00020	0.00008
Potato	0.00164	0.00393	0.00092
		Learning Rate	
Learning Rate	10^{-3}	10^{-4}	10^{-5}
Tomato	0.00005	0.00008	0.00571
Potato	0.00193	0.00092	0.02338
		Decay	
Decay	10^{-3}	10^{-4}	10^{-5}
Tomato	0.00006	0.00007	0.00005
Potato	0.00140	0.00107	0.00092

Each model was trained for a maximum of 500 epochs, and each epoch lasted two seconds. As mentioned earlier, early stopping was used to prevent overfitting. In this method, training is stopped when the validation loss does not improve for a certain number of epochs. Patience is a hyperparameter in the early stopping method that controls the number of epochs in which the validation loss no longer improves [30]. The exact amount of patience varies by model and problem. Examining plots of model performance measures can be used to determine patience. In this work, by examining the plot, patience was determined to be 30 and 50 for the tomato and potato models, respectively. As shown in Figure 9, training of the tomato and potato yield prediction models was completed after 360 and 250 epochs, respectively. The tomato yield prediction model was trained with more samples in the training set due to larger availability of experimental data, which may result in the training of this model being more stable than that of the potato yield prediction model.

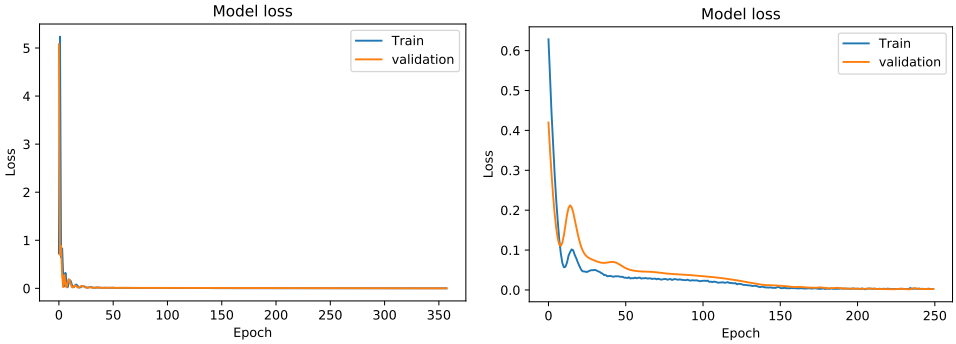

Figure 9. Model loss during training. The left side shows the loss of tomato yield prediction model and the right side shows potato yield prediction model.

Since the test dataset was randomly selected, there were different seasons with different amounts of irrigation in each test dataset. Table 8 shows the performance of the models on the test dataset, and Figure 10 shows the actual value of yield compared to the values predicted by the models. The model predicting tomato yield with an MSE of 0.017 performed better on the test dataset than the model predicting potato yield with an MSE of 0.039. This result could be due to the fact that the standard deviation of tomato yield is smaller than the standard deviation of potato yield (see Table 3) and also, as mentioned earlier, the model used to estimate tomato yield was trained with a larger training set.

As shown in Figure 10, the tomato test dataset included the 2010 season under four different irrigation levels. The irrigation amounts were 0, 10, 20, and 60, and the model was able to achieve an MSE of 0.02 in this season. The same result was true for the potato crop estimation model, and the model achieved an MSE of 0.09 to predict the 2012 crop under four irrigation amounts. These results show that the model not only captures the relationship between climate data and yield but also can accurately predict yield under different irrigation amounts in a season.

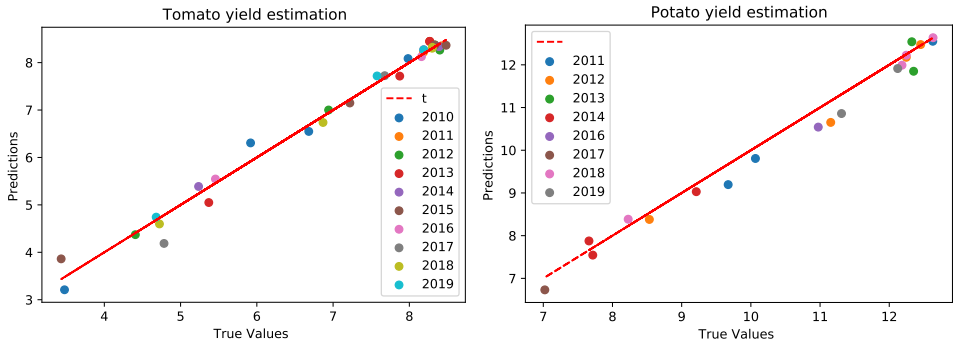

Figure 10. Predicted values vs. true values.

