computers

MDPI

Article

Comparison of On-Policy Deep Reinforcement Learning A2C with Off-Policy DQN in Irrigation Optimization: A Case Study at a Site in Portugal

Khadijeh Alibabaei [1,2], Pedro D. Gaspar [1,2], Eduardo Assunção [1,2], Saeid Alirezazadeh [3], Tânia M. Lima [1,2], Vasco N. G. J. Soares [4,5,*] and João M. L. P. Caldeira [4,5]

1 C-MAST Center for Mechanical and Aerospace Science and Technologies, University of Beira Interior, 6201-001 Covilhã, Portugal; k.alibabaei@ubi.pt (K.A.); dinis@ubi.pt (P.D.G.); eduardo.assuncao@ubi.pt (E.A.); tmlima@ubi.pt (T.M.L.)
2 Department of Electromechanical Engineering, University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
3 C4—Cloud Computing Competence Centre (C4-UBI), University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal; saeid.zadeh@ubi.pt
4 Polytechnic Institute of Castelo Branco, Av. Pedro Álvares Cabral nº 12, 6000-084 Castelo Branco, Portugal; jcaldeira@ipcb.pt
5 Instituto de Telecomunicações, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
* Correspondence: vasco.g.soares@ipcb.pt

check for updates

Citation: Alibabaei, K.; Gaspar, P.D.; Assunção, E.; Alirezazadeh, S.; Lima, T.M.; Soares, V.N.G.J.; Caldeira, J.M.L.P. Comparison of On-Policy Deep Reinforcement Learning A2C with Off-Policy DQN in Irrigation Optimization: A Case Study at a Site in Portugal. *Computers* **2022**, *11*, 104. https://doi.org/10.3390/computers11070104

Academic Editors: Phivos Mylonas, Katia Lida Kermanidis and Manolis Maragoudakis

Received: 30 May 2022
Accepted: 21 June 2022
Published: 24 June 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Abstract: Precision irrigation and optimization of water use have become essential factors in agriculture because water is critical for crop growth. The proper management of an irrigation system should enable the farmer to use water efficiently to increase productivity, reduce production costs, and maximize the return on investment. Efficient water application techniques are essential prerequisites for sustainable agricultural development based on the conservation of water resources and preservation of the environment. In a previous work, an off-policy deep reinforcement learning model, Deep Q-Network, was implemented to optimize irrigation. The performance of the model was tested for tomato crop at a site in Portugal. In this paper, an on-policy model, Advantage Actor–Critic, is implemented to compare irrigation scheduling with Deep Q-Network for the same tomato crop. The results show that the on-policy model Advantage Actor–Critic reduced water consumption by 20% compared to Deep Q-Network with a slight change in the net reward. These models can be developed to be applied to other cultures with high production in Portugal, such as fruit, cereals, and wine, which also have large water requirements.

Keywords: agriculture; deep learning; on-policy deep reinforcement learning; irrigation optimization

1. Introduction

Water deficiency directly or indirectly affects all physiological processes in plants, some of which have a major impact on crop growth, development, and productivity [1,2]. The effect of water stress on transpiration, photosynthesis, and the subsequent absorption of water and nutrients by plants has a profound impact on crops and their potential productivity [1,2].

The Food and Agriculture Organization (FAO) reports that agriculture is the sector where the greatest need for action is to reduce water consumption, as about 60% of the water used for irrigation is lost as waste [3]. The same studies indicate that reducing this loss by 10% would be enough to supply twice the current world population, based on statistical averages [3]. Hence, an efficient water management system is essential. With the use of the Internet of Things (IoT) [4] in agriculture, systems are being developed to effectively manage fields [5,6]. The IoT sensors enable monitoring of light, humidity, temperature, soil moisture, and analysis of water among other parameters [5–9].

The huge amount of data provided by these sensors must be analyzed to develop an automated decision-making system. Machine-learning methods are an important tool to analyze this data [10–12]. Machine learning is a topic that has become increasingly important recently. The learning that gives the term "machine learning" its name consists of running algorithms that automatically build knowledge representation models based on a dataset [13,14]. The idea behind this learning is that the machines are trained by giving them access to historical data and one or more performance measures and letting the algorithm "learn", i.e., iteratively adjust the knowledge representation model so that it improves its performance [13,14]. After this training, the model has the potential to make high-quality predictions in future situations related to historical patterns [10–14].

Zia et al. [15] compared traditional irrigation calculations by farmers with an IoT-based irrigation method on a lemon farm. IoT sensor data were collected wirelessly through the cloud and a mobile application, and a Decision Support System (DSS) provided irrigation recommendations. The DSS system is based on temperature and humidity, real-time sensor data from the IoT device used in the farm, and plant data (Kc and plant type). The results show water savings of over 50% while increasing yields by 35% compared to the traditional irrigation method.

Tseng et al. [16] estimated the soil moisture from the images using a convolutional neural network (CNN). The results showed that CNN outperformed traditional machine-learning methods, such as support vector machine (SVM), random forest (RF), and two-layer neural networks (ANN). Song et al. [17] combined the macroscopic cellular automata (MCA) model with a deep belief network (DBN) to estimate soil water content in the field. The DBN-MCA model performed better compared to the multilayer perceptron model by reducing the mean square error by 18%.

Saggi and Jain [18] estimated daily reference evapotranspiration using a deep-learning multi-layer perceptron (MLP) for Hoshiarpur and Patiala districts in Punjab. The MLP outperformed traditional machine-learning methods, such as RF, Generalized Linear Models (GLM) and Gradient Boosting Machine (GBM) with a mean square error of 0.0369 to 0.1215.

De Oliveira and Lucas et al. [19] employed three CNN models to predict daily reference evapotranspiration. Performance was compared between CNN and AutoregRessive Integrated Moving Average (ARIMA) and the seasonal Naive model. The CNN model performed better in terms of accuracy. Ahmed et al. [20] developed a deep-learning approach for two-stage daily surface soil moisture prediction (SSM) using a Gated recurrent unit (GRUs)-based recurrent neural network. The model was built by integrating MODIS sensors (satellite-based data), ground-based observations, and climate indices tested at stations in Australia's MurrayDarling Basin. The model achieved low Mean Absolute Error (MAE) values between 0.013 and 0.113 kg m^{-2} for the first, fifth, and seventh-day predictions.

Adab et al. [21] used four different types of machine-learning models to predict near-surface (5 cm) soil moisture in the field on different plots. The Random Forest model performed better than the other three methods (ANN, SVM, and elastic net regression algorithm) in predicting soil moisture in the test cases. Although the prediction of soil water content and reference evapotranspiration is critical for irrigation scheduling, further analysis is needed to predict the exact timing and amount of water for irrigation.

Jimenez et al. [22] two recurrent neural network (RNN) models were used to estimate irrigation effort. Data were collected from 2017 to 2019 on a corn farm in Samson, Alabama. Hourly weather data and soil matric potential (SMP) data measured at three soil depths from 13 sensor probes installed on a loamy fine sand soil and a sandy clay loam soil were used for the study. Two neural network methods and Long Short-Term Memory (LSTM) models were used to predict irrigation schedules. The results showed that both RNN models performed well in predicting irrigation prescriptions for the soil types studied, with a coefficient of determination of R^2 greater than 0.94 and a root mean square error (RMSE) less than 1.2 mm.

