UNIVERSIDADE BEIRA INTERIOR

Application of Artificial Intelligence algorithms to support decision-making in agriculture activities

Khadijeh Alibabaei

Tese para obtenção do Grau de Doutor em
Engenharia e Gestão Industrial
(3º ciclo de estudos)

Júri
Doutor António João Marques Cardoso, professor catedrático da Universidade da Beira Interior
Doutor Hugo Pedro Martins Carriço Proença, professor catedrático da Universidade da Beira Interior
Doutor Carlos Manuel Antunes Lopes, professor associado do Instituto Superior de Agronomia da Universidade de Lisboa
Doutora Isabel Maria do Nascimento Lopes Nunes, professora associada da Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa
Doutor Pedro Miguel de Figueiredo Dinis Oliveira Gaspar, professor auxiliar da Universidade da Beira Interior

Data de realização das provas: 31/05/2023

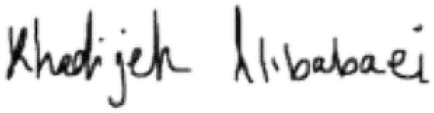
Orientador: Pedro Miguel de Figueiredo Dinis Oliveira Gaspar
Co-orientador: Tânia Daniela Felgueiras de Miranda Lima

Junho de 2023

Declaração de Integridade

Eu, Khadijeh Alibabaei, que abaixo assino, estudante com o número de inscrição D2740 do 3ºCiclo em Engenharia e Gestão Industrial, da Faculdade de Engenharia, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o Código de Integridades da Universidade da Beira Interior. Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, ex- tratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 2022 /06 /09



To my parents Zahra and Hossein

Agradecimentos

The research contained in this dissertation could not have been accomplished without the help, patience, and support of many individuals. First, I would like to express my sincere gratitude to Prof. Pedro D. Gaspar for teaching, guiding, and supervising me throughout these years. I sincerely thank him for his confidence in me and his comments to improve my work. I am also grateful for support of my friend and college Eduardo Assunção.
Finally, and most importantly, this thesis is dedicated to my husband, Saeid, who has always supported, encouraged and loved me during the challenges of my studies. I am truly grateful to have him in my life. I thank my parents, Zahra and Hossein, for their trust in me and for allowing me to be as ambitious as I wanted to be. Under their watchful eyes, I developed so much drive and the ability to overcome challenges.
This research was 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). It was also supported by 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.

Resumo

Nos últimos anos, a técnica de aprendizagem profunda (Deep Learning) foi aplicada com sucesso ao reconhecimento de imagem, reconhecimento de fala e processamento de linguagem natural. Assim, tem havido um incen tivo para aplicá-la também em outros sectores. O sector agrícola é um dos mais importantes, em que a aplicação de algoritmos de inteligência artificial e, em particular, de deep learning, precisa ser explorada, pois tem impacto direto no bem-estar humano. Em particular, há uma necessidade de explorar como os modelos de aprendizagem profunda para a tomada de decisão podem ser usados como uma ferramenta para cultivo ou plantação ideal, uso da terra, melhoria da produtividade, controlo de produção, de doenças, de pragas e outras atividades. A grande quantidade de dados recebidos de sensores em explorações agrícolas inteligentes (smart farms) possibilita o uso de deep learning como modelo para tomada de decisão nesse campo. Na agricultura, não há dois ambientes iguais, o que torna o teste, a validação e a implementação bem-sucedida dessas tecnologias muito mais complexas do que na maioria dos outros setores. Desenvolvimentos científicos recentes no campo da aprendizagem profunda aplicada à agricultura, são revistos e alguns desafios e potenciais soluções usando algoritmos de aprendizagem profunda na agricultura são discutidos. Maior desempenho em termos de precisão e menor tempo de inferência pode ser alcançado, e os modelos podem ser úteis em aplicações do mundo real. Por fim, são sugeridas algumas oportunidades para futuras pesquisas nesta área. A capacidade de redes neuronais artificiais, especificamente Long Short-Term Memory (LSTM) e LSTM Bidirecional (BLSTM), para modelar a evapotranspiração de referência diária e o conteúdo de água do solo é investigada. A aplicação destas técnicas para prever estes parâmetros foi testada em três locais em Portugal. Um BLSTM de camada única com 512 nós foi selecionado. A otimização bayesiana foi usada para determinar os hiperparâmetros, como taxa de aprendizagem, decaimento, tamanho do lote e tamanho do ”dropout”. O modelo alcançou os valores de erro quadrático médio na faixa de 0,014 a 0,056 e R^2 variando de 0,96 a 0,98. Um modelo de Rede Neural Convolucional (CNN – Convolutional Neural Network) foi adicionado ao LSTM para investigar uma potencial melhoria de desempenho. O desempenho decresceu em todos os conjuntos de dados devido à complexidade do modelo. O desempenho dos modelos também foi comparado com CNN, algoritmos tradicionais de aprendizagem máquina Support Vector Regression e Random Forest. O LSTM obteve o melhor desempenho. Por fim, investigou-se o impacto da função de perda no desempenho dos modelos propostos. O modelo com o erro quadrático médio (MSE) como função de perda teve um desempenho melhor do que o modelo com outras funções de perda. Em seguida, são investigadas as capacidades desses modelos e sua extensão, BLSTM e Bidirectional Gated Recurrent Units (BGRU) para prever os rendimentos da produção no final da campanha agrícola. Os modelos usam dados históricos, incluindo dados climáticos, calendário de rega e teor de água do solo, para estimar a produtividade no final da campanha. A aplicação desta técnica foi testada para os rendimentos de tomate e batata em um local em Portugal. A rede BLSTM superou as redes GRU, LSTM e BGRU no conjunto de dados de validação. O modelo foi capaz de captar a relação não linear entre dotação de rega, dados climáticos e teor de água do solo e prever a produtividade

com um MSE variando de 0,07 a 0,27 $(\mathrm{mm\ d^{-1}})^2$ para ETo (Evapotranspiração de Referência) e de 0,014 a 0,056 $(\mathrm{m^3 m^{-3}})^2$ para SWC (Conteúdo de Água do Solo), com valores de R^2 variando de 0,96 a 0,98. O desempenho do BLSTM no teste foi comparado com o método de aprendizagem profunda CNN, e métodos de aprendizagem máquina, incluindo um modelo Multi-Layer Perceptrons e regressão Random Forest. O BLSTM superou os outros modelos com um R^2 entre 97% e 99%. Os resultados mostram que a análise de dados agrícolas com o modelo LSTM melhora o desempenho do modelo em termos de precisão. O modelo CNN obteve o segundo melhor desempenho. Portanto, o modelo de aprendizagem profunda tem uma capacidade notável de prever a produtividade no final da campanha. Além disso, uma Deep Q-Network foi treinada para programação de irrigação para a cultura do tomate. O agente foi treinado para programar a irrigação de uma plantação de tomate em Portugal. Dois modelos LSTM treinados anteriormente foram usados como ambiente de agente. Um prevê a água total no perfil do solo no dia seguinte. O outro foi empregue para estimar a produtividade com base nas condições ambientais durante uma o ciclo biológico e então medir o retorno líquido. O agente usa essas informações para decidir a quantidade de irrigação. As redes LSTM e CNN foram usadas para estimar a Q-table durante o treino. Ao contrário do modelo LSTM, a RNA e a CNN não conseguiram estimar a tabela Q, e a recompensa do agente diminuiu durante o treino. A comparação de desempenho do modelo foi realizada entre a irrigação com base fixa e a irrigação com base em um limiar. A aplicação das doses de rega preconizadas pelo modelo aumentou a produtividade em 11% e diminuiu o consumo de água em 20% a 30% em relação ao método fixo. Além disso, um modelo dentro da táctica, Advantage Actor–Critic (A2C), é foi implementado para comparar a programação de irrigação com o Deep Q-Network para a mesma cultura de tomate. Os resultados mostram que o modelo de táctica A2C reduziu o consumo de água consumo em 20% comparado ao Deep Q-Network com uma pequena mudança na recompensa líquida. Estes modelos podem ser desenvolvidos para serem aplicados a outras culturas com elevada produção em Portugal, como a fruta, cereais e vinha, que também têm grandes necessidades hídricas. Os modelos desenvolvidos ao longo desta tese podem ser reavaliados e treinados com dados históricos de outras culturas com elevada importância em Portugal, tais como frutas, cereais e uvas, que também têm elevados consumos de água. Assim, poderão ser desenvolvidos sistemas de apoio à decisão e de recomendação aos agricultores de quando e quanto irrigar. Estes sistemas poderão ajudar os agricultores a evitar o desperdício de água sem reduzir a produtividade. Esta tese visa contribuir para os passos futuros na evolução da agricultura de precisão e da robótica agrícola. Os modelos desenvolvidos ao longo desta tese são relevantes para apoiar a tomada de decisões em atividades agrícolas, direcionadas à otimização de recursos, redução de tempo e custos, e maximização da produção.

Palavras-chave

Agricultura, Inteligência Artificial, Aprendizagem profunda, Long-Short Term Memory (LSTM), Gestão da irrigação, Aprendizagem profunda por reforço, Estimativa de rendimento, Agricultura inteligente, Algoritmos de apoio á tomada de decisão, Internet of Things (IoT), robótica, Detecção de tronco.

Abstract

Deep Learning has been successfully applied to image recognition, speech recognition, and natural language processing in recent years. Therefore, there has been an incentive to apply it in other fields as well. The field of agriculture is one of the most important in which the application of artificial intelligence algorithms, and particularly, of deep learning needs to be explored, as it has a direct impact on human well-being. In particular, there is a need to explore how deep learning models for decision-making can be used as a tool for optimal planting, land use, yield improvement, production/disease/pest control, and other activities. The vast amount of data received from sensors in smart farms makes it possible to use deep learning as a model for decision-making in this field. In agriculture, no two environments are exactly alike, which makes testing, validating, and successfully implementing such technologies much more complex than in most other sectors. Recent scientific developments in the field of deep learning, applied to agriculture, are reviewed and some challenges and potential solutions using deep learning algorithms in agriculture are discussed. Higher performance in terms of accuracy and lower inference time can be achieved, and the models can be made useful in real-world applications. Finally, some opportunities for future research in this area are suggested. The ability of artificial neural networks, specifically Long Short-Term Memory (LSTM) and Bidirectional LSTM (BLSTM), to model daily reference evapotranspiration and soil water content is investigated. The application of these techniques to predict these parameters was tested for three sites in Portugal. A single-layer BLSTM with 512 nodes was selected. Bayesian optimization was used to determine the hyperparameters, such as learning rate, decay, batch size, and dropout size. The model achieved mean square error (MSE) values ranging from 0.07 to 0.27 $(\mathrm{mm\ d^{-1}})^2$ for ETo (Reference Evapotranspiration) and 0.014 to 0.056 $(\mathrm{m^3 m^{-3}})^2$ for SWC (Soil Water Content), with R^2 values ranging from 0.96 to 0.98. A Convolutional Neural Network (CNN) model was added to the LSTM to investigate potential performance improvement. Performance dropped in all datasets due to the complexity of the model. The performance of the models was also compared with CNN, traditional machine learning algorithms Support Vector Regression, and Random Forest. LSTM achieved the best performance. Finally, the impact of the loss function on the performance of the proposed models was investigated. The model with the mean square error (MSE) as loss function performed better than the model with other loss functions. Afterwards, the capabilities of these models and their extension, BLSTM and Bidirectional Gated Recurrent Units (BGRU) 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 BLSTM network outperformed the GRU, the LSTM, and the BGRU networks 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 $\mathrm{kg/ha}$. The performance of the BLSTM in the test was compared with the most commonly used deep learning method called CNN, and machine learning methods including a Multi-Layer Perceptrons model and Random Forest regression. The BLSTM out-

performed the other models with a R^2-score between 0.97 and 0.99. The results show that analyzing agricultural data with the LSTM model improves the performance of the model in terms of accuracy. The CNN 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. Additionally, a Deep Q-Network was trained for irrigation scheduling. The agent was trained to schedule irrigation for a tomato field in Portugal. Two LSTM models trained previously were used as the agent environment. One predicts the total water in the soil profile on the next day. The other one was employed to estimate the yield based on the environmental condition during a season and then measure the net return. The agent uses this information to decide the following irrigation amount. LSTM and CNN networks were used to estimate the Q-table during training. Unlike the LSTM model, the ANN and the CNN could not estimate the Q-table, and the agent's reward decreased during training. The comparison of the performance of the model was done with fixed-base irrigation and threshold-based irrigation. The trained model increased productivity by 11% and decreased water consumption by 20% to 30% compared to the fixed method. Also, an on-policy model, Advantage Actor–Critic (A2C), was implemented to compare irrigation scheduling with Deep Q-Network for the same tomato crop. The results show that the on-policy model A2C 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 importance in Portugal, such as fruit, cereals, and grapevines, which also have large water requirements. The models developed along this thesis can be re-evaluated and trained with historical data from other cultures with high production in Portugal, such as fruits, cereals, and grapes, which also have high water demand, to create a decision support and recommendation system that tells farmers when and how much to irrigate. This system helps farmers avoid wasting water without reducing productivity. This thesis aims to contribute to the future steps in the development of precision agriculture and agricultural robotics. The models developed in this thesis are relevant to support decision-making in agricultural activities, aimed at optimizing resources, reducing time and costs, and maximizing production.

Keywords

Agriculture, Deep learning, LSTM, irrigation management, Deep Reinforcement Learning, yield estimation, Smart farming, support decision-making algorithms, IoT, Robot, Trunk detection.

Contents

List of Figures

List of Tables

Acronym List

A2C	Advantage Actor–Critic
AC	Accuracy
AI	Artificial intelligent
AP	Average Precision
AUC	Area Under the ROC Curve
BLSTM	Bidirectional LSTM
CNN	convolutional neural network
DSSAT	Decision Support System for Agrotechnology Transfer
DL	Deep Learning
DQN	deep Q-learning
ETo	reference evapotranspiration
FAO	Food and Agriculture Organization
FCN	fully Convolutional Networks
FN	False Negative
FP	False Positive
GBM	Gradient Boosting Models
GMM	Gaussian Mixture Model
GLM	Generalized Linear Model
GRU	Gated Recurrent Units
$\mathrm{HR}_{\mathrm{Avg}}$	average humidity
IoT	Internet of Things
IOU	Intersection Over Union
irr	irrigation
MAE	Mean Absolute Error
MBE	Mean Bias Error
MDP	Markov Decision Process
mAP	Mean Average Precision
MSE	Mean Square Error
ML	Machine Learning
MLP	multilayer perceptrons
NIR	near-infrared
NAG	Nesterov Accelerated Gradient
NN	Neural Network
LSTM	long term short term memory
P	Precision
R	Recall

RF	Random Forest
RMSE	Root Square Error
RMSprop	Root Means Square propagation
RNN	Recurrent Neural Networks
RL	Reinforcement Learning
RLU	Rectified Linear Units
PPN	Pooling Pyramid Network
Prec	precipitation
SGD	Stochastic Gradient Descent
SSD	Single Shot MultiBox Detector
SGM	Semi Global Matching
$\mathrm{SR_{Avg}}$	solar radiation
SVM	Support Vector Machines
SVR	Support Vector Regression
SWTD	total soil water in profile
WCTot	total soil profile
TD	temporal difference method
$\mathrm{T_{min}}$	Minimum Temperature
$\mathrm{T_{max}}$	Maximum Temperature
TP	True Positive
UAV	Unmanned Aerial Vehicles
XGBoost	Extreme Gradient Boosting
VIF	Variance Inflation Factor
YOLO	You Look Once
$\mathrm{WS_{Avg}}$	average wind speed

Chapter 1

Introduction

Today, about half of the EU territory is arable land, and agriculture remains the main economic activity in most rural areas. However, agriculture has environmental impacts, such as air pollution and the release of greenhouse gases that contribute to climate change, water consumption, waste production, erosion and soil degradation, resource pollution and its impacts on populations, communities, and ecosystem services, fertilizer use, resource pollution, ecosystem acidification, and more [1, 2, 3]. Moreover, the world's population living in urban areas will increase by 13% by 2050 [4]. On the other hand, with the current agricultural production method, the capacity of the Earth is already exceeded [5]. Therefore, improving agricultural production and feeding the world, taking into account climate change, temperature rise leading to a reduction in water availability, plagues, pests and diseases, remains the main problem in agriculture.

Precision agriculture is an agricultural management system based on the spatial and temporal variability of the unit of production, which allows for a more rational exploration of production systems, leading to optimization of input use, increased profitability, and sustainability, and minimization of environmental impact [6, 7]. Precision agriculture enables small, medium, and large producers to manage their land by using inputs at the right time, in the right place, and in the right quantity, thereby increasing productivity and sustainability [8, 6, 7, 9].

Smart farming is part of precision agriculture that include farm management systems that use new technologies to increase food safety, quality, and quantity and reduce environmental impact. Increased control of production leads to better cost management and reduced waste [5, 10]. These farms use sensors to monitor and control animal behavior and plant [5, 10, 11]. With the help of sensors in the field, we can collect data such as soil mapping, climate parameters, and weather data, among others [5, 10, 11]. The combined analysis of this data can provide information to make the appropriate management decisions. Artificial Intelligence (AI) receives significant attention in the development of decision-making algorithms. AI is any technique that enables machines to learn from experience, adapt to new inputs, and mimic human behavior [12, 13]. Since the term was first used in the 1950s, artificial intelligence has spread to several industries, including transportation, medicine, manufacturing and agriculture. Strategies have also changed, starting with the basic AI algorithms of the 1950s, through symbolic algorithms, the development of expert systems, the introduction of machine learning in the 1990s, and the development of deep learning algorithms in the 2010s [14, 15].