The ability to hold water depends on the soil type. As mentioned in the Data Collection section, the soil type of Fadagosa is sandy loam, and these models are designed for a sandy loam soil type. Therefore, these models work well on this soil type. Moreover, the models were trained with simulated data. However, the combination of simulated data and real data may give better results.

The performance of BLSTM models was compared with CNN, MLP, and the traditional machine learning algorithm Random Forest (RF).

MLP is a computational model inspired by the human nervous system. It is able to detect patterns in a mass of data in order to categorize it or regress a value. An MLP model consists of an input layer, a stack of fully connected layers (hidden layers), and an output layer. The fully connected layers connect every neuron in one layer to every neuron in the next layer. Mathematically, these layers are a linear function where each layer takes the output of the previous layer and adds weights and biases to it. To add nonlinearity to the model,

the activation function (e.g., ReLU, tanh) is used after each fully connected layer [42]. Again, to avoid overfitting, the number of nodes in each layer of the MLP model was kept below 513 nodes, and the model with 512 neurons achieved the best performance in each dataset. An MLP with 512 neurons and a different number of layers was implemented. The performance improvement of the MLP model with five layers stopped on both datasets, so the number of layers was kept below 5. Table 9 shows the performance of these models. The models with three layers and four layers achieved the best performance in predicting tomato and potato yields, with R^2 scores of 0.89 and 0.71, respectively.

CNN is a deep learning algorithm. A CNN model consists of a stack of convolutional layers, nonlinear activation functions, pooling layers (e.g., maximum pooling, average pooling), Flatten layers, and fully connected layers [24]. Similar to the fully connected layers, a convolutional layer is a linear function, and it contains a set of kernels. A kernel is a matrix used for a matrix multiplication operation. This operation is applied multiple times in different input regions, and extracts feature maps from the input. Pooling is a reduction process. It is a simple process to reduce the dimension of feature maps and hence the number of parameters trained by the network. A Flatten layer is usually used in splitting convolutional layers and the fully connected layers. It basically performs a transformation in the output of the convolutional layer and changes its format to an array. The fully connected layer takes the feature maps from the Flatten layer and applies weights and biases to predict the correct label or regress a value [24]. The number of kernels in each convolutional layer and the number of convolutional layers was chosen manually. For the same hardware reason, the number of kernels in each convolutional layer is kept as a power of two. The models with a number of four convolutional layers or more than 512 kernels in each layer start to overfit. Therefore, the number of kernels and layers is kept below 513 and four, respectively. All combinations of the number of layers and kernels were implemented. The CNN model with 512 kernels and a number of layers of 2 and 3 achieved better validation loss. The batch size, learning rate, and decay time from the BLSTM model were used for the CNN models. Padding was used in each convolutional layer to ensure that the output had the same shape as the input. The kernel size is usually chosen as an odd number. The model with a kernel size greater than 11 starts with overfitting, and below three, the model performance was poor, so the kernel size was chosen from $\{5, 7, 11\}$. Table 9 shows the architecture of the CNN model.

Table 8. Performance of the models on the test set. L shows the number of layers, and K shows the kernel size.

Model Architecture	Crops	MSE	R^2 Score	Trainable Parameters
BLSTM	Potato	0.039	0.988	665,601
	Tomato	0.017	0.99	6,853,633
CNN (2L-5k)	Potato	0.791	0.792	17,061,889
	Tomato	0.099	0.910	15,751,169
CNN (2L-7k)	Potato	0.499	0.882	17,594,369
	Tomato	0.087	0.925	16,283,649
CNN (2L-11k)	Potato	0.443	0.915	18,659,329
	Tomato	0.102	0.912	17,348,609
CNN (3L-5k)	Potato	0.615	0.792	18,373,121
	Tomato	0.075	0.934	17,062,401
CNN (3L-7k)	Potato	0.965	0.600	19,429,889
	Tomato	0.169	0.854	18,119,169
CNN (3L-11k)	Potato	0.515	0.860	21,543,425
	Tomato	0.141	0.878	20,232,705
MLP (3L)	Potato	0.870	0.601	1,021,953
	Tomato	0.126	0.891	976,897
MLP (4L)	Potato	0.643	0.711	1,284,609
	Tomato	0.133	0.884	1,239,553
MLP (5L)	Potato	1.270	0.428	1,547,265
	Tomato	0.143	0.8765	1,502,209
RF	Potato	0.5	0.872	-
	Tomato	0.107	0.901	-

Table 9. CNN architecture.