Bu and Wang [23] mentioned that Deep Reinforcement Learning is a promising model for building smart farms. Deep Reinforcement Learning is a combination of reinforcement learning and deep learning (DL). DL is a sub-field of machine learning where the algorithms are deeper in terms of the number of hidden layers [13,14]. DL algorithms are created and function similarly to machine learning.

However, these algorithms have numerous layers, each providing a different interpretation of the data. Neural networks attempt to mimic the function of human neural networks in the brain [13,14]. Reinforcement learning is an area of machine learning that involves taking appropriate actions to maximize reward in a given situation. It is used by various software programs and machines to find the best possible behavior or path in a given situation [23].

Chen et al. [24] used a Deep-Q Learning (DQN) model for an irrigation decision strategy based on short-term weather forecasts for rice. The results of the DQN irrigation strategy compared with those of the conventional irrigation strategy showed a significant reduction in irrigation water volume, irrigation timing, and drainage water without yield loss. In Alibabaei et al. [25], the DQN was used to estimate the timing and amount of water for irrigation. The objective of the paper was to minimize water consumption without affecting the net return of the farmer.

The model was trained using the environmental conditions, such as the temperature, humidity, reference evapotranspiration, soil moisture, and the last irrigation amount and decided the timing and amount of water for the next irrigation. The DQN agent model increased productivity by 11% and avoided water waste by 20–30% compared to a fixed irrigation amount and threshold methods.

The DQN model is an off-policy method, i.e., in the DQN algorithm, the updating policy (strategy to select the best action) is different from the behavioral policy. The on-policy Advantage Actor–Critic (A2C) method outperformed the DQN method in the Atari domain and on a variety of continuous motor control problems as well as for navigating random 3D mazes with visual input [26]

To determine how well the A2C model performs in this task of irrigation scheduling for a tomato field, this paper compares the performance of the model with that of DQN. The model simply estimates and tells farmers when and how much water is needed for the next irrigation. The performance of the model is compared in terms of productivity and water use with the DQN model and threshold method. Moreover, the total soil water content is compared when using the DQN and A2C models for irrigation scheduling.

The remainder of this paper is organized as follows. In Section 2, materials and methods, the general framework of the work is explained, and in the subsections, each step is explained in detail through the sequence of data set collection, 2.1, data processing, 2.2, an overview of the models used in the work, 2.3, and experimental setup, 2.4. In Section 3, the results are described and discussed. The summary of the work is included in Section 4.

2. Materials and Methods

The framework of this paper is shown in Figure 1. The first two steps are the same as in [25], and, in the last step, the DQN algorithm is replaced by the A2C algorithm to investigate the potential of this model for irrigation scheduling. In the first step of the framework, the big data are collected from the weather station at a site in Portugal and simulated using Decision Support System for Agrotechnology Transfer (DSSAT) software [27,28]. In the second step, two DL models, called Long Short-Term Memory (LSTM), are trained to estimate the total soil water in the soil profile (mm) (SWTD) and tomato yield at the end of the season. In the third step, these trained models are used as the environment for the agent. The agent acts (chooses the amount of water), and the environment responds to this action by calculating the SWTD and tomato yield. The agent and the environment interact until the best strategy for irrigation is found.

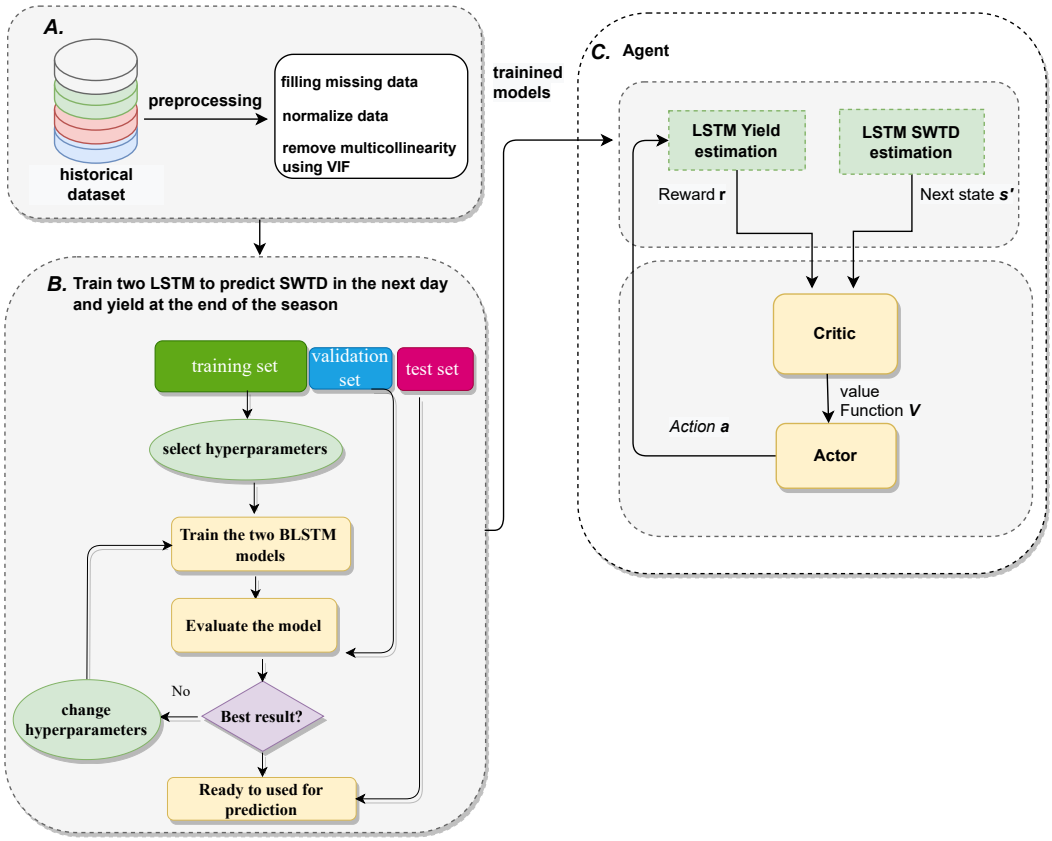

Figure 1. Framework of this paper. Modified version of [25].

Each step of the framework is explained in the following subsections.

2.1. Data Collection

To compare the potential of DQN and A2C models in irrigation scheduling, the same data from [25] were used in this work. Climate Big data were collected by the government agency of the Ministry of Agriculture and the Sea, Direção Regional de Agricultura e Pescas do Centro, Portugal (www.drapc.gov.pt (accessed on 4 March 2020)) for the Fadagosa site in Portugal from 2010 to 2019. The soil texture of Fadagosa is either sandy or sandy loam, and the climate type is Mediterranean hot summer climate (Csa). Figure 2 shows the Fadagosa region from Google Earth.