Machine Learning (ML) is a sub-area of AI. The field of ML began to move from a knowledge-based strategy to a data-based approach, paving the way for today's machine learning models. This transition was fueled by the growth of the Internet and the increasing availability of actionable data [15]. ML algorithms uses computational algorithms to convert raw data

from the real world into useful models and decision advice. By using ML models, the system can automatically learn from previous experiences and improve itself. The direction of machine learning development changed in the 1990s. ML techniques include Support Vector Machines (SVM), decision trees, Bayesian learning, K-means clustering, association rule learning, regression, neural networks, and many others [12]. Liakos et al. [16] presented an overview of the application of the ML model in various agricultural activities.

Deep Learning (DL) is a subfield of machine learning. DL algorithms can be used throughout the growing 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 [12]. Deep here refers to the number of hidden layers of the neural network. The performance of the model improves as the network becomes deeper [12]. In 2012, the LSVRC classification competition was won by the DL model known as AlexNet [17]. Sermanetet al. [18] showed that DL algorithms could be used for classification, recognition, and localization and achieved excellent results. These successes motivate researchers to apply DL models to other areas of human activity, such as agriculture. In Kamilaris and Prenafeta-Boldú [19], as of 2018, 43 papers have been listed that deal with the application of the DL technique in agriculture including species classification, fruit detection and yield estimation, soil moisture prediction, weather prediction and more.

Regarding the diversity of environments in agriculture, the result of a trained model in a given environment may not be transferable to others [20]. This makes the application of deep learning models in agriculture more challenging and requires more attention. Chapter 2 aims to review newly published papers to find new techniques, challenges, and potential solutions for the application of Deep Learning in agriculture since 2018. Forty-six recent articles were reviewed in this chapter. Of these, ten articles were related to plant diseases, eleven articles were related to fruit detection, five articles were related to weed detection, six articles were related to species detection, three articles were related to soil management, five articles were related to water management, and six articles were related to automation in agriculture. This work was published in [21].

In Chapter 3, a Long Short-Term Memory (LSTM) was developed to predict two critical parameters: Evapotranspiration (ET) and Soil Water Content (SWC). Soil Water Content is the volume of water per unit volume of soil and ET is the sum of the water that evaporates from the soil surface with the water transpired by the plants. Solar radiation, wind speed, temperature, and relative humidity all influence daily ET, and this effect is highly non-linear. The model in Chapter 3 forecasts one day ahead ET and Soil Water Content using eight days of historical data. The various evaluation metrics such as Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Bias Error (MBE), and R^2 were used to demonstrate the effectiveness of the models. Each metric has its advantages and disadvantages, and the results of this chapter shows that using only one metric can lead to misleading results. The Convolutional Neural Network (CNN) is capable of extracting features from raw data [22]. A CNN model is superimposed on the LSTM model to investigate the performance of CNN- LSTM models compared to LSTM. The performance of the model

is also compared with CNN, and traditional machine learning algorithms such as Support Vector Regression (SVR) and Random Forest [23, 24]. Finally, the impact of the loss function on the training of the model is investigated. The most commonly used loss functions for regression problems are selected, including MSE, RMSE, MAE, and the Huber function. The Huber function is basically MAE, which becomes MSE when the error is small. Unlike MAE, the Huber function is differentiable at 0 and is less sensitive to outliers in the data than MSE [25]. Each of these functions yields a different error for the same prediction and affects the performance of the model on the test set. This work was published in [26].

In Chapter 4 , recurrent neural networks (RNN), including the LSTM model and Gated Recurrent Units (GRU) model and their extension 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, Reference Evapotranspiration (ETo), the water content in the soil profile, and irrigation scheduling during a season [1]. The performance of the model is 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). This work was published in [27].

In Chapter 5, a deep Q(LSTM) network model was trained that simply tells farmers when and how much to irrigate to achieve the best productivity without wasting water for a tomato field. Historical data from 3 and 4 were used as input to the models. Then, two LSTM models from 3 and 4 are used to predict soil water content for the next day and tomato yield at the end of a season, respectively. Training the LSTM models is a unique process and after training, they are used as a feature in the DRL training environment, which takes the current state (s) (historical climate data) and action a (amount of irrigation), and then returns the next state s' and reward r. As the agent trains, it selects an action for the next irrigation based on the current state of the field and evaluates that action using a function called Q-value. The environment receives the current state and the action chosen by the agent and indicates the next state and the reward. This interaction between the DRL agent and training environment is repeated until the DRL agent converges to an optimal strategy for selecting the next day's irrigation amount. This work was published in [28].

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 [29].

To determine how well the A2C model performs in this task of irrigation scheduling for a tomato field, Chapter 6 compares the performance of the model with that of DQN. The model

[1]The codes are available at the following links:
https://github.com/falibabaei/tomato-yieldestimation/blob/main/main
https://github.com/falibabaei/potato-yield-estimation/tree/main

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. This work was published in [30].

Our goal is to explore the potential of precision agriculture and machine learning techniques to contribute to sustainable agriculture. Specifically, we aim to develop and evaluate deep learning models such as convolutional neural networks (CNN), recurrent neural networks (RNN) and their combination (CNN-LSTM), and deep reinforcement learning (DRL) methods. These models have the potential to improve agricultural efficiency, increase productivity, reduce environmental impact, and address climate change challenges.

In this work, we will apply these models to various agricultural scenarios, including predicting evapotranspiration (ET) and soil water content (SWC), estimating crop yields, and optimizing irrigation methods. Using these models, farmers can make informed decisions about when, where, and how much to irrigate, leading to reduced water waste, more efficient water use, and conservation of valuable water resources.

Chapter 2

A Review on challenges of using Deep Learning Algorithms to Support Decision-Making in Agricultural Activities

Abstract[1]

Deep Learning has been successfully applied to image recognition, speech recognition, and natural language processing in recent years. Therefore, there has been an incentive to apply it in other fields as well. The field of agriculture is one of the most important in which the application of deep learning needs to be explored, as it has a direct impact on human well-being. In particular, there is a need to explore how deep learning models can be used as a tool for optimal planting, land use, yield improvement, production/ disease/ pest control, and other activities. The vast amount of data received from sensors in smart farms makes it possible to use deep learning as a model for decision- making in this field. In agriculture, no two environments are exactly alike, which makes testing, validating, and successfully implementing such technologies much more complex than in most other industries. This paper reviews some recent scientific developments in the field of deep learning, applied to agriculture, and highlights some challenges and potential solutions using deep learning algorithms in agriculture. The results in this paper indicate that by employing new methods from deep learning, higher performance in terms of accuracy and lower inference time can be achieved, and the models can be made useful in real-world applications. Finally, some opportunities for future research in this area are suggested.

Keywords

Agriculture, Deep learning, Smart farm, support decision-making algorithms.

2.1 Introduction

European landscapes have been transformed primarily by human activities so that pristine vegetation and real wilderness are nowadays extremely rare in the European Union (EU).

[1]This work was published in [21], K. Alibabaei, P. D. Gaspar, T. M. Lima, R. M. Campos, I. Girão, J. Monteiro, and C. M. Lopes, "A review of the challenges of using deep learning algorithms to support decision-making in agricultural activities," Remote Sensing, vol. 14, no. 3, 2022. [Online]. Available:https://www.mdpi.com/2072-4292/14/3/638

Agriculture plays a particularly important role in the development of European landscapes, as it now accounts for almost half of the total area of the EU [31]. On the other hand, we are already exceeding the Earth's capacity with the current type of agricultural production. Climate change, degradation of soils, pollution, rising costs of groundwater and pump irrigation, the transition from a fuel-based to a bio-based economy, the scarcity of freshwater as demand increases, and other environmental and economic issues will make access to fresh food an ever-greater challenge [5, 32].

Moreover, it has already been shown that some of the aforementioned practices such as climate changes and scarcity of freshwater lead to stagnation and sometimes even a decline in production. Thus, the question arises whether transforming all agricultural systems into high-intensity farming systems alone is not counterproductive in an attempt to solve the world's food problems. It is understandable, then, that the implementation of practices adapted to the reality of a given production will have positive effects in terms of better use of resources and even a different approach to their use.

One way to address this challenge is to use new technologies, supplement workers with robots and machines, and better address biodiversity parameters and functions as well as ecosystem state. Smart farms include farm management systems that use new technologies to increase food safety, quality, and quantity and reduce environmental impact. Increased control of production leads to better cost management and reduced waste [5, 10]. These farms use sensors to monitor, control, and treat animals and plants [5, 10, 11]. With the help of sensors in the field, we can collect data such as soil mapping, climate changes, weather data, among others [5, 10, 11]. The combined analysis of this data can provide information to make the appropriate management decisions. Artificial Intelligence (AI) receives significant attention in the development of decision-making algorithms. AI is any technique that enables machines to learn from experience, adapt to new inputs, and mimic human behavior [12, 13].

Machine Learning (ML) is a sub-area of AI that uses computational algorithms to convert raw data from the real world into useful models and decision advice. By using ML models, the system can automatically learn from previous experiences and improve itself. ML techniques include Support Vector Machines (SVM), decision trees, Bayesian learning, K-means clustering, association rule learning, regression, neural networks, and many others [12]. Liakos et al. [16] presented an overview of the application of the ML model in various agricultural activities of agriculture.

Deep learning (DL) is a subfield of ML. The algorithms of DL are more complex than traditional ML models. In a network, the layers between input and output are called hidden layers. A shallow network has one hidden layer, while a deep network has more than one. With multiple hidden layers, deep neural networks can learn the features of data and solve more complex problems [12, 13]. Since the models of DL are faster and more efficient compared to the shallow algorithms in ML, and unlike them, they can automatically extract features from the input data, they have been the most widely used models in recent years [12, 13]. In 2012, AlexNet [33] won the LSVRC competition for classification. Sermanet et al. [18] showed that DL models could be used for classification, recognition, and localization and achieved excellent results. These achievements encourage researchers to apply DL models to other areas

of human life, including agriculture.
Liu et al. [34] reviewed advances in combining edge intelligence and remote sensing with Unmanned Aerial Vehicles (UAV) in precision agriculture. This paper presented a list of recent works using UAVs in combination with DL models. Since the focus of the paper was on UAV devices, this work was listed based only on the application and model used in the paper and did not include reviews and analyses of the work.
Regarding the diversity of environments in agriculture, the result of a trained model on one environment may not be transferable to others [20]. This makes the application of deep learning models in agriculture more challenging and requires more attention. This paper aims to review newly published works to find new techniques, challenges, and possible solutions for using deep learning in agriculture since 2018. The following are the main objectives of this paper:

- To provide an overview of recent work and identify challenges related to data acquisition and preparation for deep learning models;
- To review of the new DL model algorithms used in agriculture;
- Highlighting the challenges in training the model;
- Examine the challenges associated with applying the trained model in the real world;
- Review the new edge devices used to implement the trained models.

2.2 Materials and Methods

The two most reputable databases for scientific papers, the Web of Science and ScienceDirect, were searched for relevant journal articles and conference proceedings. The term ”deep learning” was used in the keywords of the screening titles with one of the following keywords: ”agriculture”, ”plant diseases”, ”weed detection”, ”yield estimation”, ”fruit detection”, ”species identification”, ”crop classification”, ”soil management”, ”automation and robotics”, and ”irrigation”.
The articles that did not deal with Deep Learning in agriculture were excluded. In the end, forty-six articles remained. Of these, ten articles were related to plant diseases, eleven articles were related to fruit detection, five articles were related to weed detection, six articles were related to species detection, three articles were related to soil management, five articles were related to water management, and six articles were related to automation in agriculture.
The following criteria were considered in the analysis of the articles:

- The way the dataset was collected for training the model and the challenges regarding the dataset.
- The DL models/architectures used in the article and the performance of the models.
- The metrics that were used to evaluate the model.

- The inference time of the model (if specified), as it is very important and critical for the use of the model in the real-time application.
- The analysis of the failure predicted by the model.
- Whether the authors used a low-cost device to deploy the trained model.

These articles based on the above criteria are discussed in Section 2.3.8 and listed in the tables 2.1 to 2.9.

2.3 An overview of Deep Learning

DL models were inspired by human neural networks in the brain. The word ”deep” refers to the number of hidden layers through which the data is transformed. Through training, they send the input through a deep network with different layers, where each layer hierarchically extracts specific features at different scales or resolutions from the data and combines them into a higher-level feature, and uses these features to make predictions [12].
DL models are divided into supervised learning, unsupervised learning, and reinforcement learning [12]. Supervised learning is the task of learning a function from labeled training data. In supervised learning , all data in the dataset is a pair consisting of an input object and the desired output value. A supervised learning model analyzes the training data and produces a function that can be used for prediction [12]. The most commonly used DL models included in this review are Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN). Unsupervised learning was used in three papers.

2.3.1 Convolutional Neural Network

The Convolutional Neural Network (CNN) model is a supervised learning model specifically designed for classification, recognition, and segmentation. A CNN model consists of a stack of convolutional layers, nonlinear functions, pooling layers, and fully connected layers. A convolutional layer contains a set of kernels whose parameters must be learned and is used to extract features from the data. Each kernel slides over the height and width of the input, and the dot product between the kernel entries and each position of the input is computed (Figure 2.1). The output of a convolutional layer can be calculated by Equation (2.1):

$$y_i = X * W_i + b_i, \qquad i = 1, \cdots, m \quad \textbf{(2.1)}$$

where X is the input of the layer, y_i is the output corresponding to the i-th convolutional kernel, W_i is a matrix of the trainable weights of the i-th kernel, b_i is the i-th bias, and m is the number of kernels.
After each convolutional layer, a nonlinear activation function is used to introduce nonlinear features into the model. Rectified Linear Units (ReLU) have become state-of-the-art and are the most commonly used activation function in Deep Learning models [12]. A pooling

layer is usually located between two convolutional layers and is used to reduce the number of parameters to minimize overfitting [12, 13].

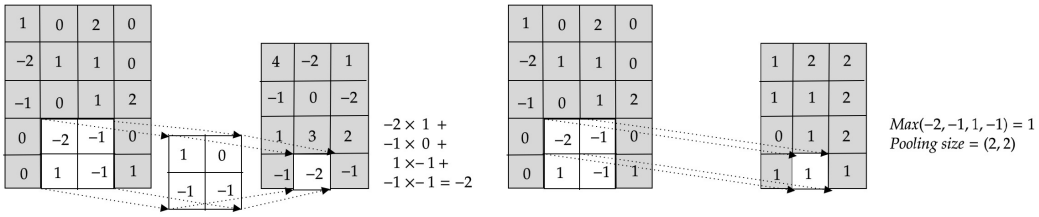


Figure 2.1: The left side image is the example of the convolutional operation, and the right side is the example of a Max pooling operation with pooling size (2,2).

The final layers consist of fully connected layers that take the features extracted by the previous layer and generate class probabilities or scores. These layers are fully connected to all neurons in the previous layer. Figure 2.2 shows the architecture of the well-known CNN model called VGG19 and the visualization of the last layer of the VGG19 model in plant disease classification, respectively.

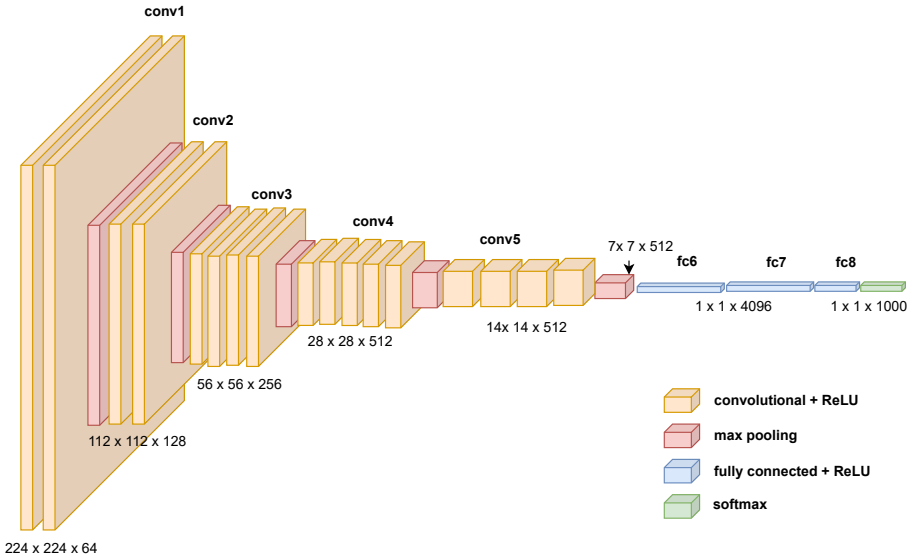


Figure 2.2: Architecture of the well-known CNN model VGG19. The VGG19 network contains convolutional layers, max-pool layers, fully connected layers, and a soft-max classification layer.

2.3.1.1 Classification/Regression by CNN

The CNN model can be used for classification or regression problems. For the classification task, the last layer of the model is chosen as a fully connected layer with a softmax activation function, and for the regression task as a fully connected layer, usually with a linear function. The most popular CNN architectures for classification include AlexNet [33], GoogleNet [35], VGG [36], ResNet [37], the new lightweight models like MobileNet [38], EfficientNet [39]. The lightweight models enable to use mobile and low-cost edge devices like Jetson nano (`https://developer.nvidia.com/embedded/jetson-nano-developer-kit`), Raspberrypi (`https://www.raspberrypi.com/`) and Google TPU Coral (`https://coral.ai/`) to deploy

the trained model in robots and use them in the field [40, 34] (Figure 2.3).