Layer (Type)	Output Shape
Input	(Batch size, time-steps, number of features)
(Conv1D) (kernel-size)	(Batch size, time-steps, 512)
Dropout	(Batch size, time-steps, 512)
Average-pooling (pool-size = 2)	(Batch size, (time-steps/pool-size), 512)
Flatten	(Batch size, (time-steps/pool-size) × 512)
Dense	(Batch size, 512)
Dropout	(Batch size, 512)
Dense	(Batch size, 1)

The CNN with two layers and three layers and a kernel size of 5 and 11 (tomato and potato, respectively), with an R^2 score of 0.96 and 0.933, performed better in predicting yield than models with other combinations of the number of layers and kernel sizes.

RF is one of the most powerful machine learning methods. The Random Forest consists of several Decision Trees. Each individual tree is a very simple model that has branches, nodes where a condition is verified, and if it is satisfied, the flow goes through one branch, otherwise through the other, always to the next node until the tree is finished. As Table 8 shows, the RF model outperformed the MLP model in predicting yield, with an R^2 score of 0.87 to 0.90.

Computation time in the training process for each epoch lasted two, one, and less than one second for BLSTM, CNN, and MLP, respectively. Although the computation time in training BLSTM was higher than the other models, BLSTM achieved the best accuracy in the test dataset.

As mentioned earlier, one of the applications of this model is to create a decision-making system that decides when and how much to irrigate to avoid wasting water without affecting productivity. The yield prediction model is used to calculate the net yield at the end of the season. The net return in agriculture is calculated using Equation (16).

$$R = Y * P_y - W * P_w \tag{16}$$

where Y is the yield at the end of the season, P_y is the price of the yield, W is the total amount of water used for irrigation, and P_w is the price of the water.

To show an application of the model, the net return for tomato yield in 2018 and 2019 was calculated under random irrigation every five days. An RF model was implied to predict WcTot after each irrigation. The model receives five days of climate data and irrigation rate and predicts WcTot for the next day. The RF model predicts WcTot with an R^2 score of 0.80. Algorithm 1 was used to calculate the net return at the end of the season under random irrigation. To calculate the net return, the cost of irrigation per 1 ha-mm/ha was assumed to be nearly 0.5 USD [43], and tomato prices were assumed to be nearly 728.20 USD/ton (www.tridge.com (accessed date 22 May 2021)).

Algorithm 1: Algorithm

```
Env step(current_state, action):
    season_state = List ();
    next_WcTot = RF_WcTot.predict (current_state, action);
    season_state.append (current_state, action);
    if time_passed = end_of_season then
        done = True;
        Y = BLSTM _yield.predict (season_state);
        calculate reward from Equation (16);
    else
        done = False;
        reward = 0;
    time_passed = time_passed+1;
    return next_WcTot, reward, done
steps = 5;
n_steps = 2;
env = Env ();
for i = 1 to n_steps do
    state = state [0];
    action = 0;
    done = False;
    while done = False do
        for k = steps to steps + 5 do
            if k = steps then
                action = random_action (0, 60);
            else
                action = 0 ;
        new_WcTot, reward, done = env.step (states, action, k);
        state = new_WcTot + climate_data
```

Tables 10 and 11 show the net returns and the parameters used in Algorithm 1.

Table 10. The net return of random irrigation.

Irrigation	Yield (ton/ha)	Total Irrigation (ha-mm/ha)	Net Return (USD/ha)
(2018)	5.04	660	3344
(2019)	3.84	770	2415

Table 11. The parameters used in Algorithm 1.