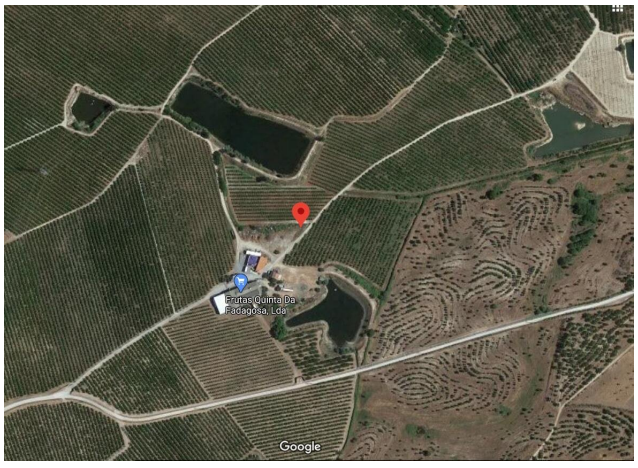

Figure 2. Map of the Fadagosa region.

Table 1 shows the details of the climate variables retrieved from the weather station, and Figure 3 shows the daily climate variables from 2010 to 2019.

Table 1. Dataset details.

Variables	Unit	Data Source	Max	Min	Mean	SD
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
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
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
SR_{Avg}	wm^{-2}	DRAP-Centro	346.66	6.35	172.02	89.25
ET0	mm d^{-1}	Penman-Monteith equation (AquaCrop calculator)	9.8	0.2	3.68	2.088
Tomato yield	kg/ha	DSSAT	8387	974	4391	2564

The abbreviations stand for the following: SD: standard deviation, Min: minimum, Max: maximum, Avg: average, HR: relative humidity, T: temperature, WS: Wind Speed, Prec: precipitation, SR: Solar Radiation, and ET0: Reference Evaporation [29].

As it is difficult to record the tomato yield at different irrigation rates (e.g., the recording yield at no irrigation), to ensure data availability for training the model, Decision Support System for Agrotechnology Transfer (DSSAT) [27,28] was used to estimate the tomato yield at different irrigation rates. The study was conducted using the cropping simulation model, and the same calibration of DSSAT software was used as in [25]. For the calculation of the irrigation regime, we considered a fixed interval of four days as a time parameter. The depth criterion was also considered as a fixed value in the interval of 0 and 60 mm. The ET0 calculator developed by the Land and Water Division of FAO was used to estimate Reference Evaporation (ET0) [30].

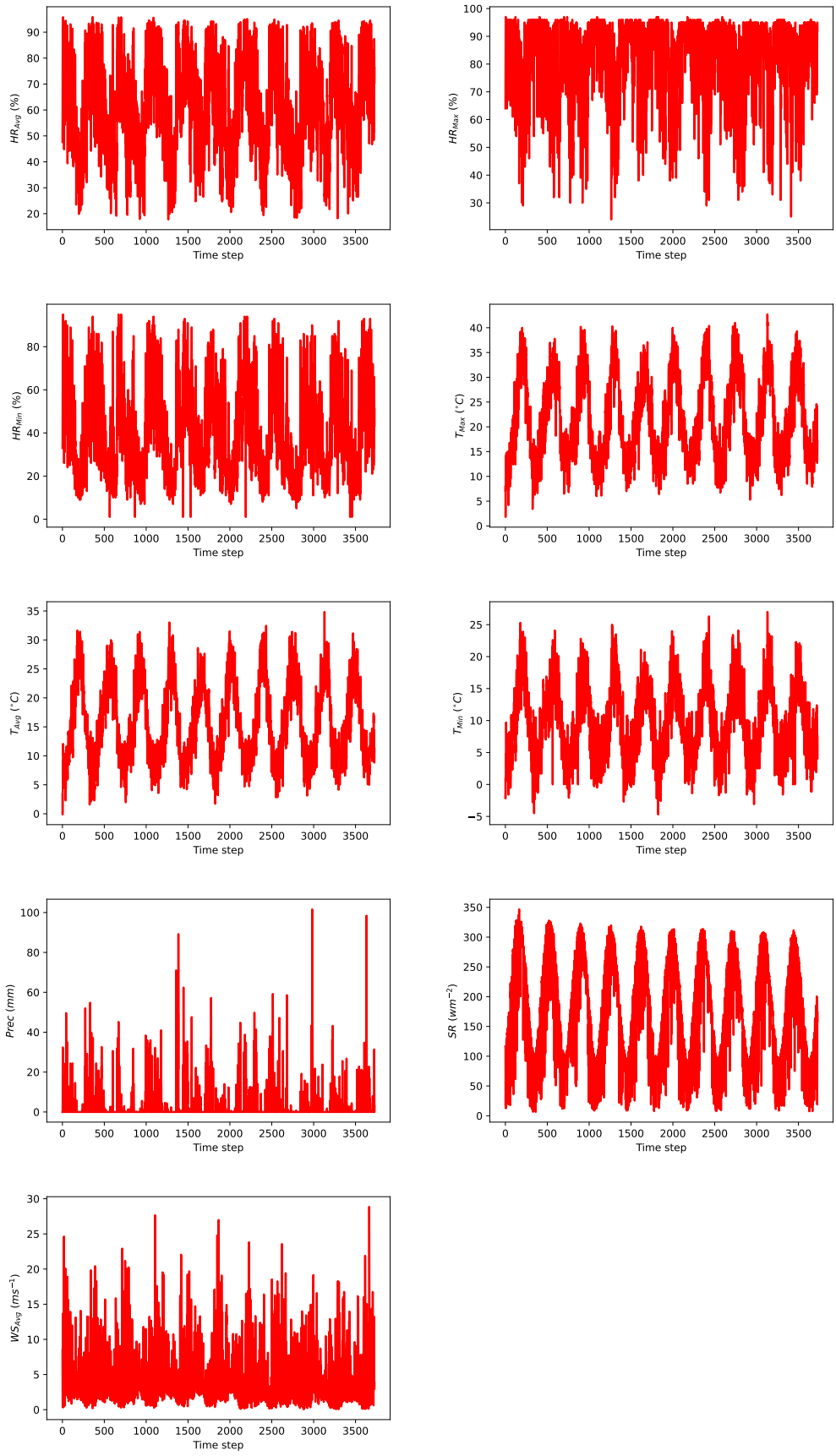

Figure 3. Fadagosa daily dataset [29]. The abbreviations are the same as in Table 1.

2.2. Data Pre-Processing

As in [25], the moving average was used to fill in the missing big data [31]. Then, the multicollinear parameters were removed from the data set using the variance inflation factor (VIF) [32]. The same information contained in the multicollinear parameters leads to calculation and interpretation problems.

Normalization is a technique generally applied as part of data preparation for machine learning. The purpose of normalization is to change the variables in the data set to use a single scale without distorting differences in value ranges or losing information. The values of the scaling coefficients must be calculated for the training data set and used to re-scale the test data set and the predictions. This avoids contaminating the experiment with knowledge about the test data set. In this work, the min-max normalization method was used, which scales the variables in the data set between zero and one using Equation (1).

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}} \tag{1}$$

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

2.3. Model Used

2.3.1. Bidirectional LSTM Structure

Recurrent Neural Networks (RNNs) are a family of neural networks designed to process sequential data [13,33]. A variant of conventional RNNs is Long Short-Term Memory (LSTM), which solves a well-known problem called a vanishing gradient [13,33]. This occurs when the weights computed in the initial parts of the sequence lose influence as iterations progress and they respond to new inputs. As a result, the range of contextual information that can be captured by conventional RNNs is usually quite limited. Such a limitation causes these architectures to perform very poorly for longer sequences [13,33]. The LSTM was developed to address this problem. It is an RNN with substructures that help to manage the memory of the recurrent neural network. Figure 4 shows the cell of an LSTM.