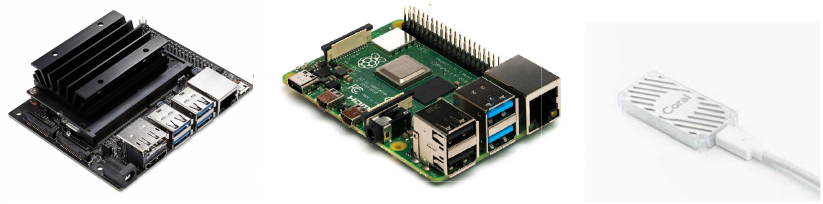


Figure 2.3: From left to right: Jetson nano, Raspberry pi, and Google.

2.3.1.2 Detection by CNN

The CNN model can also be used for object detection. There are two types of modern object detection algorithms, the two-stage detectors such as R-CNN [41], Fast R-CNN [42], Faster R-CNN [43] and single-stage detectors such as You Look Once (YOLO) [44], Single Shot multi-box Detector (SSD) [45], and LedNet [46]. In the two-stage detector , regions of interest are generated in one stage using a Region Proposal Network (RPN). In the other stage, the proposed regions are evaluated for object classification and bounding box regression using convolutional layers. Such models achieve high accuracy rates but are very slow [45]. Single-stage detectors , on the other hand, contain a single feed-forward convolutional network that directly provides the bounding boxes and object classification. The single-stage detectors are faster in detection but less accurate than two-stage detectors, and it depends on the application which model should be chosen. Faster R-CNN and SDD architecture are shown in Figure 2.4.

2.3.1.3 Segmentation by CNN

The segmentation task can also be performed with CNN models. In segmentation , an image is divided into groups of pixels, and each group is assigned a class. Figure 2.6 shows the segmentation of the weed and crop in the image.
The fully Convolutional Networks (FCN) [49], Mask R-CNN [50] and Deeplab [51] are DL models for segmentation. Lottes et al. [52] used an encoder-decoder Fully Convolutional Network (FCN) model to identify crops and weeds during field operations. Ghiani et al. [53] used Mask R-CNN with ResNet101 which was trained with the dataset COCO, as a backbone for detecting grape branches in the tree.

2.3.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is specially designed for handling sequential data. In RNN, the output from the previous step is fed into the current step as input. The output of an RNN unit is determined using a tanh function by Equation (2.2):

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_t) \quad (2.2)$$

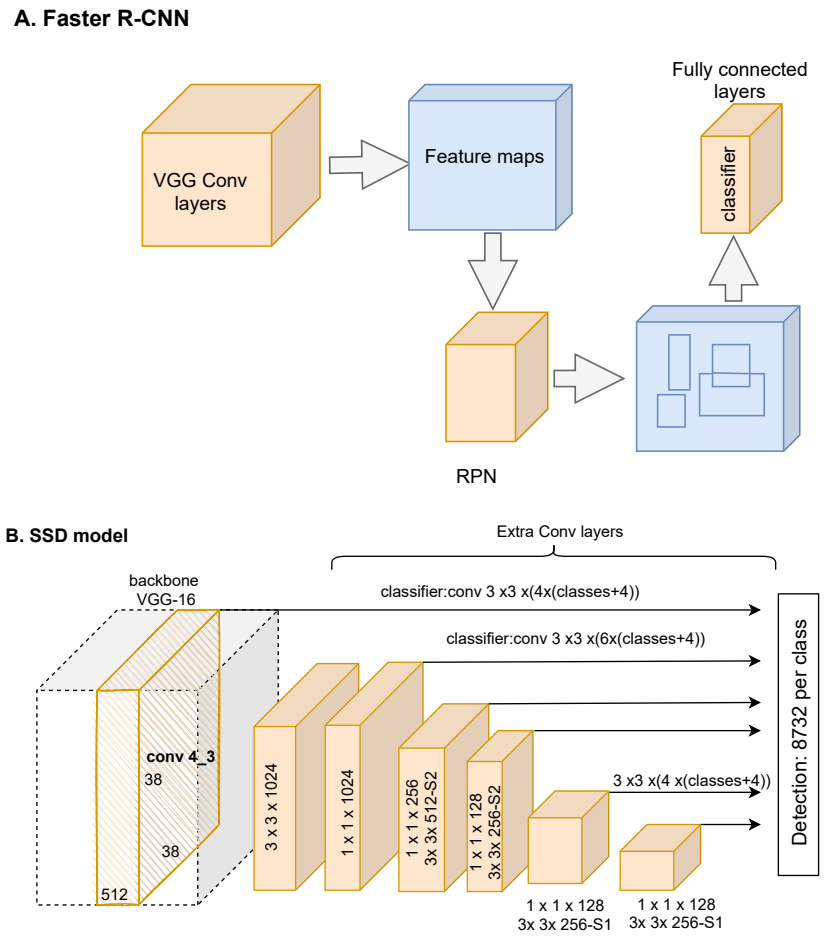


Figure 2.4: A. Faster R-CNN model, B. SSD model with VGG16 as the backbone. In the faster R-CNN, a series of convolutional layers (Conv layer) is used to extract features from the input, and then an RPN algorithm is used to find the region of interest, but in the SSD model, the features and bounding box are extracted using only convolutional layers.

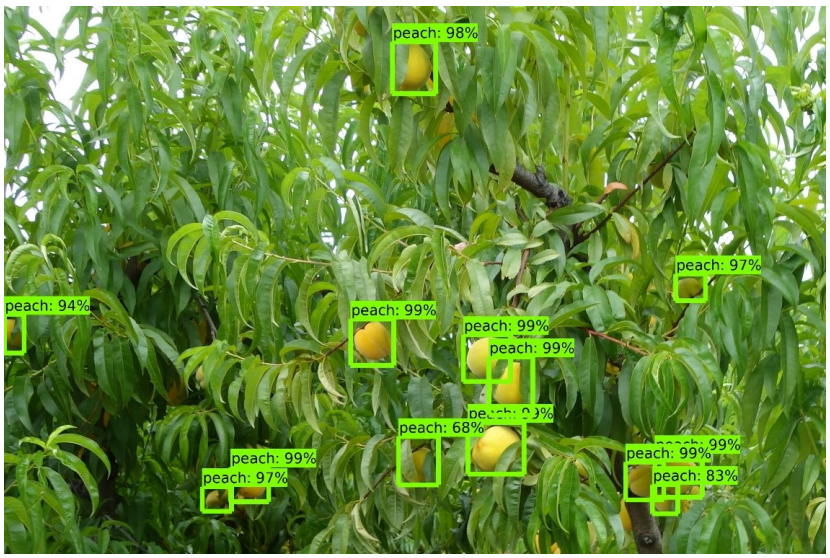


Figure 2.5: Detection of peaches in the image using Faster R-CNN model. Each bounding box contains peach and the score above the box shows the confidence score for each detected object [47].

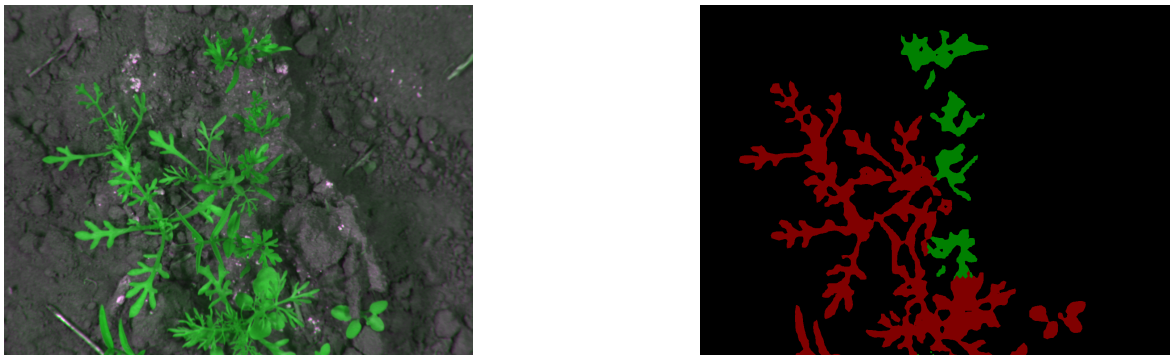


Figure 2.6: The left side shows the orginal image and the right side shows the task of segmentation for weed (red) and crop (green) [48].

The main problem with RNN is the vanishing gradient , i.e., the gradient of the loss function approaches zero [54]. Long Short-Term Memory Networks (LSTM) [55] is a special type of RNN created to solve the vanishing problem in RNN. A standard LSTM unit consists of an input gate; an input gate (2.3), forget gate (2.4), an output gate (2.7), and a memory cell (2.6). Figure 2.7 shows an LSTM unit and an RNN unit.

$$\mathrm{i_t} = \sigma(\mathrm{W_{ix}x_t} + \mathrm{W_{ih}h_{t-1}} + \mathrm{b_i}), \tag{2.3}$$

$$\mathrm{f_t} = \sigma(\mathrm{W_{fx}x_t} + \mathrm{W_{fh}h_{t-1}} + \mathrm{b_f}), \tag{2.4}$$

$$\mathrm{z_t} = \tanh(\mathrm{W_{zx}x_t} + \mathrm{W_{zh}h_{t-1}} + \mathrm{b_z}), \tag{2.5}$$

$$\mathrm{c_t} = \mathrm{f_t} * \mathrm{c_{t-1}} + \mathrm{i_t} * \mathrm{z_t}. \tag{2.6}$$

$$\mathrm{o_t} = \sigma(\mathrm{W_{ox}x_t} + \mathrm{W_{oh}h_{t-1}} + \mathrm{b_o}) \tag{2.7}$$

The weights (W) and biases (b) are trainable parameters that are adjusted and optimized during the training of the model.

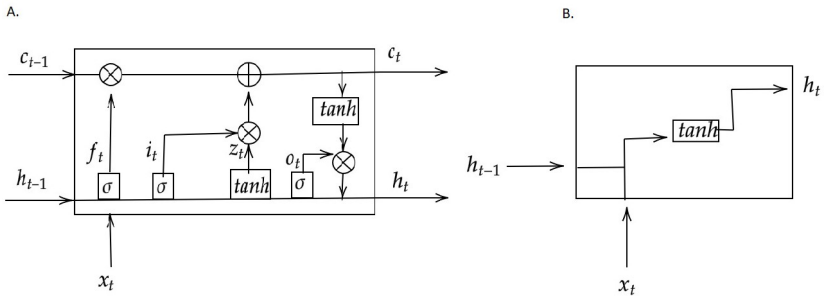


Figure 2.7: A. LSTM unit diagram, B. regular RNN unit. In an RNN unit, all input data is used for output. However, in an LSTM unit, a set of gates is used to control the information in memory and decide how long and how much information may be stored.

2.3.3 Unsupervised Learning

Unsupervised learning is a type of learning in which hidden structures are found in unlabeled data with minimal human supervision [12]. The primary method of unsupervised learning

is cluster analysis, such as K-Means clustering. Cluster analysis groups or segments datasets with common attributes. In supervised learning, labeling a large amount of agricultural data to train the model is expensive and time-consuming. Deep clustering models can be used to avoid labeling and make the model more robust. Kang & Chen [56] used automatic label generation by a clustering RCNN (C-RCNN) model to identify the apple on the trees. Tang et al. [57] used K-means clustering to label the dataset and identify weeds in the field. Ferreira et al. [58] used DeepCluster and Joint Unsupervised Learning from Deep Representations and Image Clusters (JULE) to identify weeds in the field. However, many other new deep clustering algorithms can be explored to avoid labeling in agriculture [59].

2.3.4 Reinforcement Learning

Reinforcement Learning (RL) is concerned with how an agent performs appropriate actions in an environment to maximize the rewards in a given situation. RL agents must determine the best strategy to maximize their expected cumulative rewards. An interactive environment receives the current state and the action chosen by the agent and indicates the next state and reward (see Figure 2.8). This interaction between the DRL agent and the training environment is repeated until the DRL agent converges to an optimal strategy for choosing an action.

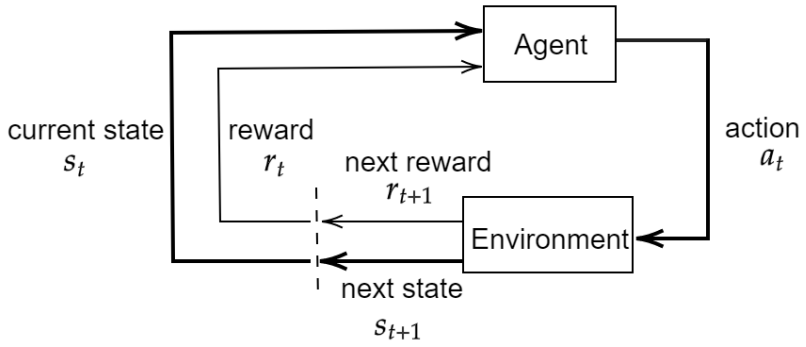


Figure 2.8: Interaction of the agent with the environment [28]. The agent chooses an action a_t in the current state s_t based on a policy. The environment receives the action and the current state and outputs the next state and the reward.

Deep reinforcement learning (DRL) combines deep learning and reinforcement learning [60]. As Bu & Wang [60] mentioned, DRL is a promising model for building smart farms. Chen et al. [61] used Deep Q-Network to build a decision-making system for rice irrigation based on weather forecasts.

2.3.5 Data Acquired and Data pre-paration

AI models convert raw data into usable models, so a training dataset is required to train a model. The more data available, the better results are obtained. A dataset can be collected using advanced technologies such as sensors mounted on ground-based platforms, satellites, and unmanned aerial vehicles (UAVs) [62, 63, 64, 65, 66, 67].
A sensor node is a specialized device capable of sensing physical parameters and processing

the sensed parameters using a microcontroller. In addition to processing, it can also transmit the processed data wirelessly to neighboring nodes or router nodes using transceiver [68]. Sensors mounted on ground-based platforms are divided into hand-held, free-standing in the field (Figure 2.9), and agricultural machinery-mounted sensors that can measure soil, plant, and weather characteristics [62, 63, 69]. The advantages of ground-based sensors are that they are less expensive than other sensors, they are not sensitive to humid conditions, and they are more suitable for small farms. The disadvantage of these types of sensors is that they are very time-consuming and, even when attached to farm machinery, the information is very inefficient for larger fields. Also, some of these sensors are inflexible in terms of the type of spectral data that can be collected [70].

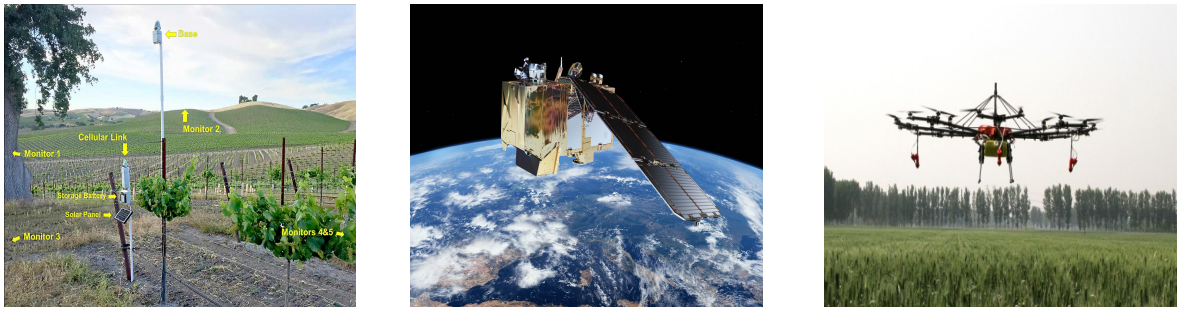


Figure 2.9: From left to right: sensors mounted on ground-based platforms to test vineyard stress (`https://www.goodfruit.com/using-sensors-to-test-vineyard-stress/`), Sentinel2 satellites (`https://www.esa.int`), and unmanned aerial vehicles (UAVs: `https://precisionagricultu.re/12-potential-uses-for-uas-in-agriculture/`).

Satellite remote sensing provides synoptic, objective, and homogeneous data that can be captured geographically and temporally, and therefore could be an efficient tool to provide high-quality information about agriculture over large areas, even across national borders [71]. Depending on the spatial and spectral resolution, repetition frequency, and cost, the multitude of satellites and sensors orbiting the Earth can be used to provide data for many types of applications [69]. The main disadvantage of satellite imagery is that it is expensive at high spatial resolution. Another disadvantage is their fixed schedule, so the data cannot be acquired in a specific time step. For example, the time step for Sentinel2 is five days. Another disadvantage of satellite images is that they highly depend on climate and cannot be post-processed on cloudy days [69]. For a list of satellites launched since 1970, we refer to [69, 72].

An aircraft system with no pilot on board that, once programmed, allows no outside intervention during flight. Originally intended for military use, they are now used in agriculture to monitor crops and collect images and data from anywhere in the field [70, 34]. Unlike satellites, unmanned aerial systems can collect data at a specific time step, and the results of the collected data have a higher resolution than that of satellites. Most unmanned aerial systems can fly in precipitation, but cloud cover can affect the data unless a second sensor is used to measure the amount of sunlight reaching the aircraft. The unmanned aerial systems can also be expensive, but there are models that are inexpensive. The disadvantages of unmanned aerial systems are the requirement of a human operator [70], limited flight duration, the limited weight of cargo carried, energy consumption [62], and dependence on weather

conditions [62, 73]. In addition, the processing of data collected by UAS is higher than that of satellite imagery [70].