Param	Explanation
action	volume of water
Done	a Boolean value represent whether a season is complete
n_steps	number of season
steps	time step between irrigation
state	climate big data and WcTot
new_WcTot	WcTot after irrigation
time_passed	time elapsed after the start of the season
state [0]	the data from the first 5 days of the season

In this example, the irrigation amount was randomly selected, but in future work, a reinforcement learning agent is trained to select the best irrigation amount, and the random

action (water amount) is replaced by the model. In Deep Reinforcement Learning algorithms, there is an environment that interacts with an agent. During the training, the agent chooses an action based on the current state of the environment, and the environment returns the reward and the next state to the agent. The agent tries to choose the action that maximizes the reward [44]. In the agricultural domain, the state of the environment can be defined as the climate data and the water content of the soil; the action is the amount of irrigation, and the reward is the net return. The function Env() in Algorithm 1 is used as the environment. An agent can be trained to select the amount of irrigation based on the condition of the field. Therefore, the yield estimation model can be used as part of the environment of Deep Reinforcement Learning to calculate the reward of the agent. This system can be trained end-to-end.

4. Conclusions

Irrigation and storage are closely related to the use of energy. Efficient irrigation minimizes unnecessary water use, which contributes to energy conservation. Yield estimation models help to reduce energy consumption, increase productivity, and estimate labor requirements for harvesting and storage requirements. RNN models offer several advantages over other deep learning models and traditional machine learning approaches. The most important aspect is their ability to process time-series data such as agricultural datasets. In this work, the ability of RNN models to predict tomato and potato yields based on climate data and irrigation amount was investigated. The LSTM, GRU, and their extension BLSTM and LSTM models were trained on sandy loam soil for crop yield prediction. The results show that the use of BLSTM models outperformed the simple LSTM, GRU, and BGRU models on the validation set. In addition, the LSTM model performed better than the GRU model in the validation set. Therefore, removing the cell state from the LSTM nodes could be problematic in our context.

The BLSTMs achieved an R^2 score of 0.97 to 0.99 on the test set. The results show that BLSTM models can automatically extract features from raw agricultural data, capture the relationship between climate data and irrigation amount, and convert it into a valuable model for predicting future field yields. One drawback of these models is that a sufficient amount of clean data is needed to train the model. With more data, the model can make better predictions. In this work, the simulated yield dataset was used to overcome this disadvantage.

The performance of the BLSTM was compared with the CNN model, MLP, and RF, and it was found that the BLSTM outperformed the MLP networks and CNN and RF in yield prediction. The CNN model achieved the second-best performance and MLP the worst performance. The results show that past observations of a season are important in yield prediction. The BLSTM model can capture the relationship between past observations and the new observations and predict the yield more accurately. One disadvantage of the BLSTM model was that the training time of the BLSTM model was higher than other implemented models.

One of the applications of the yield prediction model is to develop an end-to-end decision support system that automatically decides when and how much to irrigate. Deep Reinforcement Learning models are used to build such a system. An agent can be trained to select the amount of irrigation based on the condition of the field. To train such a model, a reward function must be developed. In the agricultural domain, the net reward is used as the reward for the agent. Since it is difficult and time-consuming to work with a simulation system such as Aquacrop to train a deep learning model, the yield estimation model is used to determine the net return of the agent at the end of each season, and the agent can decide based on this reward. This system can help farmers to decide when and how much to irrigate and reduces water consumption without affecting productivity.

Author Contributions: K.A.: investigation, methodology, writing—original draft, and writing—review. P.D.G.: supervision, writing—review, project administration, and funding acquisition. T.M.L.: writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the project Centro-01-0145-FEDER000017-EMaDeS-Energy, Materials, and Sustainable Development, co-funded by the Portugal 2020 Program (PT 2020), within the Regional Operational Program of the Center (CENTRO 2020) and the EU through the European Regional Development Fund (ERDF). Fundação para a Ciência e a Tecnologia (FCT—MCTES) also provided financial support via project UIDB/00151/2020 (C-MAST).