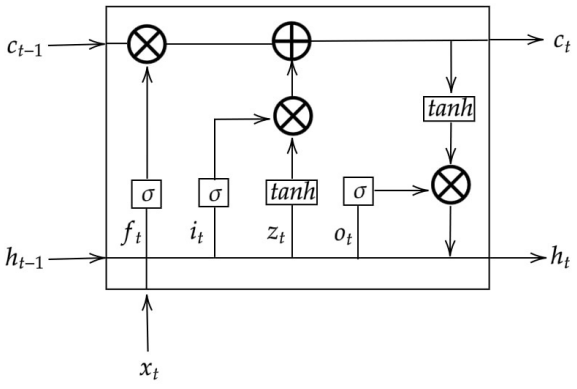

Figure 4. LSTM cell [29].

First, the LSTM must decide what information to disregard in the cell. This is done by the forget gate using the sigmoid function (σ). It analyses the output of the previous cell h_{t-1} and the input of the current cell x_t and produces an output of numbers between 0 and 1, where 1 represents the complete retention of that information (Equation (2)).

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2}$$

It then decides what new information it will retain. To this end, it performs two processes: First, the input gate i_t formed by the sigmoid function decides which value will

be updated, and then the activation function tanh generates a vector of new candidates $\tilde{C}_t$ (Equation (3)) that can be added to the state.

$$\tilde{C}_t = \tanh((W_C[h_{t-1}, x_t] + b_C) \tag{3}$$

Then, the forget gate f_t is multiplied by C_{t-1} so that the information deemed unnecessary is forgotten, and i_t is multiplied by C_t to retain the new information that is useful (Equation (4)). Then, add the result of these two multiplications. Finally, the output is determined. To do this, the sigmoid layer (Equation (5)) decides which parts of the cell state to output, and then multiplies this by the cell state $\tanh(C_t)$ so that only the information that the network has learned is important (Equation (6)).

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4}$$

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

$$h_t = o_t \odot \tanh(C_t) \tag{6}$$

W and b are the weights and biases of the specific gate in Equations (2)–(6), which should be adjusted when training the model to minimize the loss function.

Bidirectional LSTMs (BLSTM) [34] are a complement to regular LSTMs that are used to improve model performance in sequence classification problems. BLSTMs use two LSTMs to train on sequential inputs. The first LSTM is used unchanged in the input chain. The second LSTM is used in a reverse representation of the input sequence.

2.3.2. Advantage Actor–Critic Network

A Markov decision process (MDP) contains a tuple (S, A, R, P), where [35]

- States S: is the set of environment states.
- Action (A): a set of all possible actions.
- A real-valued function R of $S \times A \times S$ is called a reward function, which is an incentive mechanism that tells the agent which action is more valuable.
- A transition function P from $S \times A \times S$ to $[0, 1]$, where $P(s, a, s')$ captures the probability of changing from state s to s' after executing action a.

In an MDP model, the conditional probability distribution of the next states of the process depends only on the current state, not on the sequence of events that preceded it [23,35].

The Policy π is a mapping from the states S to the set of actions A and determines the action based on the current state [23,35]. The objective of reinforced learning is for the agent to learn an optimal or near-optimal policy that maximizes the reward function. The long-term reward at time t is defined by Equation (7) [23,35]:

$$\begin{aligned} G_t &= R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} \cdots + \gamma^{T-1} R_T \\ &= R_t + \gamma\left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots + \gamma^{T-2} R_T\right) \end{aligned} \tag{7}$$

where R_T indicates the reward of the final state and γ is a real number between 0 and one, called the discount factor, added because of the uncertainty of the future states. Equation (8) results from the definition of G_t and Equation (7).

$$G_t = R_t + \sum_{n=1}^{T} \gamma G_{t+1} \tag{8}$$

The value function (V-function) measures how good it is for an agent to be in a state s following a policy and is calculated by Equation (9) [35]:

$$\mathrm{V}_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] \quad \forall s \in S \tag{9}$$

where $\mathbb{E}_\pi$ denotes the expected value if the agent follows strategy π, and s_t is the state at time t. The optimal policy is defined using the value function as:

$$\mathrm{V}^*(s) = \max_\pi \mathrm{V}_\pi(s) \tag{10}$$

The optimal policy π^* is computed using an iteration algorithm over the V function. The Bellman Equation (11) is applied to $V(s)$ for any state s until $V(s)$ reaches the maximum value, denoted as $V^*(s)$.

$$\mathrm{V}^{(i)}(s) = \sum_{s' \in S, a \in A} T(s'|s;a) \left[R(s;a;s') + \gamma \max_{s'} \mathrm{V}^{(i-1)}(s') \right] \tag{11}$$

$$\lim_{i \to \infty} \mathrm{V}^{(i)}(s) = \mathrm{V}^*(s) \tag{12}$$

where $T(s'|s;a)$ is the transition probability from state s to state s' when the agent chooses an action a, $R(s;a;s')$ is the immediate reward from state s to state s' when the agent chooses an action a, and γ is the discounted rate.

DL algorithms estimate a nonlinear function between the dependent variables and the independent variables. If the transition or reward function for an MDP problem is not known, a deep-learning model can be used to estimate it and solve the MDP. The Advantage Actor–Critic (A2C) [26] is a DRL method that uses two different deep-learning models to perform the learning (see Figure 5). The first model is the actor, which is used to define the policy of the applied actions.

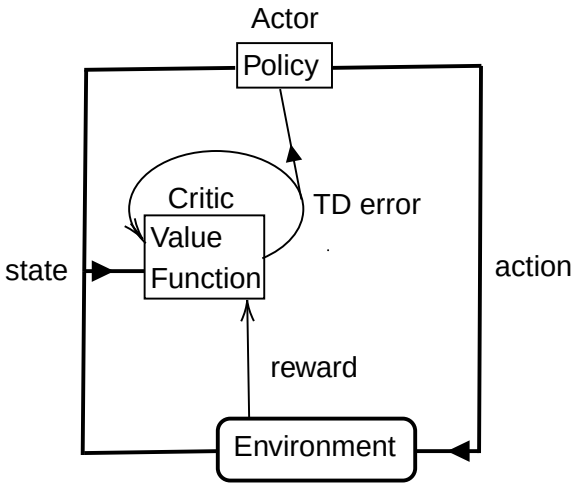

Figure 5. The A2C model interacting with environment.

The output of the actor is the probability of executing each action from state s_t at time t. The second model is the critic, which estimates the value function V and evaluates all actions performed by the actor [26,36].

The value function V is the basis for choosing the optimal policy; however, this function is unknown to the agent, and thus it must learn to estimate it from the rewards it receives from interactions with the environment. The temporal difference method (TD) uses the rewards received to estimate the V function and allows the agent to learn and improve its behavior with each action performed [26]. The TD error can be calculated using Equation (13):

$$\delta_t = r_{t+1} + \gamma \mathrm{V}(s_{t+1}) - \mathrm{V}(s_t) \tag{13}$$

where s_{t+1} represents the state reached after performing an action starting from a state s_t and receiving a corresponding reward r_{t+1}. After receiving this new reward, the value

of the new state is used to update that of the previous state. An important parameter of learning is the discount factor, which is limited to the range of 0 and 1 and determines the importance of the future rewards; the lower the discount factor, the more important the short-term rewards are and the less important the future rewards are. When the TD error is positive, it indicates that the tendency to choose the action a_t should be strengthened for the future, while when the TD error is negative, it indicates that the tendency should be weakened [26]. When using experience through interactions, the temporal difference method eliminates the need for an explicit model of the system that can be applied to systems with unknown parameters or dynamics [37].