In addition to using Sensor, the dataset can also be collected from free online datasets such as ImageNet, Microsoft COCO, PlantVillage [67] or images can be generated synthetically as described in [74] or using simulation software in agriculture such as AguaCrop and DSSAT[27]. For DL models, a large dataset is required for training to extract appropriate features [75]. Collating and labeling a large amount of data is time-consuming. One strategy to overcome these challenges is to increase the size of the dataset using augmentation techniques [75]. Augmentation techniques artificially create different versions of a real dataset to increase its size. These include rotation, blurring, scaling, filliping of images in the dataset [75, 76].

Another strategy is transfer learning and fine-tuning the model [77]. In transfer learning , the model is trained with a large dataset such as ImageNet or COCO and then the same model is reused and re-trained for a similar task. If the dataset does not contain enough data to train a model from scratch, this strategy can be used. Ghazi et al. [78] investigated how fine-tuning pre-trained models affects plant classification results. For this purpose, the AlexNet, GoogLeNet, and VGGNet models were trained, once from scratch, and another time using fine-tuning from Caffe Models. The results showed that fine-tuning the models improved their performance. Barbedo [75], however, investigated how the size and diversity of the dataset affect the performance of DL techniques applied to plant diseases. The results showed that even with transfer learning and augmentation techniques, CNN might require a large number of images to extract useful features from the data. Figure 2.10 shows the transfer learning task.

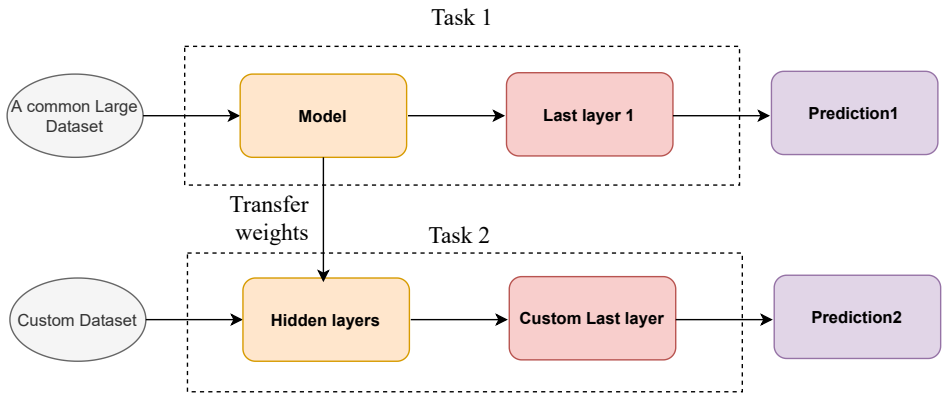


Figure 2.10: Transfer learning task. The weights and biases of a previously trained model are transferred to a similar model, the final layers are replaced with modified new layers, and then the model is trained with a custom dataset.

The next challenge is how well the trained datasets represent the real world. Ramcharan et al. [20] trained the SSD object detection model to identify three diseases in cassava. Although the model was trained on the dataset collected in the field with a camera, the F_1-score decreased by 32% on the images from other sources. Therefore, the main focus when collecting datasets should be how well the dataset represents the field study.

2.3.6 Training and Learning

In the training step of a DL model, the dataset is divided into the training set, the validation set, and the test set. The training set is used to train the model. The validation dataset is a sample of data retained by the training model and used for model selection. The model structure is determined using hyperparameters. Hyperparameters are parameters whose values are set before the learning process begins, such as the number of hidden layers in the model, the number of kernels in the CNN model, and the number of nodes in each LSTM layer [54]. A test set is used to test a machine learning model after it has been trained on an initial training dataset. When the test set is used during the training of the model, it does not provide information about how the network will perform on unseen data, and the evaluation of the result has no statistical significance. [54]. Figure 2.11 shows the general steps from preprocessing the data to finalizing the model for prediction.

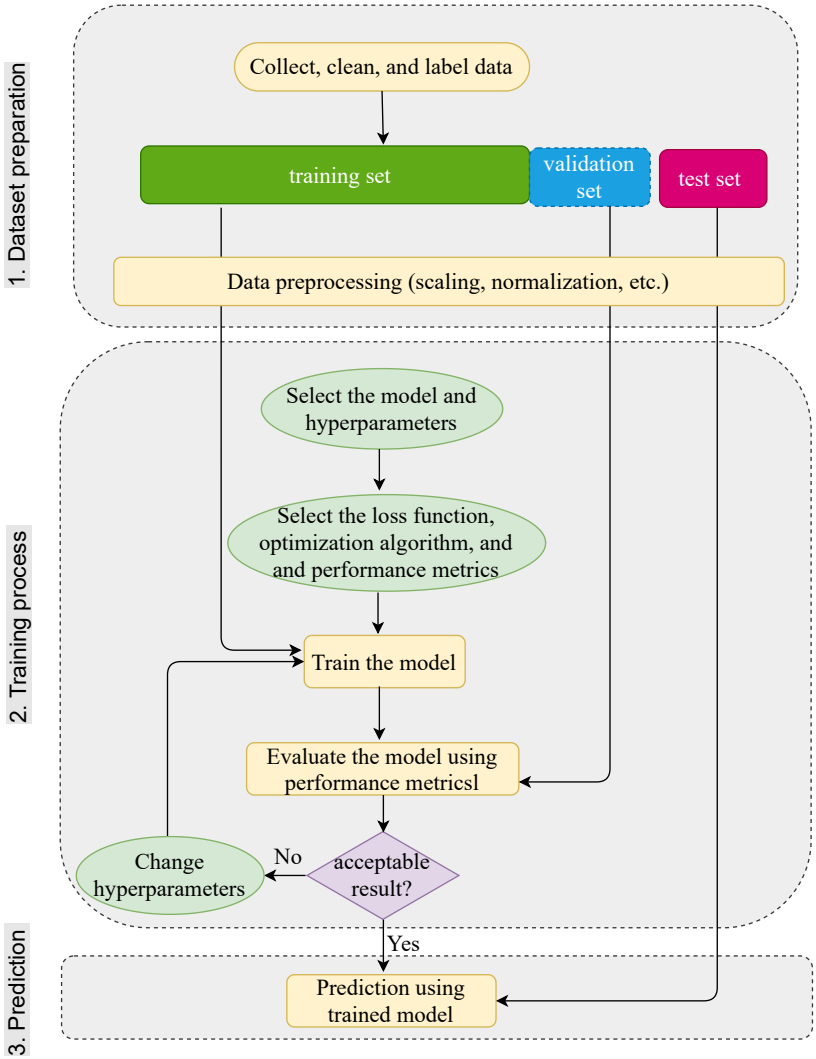


Figure 2.11: Flowchart for DL algorithm tasks. The test data is obtained by splitting the original dataset, which is not used for training and validation. The training data is passed to the model to perform automatic feature extraction and validation to evaluate the model during training and tune the hyperparameters.

The error of a model is defined as the difference between the actual value the predicted output, and it must be minimized in the training step. The function that is used to compute

this error is known as the loss function . It can be categorized into two types: Classification and Regression Loss. The loss function used for the classification contains Hinge loss, Expositional loss, cross-entropy, Log loss, and many more. For the regression problems, the loss function such as Mean Square Error (MSE), Mean Absolute Error (MAE), Huber loss, and log cosh loss can be used. Each of these gives a different error for the same prediction. The choice of a loss function depends on the algorithm. Training DL models are based on backpropagation [79]. Backpropagation uses the chain rule and computes the loss function's gradient concerning the network's weights and biases for a single input-output example, adjusted network weights, and biases. The optimization algorithm must be selected to minimize the loss function. The choice of optimization algorithm and loss functions are critical. The same as loss function, there are several ways to optimize the error, such as Stochastic Gradient Descent (SGD), SGD with momentum, Root Means Square propagation (RMSprop), Adam [12]. Zhang et al. [80] built up an experience to compare the performance between SGD and Adam in identifying tomato leaf disease. The ResNet with the SGD optimization method obtained the highest test accuracy of 96.51%. Figure 2.12 shows the learning steps of a DL model.

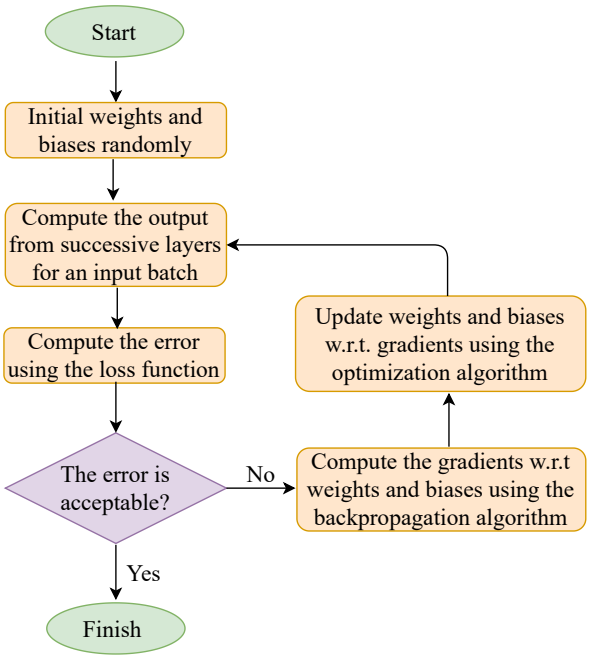


Figure 2.12: During training, the parameters of the model (weights and biases) are modified using the backpropagation and optimization algorithms to minimize loss.

2.3.7 Evaluating the models

Once the model is trained, the most crucial question is how good the model is. There are several metrics for evaluating DL models. The most commonly used metrics in this review for the regression problems are:

- Mean Square Error (MSE): is the average of the squared differences between prediction and actual observation.

- Root Mean Square Erro (RMSE): is the square root of MSE.
- Mean absolute Error (MAE): is the average of the absolute differences between prediction and actual observation.
- R^2-score: is a statistical measure calculated from the variance explained by the model versus the total variance. The higher R^2 indicates the smaller the differences between the observed data and the fitted values.

The metrics that can be used in evaluating a classification or segmentation problem are as follows:

- Accuracy (AC) is the ratio of correctly predicted observation of the total observations.
- Recall (R): also known as sensitivity, measures how many actual positives are captured by the model correctly.
- Precision (P): also known as a positive predictive value, measures how accurate the prediction is in classification, i.e., the correct prediction.
- The Average Precision (AP) is given by the area under the precision-recall curve above. Since Precision and Recall always lie between 0 and 1, AP also lies between 0 and 1. The Mean Average Precision (mAP) is calculated by taking the mean AP over all classes.
- F_1-score : is the weighted average of Precision and Recall.
- Intersection Over Union (IOU) : is an evaluation metric used to measure an object detector's accuracy on a particular dataset.
- Area Under the ROC Curve (AUC): measures the area underneath the entire ROC curve.

2.3.8 Brief review of papers

2.3.8.1 Disease detection

Plant diseases cause losses in agricultural production and endanger food security. The most common practice in pest and disease control is to spray the crop area evenly with pesticides. This method requires significant amounts of pesticides, which results in high financial and human resources and significantly changes the environment. The use of a deep learning algorithm that detects the exact time, location, and affected crops and sprays the pesticides only on the affected plants can reduce resource consumption and environmental changes.
Kerkech et al. [81] used the SegNet model to segment and detect grapevine diseases using images with RGB and infrared ranges. The dataset was acquired using a UAV device with two MAPIR Survey2 camera sensors, including a visible light (RGB) sensor set for automatic illumination and an infrared sensor. The dataset was labeled using a semi-automatic method, i.e., a sliding window to identify potentially diseased areas. Then, each block was classified by a LeNet5 network for pre-labeling. In the end, the labeled images were manually corrected. The SegNet recognizes four classes including the shaded areas, soil, healthy and symptomatic

vines. Two models were trained, one for the RGB images and the other for the infrared images. The segmentation results of the two models were also fused in two ways. The first case was called "Fusion AND", which means that the symptom is considered detected if it is present in both the RGB and infrared images. The second case is called "fusion by the union" and has the symbol "fusion OR", which means that the symptom is considered detected if it is present in either the RGB or the infrared image. The model trained with RGB images (Acc=85.13) outperformed the model trained with infrared images (78.72). The fusion AND had the best performance, and the fusion OR had the worst accuracy. The runtime of SegNet on a UAV image was reported to be 140 s for visible and infrared images. The fusion between the two segmented images takes less than 2s.

Kerkech et al. [82] used a CNN model to detect Esca disease in grapevine using UAV RGB images. A CNN model was trained with different combinations of patch sizes 16×16, 32×32 and 64×64 with different color spaces including RGB, HSV, LAB, YUV, and vegetation indices such as ExG, ExR, ExGR, GRVI, NDI, RGI. All the different color spaces and vegetation indices used in this study can be calculated from the RGB images. The results show that the CNN model trained with the RGB and YUV color spaces has a better performance compared to the models trained with HSV and LAB. It was pointed out that the lower accuracy of the models trained with HSV could be due to the Hue (H) channel in HSV, which combines all color information into a single channel and is less relevant for the network to learn the best color features. The LAB color space has one luminance channel (L) and two chrominance channels, which do not reproduce the colors of the diseased vineyard well.

In the next experiment, vegetation indices were added to the RGB and YUV data. Combining vegetation indices with RGB and YUV improved classification results in most cases. The final investigation concerned the models trained by combining the vegetation indices alone. The combination of ExR, ExG, and ExGR vegetation indices with a size of 16×16 gave the best performance among the other inputs (including color space) and sizes with an accuracy of 95.80%. Also, the combination of YUV and ExGR vegetation indices with sizes of 32×32 and 64×64 achieved similar performance but the run-time was longer.

From the results, the color of the images and leaves is very important in the detection of grapevine diseases. Also, the number of channels in the input does not affect the run-time of the model, but the size of the input and the model structure do.

Barbedo [75] investigated how dataset size and diversity affect the performance of DL techniques applied to plant diseases. An image dataset of leaves with a small number of samples for the CNN model was used to investigate the behavior of GoogLeNet under different conditions. Data augmentation and transfer learning methods were used to train the CNN. The results showed that even with transfer learning and augmentation techniques, CNN requires a large number of images to extract useful features from the data. Even though a large number of images can be easily acquired with the new technology, labeling the dataset is time-consuming. One option proposed in this paper was to share the dataset, but as mentioned earlier, the two environments are not the same in agriculture. The effect of removing the background of the images on the accuracy of the model was also investigated. The model has trained again with the images without background. The model has trained again with

the images without background. The results show different behaviors with respect to the accuracy of the model, including no significant effect, a significant improvement in accuracy in some cases, a significant decrease in accuracy, and mixed results (improved accuracy in some diseases while the error rate increased in others). From the significant decrease in accuracy for some plants, it was inferred that the CNN model sometimes uses the background of the model to classify the images. An attempt that can be made here is to train the model with both datasets (with and without background) and investigate the accuracy of the model in this case.

Ferentinos (2018) [83] used AlexNet, AlexNetOWTBn [84], GoogLeNet, Overfeat [18], and VGG models to identify 25 plant disease in 58 different classes of (plant, disease) combinations, including some healthy plants. The dataset used was images of healthy and infected leaves of the plants from the Plantvillag database (`https://github.com/spMohanty/PlantVillage-Dataset`). More than 37.3% of the images in the dataset were taken under real cultivation conditions in the field, and the other images were taken under laboratory conditions. In the first experiment, the number of images acquired under laboratory conditions and real conditions was kept similar in the training and test set, and the models were trained using this dataset. The VGG model performed best on the test set with an accuracy of 99.53%.

They also investigated the significance of the presence of field-captured images in the training set. From the 58 available classes of the form [plant, disease], the 12 that contained images of both types were selected. Two experiments were conducted with these 12 classes, and two CNN models were developed: one was trained with images under laboratory conditions and tested with images under field conditions, and another was trained with images under field conditions and tested with images under laboratory conditions. Although the number of images acquired under field conditions was less than the number of images acquired under laboratory conditions, the CNN model trained only with images under field conditions performed better with an accuracy of 68% than the CNN model trained only with images under laboratory conditions with an accuracy of 33%. This result shows the importance of the presence of the images acquired under field conditions. From the misclassification image, they point out some problematic situations, including images with extensive partial shading on the leaves, images with multiple objects in addition to the image, images where the leaf occupies a very small and non-centric part of the frame, image without leaves.

Jiang et al. [85] implemented the CNN models to detect five common apple leaf diseases using images from the field. By applying data augmentation such as rotational transformations, horizontal and vertical flips, intensity disturbances, and Gaussian noise, 26,377 disease images were generated. The problem with SDD is that it cannot detect a small object, and also, an object can be detected multiple times. To overcome these drawbacks, they develop an SSD model with a VGG-INCEP as the backbone, where two GoogLeNet inception layers replace two convolutional layers of the VGG model. Moreover, the structure of feature extraction and fusion is designed by applying the Rainbow concatenation method instead of the pyramid (which is used in the SSD model) to improve the performance of feature fusion. The results showed that the data augmentation improved the accuracy of the model by 6.91% compared to the original dataset. Moreover, the proposed model achieved the best perfor-

mance compared to faster R -CNN and SSD with VGG and Rainbow SSD (RSSD) model. To investigate the effect of using a deep model as a backbone, ResNet-101 was used as a feature extractor for SSD. The results show that ResNet-101 does not lead to any improvement. In terms of speed, Faster RNN was the slower model with more accuracy. The proposed model with 78.80% mAP and 23.13 Frames Per Second (FPS) was more accurate than SDD and RSSD, but SSD outperformed the other models in terms of time inference. By examining the misclassified images, it was indicated that similarity between diseases, misclassified backgrounds, and light conditions were the challenges in classification.