Acknowledgments: We would like to express our sincere gratitude for the support provided by AppiZêzere and DRAP-Centro with the data from the meteorological stations near Fadagosa. P.D.G. and T.M.L. acknowledge Fundação para a Ciência e a Tecnologia (FCT—MCTES) for its financial support via the project UIDB/00151/2020 (C-MAST).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sundmaeker, H.; Verdouw, C.N.; Wolfert, J.; Freire, L.P. Internet of Food and Farm 2020. In *Digitising the Industry*; Vermesan, O., Friess, P., Eds.; River Publishers: Gistrup, Denmark, 2016; pp. 129–150.
2. Nguyen, G.; Dlugolinsky, S.; Bobak, M.; Tran, V.; Garcia, A.L.; Heredia, I.; Malik, P.; Hluchy, L. Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: A survey. *Artif. Intell. Rev.* **2019**, *52*, 77–124, [CrossRef]
3. Hayes, M.J.; Decker, W.L. Using NOAA AVHRR data to estimate maize production in the United States Corn Belt. *Int. J. Remote Sens.* **1996**, *17*, 3189–3200. [CrossRef]
4. Bargoti, S.; Underwood, J.P. Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards. *J. Field Robot.* **2017**, *34*, 1039–1060. [CrossRef]
5. Habaragamuwa, H.; Ogawa, Y.; Suzuki, T.; Shiigi, T.; Ono, M.; Kondo, N. Detecting greenhouse strawberries (mature and immature), using deep convolutional neural network. *Eng. Agric. Environ. Food* **2018**, *11*, 127–138. [CrossRef]
6. Kang, H.; Chen, C. Fast implementation of real-time fruit detection in apple orchards using deep learning. *Comput. Electron. Agric.* **2020**, *168*, 105108. [CrossRef]
7. Koirala, A.; Walsh, K.B.; Wang, Z.; McCarthy, C. Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'MangoYOLO'. *Precis. Agric.* **2019**, *20*, 1107–1135. [CrossRef]
8. Liang, Q.; Zhu, W.; Long, J.; Wang, Y.; Sun, W.; Wu, W. A Real-Time Detection Framework for On-Tree Mango Based on SSD Network. In *Intelligent Robotics and Applications, Proceedings of the 11th International Conference, ICIRA 2018, Newcastle, NSW, Australia, 9–11 August 2018*; Springer International Publishing: Cham, Switzerland, 2018; pp. 423–436.
9. Stein, M.; Bargoti, S.; Underwood, J. Image Based Mango Fruit Detection, Localisation and Yield Estimation Using Multiple View Geometry. *Sensors* **2016**, *16*, 1915. [CrossRef]
10. Tian, Y.; Yang, G.; Wang, Z.; Wang, H.; Li, E.; Liang, Z. Apple detection during different growth stages in orchards using the improved YOLO-V3 model. *Comput. Electron. Agric.* **2019**, *157*, 417–426. [CrossRef]
11. Apolo-Apolo, O.; Martínez-Guanter, J.; Egea, G.; Raja, P.; Pérez-Ruiz, M. Deep learning techniques for estimation of the yield and size of citrus fruits using a UAV. *Eur. J. Agron.* **2020**, *115*, 126030. [CrossRef]
12. Maimaitijiang, M.; Sagan, V.; Sidike, P.; Hartling, S.; Esposito, F.; Fritschi, F.B. Soybean yield prediction from UAV using multimodal data fusion and deep learning. *Remote Sens. Environ.* **2020**, *237*, 111599. [CrossRef]
13. Yang, Q.; Shi, L.; Han, J.; Zha, Y.; Zhu, P. Deep convolutional neural networks for rice grain yield estimation at the ripening stage using UAV-based remotely sensed images. *Field Crop. Res.* **2019**, *235*, 142–153. [CrossRef]
14. Chen, Y.; Lee, W.S.; Gan, H.; Peres, N.; Fraisse, C.; Zhang, Y.; He, Y. Strawberry Yield Prediction Based on a Deep Neural Network Using High-Resolution Aerial Orthoimages. *Remote Sens.* **2019**, *11*, 1584. [CrossRef]
15. Zhou, Z.; Song, Z.; Fu, L.; Gao, F.; Li, R.; Cui, Y. Real-time kiwifruit detection in orchard using deep learning on Android™ smartphones for yield estimation. *Comput. Electron. Agric.* **2020**, *179*, 105856. [CrossRef]
16. Rahnemoonfar, M.; Sheppard, C. Deep Count: Fruit Counting Based on Deep Simulated Learning. *Sensors* **2017**, *17*, 905. [CrossRef] [PubMed]
17. Ma, J.W.; Nguyen, C.H.; Lee, K.; Heo, J. Regional-scale rice-yield estimation using stacked auto-encoder with climatic and MODIS data: A case study of South Korea. *Int. J. Remote Sens.* **2019**, *40*, 51–71. [CrossRef]
18. Han, J.; Zhang, Z.; Cao, J.; Luo, Y.; Zhang, L.; Li, Z.; Zhang, J. Prediction of Winter Wheat Yield Based on Multi-Source Data and Machine Learning in China. *Remote Sens.* **2020**, *12*, 236. [CrossRef]
19. Kim, N.; Ha, K.J.; Park, N.W.; Cho, J.; Hong, S.; Lee, Y.W. A Comparison Between Major Artificial Intelligence Models for Crop Yield Prediction: Case Study of the Midwestern United States, 2006–2015. *ISPRS Int. J. Geo-Inf.* **2019**, *8*, 240. [CrossRef]
20. Abbas, F.; Afzaal, H.; Farooque, A.A.; Tang, S. Crop Yield Prediction through Proximal Sensing and Machine Learning Algorithms. *Agronomy* **2020**, *10*, 1046. [CrossRef]
21. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90. [CrossRef]
22. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*; Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2012; pp. 1097–1105.