The critic model uses the TD error, Equation (13), to improve its estimation and the critic's policy. The critic error is defined as the squared of δ_t. The actor loss is defined by Equation (14). The log probability of the action is scaled by the advantage (TD error), making the variance of the error smaller and the learning process more stable [38].

$$L_{\pi_\theta} = -\sum_{t=0}^{T} (\log \pi_\theta(a_t|s_t))\delta_t \tag{14}$$

where θ is the weights of the actor model.

A learning agent is subject to the trade-off between exploration and exploitation. Exploration means repeating the actions that are known to give good results, and exploitation means trying new actions to learn new things [35]. Exploration without exploitation leads to a sub-optimal solution. In the A2C model, the entropy bonus is usually added to the loss to improve exploration [39]. The role of entropy regularization is to promote exploration through various actions. A more uniform action distribution of a policy has higher entropy and, as a result, more random action; the lower the entropy, the more ordered the action. In the case of the A2C model, the entropy for the Softmax policy action $\pi(a_t|s_t)$ is calculated at the neural network output according to Equation (15).

$$E = -\beta \sum_{t=0}^{T} \pi_\theta(a_t|s_t) \log \pi_\theta(a_t|s_t) \tag{15}$$

where $\beta > 0$ is the entropy regularization weight that determines the trade off between exploration and exploitation. The entropy regularization weight is a hyperparameter that should be determined before training. In this paper, β was chosen to be equal to 0.001.

The differences between DQN and A2C are:

- The DQN model is an off-policy method, and the A2C model is an on-policy method, i.e., unlike A2C, in the DQN algorithm, the updated policy is different from the behavioral policy [26,40].
- Unlike DQN, A2C does not use the Replay Buffer but learns the model using state, action, reward, and next state obtained at each step [26,40].
- In DQN, the function Q is estimated; however, in A2C, the value function V and policy π are estimated [26,40].

2.4. Experimental Setup

A computer system with an Intel Core i7-9700 CPU, 32.0 GB RAM, and an NVIDIA GEFORCE RTX 2080 graphics card was used for the work. The models were implemented using the Python language. Tensorflow [41] and Keras [42] libraries were used to implement the deep-learning models. Tensorflow is an open-source library developed for machine learning, numerical computation, and many other tasks. Keras is also an open-source neural network library written in Python. It can be built on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. It is designed to enable rapid experimentation with deep neural networks and focuses on being easy to use, modular, and extensible.

2.4.1. States and Actions Setup

The state and actions were selected as in [25] to ensure comparability of the models. Table 2 shows the action and state sets.

Table 2. State and action sets.

Environment states	HR_{Avg}, T_{Avg}, $Prec$, WS_{Avg}, $ET0$, $SWTD$, irr
Set of actions	0, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60

2.4.2. Environmental Setup

Two BLSTM models were implemented in [29,43], to predict tomato yield using climate big data, irrigation amount, and soil profile water content, and to estimate SWTD and ET0 from climate data, respectively. Before training a neural network, hyperparameters should be established. The hyperparameters determine the model structures and training strategy [44]. For the BLSTM model, the following hyperparameters were set: the number of layers, the number of hidden units, the dropout size, the learning rate, the learning rate decay, and the batch size. These parameters are set during training based on what is called a validation set, which is used to evaluate the model during training, selecting the hyperparameters with the best validation metric [44]. Table 3 shows the parameters used for each BLSTM model.

Table 3. Selected hyperparameters for the tomato yield estimation model (BLSTM1) and soil moisture estimation model (BLSTM2).

Model	No. Layers	No. Hidden Layers	Batch Size	Learning Rate	Decay	Drop Out Size Out Size
BLSTM1	2	512	64	10^{-3}	10^{-5}	0.3
BLSTM2	1	512	124	10^{-4}	10^{-5}	0.2

As in [25], these BLSTM models were used to set up the agent's environment. The BLSTM1 was used as a function to estimate the net return at the end of the season using Equation (16):

$$r = (yield) * p_{yield} - (water) * p_{water} \quad (16)$$

where p_{water} is the price of 1 mm over 1 ha of water and and p_{yield} is the price of 1 kg of yield. The price of irrigation per 1 mm over 1 ha and tomato prices were nearly 0.5$ and 728.2$/tonne$, respectively, ([45] and www.tridge.com (accessed on 15, 07, 2021)).

The BLSTM2 was deployed to estimate the soil water content for the next day and to determine the next state of the agent. Algorithm 1 shows the environment created for the A2C agent, and Table 4 explains the parameters used in Algorithm 1.

Table 4. The parameters used in Algorithm 1.

Param	Explanation
action	Amount of water for irrigation
state	Climate data and SWTD
Done	A boolean value. If true, it indicates the end of the season
next_SWTD	SWTD after irrigation
time_step	The day of the season

Algorithm 1: A2C training environment [25]

```
ENV step(state, action):
    state_of_ season=[ ];
    next_SWTD=BLSTM2.predict(state, action);
    state_of_ season.append(state, action);
    if time_step = length_of_a_Season then
        Done=True;
        Y=BLSTM1.Predict(states_of_ season);
        compute net return using Equation (2);
    else
        Done=False;
        net return=0;
    time_step+=1;
    return next_SWTD, net return, done
```

2.4.3. Training Configuration of Agent

An A2C agent was implied as shown in Algorithm 2. The states are four days of historical climate data (see Table 2) after the last irrigation chosen by the agent. During these four days, the action is zero for the first three days and is selected by the agent on the fourth day. Since the states are time series, a two-layer LSTM with 256 nodes was used to estimate the value function V and the policy. The LSTM model receives the current state of the environment and outputs two values. One is the probability of executing each action from the current state, and the other is the value of the action executed by the agent.

For the training set, the first seven years of data were selected, and for the test set, the last two years of data were selected.

Algorithm 2: A2C Algorithm

```
initialize training environment env=LSTM();
randomly initialize critic model with random weight ω;
randomly initialize actor policy model with random weight θ;
for episode=1 to max_episode do
    done=False;
    while done=False do
        sample action a_t according to the current policy;
        next_SWTD, reward, done=env(current_state, action);
        next_state=Concatenate(next_SWTD, next_climate_data);
        if done=True and episode=3 then
            calculate TD error;
            update critic by minimizing δ_t^2;
            update actor by minimizing loss from Equation (14)
        end
        current_state=new_state
    end
end
```

3. Results and Discussions

3.1. BLSTM Models Evaluation

In this work, the trained BLSTM models of [29,43] were used as features in the agent's environment. The BLSM model for tomato yield achieved an R^2-score of 0.97 and a Root Mean Square Error (RMSE) of 366 (kg/ha) on the test data set, and the BLSTM model for predicting SWTD achieved an RMSE of 6.841 mm and an R^2-score of 0.98 on the test data set for tomato yield.

3.2. Evaluation of the A2C Agent

Figure 6 shows the actor and critic loss during training. Each episode lasted 7 s. The actor's loss at the end of the training tends to zero, and the agent chooses the action that has the best reward.