Table 2.1: Feature descriptions of recently published papers in the field of "Disease Detection".

Reference	Application	Data used	Model	Metric used	Model performance
Kerkech et al. [81]	Detect Esca disease in grapevine using UAV RGB images.	The images were collected using a UAV system with an RGB sensor.	A CNN model was trained with different combinations of patch sizes and different color spaces.	AC	The CNN model trained with the RGB and YUV color spaces has a better performance compared to the models trained with HSV and LAB. Moreover, The model obtained by combining YUV and RGB trained with vegetation indices gave better performances than the models trained with YUV and RGB.
Kerkech et al. [82]	Segmentation of the plant symptomatic area	A Quadcopter drone with two camera sensors	SegNet	P, R, F_1, ACC	The model trained with RGB images outperformed the model trained with infrared images.
Barbedo [75]	Disease detection on 12 plants with a variety of diseases.	The images were taken with a variety of digital cameras and mobile devices.	GoogleNet	Ac	CNN's AC using the original image as the dataset was 84% and using the removed background was 87%.
Ferentinos (2018) [83]	Identification of 25 plant disease in 58 different classes of (plant, disease).	`https://github.com/spMohanty/PlantVillage-Dataset`	AlexNet, AlexNetOWTBn, GoogLeNet, Overfeat, and VGG models	ACC	The VGG model performed best on the test set. Also, the CNN model trained only with images under field conditions performed better than the CNN model trained only with images under laboratory conditions.
Jiang et al. [85]	Identification of apple leaf diseases.	Images were taken in the field, and some of them were in the laboratory.	SSD with base VGG-INCEP.	mAP	SSD with VGG-INCEP as base achieved better performance with an mAP of 78.80% compared to FR-CCN and SSD with VGG and ResNet as the base.

Karthik et al. [86] proposed a Residual Attention Network for disease detection in tomato leaves. The main idea of attention is to focus on the relevant parts of the input to produce outputs. The dataset was collected from the open-source website Plantvillage and contained one healthy class and three diseases. They implemented three models. One was a traditional CNN model, the second used the three residual layers introduced into the CNN model as part of the ResNet architecture, and the last used an attention mechanism on top of the residual CNN for effective feature learning. The traditional CNN-based methods focus on ordered feature learning, starting from basic image-level features such as edges, color, etc., to complex texture-based differences [86]. In the deeper layer, some important features extracted in the first layer may be lost. The residual layers are designed to avoid this problem [86].

They concatenate the extracted features from the earlier layers with the deeper layers. In addition, the attention mechanism is used to extract the relevant parts of the feature maps. The Residual Attention Network CNN performed better with an overall accuracy of 98% than the Vanilla Residual Network with an accuracy of 95% and the traditional CNN model with an accuracy of 84%.

Liu et al. [87] implemented the model Cascade Inception to detect four common apple leaf diseases in images captured in the field. The Cascade Inception was a modified AlexNet model with inception layers from GoogleNet. Various data enhancement methods such as image rotation, mirror symmetry, brightness adjustment, and PCA jittering were applied to the training images. Moreover, the fully connected layers were replaced by convolutional layers, which results in fewer parameters and avoids overfitting. The proposed model was trained using the optimization method Nesterov Accelerated Gradient (NAG) and achieved an accuracy of 97.62%. The performance of the proposed model was compared with SVM and BP neural networks, standard AlexNet, GoogLeNet, ResNet-20, and VGGNet-16. Transfer learning method was used to train VGGNet-16 and achieved 96% accuracy. Standard AlexNet, GoogLeNet, and ResNet-20 were trained from scratch using SGD optimization and achieved a maximum accuracy of 95.69%. The SVM and BP, which achieved an accuracy of less than 60%, show that the traditional approaches rely heavily on the expert-developed classification features to improve the recognition accuracy. They also investigated the effect of data augmentation methods and optimization algorithms on accuracy. The model with the SGD optimizer achieved 93.32% accuracy, while the model with NAG achieved 97.62% accuracy. The data augmentation methods improved the performance of the model by 10.83%. The advantage of the model is that it outperformed other CNN models in terms of training time and memory required. Also, the number of parameters of the model was less than AlexNet and GoogleNet. One point that emerges from the paper is that GoogleNet and AlexNet were trained using the SGD optimizer, and the proposed model was trained using the NAG method, but as mentioned in the paper, SGD has the ”local optimum” problem. In addition, the models were not compared based on the inference time.

Ramcharan et al. [20] trained the SSD object detection model to identify three diseases, two types of pest damage, and nutrient deficiency in cassava at the mild and pronounced stages. A dataset was collected from the field, which was divided into 80-20 training and testing sets. The model achieved 94 ± 5.7% mAP on the test dataset. The trained model was used on a mobile phone to investigate the performance of the model in the real world. One hundred twenty images of leaves were captured using a mobile device of the study experiment, and the model inference was run on a desktop and mobile phone to calculate the performance metrics of the trained model. The results show that the average precision of the model on the real dataset decreases by almost 5%, but the average recall decreases by almost half, and the F_1-score decreases by 32%. Furthermore, the results show that the model performs better on the leaves with pronounced symptoms than on the leaves where the symptoms are only mildly pronounced.

Picon et al. [88] used DL model to detect seventeen diseases of five crops (wheat, barley, corn, rice, and rape-seed) in images captured in the field. The dataset contained several challenges,

such as multiple diseases on the same plant, similar visual symptoms among diseases, images of early and early-stage diseases, and diseases of leaves, and stems. To improve the accuracy of the model, the crop ID was used in the network. The crop ID was defined as a categorical vector with K components, where K is the number of crops in the model. Several models were trained. The first approach was to train an independent model for each crop and the second approach was to train a single model for the entire dataset. The results show that the multi-crop model had a similar performance to splitting the training dataset into the different crops (1% increased accuracy). However, the class of diseases with fewer images in the training set may benefit from the multi-crop model. The second approach was to add the crop ID to the multi-crop model. The results show that by adding crop id, the model can still benefit from more images for training, while crop ID information helps the network to discriminate between similar diseases.
Chen et al. [89] used pre-trained MobileNet V2 networks to identify twelve rice plant diseases. The dataset was collected from online sources and real agricultural fields. An attention mechanism was added to the model, and transfer learning was performed twice during model training to achieve better performance. The MobileNet-V2 achieved 94.12% accuracy on the PlantVillage dataset, while the MobileNet-V2 with attention and transfer learning achieved 98.26%. The five CNNs such as MobileNetv1, MobileNet-V2, NASNetMobile, EfficientNet-B0, and DenseNet121 were selected for comparison with the proposed models. In addition, two other networks named MobileNetv2-2stage and MobileAtten-alpha were trained. In the MobileNetv2-2stage model, transfer learning was performed twice to identify the images of plant diseases.
Similarly, in MobileAtten-alpha, the attention method was used instead of transfer learning twice. The proposed model (using attention and transfer learning twice) achieved the the second best accuracy of 98.84%. The DenseNet121 with an accuracy of 98.93% outperformed other models. MobileNetv2-2stage and MobileAtten-alpha achieved an accuracy of 98.68% and 96.80%, respectively. The proposed model was trained with images from the field and achieved an accuracy of 89.78%.

2.3.8.2 Fruit detection and Yield Forecast

Yield forecasting is one of the most important and popular topics in precision agriculture because it is the basis for yield mapping and estimation, supply and demand matching, and crop and harvest management. Modern approaches go far beyond simple forecasting based on historical data but incorporate methods from DL to provide a comprehensive multidimensional analysis of crop, climatic and economic conditions to maximize yields.
Silver & Monga [91] trained five CNN models to estimate grape yield from RGB images taken in a vineyard on harvest day using the camera of a Samsung Galaxy S3. The images of 40 grapevines were split into two parts, one for the left cordon and one for the right cordon, resulting in a total of 80 cordons. A simple CNN was trained by inputting the RGB images of the left and right cordons and estimating the grape yield. The second model was the same as the first model, but the input to the model was the right cordon images and the inversion of the left cordon images to look like right cordon images. The third model, an autoencoder

Table 2.2: Feature descriptions of recent published papers in the field of "Disease Detection".

Reference	Application	Data used	Model	Metric used	Model performance
Karthik et al. [86]	Identifying the type of infection in tomato leaves.	Plant Village Dataset.	Residual CNN with attention.	AC	Residual CNN with attention with an ACC of 98% achieved better performance compared to CNN and residual CNN.
Liu et al. [87]	Identification of 4 apple leaves diseases.	Images were taken in field using digital color camera.	CNN (AlexNet+ +Inception layers from GoogleNet).	AC	Their model with an AC of 97.62% performed better than GoogLeNet, ResNet-20, VGGNet16, SVM, and BP, AlexNet.
Ramcharan et al. [20]	Identification of cassava pests and diseases.	dataset from [90]	SSD	F_1-score	The F_1-score decreased by 32% when moving from the test set to real images.
Picon et al. [88]	Identification of 17 diseases and five crops by adding metadata to the model.	Images were Taken by cell phone in real field wild conditions.	CNN (ResNet50	ACC, R^2, Negative and positive predictive value	.Adding plants species information to the model by concatenating plant information in the embedding layer improved the performance of the model (AC= 0.98)
Chen et al. [89]	Identify rice plant diseases from collected real-life images.	online sources and agricultural fields.	MobileNet-V1, MobileNet-V2, DenseNet-121, NASNetMobile, EfficientNet-B0, proposed model.	ACC, R, P, F_1-score	The proposed model with an accuracy of 0.98 out- performed the other models.

network, is trained to a high level of accuracy and the CNN encoder weights are transferred as starting weights into the CNN model for the yield estimation model. In the fourth model, an autoencoder model was trained to output the density map of the grapes in the image. Then, the weights of the encoder are transferred as initial parameters into the CNN model for yield estimation. The last model was trained with the output of the autoencoder for the density map as input to the CNN model for yield estimation instead of the RGB images. The CNN models with flipped images outperformed the simple CNN model with an MAE % of 15.43. The models of transfer learning from Density Map Network Representation with MAE % of 11.79 achieved the best performance among the other models. The last model with MAE % of 15.99 did not perform as well as the fourth model because the accuracy of the density map estimates was quite low. The results show that transfer learning, when used properly, can improve recognition accuracy.

Aguiar et al. [92] trained SSD MobilenetV1 and the Inception model to detect Grape Bunch in Mid Stage and early stages and then transferred the trained model to the TPU-Edge device to investigate the temporal accuracy of the model. The same strategy in [40] was used to collect the dataset and it is publicly available (`https://doi.org/10.5281/zenodo.5114142`) with 1929 vineyard images and their annotation. Overall, SSD MobileNet-V1 performed better than the Inception model, as it had a higher F1 score, AP, and mAP. The early stage was more difficult to detect than the middle growth stage. The first class represented smaller clusters that were more similar in color and texture to the surrounding foliage, making them more difficult to detect. SSD MobileNet-V1 showed an AP of 40.38% in detecting clusters at an early growth stage and 49.48% at a medium growth stage. In terms of time, SSD MobileNet-V1 was more than four times faster than the Inception model on TPU-Edge Device.

Ghiani et al. [53] used Mask R-CNN with ResNet101 as a backbone which was pre-trained

with the dataset COCO (https://cocodataset.org/#home) for detecting grape branches on the tree. An open-source dataset GrapeCS-ML [93] containing more than 2000 images without annotation of fifteen grape varieties at different stages of development in three Australian vineyards was used to train the model. In addition to the GrapeCS-ML dataset, 400 images were collected from the island of Sardinia (Italy). The result obtained by applying the detector to the test samples was an mAP value of 92.78%. To investigate the generalizability of the proposed model, the model trained on the GrapeCS-ML dataset was tested on its internal dataset. The dataset contained different grape varieties, vegetation, and colors than the GrapeCS-ML dataset. An mAP of 89.90% was obtained with the internal dataset, indicating that the model can be used in other fields. To investigate the significance of the size of the dataset used for training and the importance of the augmentation techniques, the size of the original dataset was reduced to 10% of the training images in one case with augmentation and the other case without augmentation of the dataset. The mAP was reduced by up to 5%, especially in the experiments performed without augmentation. It was observed that the recognition accuracy decreased for images with overlap between clusters.

In Milella et al. [94], a system for the automatic estimation of harvest volume and for detecting grapes in vineyards using an RGB-D sensor on board an agricultural vehicle has been proposed. An RGB-D sensor is a special type of depth detection device that works in conjunction with an RGB camera and is able to add depth information to the conventional image pixel by pixel. The approach to determine the crop volume involved three steps: the 3D reconstruction of the vine rows, the segmentation of the vines using a deep learning method, and the estimation of the volume of each plant. In the first step, the depth output from the camera was not used because the parameters of the algorithm are fixed and cannot be configured. Instead, the semi Global Matching (SGM) algorithm was used, which is a computer vision algorithm for estimating a dense disparity map from a rectified stereo image pair. A box-grid filter is then used to merge the point clouds.

In segmentation, a segmentation algorithm was first used to separate the canopy from the background using the green-red vegetation index (GRVI), and then k-means were used to identify each plant. Based on the results of the clustering algorithm, different plants were calculated and an estimate of the volume per plant was performed. Finally, the pre-trained AlexNet, VGG16 and VGG19, and GoogLeNet were trained to perform grape cluster detection in 5 different classes (grapes, vineyard stakes, vine stems, cordons, canes, leaves, and background). The VGG model performed the best with an accuracy of 91.52.

Santos et al. [95] used deep learning models and computer vision models to estimate grape wine yield from RGB images. Images were captured with a Canon EOS REBEL T3i DSLR camera and a Motorola Z2 Play smartphone from five different grape varieties. The dataset named Embrapa Wine Grape Instance Segmentation Dataset (WGISD) with 300 RGB images is publicly available (https://doi.org/10.5281/zenodo3361736). Models from DL such as Mask R-CNN, YOLOv2, and YOLOv3 were trained to detect and segment grapes in the images. Then, an image processing algorithm called Structure-from-Motion (SfM) was used to perform spatial registration, integrating the data generated by the CNN-based step. In the final step, the results of the CV model were used to remove the clusters de-

tected in different images to avoid counting the same clusters in several images. The Mask R-CNN with ResNet101 as the backbone outperformed YOLOv2 and YOLOv3 in terms of object detection, but the YOLO model outperformed the Mask R-CNN in terms of detection time. The worst performance was obtained with YOLOv3. To verify the performance of Mask R-CNN+SfM, 500 key-frames of a video were used, and the result is shown in a video at `https://youtu.be/1Hji3GS4mm4`.

Palacios et al. [96] applied the method of deep learning to detect the flowering of the vine and used it for the estimation of early yield estimation. Images of six grapevine varieties were acquired using a mobile platform developed at the University of La Rioja. The RGB camera was a Canon EOS 5D Mark IV RGB with a full-frame CMOS sensor. The ground truth was acquired using the algorithm in [97]. This algorithm was developed to process only images with a single inflorescence and a dark background. In the first step, SegNet was used with VGG (VGG16 and VGG19) as a backbone to segment and extract the inflorescences contained in the images. Then, these regions were used to count the flowers in each inflorescence using three algorithms, including SegNet, Watershed Flower Segmentation, and a linear model. The SegNet with VGG19 as backbone outperformed the model with VGG16 in terms of IOU and F_1- score. For flower recognition, the SegNet model with VGG19 was trained to classify a group of flowers per image into three classes including contour, center, and background. After segmentation, false-positive filtering of flower segmentation was performed. Here, the flowers whose center and contour was segmented, and whose contour surrounded the center above a certain threshold were considered as true positives. The F_1-score reached its highest value when contour filtering was set to 50%, resulting in an F_1-score of 0.729. The best F_1-score for the watershed approach was 0.708 and the worst performance in counting flowers was obtained with linear regression in the form of the Root Mean Square metric. The number of flowers counted using SegNet-VGG19 for inflorescence extraction and flower detection, and flower counting using the algorithm of [97] showed a correlation with R^2 close to or above 0.75 for all cultivars.

Kang & Chen [56] implemented DaSNet-v2, which is an 'Encoder-Decoder with atrous convolution developed in Deeplab-v3+ to detect and segment the apple in an orchard for harvesting by a robot. Atrous convolution is a type of convolutional layer that allows control of the resolution of the features computed by the CNN. The dataset was collected from an apple orchard as RGB-D and RGB. The RGB-D was used to visualize the environment. A lightweight model, Resnet 18, was used as the backbone of the DaNet-v2 to ensure its deployment on the Jetson-TX2 with limited computing capacity. In addition, the model was trained with the Resnet50 and Resnet101 backbones. The performance of the model with the Resnet101 backbone was compared with DaSNet-v1, YOLO-v3, faster-RCNN, and the Mask-RCNN. DaSNet-v2 and Mask-RCNN with F_1- score of 0.873 and 0.868 respectively, outperformed the other models. However, DaSNet-v2 outperformed mask-RCNN with a computational efficiency between 306 and 436ms with a time of 1.3s. The results also show that single-stage detection models such as Yolo have better computational efficiency compared to two-stage detectors. The model was implemented on Jetson-TX2, a lightweight backbone of Resnet-18, and the experimental results show that DaSNet-v2 with Resnet-18 can achieve similar performance

Table 2.3: Feature descriptions of recent published papers in the field of "Fruit detection and Yield estimation".