23. Sermanet, P.; Eigen, D.; Zhang, X.; Mathieu, M.; Fergus, R.; LeCun, Y. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *arXiv* **2013**, arXiv:1312.6229.
24. Patterson, J.; Gibson, A. *Deep Learning: A Practitioner's Approach*; O'Reilly: Beijing, China, 2017.
25. Hochreiter, S.; Schmidhuber, J. Long Short-Term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef]
26. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014; Association for Computational Linguistics: Doha, Qatar, 2014; pp. 1724–1734.
27. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. In Proceedings of the NIPS 2014 Workshop on Deep Learning, Montreal, QC, Canada, 8–13 December 2014.
28. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]
29. Gal, Y.; Ghahramani, Z. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16*; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 1027–1035.
30. Prechelt, L. Early Stopping—But When? In *Neural Networks: Tricks of the Trade: Second Edition*; Montavon, G., Orr, G.B., Müller, K.R., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; pp. 53–67.
31. Raes, D.; Steduto, P.; Hsiao, T.C.; Fereres, E. AquaCrop—The FAO Crop Model to Simulate Yield Response to Water: II. Main Algorithms and Software Description. *Agron. J.* **2009**, *101*, 438–447. [CrossRef]
32. Allen, R.G.; Pereira, L.S.; Raes, M.S.D. *Crop eVapotranspiration—Guidelines for Computing Crop Water Requirements FAO Irrigation and Drainage Paper 56*; FAO—Food and Agriculture Organization of the United Nations: Rome, Italy, 1998.
33. Montgomery, D.C.; Jennings, C.L.; Kulahci, M. *Introduction to Time Series Analysis and Forecasting*; Wiley Series in Probability and Statistics; Wiley: Hoboken, NJ, USA, 2011.
34. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning: With Applications in R*; Springer Publishing Company, Incorporated: Berlin/Heidelberg, Germany, 2014.
35. Willmott, C.J.; Ackleson, S.G.; Davis, R.E.; Feddema, J.J.; Klink, K.M.; Legates, D.R.; O'Donnell, J.; Rowe, C.M. Statistics for the evaluation and comparison of models. *J. Geophys. Res. Ocean.* **1985**, *90*, 8995–9005. [CrossRef]
36. Zhang, J.; Zhu, Y.; Zhang, X.; Ye, M.; Yang, J. Developing a Long Short-Term Memory (LSTM) based model for predicting water table depth in agricultural areas. *J. Hydrol.* **2018**, *561*, 918–929. [CrossRef]
37. Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G.S.; Davis, A.; Dean, J.; Devin, M.; et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Software. Available online: tensorflow.org (accessed on 3 March 2020).
38. Chollet, F. Keras. 2015. Available online: https://keras.io (accessed on 3 March 2020).
39. Dreyfus, S. The numerical solution of variational problems. *J. Math. Anal. Appl.* **1962**, *5*, 30–45. [CrossRef]
40. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
41. Vanhoucke, V.; Senior, A.; Mao, M.Z. Improving the speed of neural networks on CPUs. In Proceedings of the Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011, Granada, Spain, 12–17 December 2011.
42. Murtagh, F. Multilayer perceptrons for classification and regression. *Neurocomputing* **1991**, *2*, 183–197. [CrossRef]
43. Rodrigues, L.C. Water Resources Fee in Portugali, 2016. Led by the Institute for European Environmental Policy. Available online: https://ieep.eu/ (accessed on 3 March 2021).
44. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018.