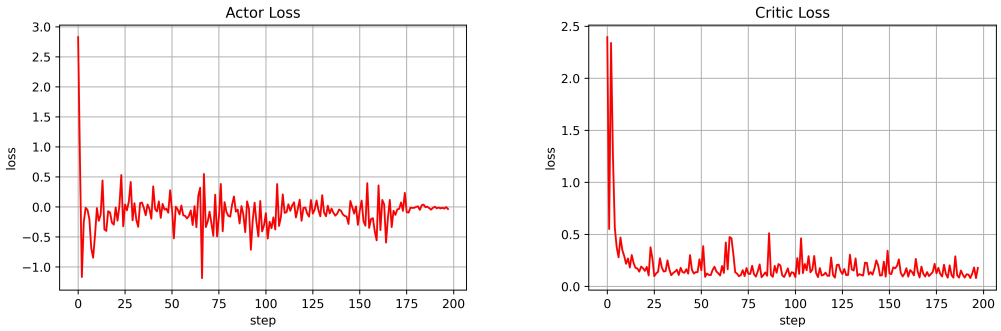

Figure 6. Actor–Critic loss during training.

The logarithm of the Equation (2) was used to accelerate the convergence of the agents. Figure 7 shows the average rewards of the A2C and DQN agents during training. Every fifty episodes, the average is calculated. Upon the improvement of the average reward, the weights of the Actor–Critic network are saved. As can be seen in Figure 6, the training of the A2C agent is more stable than that of the DQN agent. This is because the DQN agent selects the action based on the epsilon greedy method, and the training of the Q function is independent of the action selected by the agent, while the A2C agent is an on-policy model, and during training, the policy is also trained to select the best possible action.

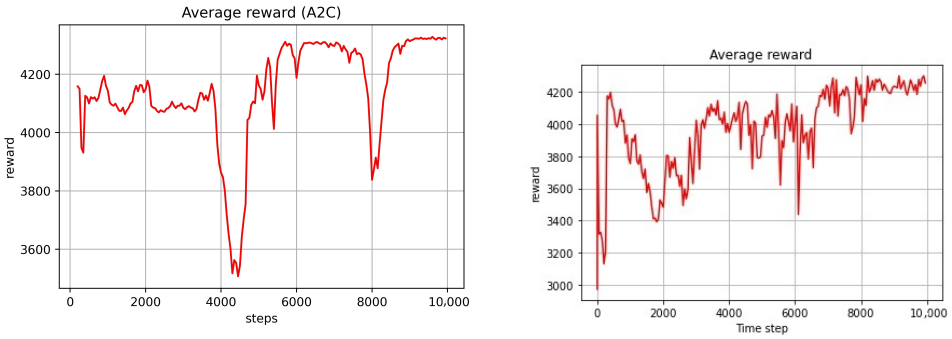

Figure 7. From left to right, the A2C and DQN rewards during training.

The agent was tested with the 2018 and 2019 datasets. Figures 8 and 9 show the SWTD when the A2C model is used for irrigation and the volume of water selected by the agent for irrigation in 2018 and 2019 compared to the DQN agent. The A2C agent removed irrigation early in the season and begins irrigation in mid-season. In the results, the SWTD predicted by the environment of the A2C model is lower than the SWTD predicted by the environment in the DQN model.

Table 5 shows the comparison between the trained DQN agent and the A2C agent and the best result in terms of net return of the threshold method irrigation with a fixed amount in [25]. In the threshold method, SWTD is calculated every four days and if it is below a threshold, a fixed amount of water is used for irrigation. Although the productivity in the case of the DQN model is higher than productivity in the case of the A2C model, water consumption is on average 21.5% lower with the A2C model. In addition, the net yield with the DQN method is on average 3.5% higher than with the A2C model. Thus, the A2C method uses less water; however, the net return is slightly lower.

As we mentioned in the introduction, the objective of the work [25] was to minimize water consumption without affecting the net yield of the farmer. The trained model in-

creased the farmer's net yield by 11% and reduced water consumption by 20–30% compared to a fixed and a threshold method. The main objective of this work was to compare the on-policy model with the off-policy model with the same goal as [25], namely to reduce water consumption without affecting the net return to the farmer. As can be seen from Table 5, both automatic methods (DQN and A2C) performed better in terms of water consumption and net return compared with the threshold method. Moreover, the average rainfall in 2018 was lower than the average rainfall in 2019, and both automatic models learned to irrigate more when rainfall was lower and to adjust the irrigation to climatic changes.

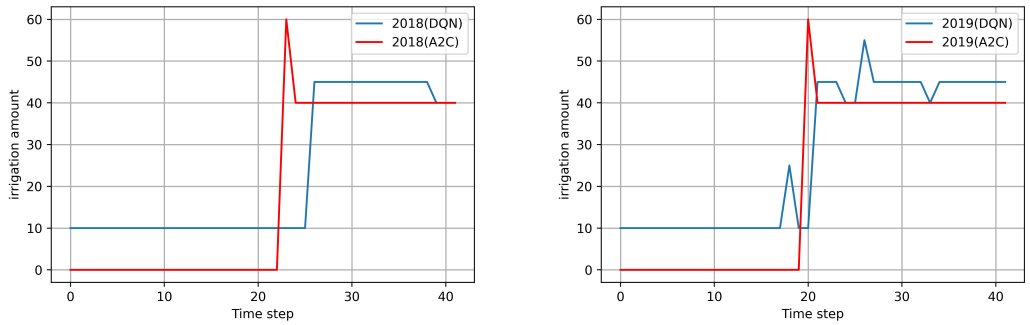

Figure 8. Comparison of the irrigation amount of the trained DQN and A2C models. The time step starts from the beginning of the season to the end of the season every four days.

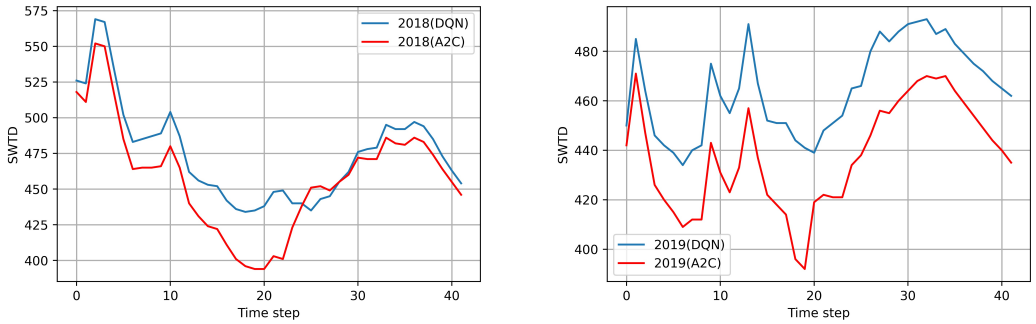

Figure 9. Comparison of SWTD of the trained DQN and A2C models.

Table 5. Comparison of net return of the DQN and A2C agent. The arrow next to each value indicates the increase or decrease of that value compared to the A2C method.