Reference	Application	Data used	Model	Metric used	Model performance
Silver & Monga [91]	estimate grape yield from RGB images.	Images were taken in of a vineyard on harvest day using the camera of a Samsung Galaxy S3.	Five CNN models from scratch trained with five different techniques for training	MAE	The models of transfer learning from Density Map Network Representation. with MAE% of 11.79 achieved the best performance among the other models.
Aguiar et al. [92]	Detect Grape Bunch in Mide Stage and early stages.	The images were acquired from four different vineyards in Portugal.	SSD Inception-V2, SSD MobileNet- V1	IOU, mAP, R	The SSD Inception-V2 had higher precision than the SSD MobileNet-V1 at all confidence values, but lower recall. In terms of time, SSD MobileNet-V1 was more than four times faster than the Inception
Ghiani et al. [53]	Detecting grape branches in the tree	GrapeCS-ML [93] and 400 images were collected from field	Mask R-CNN with ResNet101 as back- bone	IOU, mAP	The model achieved an mAP value of 92.78%
In Milella et al. [94]	Estimation of harvest volume and for detecting grapes in vineyards	Images was collected from field using RGB-D sensor	AlexNet, VGG, GoogleNet	P, R, AC, TP (true positive)	The VGG model performed the best with an accuracy of 91.52.
Santos et al. [95]	Estimate grape wine yield from RGB images.	Images were captured with a camera from five different grape varieties.	Mask R-CNN and YOLOv2 and YOLOv3	P, R, F_1-score	The Mask R-CNN with ResNet 101 as the back-bone outperformed YOLOv2 and YOLOv3 in terms of object detection, but the YOLO model outperformed the Mask R-CNN in terms of detection time.
Palacios et al. [96]	Detect the flowering of the vine, and used it for the estimation of early yield estimation	Images of six grapevine varieties were acquired using a mobile platform	SegNet model with VGG19	IOU and F_1-Score	The number of flowers counted using SegNet-VGG19, and flower counting using the algorithm of [97] showed a correlation with R2 close to or above 0.75 for all cultivars.

in recall and precision of detection compared to YOLO-v3. Environmental factors such as strong sunlight reflection, shadows, and appearance variations of fruits in color, shape, occlusion, or viewing angle could lead to false-negative detection results. RGB-D images were processed using DaSNet-v2 and the PPTK toolbox, a Python package for image visualization, to deploy the robot in the orchard.

Koirala et al. [98] developed a DL model, called Mango-YOLO, based on YOLO-v3 and YOLO-v2 (tiny) for counting mangoes on trees. YOLOv2 (tiny) has a small architecture with only nine convolutional layers, six pooling layers, and one detection layer, sacrificing accuracy for speed. YOLOv3 [44] is based on Darknet-53 and improves upon YOLOv2 [99]. The Mango YOLO had 33 layers compared to 106 layers in YOLO-v3 and 16 layers in YOLO-v2 (tiny). The reduction in the number of layers is expected to reduce computation time and detect mangoes more accurately. The model was trained on the dataset collected from five orchards. The Mango- YOLO achieved better performance with an accuracy of 0.967% compared to YOLO-v2 (tiny) (0.9%) and YOLO-v3 (0.951%). In terms of time inference, the Mango YOLO with 15ms was faster than YOLO-v3 (25ms) and slower than YOLO-v2 (tiny)

(10ms). Moreover, Mango-YOLO was trained once from scratch on the augmented dataset, and the second time transfer learning was used using the COCO dataset. The models had the same performance on the test set, and the reason was explained by the fact that the COCO dataset does not contain Mango images. The false detection over images taken with a Canon camera shows resizing of the images and results in image distortion with leaves taking a curved shape resembling the fruit and overexposed areas on branches, trunks, and leaves.

Liang et al. [100] applied the SSD network to detect mango and almond on tree fruits. The dataset in [101] was used to train the model. The SSD model used the original and sampled a patch such that the minimum Jaccard overlap with the objects is 0.1, 0.3, 0.5, 0.7, or 0.9, and finally randomly sampled the input image. The size of each sampled area and the minimum Jaccard overlap are critical for object detection. These two parameters were changed to adopt the SSD model for small mango detection. Using VGG16 as the basic framework for the SDD outperformed the SSD with ZFNet. Also, using the input image size of 400×400 and 500×500 as the input of the model, the model achieved better performance than SSD with an input size of 300×300. The model outperformed the faster RCNN in terms of speed and accuracy. The challenges related to mango detection on the tree were considered in the paper: the size of the mango on the whole image, blocking the mango with leaves, and branches, resizing the original image, making the mango smaller, and the similarity between mango and background.

Tian et al. [102] developed YOLO-V3 with DenseNet as the backbone to detect apples on trees. They used two datasets for training. The first one contained images of apples at one growth stage, and the second one contained images taken at different growth stages. The model showed better performance for mature apples than for young and growing apples because the color features were more prominent, the individual volume was larger, and there was less overlap. The results also 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. The performance of the trained model decreased for images with partial occlusion of apples with branches and leaves but is still an acceptable result (IOU =0.889 for mature apples).

The model achieved the best performance compared to Faster R-CNN, YOLOV2, and the original YOLOv3. In terms of time inference, the model was faster than Faster R-CNN and similar to YOLOv3. The F_1- score and IOU of the model without data augmentation methods decreased by 0.033 and 0.058, respectively.

Zhou et al. [103] implemented an SSD model with two lightweight backbones MobileNetV2 and InceptionV3, to develop an Android APP called KiwiDetector that detects kiwis in the field. Quantization is a technique for performing computations and storing tensors with bit widths smaller than floating-point numbers. When training a neural network, 32-bit floating-point weights and activation values are typically used. A quantized model performs some or all operations with tensors using integers instead of floating-point values. This allows the computational complexity to be reduced and the trained model can be used on devices with lower resource requirements. The quantization method was used to compress the model size and improve the detection speed by quantizing the weight tensor and activation function data of convolutional neural networks from 32-bit floating-point numbers to an

8-bit integer. The results showed that MobileNetV2, quantized MobileNetV2, InceptionV3, and quantized InceptionV3 achieve a true detection rate of 90.8%, 89.7%, 87.6%, and 72.8%, respectively. The quantized MobileNetV2 on the Huawei P20 smartphone outperformed the other models in terms of detection speed (103 ms/frame) and size. Although the SSD with MobileNetV2 was more accurate than the SSD with quantized MobileNetV2, the SSD with quantized MobileNetV2 was 37% faster. The problem with kiwi detection was that overlapping kiwis were reported, which the model counts as one kiwi.

Table 2.4: Feature descriptions of recently published papers in the field of "Fruit detection and Yield estimation".

Reference	Application	Data used	Model	Metric used	Model performance
Kang & Chen [56]	Counting apple on the tree.	The images were taken in the field with a camera.	C-RCNN+LedNet with a lightweight (LW) as the backbone, resnet-50, resnet-101, darknet-53.	IOU, P, R, F_1-score	LedNet with resnet-101 achieved an AC of 0.864 and LedNetwith LW achieved an Ac of 0.853.
Koirala et al. [98]	Counting mango on trees.	The images were taken in the field.	CNN (Mango-YOLO)	F_1-score	Mango-YOLO with an $F_1 - \text{score}$ of 0.97 achieved better performance compared to faster R-CNN, SSD, and YOLO.
Liang et al. [100]	Detection of mango and almond fruits on the tree.	https://arxiv.org/abs/1610.03677	SSD with VGG-16 and ZFNet	IOU, F_1-score	The model outperformed the faster RCNN in terms of speed and accuracy.
Tian et al. [102]	Detection of apple in image and yield estimation.	Image capture was performed with a camera in the field.	Improved YOLO-V3 with DenseNet as the backbone.	F_1-score	The proposed model with an F_1-score of 0.817 had better performance compared to YOLO-V2, YOLOV3, and Faster R-CNN.
Zhou et al.[103]	Detection of kiwi fruit in the orchard with android smartphones.	Collected from the orchard.	MobilNetV2, quantized MobileNetV2, InceptionV3, quantized InceptionV3.	IOU, TDR, FDR	Quantized MobileNetV2 outperformed the other models in terms of accuracy, speed, and size.

2.3.8.3 Weed Detection

Besides disease, weeds are considered to be a prevalent threat to agricultural production. These are plants considered undesirable in a particular situation, as they may compete with crops for sunlight and water, resulting in crop and economic loss. One significant problem in weed control is that they are difficult to detect and distinguish from crops. DL algorithms can improve weed detection and discrimination at a lower cost with reduced environmental problems and side effects. These technologies could power robots that detect and remove weeds.

Bah et al. [104] proposed a CNN model with unsupervised training dataset annotation collection for weed detection in UAV images of bean and spinach fields. They assumed that crops are grown in regular rows and that plants growing between the rows are considered weeds. The Hough transform was applied to the skeleton to detect the rate of plant rows, and then a simple linear iterative clustering (SLIC) algorithm was applied to create a marker and de-

lineate the plant rows. This algorithm generated superpixels based on k-mean clustering. After row detection, a blob coloring algorithm was used to identify the weeds. The unsupervised training dataset was used to train ResNet18 for weed detection in the images. The supervised learning method performed 6% better than the unsupervised learning method in the bean field and about 1.5% better in the spinach field. The low number of weeds between rows may explain the difference in performance in the bean field. The performance of the model was compared with SVM and RF. In general, ResNet18 shows better performance in supervised and unsupervised learning methods than SVM and RF.

Ferreira et al. [58] implemented the unsupervised learning models JULE and DeepCluster to detect weeds in the field. The JULE consists of stacked multiple combinations of convolutional layer, batch normalization layer, ReLU layer, and pooling layer. AlexNet and VGG16 were implemented as the basis of the DeepCluster model to extract features, and then K-means is used as the clustering algorithm. Two datasets, Grass-Broadleaf [105] and DeepWeeds [106] were used in this work. The DeepCluster model performed better than the JULE model on both datasets with a large number of clusters. The DeepCluster with a base of Alexnet and VGG achieved similar performance, with Alexnet performing better on DeepWeed and VGG on the other dataset. To investigate the effect of transfer learning on unsupervised learning, the pre-trained VGG and AlexNet were used on ImageNet. The pre-trained model did not improve the accuracy of Grass-Broadleaf, but it did improve the accuracy of DeepWeed. The data augmentation also improved the accuracy of the unsupervised learning methods. They also used semi-supervised data labeling. Semi-supervised learning is a machine learning approach that deals with the use of labeled and unlabeled data. In the semi-supervised method, labeled images from the DeepCluster model were used to train Inception-V3, VGG, and ResNet. Inception-V3 and VGG outperformed ResNet on the Grass-Broadleaf and DeepWeeds dataset, respectively.

Milioto et al. [107] modified Encoder-Decoder CNN architecture in [108] to distinguish weeds from crops and soil. The number of convolutional layers was decreased to reduce time inference, and additional vegetation indices (14 channels) were included in the input for more accurate detection. The dataset of three fields was used. Three networks were trained with different inputs: one with RGB images, another with RGB and near-infrared (NIR) images, and the last one was trained with 14 channels such as RGB, Excess Green (ExG), Excess Red (ExR), color index of Vegetation Extraction (CIVE), and Normalized Difference Index (NDI). To investigate the generalization ability of the proposed model, the model was trained on the images of one field and tested on images of all fields. The results indicate that feeding these 14-channels into the input can speed up the training and improve the performance of the model on the unseen dataset compared to the model trained on RGB images and RGB+NRI but still, the recall and precision of the model on the unseen field drop sharply (11-50%).

Another experience was conducted to investigate the generalization capacity of the proposed model. The trained model was retrained on the unseen dataset with 10, 20, 50, 100 images in the training set. The performance of the RGB network when retrained with 100 images is almost the same as the performance of the proposed model trained with ten images. The inference time of the proposed model on PC and Jetson TX2 platform was also found to be

44 ms and 210 ms, respectively, which is slower than the model trained with RGB images with 31 ms and 190 ms, respectively.
Lottes et al. [52] developed an encoder-decoder Fully Convolutional Network (FCN) with the sequential model for weed detection in sugar beet fields. The encoder-decoder FCN was used to extract features from the input images, and the sequential model processed the five images in a sequence using 3D convolution and output a sequence code that was used to learn sequential information about the weeds in five images in a sequence. The results showed that the encoder-decoder with a sequential model improved the F_1-score of the module by almost 11-14% compared to the encoder-decoder FCN. The results indicated that the changes in the visual appearance of the images in the training and test dataset could lead to a decrease in model performance, and adding additional information, such as vegetation indices, leads to better generalization for other fields.
Wang et al. [109] used the encoder-decoder with Atrous Convolution for pixel-wise semantic segmentation of crops and weeds. The two datasets for sugar beet and oilseed included in the paper were taken under completely different lighting conditions. To mitigate the effects of the different lighting conditions, three image enhancement methods were evaluated, including Histogram Equalization (HE), Auto Contrast, and Deep Photo Enhancer. The models were also trained with various inputs, including YCrCb and YCgCb color spaces and vegetation indices such as Normalized Difference Index (NDI), Normalized Difference Vegetation Index (NDVI), Excess Green (ExG), Excess Red (ExR), Excess Green minus Excess Red (ExGR), Color Index of Vegetation (CIVE), Vegetative Index (VEG), and Modified Excess Green Index (MExG), Combined Indices (COM1), and COM2. For the sugar beet dataset, the model trained with NIR images enhanced by Auto Contrast outperformed the other models with a mean IOU of 87.13%. For the Oilseeds dataset, the models were trained with RGB images only, and the model trained with images enhanced by Deep Photo Enhancer outperformed the other models (mIOU=88.91%). Moreover, the auto-contrast method outperforms other methods in terms of time inference.

2.3.8.4 Species recognition

The classification of species (e.g., insects, birds, and plants) is another critical aspect of agricultural management. The traditional human approach to species classification requires specialists in the field and is time-consuming. Deep learning can provide more accurate and faster results by analyzing real-world data.
Rußwurm & Körner [111] trained LSTM and GRU models for multitemporal classification, which achieved high accuracy in crop classification tasks for many crops. Images were collected from SENTINEL 2A images between 31 December 2015 and 30 September 2016. For consistency with the LANDSAT series, the blue, green, red, near-infrared, and shortwave infrared 1 and 2 bands were selected for this assessment. The performance of the LSTM model was compared to the RNNs, the CNN model, and a baseline SVM. The LSTM model outperformed the other models in terms of accuracy in land cover classification.
Lee et al. [112] used CNN to classify 44 plants based on leaf images. Two datasets were prepared. One contained the entire image of a leaf, and for the other, each leaf image was man-

Table 2.5: Feature descriptions of recent published papers in the field of "Weed Detection".

Reference	Application	Data used	Model	Metric used	Model performance
Bah et al. [104]	Weed detection in bean and spinach fields.	The images were taken with a DJI Phantom 3 Pro drone.	Simple linear iterative clustering (unsupervised)+ CNN (Resnet18)	AC	CNN trained with unsupervised labeling achieved an AC of 88.73 (spinach) and 94.34 (bean), and with supervised labeling achieved an AC of 94.84 (spinach) and 95.70 (bean).
Ferreira et al. [58]	Identification of weeds with unsupervised clustering.	The grass-Broadleaf Dataset in [105] and DeepWeed in [106] .	JULE, DeepCluster	AC, NMI	DeepCluster achieved better performance than JULE.
Milioto et al. [107]	Identification of weeds (additional vegetation indices added to the input).	Data was taken with a 4-channel RGB+NIR camera.	Mask R-CNN.	IOU, mIOU	The model achieved better performance by adding an extra channel at the input of the model (IOU=80.8%)
Lottes et al. [52]	Crop and weed detection in the sugar field.	Dataset was taken with robots equipped with a 4-channel with RGB+NIR camera	Encoder-Decoder FCN (FC-DenseNet) sequential model.	average F_1-score	The proposed model achieved better performance with a F_1-score of 92.3 compared to encoder-decoder FCN without a sequential model, random forests, and vanilla FCN.
Wang et al. [109]	Pixel-wise segmentation of field images into soil, crop, and weeds.	Two datasets used: the first Dataset in [110] and oilseed image acquired in the field	Encoder-decoder CNN with different inputs.	AC, IOU	The model with RGB input with an AC of 96.06 on dataset1 and 96.12 on dataset2 achieved the best performance. Image enhancement improved the results.

ually cropped. The accuracy of the CNN model trained with the second dataset was higher than that obtained with the first dataset. The results showed that CNN can extract high-level features such as structural subdivisions, leaf tip, leaf base, leaf margin, etc., and is not limited to shape, color, and texture.

Ayhan et al. [113] implement DeeplabeV3+, a CNN model developed from scratch and a machine learning method that uses NDVI (NDVI+ML) to segment vegetation and non-vegetation in the images. The dataset of [114] belongs to two studied sites, Vasiliko in Cyprus and Kimisala in Rhodes Island, were used. The images were acquired using a UAV and a modified, uncalibrated near-infrared camera. The image resolution of the Vasiliko image is 20 cm per pixel and these images were used to train the models. On the other hand, the Kimisala dataset was used as a test set. The images in this dataset contain two different resolutions of 10 cm per pixel and 20 cm per pixel.