Irrigation	Yield (kg/ha)	Total Irrigation (ha-mm/ha)	Net Return (Dollars/ha)
A2C (2018)	4475	780	3202
DQN (2018)	4675(3% ↑)	965 (20% ↑)	3270 (3% ↑)
threshold of 480 with Fixed 50 mm (2018)	5000 (7.5% ↑)	1400 (45% ↑)	3306 (3% ↑)
A2C (2019)	3787	900	2559
DQN (2019)	4046.6 (7% ↑)	1165(23% ↑)	2666(4% ↑)
threshold of 480 with Fixed 50 mm (2019)	4395 (14% ↑)	1800 (50% ↑)	2628 (2.6% ↑)

4. Conclusions

Most water is used in agriculture, and much of that water is wasted due to the lack of efficient irrigation systems. Water has become a scarce resource. Therefore, it is important to create an efficient irrigation system without compromising productivity.

In [25], the ability of the DQN model to schedule the irrigation of an agricultural field was studied. In this work, the A2C model was trained in the same way to compare the performance of A2C with DQN in scheduling irrigation. The goal was to train the agent to achieve high crop productivity with efficient water use. The agent decided when and how much water was needed for irrigation. The models were trained with seven years of data and tested with two years of data. A disadvantage of the deep-learning method is that a large amount of data is needed to train a model. To overcome this problem, the simulation software DSSAT was used. The tomato yield was simulated based on the different irrigation schedules.

The same environment for DQN was used for the A2C model. The trained A2C agent reduced water consumption by up to 20% compared to the DQN agent; however, productivity decreased slightly. These results show that an on-policy model, such as the A2C model, can achieve better performance in terms of water savings compared with an off-policy model, such as the DQN model. Therefore, the on-policy model is more appropriate than the off-policy DQN model for regions with limited water sources. Both automatic models learn to irrigate more when there is less rain in a year, and they adjust irrigation to the changing climate.

They also both outperformed the threshold method. However, training the models with both simulations and real data sets makes the model more reliable. For a real-world application, real-world data should be collected to re-train the models. In addition, the evaluation of the net return is based on the prices of these years. In this sense, these results will be incorporated into the BioD'Agro project (https://biodagro.com (accessed on 06, 05, 2022)). The main objective of the BioD'Agro project is to develop an information system for remote monitoring of a vineyard and to assist producers in making decisions that promote agrobiodiversity.

To this end, BioD'Agro will work by combining data from in situ sensors that integrate the parameters of the vine (health and water status), the environment (climate and soil), and functional biodiversity (flora, arthropods, and bats) with earth observation imagery and, through the use of machine learning and artificial intelligence, provide a web platform where growers can monitor the water status of their vines or the presence of pests in real time while evaluating and deciding how to manage water efficiently or control pests in an environmentally friendly way. Thus, the data collected by the IoT sensors in the vineyard will be used to train the model developed in this paper to improve irrigation scheduling.

Author Contributions: K.A.: Investigation, Methodology, Writing—original draft Writing—review. E.A.: Investigation, reviwe. P.D.G.: supervision, writing—review, and project administration and Funding acquisition. S.A: Methodology, original draft Writing—review.T.M.L.: Writing—Reviewing and Editing. V.N.G.J.S.: Funding acquisition, Reviewing and Editing. J.M.L.P.C.: Funding acquisition, Reviewing and Editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by FCT/MCTES through national funds and, when applicable, co-funded by EU funds under project UIDB/50008/2020.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All data has been present in main text.

Acknowledgments: K.A. and P.D.G. acknowledge the R&D Project BioDAgro–Sistema operacional inteligente de informação e suporte á decisão em AgroBiodiversidade, project PD20-00011, promoted by Fundação La Caixa and Fundação para a Ciência e a Tecnologia, taking place at the C-MAST -

Centre for Mechanical and Aerospace Sciences and Technology, Department of Electromechanical Engineering of the University of Beira Interior, Covilhã, Portugal. Saeid Alirezazadeh was supported by operation Centro-01-0145-FEDER-000019-C4-Centro de Competências em Cloud Computing, co-financed by the European Regional Development Fund (ERDF) through the Programa Operacional Regional do Centro (Centro 2020), in the scope of the Sistema de Apoio à Investigação Cientifíca e Tecnológica - Programas Integrados de IC&DT. V.N.G.J.S. and J.M.L.P.C. acknowledge that this work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/EEA/50008/2020. 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. Muimba-Kankolongo, A. Chapter 3–Factors Important to Crop Production. In *Food Crop Production by Smallholder Farmers in Southern Africa*; Muimba-Kankolongo, A., Ed.; Academic Press: Cambridge, MA, USA, 2018; pp. 15–21. https://doi.org/10.1016/B978-0-12-814383-4.00003-7.
2. Fahad, S.; Bajwa, A.A.; Nazir, U.; Anjum, S.A.; Farooq, A.; Zohaib, A.; Sadia, S.; Nasim, W.; Adkins, S.; Saud, S.; et al. Crop Production under Drought and Heat Stress: Plant Responses and Management Options. *Front. Plant Sci.* **2017**, *8*. 10.3389/fpls.2017.01147.
3. FAO. World Agriculture 2030: Main Findings. 2002. Available online: http://www.fao.org/english/newsroom/news/2002/7833-en.html (accessed on 5 June 2020).
4. Verdouw, C.; Wolfert, S.; Tekinerdogan, B. Internet of Things in agriculture. *CAB Rev.* **2016**, *11*, 1–12.
5. Lohchab, V.; Kumar, M.; Suryan, G.; Gautam, V.; Das, R.K. A Review of IoT based Smart Farm Monitoring. In Proceedings of the 2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT), Coimbatore, India, 20–21 April 2018; pp. 1620–1625. doi:10.1109/ICICCT.2018.8473337.
6. Doshi, J.; Patel, T.; Bharti, S.K. Smart Farming using IoT, a solution for optimally monitoring farming conditions. *Procedia Computer Science* **2019**, *160*, 746–751. The tenth International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2019)/The ninth International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2019)/Affiliated Workshops. https://doi.org/10.1016/j.procs.2019.11.016.
7. Shakya, A.K.; Ramola, A.; Kandwal, A.; Vidyarthi, A. Soil moisture sensor for agricultural applications inspired from state of art study of surfaces scattering models & semi-empirical soil moisture models. *J. Saudi Soc. Agric. Sci.* **2021**, *20*, 559–572. https://doi.org/10.1016/j.jssas.2021.06.006.
8. Kumar Shakya, A.; Singh, S. Design of novel Penta core PCF SPR RI sensor based on fusion of IMD and EMD techniques for analysis of water and transformer oil. *Measurement* **2022**, *188*, 110513. https://doi.org/10.1016/j.measurement.2021.110513.
9. Abioye, E.A.; Hensel, O.; Esau, T.J.; Elijah, O.; Abidin, M.S.Z.; Ayobami, A.S.; Yerima, O.; Nasirahmadi, A. Precision Irrigation Management Using Machine Learning and Digital Farming Solutions. *AgriEngineering* **2022**, *4*, 70–103. https://doi.org/10.3390/agriengineering4010006.
10. Liakos, K.; Busato, P.B.; Moshou, D.; Pearson, S.; Bochtis, D. Machine Learning in Agriculture: A Review. *Sensors* **2018**, *18*, 61–69.
11. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep learning in agriculture: A survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90.