Two DeeplabeV3+ models were trained with RGB images as input and G, B bands plus NDVI as the third channel. Although the loss of the DeeplabeV3+ model trained from scratch with NDVI+GB was decreased, the test result was very poor. But the same model trained with transfer learning and NDVI+GB channels improved the accuracy of the model compared to the model trained with RGB images (almost one percent). The CNN developed from scratch was also trained with two different channels. The first was trained with RGB images and the second with four channels of RGB and NIR. The model trained with RGB and NIR outperformed the model trained with RGB images with an accuracy of 80.9% and an accuracy of

76%.

In the last experience, the NDVI and the Gaussian Mixture Model (GMM) of the machine learning model were used to classify the images. This method includes several thresholds that need to be adjusted by the user. The NDVI+ML method outperformed the models from DL with an accuracy of 87%. Note that the deep learning method in this work was trained with a dataset from one location and tested at another location with a different land cover. This could be a reason why NDVI+ML outperformed the DL models.

Bhusal et al. [115] used the pre-trained MobileNet on the ImageNet to classify bird pests in the image. Video data from a commercial vineyard captured with a GoPro Hero 4 outdoor camera with 1080P resolution was used as the dataset. In their work, a motion detection algorithm was used that is capable of detecting moving objects. For each of these moving objects, the abounded rectangle was extracted. These moving objects were cropped from the original RGB image and reduced to 32×32. Each of these cropped images was referred to as motion instance images (MIIs). More than 5000 MIIs were collected from different videos and classified as bird or non-bird. In the next step, a CNN model developed in [116] was used to improve image resolution. The images were converted from 32×32 to 64×64, 96×96, and 128×128 and denoted as MIIs (e-MIIs), 2e-MIIs, 3e-MIIs, and 4e-MIIs respectively. Five MobilNet models were trained with the MIIs, 2e-MIIs, 3e-MIIs, 4e-MIIs, and the entire Dataset. The worth results in terms of accuracy were obtained with a model trained with MIIs, and the best result was obtained with the model trained with all the datasets.

Mac Aodha et al. [117] used the CNN models from scratch to detect bats from audio files. Two CNN models called CNNFULL with three convolutional layers and 32 filters and CNNFAST consisting of two convolutional layers and 16 filters were trained. The audio files were converted to a spectrogram and used as input to the CNN model. CNNFULL and CNNFAST took 53 and 9.5 s, respectively, to run the entire detection pipeline on the 3.2-minute full-spectrum test dataset. CNNFAST showed a trade-off between speed and accuracy with slightly lower performance compared to CNNFULL.

The performance of the models was compared with three existing commercial closed-source detection systems, including SonoBat (version 3.1.7p) (`https://sonobat.com/`); SCAN'R (version 1.7.7.), and Kaleidoscope (version 4.2.0 alpha4) (`https://www.wildlifeacoustics.com/products/kaleidoscope-pro`), as well as a machine learning method RF. The CNN model significantly outperformed the other algorithms in terms of mAP.

Ramalingam et al. [118] used Internet of Things (IoT) based architecture for insect detection. The Internet is a global system of interconnected computer networks that use Transmission Control Protocol/Internet Protocol (TCP/IP) to connect billions of computers worldwide [119]. The (IoT), on the other hand, is a global network of physical objects equipped with sensors and actuators that connect to the Internet in real-time to be identified, sensed, and controlled remotely [119]. The IoT architecture used in [118] consists of four layers: Perception layer, Transport layer, Processing layer, and Application layer. In the perception layer, a smart wireless camera captures an image of a sticky insect trap. The transport layer uses WiFi communication and TCP/IP to send images to the processing layer and transmit processed data to the application layer. In the processing layer, an Fast RCNN with ResNet50 is

used to detect the insect in the images. In the final stage, smartphones and web interfaces are used to perform the application layer tasks. The experimental results show that the trained model achieves 96% accuracy in insect detection and outperforms YOLO and SSD in terms of accuracy.

Table 2.6: Feature descriptions of recent published papers in the field of "Species recognition".

Reference	Application	Data used	Model	Metric used	Model performance
Rußwurm & Körner [111]	Segmentation of land cover with sequential models.	Images were collected from the SENTINEL2A satellite.	LSTM, RNN, CNN, SVM	F_1-score, P, AC, R	LSTM networks and RNN achieved better results.
Lee et al. [112]	Classification of plants based on leaf images.	MalayaKew (MK) Leaf dataset was collected at Royal Botanic Gardens Kew, England.	CNN+SVM, CNN+MLP	AC	CNN+SVM with an AC of 0.993 achieved better performance compared to RBF, CNN+MLP.
Ayhan et al. [113]	classify of land into vegetation and non-vegetation	Images were collected using UAV.	Deeplabv3+, CNN model, GMM+NDVI	AC, IOU	GMM+NDVI outperformed DL models. The DeeplabeV3+ outperformed the CNN model developed from scratched
Bhusal et al. [115]	Classify bird pests in the image.	Video data from a commercial vineyard captured with cameras.	MobileNet with different input	AC	The best result was obtained with the model trained with all the datasets.
Mac Aodha et al. [117]	Insects detection	Images taken with a Wifi camera	F-RCNN with Resnet50	AC, R, P, F_1-score	The model achieved an accuracy of 96%.
Ramalingam et al. [118]	Insect detection using Internet of Things (IoT) base architecture.	Images from the traps.	Fast RCNN with ResNet 50, YOLO, SSD	Ac	The trained model achieves 96% accuracy in insect detection and outperforms YOLO and SSD in terms of accuracy. .

2.3.8.5 Soil management

For experts in agriculture, soil is a heterogeneous natural resource with complex processes and unclear mechanisms. Its temperature alone can provide insight into the impact of climate change on regional yields. Deep learning algorithms study the processes of evaporation, moisture, and soil temperature to understand the dynamics of the ecosystem and the implications for agriculture.

Li et al. [120] used a bidirectional LSTM model to estimate soil temperature at 30 sites under five different climate types. Soil temperature (ST) measurements were obtained from the U.S. Department of Agriculture's National Water and Climate Center, which has established more than 200 sites across the country to collect data on meteorology, soil, and solar radiation. Two models were trained with different inputs and outputs. The first model received the meteorological weather conditions including daily hours, minimum and maximum air temperature, minimum and maximum relative humidity, vapor pressure, average solar radiation, and average wind speed, and outputted the soil temperature amplitude obtained by subtracting the daily average soil temperature from the hourly soil temperature. The second model obtained the meteorological weather conditions including month, day of the month,

observed air temperature, dew point temperature, minimum and maximum air temperature, minimum and maximum relative humidity, vapor pressure, average solar radiation, average wind speed, and outputted the daily average ST. To calculate the hourly ST, the output of the first model was added with the output of the second model and called the integrated BiLSTM model. The result of the model was compared with the BiLSTM model, which directly estimates the hourly ST, the LSTM model, the deep neural network, random forest, SVR, and Linear Regression. The integrated BiLSTM model outperformed the other models in terms of MAE, RMSE, and R2. The LSTM model achieved the second-best performance.

It was also found that the performance of each model is not as good in snowy areas as in warm or dry areas and that, the accuracy of the other models increases with soil depth (except for RF). This behavior could be due to a change in the standard deviation of the soil temperature at different depths and climate types, but this is not investigated in the study.

Yu et al. [121] implement CNN (Conv2D, Conv3D), ConvLSTM to estimate the soil temperature. The difference between Conv2D and Conv3D is the size of the input channels. In the ConvLSTM model [122], there are convolution structures in both the input-to-state and state-to-state transitions. The model obtains the last ten days of historical data from spatiotemporal ST and predicts the ST one, three, and five days in advance. Each model was trained with two different input channels. The first time the raw data from ST was fed into the model, the second time, the input was processed using the Empirical Mode Decomposition (EMD) method, which is a proposed method for processing signals. In the EMD method, the number of channels was increased from one to ten. To complement the model DL, persistent prediction (PF) is used, a simple prediction method that treats the temperature of the first day as a prediction for the next day. When forecasting ST with one-day historical data, PF outperformed the models of DL with raw data input. On the other hand, when the models of DL used the EEMD- processed data as input, the prediction performance was significantly improved. Among all the models, EEMD-Conv3D performed the best in predicting the spatiotemporal ST. It could be noted that ST depends not only on the historical data of ST but also on the meteorological weather conditions, which can be used as input to the model to improve the accuracy.

Alibabaei et al. [26] used Bidirectional LSTM (BLSTM), CNN-LSTM, and a simple LSTM model to model daily reference evapotranspiration and soil water content. Meteorological weather data for three sites in Portugal were collected from the stations Póvoa de Atalaia, Estação Borralheira and Direção Regional de Agricultura e Pescas do Centro, Portugal. Evapotranspiration was calculated from meteorological data using the FAO Penman-Monteith equation and soil water content was retrieved from ERA5 land. The BLSTM model with an MSE of 0.014 to 0.056 outperformed the LSTM, CNN, and CNN-LSTM models. The performance of BLSTM was also compared with traditional machine learning methods such as Random Forest and SVR and outperformed these two models. Among the machine learning methods, RF outperformed the SVR model. The model was not tested with data measured in the field. For use in the field, it can be re-trained with a small set of field-measured data to be used with more confidence in real-world applications.

Table 2.7: Feature descriptions of recent published papers in the field of "Soil management".

Reference	Application	Data used	Model	Metric used	Model performance
Li et al. [120]	Estimation of soil temperature	https://www.wcc.nrcs.usda.gov/scan/	Integrated BiLSTM, BiLSTM, LSTM, RF, SVR, DDN, LR	RMSE, MAE, R^2, MSE	Integrated BiLSTM model outperformed the other models
Yu et al. [121]	Estimation of soil temperature	https://cds.climate.copernicus.eu/	CNN (Conv2D, Conv3D), ConvLSTM	RMSE, MAE, R^2, MSE	EEMD-Conv3D performed the best in predicting the spatiotemporal.
Alibabaei et al. [26]	Estimation of soil water content and evapotranspiration	https://cds.climate.copernicus.eu/ and weather stations in Portugal	BLSTM, CNNLSTM, simple LSTM ,CNN, RF, SVR	MAE, R^2, MSE	BLSTM outperformed other models.

2.3.8.6 Water Management

Water management in agriculture has implications for hydrological, climatological, and agronomic balance. Two papers on the topic were reviewed, where machine learning models are employed to estimate evapotranspiration, allowing for more effective use of irrigation systems, and water table detection, which helps determine crop water needs.

Saggi & Jain [123] used multilayer perceptrons (MLP) to evaluate daily evapotranspiration for irrigation scheduling. They developed their model from scratch. The developed DL model performed well in estimating evapotranspiration and outperformed ML models such as RF, Generalized Linear Model (GLM), and Gradient Boosting Models (GBM).

Zhang et al. [124] presented an LSTM-based model for water management in agricultural areas. The LSTM model with a R^2 value of 0.789-0.952 outperformed the fully connected network. As mentioned in the paper, the prediction of water table depth can help engineers and decision-makers to develop an optimal irrigation scheduling system while controlling the effects of salinity on intensive irrigation.

Loggenberg et al. [125] applied hyperspectral sensing and machine learning to model water stress in vineyards. Stem water potential (SWP) was measured in the field by using a pressure chamber to determine vine water stress status. Vines with SWP values between -1.0 MPa and -1.8 MPa were classified as water-stressed, while vines with SWP values -0.7 MPa were classified as not stressed. Images were acquired using the SIMERA HX MkII hyperspectral sensor, which detects 340 spectral wavebands in the VIS and NIR. A spectral subset consisting of 176 wavebands with a spectral range of 473-708 nm was used as input to the models. Two ML models Random Forest (RF) and Extreme Gradient Boosting (XGBoost) were used for classification. In the first experience, RF and XGBoost models used all wavebands (176 bands) as input. RF with an accuracy of 83.3% outperformed XGBoost with an accuracy of 78%. When using a subset of important wavebands (18 bands), the accuracy of RF and XGBoost was improved (RF =93.3% and XGBoost=90%). The results show that the choice of input to the model in ML is crucial and should be carefully selected. The effect of smoothing the spectral data with the Savitzky-Golay filter in the data preprocessing step was also investigated in the paper. The Savitzky-Golay filter reduces the model accuracy ranging between 0.7% and 3.3%.

Chen et al. [61] used a deep Q-learning (DQN) model for an irrigation decision strategy based on short-term weather forecasts for rice. Daily observed meteorological data of the rice-growing period at three stations, including daily minimum and maximum temperature, average temperature, average wind speed, sunshine duration, average relative humidity, and precipitation were collected. The state-space consisted of $\mathrm{P_t}$ is the predicted precipitation sequence for the next 7 days on day t, mm; $\mathrm{h_t}$ is the water depth on day t, mm; $\mathrm{h_min}$ is the lower limit of water depth, mm; $\mathrm{h_max}$ is the upper limit of water depth, mm; $\mathrm{H_p}$ is the maximum allowable water depth after precipitation. In the action space, 3 possible actions were included, representing the fraction of the irrigation quota (irrigation to $\mathrm{h_max}$) to be supplied to the field on day t. Action 0 returns 0%, action 1 returns 50%, and action 2 returns 100%. The reward is based on the rainfall use reward and the yield reward. The results of the DQN irrigation strategy compared to the results of the conventional irrigation strategy showed a significant reduction in irrigation water volume, irrigation timing, and drainage water without yield loss.
Alibabaei et al. [28] uses the DQN model to schedule irrigation of a tomato field in Portugal using climate Big Data. Historical data are collected from various sources and processed for use as input. Two LSTM models are trained on the obtained historical data to predict the total soil water in the soil profile for the next day and the tomato yield at the end of a season, respectively.
The trained LSTM models were used in the DRL training environment, which takes the current state (historical climate data) and action (amount of irrigation) and then returns the next state and reward. The reward was calculated as the net return and the Q-value was estimated using an LSTM model. The results show that the agent learns to avoid water waste at the beginning of the season and water stress at the end of the season. Compared to the threshold and fixed irrigation method, the DQN agent increases productivity by 11% and avoids water waste by 20-30%.

2.3.8.7 Automation in agriculture

Deep learning is also being used to control sensors and robots that enable automation and optimization of agricultural processes. These robots can be used for a variety of purposes, including automated seeding, pesticide, and crop nutrient application, damage repair, irrigation, weed detection and removal, harvesting, etc.
Li et al. [73] used a deep-learning algorithm to quickly and accurately detect and locate longan fruit in imagery and pick the fruit with a UAV device. An RGB-D camera on the UAV was used to collect images of the fruits. As mentioned in the paper, one of the disadvantages of UAVs is that they are easily affected by local circulation and airflow when capturing images, resulting in blurred images of the fruits. In the preprocessing of the data, a Fuzzy image processing algorithm was used to remove the blurred images from the data set.
FPN, YOLOv3, YOLOv4, MobileNet-SSD, YOLOv4-tiny, and MobileNet-YOLOv4 models were trained on the images to detect and locate string fruits, simple fruits, and fruit branches. In the final phase, the detection results are mapped onto the optimized depth image to extract the contours and spatial information of the three targets. Based on this information, the

Table 2.8: Feature descriptions of recent published papers in the field of "Water Management".

Reference	Application	Data used	Model	Metric used	Model performance
Saggi & Jain [123]	Estimation of evapotranspiration for irrigation scheduling.	Fourteen years of time-series data were used.	multi-layer DL	RMSE	MLDL achieved better performance compared to RF, GLM, and GBM.
Zhang et al. [124]	Prediction of water table depth in agriculture areas	Daily meteorological data of 31 years for Hoshiarpur and 38 years for Patiala was considered for the study.	LSTM	R^2, RMSE	LSTM achieved better performance.
Loggenberg et al. [125]	model water stress in vineyards	Stem water potential (SWP) was measured in the field using a pressure chamber. Images were acquired using the SIMERA HX MkII hyperspectral Sensor	RF and XGBoost	AC	RF outperformed XGBoost
Chen et al. [61]	Irrigation decision-making for rice base on weather forecasts	`http://data.cma.gov.cn`	DQN, CNN	The threat score (TS), missing alarm rate (MAR) and false alarm rate (FAR)	the DQN irrigation strategy compared to the results of the conventional irrigation strategy showed a significant reduction in irrigation water volume.
Alibabaei et al. [28]	Irrigation decision-making for tomato base on weather forecasts	`www.drapc.gov.pt`	DQN, LSTM	R^2-score, RMSE	Compared to the threshold and fixed irrigation method, the DQN agent increases productivity by 11% and avoids water waste by 20-30%.

drone can detect and locate the fruits and use them in harvesting. In general, the YOLO models had faster detection speed and achieved better accuracy than the FPN and SSD models. MobileNet-YOLOv4 achieved the best performance in terms of accuracy (mAP=89.73) and inference time (68s), while FPN achieved the worst performance.

To test the model in orchards, a picking platform was developed with a UAV, a Jetson TX2, an RGB-D camera, a set of scissors with clamps, and a support frame. The accuracy rate of successful harvesting in four cases was reported as 75, 75, 69.23, and 68.42.

Aghi et al. [126] presents a low-cost, energy-efficient local motion planner for autonomous navigation of robots in vineyards based on RGB-D images, low-range hardware (a low-cost device with low power and limited computational capabilities), and two control algorithms. The first algorithm uses the disparity map and its depth representation to generate proportional control for the robot platform. The second backup algorithm, based on a deep learning algorithm, takes control of the machine when the first block briefly fails and generates high-level motion primitives.