12. Alibabaei, K.; Gaspar, P.D.; Lima, T.M.; Campos, R.M.; Girão, I.; Monteiro, J.; Lopes, C.M. A Review of the Challenges of Using Deep Learning Algorithms to Support Decision-Making in Agricultural Activities. *Remote Sens.* **2022**, *14*. https://doi.org/10.3390/rs14030638.
13. 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. https://doi.org/10.1007/s10462-018-09679-z.
14. Min, E.; Guo, X.; Liu, Q.; Zhang, G.; Cui, J.; Long, J. A Survey of Clustering with Deep Learning: From the Perspective of Network Architecture. *IEEE Access* **2018**, *6*, 39501–39514.
15. Zia, H.; Rehman, A.; Harris, N.R.; Fatima, S.; Khurram, M. An Experimental Comparison of IoT-Based and Traditional Irrigation Scheduling on a Flood-Irrigated Subtropical Lemon Farm. *Sensors* **2021**, *21*, 4175. https://doi.org/10.3390/s21124175.
16. Tseng, D.; Wang, D.; Chen, C.; Miller, L.; Song, W.; Viers, J.; Vougioukas, S.; Carpin, S.; Ojea, J.A.; Goldberg, K. Towards Automating Precision Irrigation: Deep Learning to Infer Local Soil Moisture Conditions from Synthetic Aerial Agricultural Images. In Proceedings of the 2018 IEEE 14th International Conference on Automation Science and Engineering (CASE), Munich, Germany, 20–24 August 2018; pp. 284–291.
17. Song, X.; Zhang, G.; Liu, F.; Li, D.; Zhao, Y.; Yang, J. Modeling spatio-temporal distribution of soil moisture by deep learning-based cellular automata model. *J. Arid. Land* **2016**, *8*, 734–748.
18. Saggi, M.K.; Jain, S. Reference evapotranspiration estimation and modeling of the Punjab Northern India using deep learning. *Comput. Electron. Agric.* **2019**, *156*, 387–398.
19. de Oliveira e Lucas, P.; Alves, M.A.; de Lima e Silva, P.C.; Guimarães, F.G. Reference evapotranspiration time series forecasting with ensemble of convolutional neural networks. *Comput. Electron. Agric.* **2020**, *177*, 105700.
20. Ahmed, A.A.M.; Deo, R.C.; Raj, N.; Ghahramani, A.; Feng, Q.; Yin, Z.; Yang, L. Deep Learning Forecasts of Soil Moisture: Convolutional Neural Network and Gated Recurrent Unit Models Coupled with Satellite-Derived MODIS, Observations and Synoptic-Scale Climate Index Data. *Remote Sens.* **2021**, *13*, 554. https://doi.org/10.3390/rs13040554.
21. Adab, H.; Morbidelli, R.; Saltalippi, C.; Moradian, M.; Ghalhari, G.A.F. Machine Learning to Estimate Surface Soil Moisture from Remote Sensing Data. *Water* **2020**, *12*, 3223. https://doi.org/10.3390/w12113223.
22. Jimenez, A.F.; Ortiz, V, B.; Bondesan, L.; Morata, G.; Damianidis, D. Evaluation of two recurrent neural network methods for prediction of irrigation rate and timing. *Trans. ASABE* **2020**, *63*, 1327–1348. https://doi.org/10.13031/trans.13765.
23. Bu, F.; Wang, X. A smart agriculture IoT system based on deep reinforcement learning. *Future Gener. Comput. Syst.* **2019**, *99*, 500–507.
24. Chen, M.; Cui, Y.; Wang, X.; Xie, H.; Liu, F.; Luo, T.; Zheng, S.; Luo, Y. A reinforcement learning approach to irrigation decision-making for rice using weather forecasts. *Agric. Water Manag.* **2021**, *250*, 106838.
25. Alibabaei, K.; Gaspar, P.D.; Assunção, E.; Alirezazadeh, S.; Lima, T.M. Irrigation optimization with a deep reinforcement learning model: Case study on a site in Portugal. *Agric. Water Manag.* **2022**, *263*, 107480.
26. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Harley, T.; Lillicrap, T.P.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on International Conference on Machine Learning, New York, NY, USA, June 19 - 24, 2016, ICML'16, 2016; Volume 48. JMLR.org, pp. 1928–1937.
27. Hoogenboom, G.; Porter, C.H.; Boote, K.J.; Shelia, V.; Wilkens, P.W.; Singh, U.; White, J.W.; Asseng, S.; Lizaso, J.I.; Cadena, P.M.; et al. The DSSAT crop modeling ecosystem. In *Advances in Crop Modelling for a Sustainable Agriculture*; Burleigh Dodds Science Publishing (Cambridge, United Kingdom): 2019; pp. 173–216. https://doi.org/10.19103/as.2019.0061.10.
28. Hoogenboom, G.; Porter, C.; Shelia, V.; Boote, K.; Singh, U.; White, J.; Hunt, L.; Ogoshi, R.; Lizaso, J.; Koo, J.; et al. *Decision Support System for Agrotechnology Transfer (DSSAT) Version 4.7.5*; DSSAT Foundation: Gainesville, FL, USA, 2019.

29. 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*. https://doi.org/10.3390/en14113004.
30. 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.
31. Montgomery, D.C.; Jennings, C.L.; Kulahci, M. *Introduction to Time Series Analysis and Forecasting*; Wiley Series in Probability and Statistics; Wiley: Wiley 2011.
32. 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.
33. Jain, L.C.; Medsker, L.R. *Recurrent Neural Networks: Design and Applications*, first ed.; CRC Press, Inc.: Boca Raton, FL, USA, 1999.
34. 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.
35. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, second ed.; The MIT Press: Cambridge, MA, USA, 2018.
36. Konda, V.R.; Tsitsiklis, J.N. *Actor–Critic Algorithms*; 2000, MIT Press: Cambridge, United States, pp. 1008–1014.
37. Sutton, R.S. Learning to Predict by the Methods of Temporal Differences. *Mach. Learn.* **1988**, *3*, 9–44.
38. Grondman, I.; Busoniu, L.; Lopes, G.A.D.; Babuska, R. A Survey of Actor–Critic Reinforcement Learning: Standard and Natural Policy Gradients. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2012**, *42*, 1291–1307. https://doi.org/10.1109/TSMCC.2012.2218595.
39. Williams, R.J.; Peng, J. Function Optimization using Connectionist Reinforcement Learning Algorithms. *Connect. Sci.* **1991**, *3*, 241–268. https://doi.org/10.1080/09540099108946587.
40. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *NIPS Deep Learn. Workshop* **2013**, arXiv:1312.5602.
41. 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. Available online: tensorflow.org (accessed on 1 December 2020).
42. Chollet, F. and others Keras. 2015. Available online: https://keras.io (accessed on 1 December 2020).
43. Alibabaei, K.; Gaspar, P.D.; Lima, T.M. Modeling Soil Water Content and Reference Evapotranspiration from Climate Data Using Deep Learning Method. *Appl. Sci.* **2021**, *11*, 5029. https://doi.org/10.3390/app11115029.
44. Patterson, J.; Gibson, A. *Deep Learning: A Practitioner's Approach*; O'Reilly Media:Sebastopol, California 2017.
45. Rodrigues, L.C. Water Resources Fee in Portugali, 2016. Led by the Institute for European Environmental Policy. Available online: https://ieep.eu/ (accessed on 1 July 2020).