An Intel RealSense Depth Camera D435i RGB-D camera was used to capture images and compute the depth map on the platform. The video was captured in different terrain, quality, and time of day. Then, a light depth map-based algorithm processes the depth maps to

detect the end of the vineyard row and then calculates control values for linear and angular velocities using a proportional controller. Since, as in many outdoor applications, sunlight negatively affects the quality of the results and interferes with the control provided by the local navigation algorithm, MobileNet was trained to classify whether the camera was pointed at the center of the end of the vineyard row or one of its sides, and it was used when the first algorithm failed due to outdoor conditions. The CNN model was trained with transfer learning and classified the images into three classes: Middle, Left, and Right. For the middle class, the video was taken in rows with the camera pointed at the center, and for the other two classes, the videos were rotated left and right with the camera at a 45-degree angle to the long axis of the row. The accuracy of the model on the test set was one. The model was trained on a small portion of the data set to investigate the significance of the size of the data set. The accuracy of the model decreased by 6% with the small data set.

Finally, the CNN model was optimized by discarding all redundant operations and reducing the floating-point accuracy from 32 to 16 bits. The accuracy of the optimized model was the same as the original and the time inference was improved. The proposed model was implemented on a robot and the tests were performed in a new vineyard scenario. The first algorithm can detect the end of the vineyard regardless of the direction of the long axis of the robot. When the first algorithm fails, the CNN model jumps in and detects the end of the vineyard.

Badeka et al. [127] trained YOLOv3 to detect grape crates in the field for use on robots harvesting grapes. The images of the crate were taken under natural field conditions. Three data enhancement techniques were used, including rotation, noise, processing, and blur processing. The model achieved an accuracy of 99.74% (mAP%) with an inference time of 0.26 s. To use the trained model on robots, it must be deployed on edge devices and report the inference time and accuracy of the model on these devices. Another interesting problem is that the robot can detect whether the box is full or not.

Majeed et al. [128] combined the DL model with mathematical models to detect cordons in grape canopies and determine their trajectories. Images were taken so that a tree trunk was approximately in the center of the field of view of a high-resolution camera (Sony cyber-shot RX100 IV). A total of 191 images of random RGB images were captured from two different vineyard blocks with different cultivars. Each image in the dataset was classified into three classes: Background, Trunk, and Cordon (`http://hdl.handle.net/2376/16939`). Since the background pixels in the labeled images covered most of the images, a weight was assigned to each class in the preprocessing of the data, which was calculated using a median frequency class balancing method to avoid bias during training. Data augmentation and transfer learning techniques were used in this work.

A fully convolutional neural network (FCN) and SegNet with VGG and AlexNet backbones were trained to segment the cordons in the images. The FCN-VGG16 network achieved the best performance in terms of mean accuracy and F_1-score compared to the other networks. In the next step, the mathematical model including the Fourier model, Gaussian model, polynomial model, and sine sum model was applied to estimate the trajectory of the cordons. The sixth-order polynomial equation with an average R^2 value of 0.99 had the best fit for the tra-

jectories of the cordons compared to the other mathematical models tested.

Pinto de Aguiar et al. [129] trained seven different DL models for detecting grapevine stems in two vineyards in Portugal. The trained models included Single Shot Multibox (SSD) MobileNet-V1 ($\alpha = 0.75$), SSD MobileNet-V1 with different hyperparameters ($\alpha = 0.75$), SSD MobileNet-V2, Pooling Pyramid Network (PPN) MobileNet-V1, SSDLite MobileNet-V2 , SSD Inception-V2, Tiny YOLO-V3. In this work, Transfer Learning was applied using the COCO dataset. A dataset was created using a robotic platform. This dataset contains images captured in two different vineyards, using two cameras each, including the Raspberry Pi cameras and a conventional infrared filter, and the other camera Mako G-125C and an infrared filter (dataset is available at `http://vcriis01.inesctec.pt/-DS_AG_39`). After training the model, two edge devices, including Google's USB Accelerator and NVIDIA's Jetson Nano were used for real-time inference of the model in a real-world application. The advantage of NVIDIA's Jetson Nano over Google's USB Accelerator is that it supports floating points and more models. However, Google's USB Accelerator outperformed NVIDIA's Jetson Nano in terms of inference time (about 49 frames per second), average accuracy, and time to load models. The combination of edge devices and lightweight deep learning models makes the model DL applicable in practice.

Aguiar et al. [40] trained the SSD model using MobileNetv1 and MobileNetv2 as backbones to detect the stem in a vine. The model was deployed on an edge AI mode (Google's USB Accelerator) in order to investigate accuracy and penetration. The images were acquired from four different vineyards in Portugal and published under the name VineSet. They contain RGB and thermal images of a single vineyard and include the annotations for each image (`http://vcriis01.inesctec.pt/datasets/DataSet/VineSet.zip`). A robotic platform AgRob V16 with a frontal stereo RGB camera and a frontal thermal camera was used to capture a video from which the images were extracted. Data augmentation methods such as rotation, translation, scaling, flipping, hue and saturation, Gaussian noise, and random combination were used to resize the dataset to a suitable size for the seepage learning method. Two training datasets were used to evaluate the effect of the training dataset size on the performance of the detectors. The original VineSet dataset and a small subset of it containing 336 non-augmented images were used. The SSD MobileNetv1 and the SSD MobileNetv2 trained with VineSet and using transfer learning achieved similar performance, a maximum AP of 84.16%, but the SSD Inception achieved an AP of 75.78. In terms of inference time, the MobileNets on the TPU achieved an average inference time of 21.18 ms and 23.14 ms, which was faster than SSD Inception. The performance of the models trained with a small dataset dropped significantly (with a maximum of AP =34.42), reinforcing the importance of the dataset size for the performance of the deep learning model.

The SSD MobileNet-V1 trained with VineSet was published on the Google Colaboratory platform (`https://colab.research.google.com`) to help researchers automatically label the new dataset and then manually correct the labeling. Labeling the data is one of the challenges of using the DL model. In this way, labeling the images of the trunk becomes less time-consuming.

Table 2.9: Feature descriptions of recent published papers in the field of "Automation in agriculture".

Reference	Application	Data used	Model	Metric used	Model performance
Li et al. [73]	Pick longan fruits with a UAV device	A RGB-D camera on the UAV device used to capture images from the field.	FPN, YOLOv3, YOLOv4, MobileNet-SSD, YOLOv4-tiny and MobileNet-YOLOv4	mAP, ACC, R, P	The accuracy rate of successful harvesting using a picking platform in four cases was reported as 75, 75, 69.23, and 68.42.
Aghi et al. [126]	Local motion planner for autonomous navigation of robots in vineyards	A camera D435i RGB-D camera was used to capture images from the field.	MobileNet	Ac, R, P, F_1-score	The accuracy of the model on the test set was given as one and it was decreased 6% with the small dataset.
Badeka et al. [127]	Detect grape Crates in the field	The images of the crate were taken under natural field conditions	YOLOv3	AC	The model achieved an average detection rate of 99.74%.
Majeed et al. [128]	detect cordons in grape canopies and then de- determine their trajectories	Images were taken so that a tree trunk was approximately in the center of the field of view of a high-resolution camera	FCN and Seg-Net with VGG and AlexNet	R, P, F_1-score	The FCN-VGG16 network achieved the best performance in terms of mean accuracy and F1 score compared to the other networks.
Pinto de Aguiar et al. [129]	detect grapevine trunk in two vineyards in Portugal using TPU and NVIDIA's Jetson Nano	This dataset contains images captured in two different vineyards, using two cameras	FCN and Seg-Net with VGG and AlexNet, SSD MobileNet-V2, Pooling Pyramid Network, MobileNet-V1, SS- DLite MobileNet-V2, SSD Inception-V2, Tiny YOLO-V3.	IOU, F_1-score	SSD MobileNet-V2 on the TPU coral with an average precision of 52.98% and approximately 49 f/s achieved the best
Aguiar et al. [40]	detect the trunk in a vine	The images were acquired from four different vineyards in Portugal	SSD MobileNetv1, SSD MobileNetv2, SSD Inception	mAP, IOU, R	The SSD MobileNetv1 and the SSD MobileNetv2 achieved similar performance (maximum AP of 84.16%), but the SSD Inception achieved an AP of 75.78.

2.4 Discussion

Deep learning is already being used in various areas of agriculture, but it is still far from being applied in agriculture. In this paper, we reviewed 46 recent research papers to examine the challenges of using deep learning in agriculture. Figure 2.13 shows the pie chart of the researched papers based on the application domain.

The following challenges can be pointed out from the reviewed articles. The most critical point concerns the dataset. Training the model with a small dataset can lead to significant failures [92, 126, 40] and even with transfer learning and data augmentation, training the model may require a sufficient amount of data [75]. The presence of the image under field conditions in the data set is very important [83]. Therefore, collecting authentic field images and datasets under different conditions is the first and most important step. Using open-

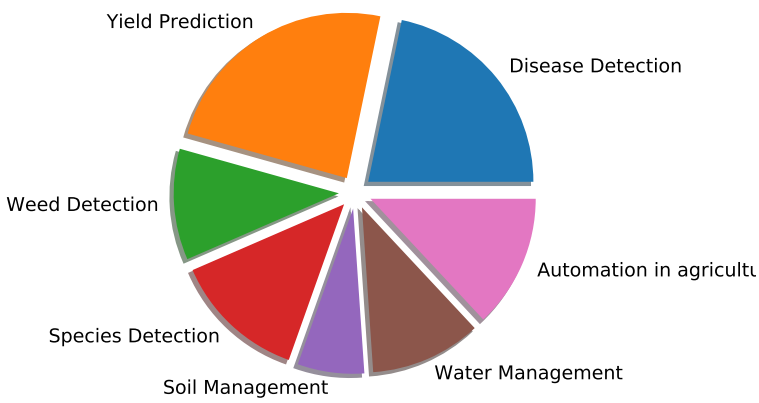


Figure 2.13: Pie chart representing the papers by application area.

source datasets, sharing the dataset with other researchers, data augmentation, and transfer learning properly can help to overcome these challenges [91, 98].

In addition, agricultural image datasets are more complex due to outdoor conditions, the object of interest usually occupying a very small and uncentered part of the image, similarity between objects and background, blocking the object with leaves and branches, multiple objects in one image, and many more [83, 85, 100, 88, 53, 103]. To be applicable in the real world, the dataset should fully represent the state of the environment. In some cases, such as variable illuminance, data augmentation may also be helpful.

The type of input affects the performance of the model [82, 75, 109, 113]. Removing background from images [75], using different color spaces and vegetarian indices as input [82], detecting crop at different growth stages [102], adding vegetarian indices to the input [107, 52], cropping the input image [112], and data from different climate types [120] change the performance of the model. The size of the input also affects the runtime and accuracy of the model [82, 100]. Therefore, it is challenging to determine the best input set for the specific task.

Labeling large amounts of data is expensive and time-consuming [75]. In addition, there are some tasks, such as those related to plant diseases, that can only be performed by specialists in the field. Even when using open-source datasets with already annotated data, there is no way to find out who actually did the annotation. As it was seen in several papers, data augmentation and transfer learning are ways to avoid labeling a large dataset, but labeling a small dataset is still time-consuming [75]. Unsupervised and semi-supervised learning methods can be of great help but need further investigation in this area [57, 58, 81, 104].

When choosing a model for a task, there is a tradeoff between accuracy and inference time [85, 87, 95, 56, 98, 117]. Depending on the application, the model can be chosen.

Neither environment in agriculture is similar, and each environment and problem requires its own data set. Therefore, the DL model may not be universally applicable, i.e., if a model is trained with a dataset from a particular site or a dataset from an open source site such as

ImageNet, it may not be able to be used at another site, or accuracy may decrease when applied to the dataset collected in the real world [20, 89]. The changes in the visual appearance of the images in the training and test dataset could lead to a decrease in model performance [52]. One way to overcome this is to retrain the already trained model with a small dataset from the new environment [107].

The design of deep models is complex and they are referred to as black boxes [13]. One of the challenges in training DL models is that a system with very high computational power (GPU) is needed [13]. Moreover, the performance of these models depends on the choice of hyperparameters, loss functions, and optimization algorithm [80, 87]. Algorithms such as Bayesian optimization [130] can help to find the right hyperparameters [26]. Google researchers used neural architecture search (NAS) algorithm [131] to find state of the art MobilenetV3 [132]. NAS is a method that searches among all possible combinations of submodules that can be repeatedly assembled to obtain the entire model with the best possible accuracy.

The other challenge relates to the real-time applicability of the models. Most deep learning models have many parameters that need to be trained, and after training the model, inference of the model is not made in real time. In some applications, such as using a robot for harvesting, time inference is essential. As we see in this work, most of the models are implemented and tested on the PC, which requires power and it is impractical in real applications. Moreover, implementation on devices like smartphones still, brings various challenges that need to be considered, like memory consumption, speed, etc. Single-stage detection models such as YOLO and SSD, lightweight classification models such as MobileNet, the development of edge devices such as Raspberry Pi and Jetson nano , and the use of cloud computing have made it possible to deploy Deep Learning models in real-world applications and practical ways [56, 129, 92, 73, 107]. The quantization method can be also used to compress the model size and improve the detection speed [103].

On small farms, the use of some methods is sometimes impractical because farmers must have some knowledge to interpret the results obtained. For example, the model that predicts soil moisture for irrigation scheduling requires a human expert to use this information to determine the best time and amount of water for irrigation. As mentioned earlier, Deep Reinforcement Learning is showing great success in several areas as a promising model for building smart farms in several areas [60]. The model can be trained to make decisions about when and how much to irrigate [28, 61]. The information generated by IoT devices also helps farmers track agricultural operations and make better decisions to improve agricultural productivity [118].

2.5 Conclusions

This paper discussed some recent work on the application of DL to agriculture and biodiversity and highlighted some challenges in this area. More sustainable agriculture and the promotion of biodiversity in agricultural systems can be achieved by reducing the use of agrochemicals, low pesticide use and organic farming, appropriate crop rotations, small-scale fields, and the preservation of natural spaces between agroecosystems [133]. As mentioned,

this can be achieved through the new IoT technology in combination with the new biodiversity algorithms and Artificial Intelligence model. They can be used to detect and control species and also to improve ecosystem state and thus productivity having to resort to the use of environmentally damaging practices.
The novelty of the application of DL models and the challenges identified in agriculture demonstrates the need for further research. CNN is the most widely used DL model in agriculture. The use of new methods, such as attention mechanisms, new lightweight models, and single-stage detection models, can improve the performance of the model, as a slight improvement in accuracy and run time can significantly improve the final results.
In the future, we expect to develop or improve models that help farmers make crop management decisions. These models can be integrated into decision support tools, and these tools can guide users through precise steps and suggest optimal decision paths. It is expected that the development of these models will also enable biodiversity monitoring. Consequently, the adoption of new sustainable practices supported by DL models and biodiversity monitoring will help manage the farm with minimal human intervention and greater effectiveness.

Chapter 3

Modeling Soil Water Content and Reference Evapotranspiration from Climate Data using Deep Learning method

Abstract[1]

In recent years, deep learning algorithms have been successfully applied in the development of decision support systems in various aspects of agriculture, such as yield estimation, crop diseases, weed detection, etc. Agriculture is the largest consumer of freshwater. Due to challenges such as lack of natural resources and climate change, an efficient decision support system for irrigation is crucial. Evapotranspiration and soil water content are the most critical factors in irrigation scheduling. In this paper, the ability of Long Short-Term Memory (LSTM) and Bidirectional LSTM (BLSTM) to model daily reference evapotranspiration and soil water content is investigated. The application of these techniques to predict these parameters was tested for three sites in Portugal. A single-layer BLSTM with 512 nodes was selected. Bayesian optimization was used to determine the hyperparameters, such as learning rate, decay, batch size, and dropout size.The model achieved mean square error (MSE) values ranging from 0.07 to 0.27 $(\mathrm{mm\,d^{-1}})^2$ for ETo (Reference Evapotranspiration) and 0.014 to 0.056 $(\mathrm{m^3m^{-3}})^2$ for SWC (Soil Water Content), with R^2 values ranging from 0.96 to 0.98. A Convolutional Neural Network (CNN) model was added to the LSTM to investigate potential performance improvement. Performance dropped in all datasets due to the complexity of the model. The performance of the models was also compared with CNN, traditional machine learning algorithms Support Vector Regression, and Random Forest. LSTM achieved the best performance. Finally, the impact of the loss function on the performance of the proposed models was investigated. The model with the mean square error as loss function performed better than the model with other loss functions.

Keywords

Agriculture; Deep learning; LSTM; support decision-making algorithms; irrigation management.

[1]This work was published in [26], K. Alibabaei, P. D. Gaspar, and T. M. Lima, "Modeling soil water content and reference evapotranspiration from climate data using deep learning method," Applied Sciences, vol. 11, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/11/5029

3.1 Introduction

The world's population living in urban areas will increase by 13% by 2050 [4]. On the other hand, with the current agricultural production method, the capacity of the Earth is already exceeded [5]. Therefore, improving agricultural production and feeding the world without exceeding the consumption such as water remains the main problem in agriculture.
Agricultural production depends on water. It is also the largest consumer of water as it consumes on average 70% of all freshwater [134]. Therefore, it is essential to optimize irrigation and make irrigation systems more efficient, especially in arid regions. Many factors need to be considered when scheduling irrigation. The amount of water available in the soil and reference evapotranspiration (ETo) are two critical factors [135, 136]. Evapotranspiration is the amount of water that evaporates from the Earth and plant surface. Solar radiation, wind speed, temperature, and relative humidity all influence daily ET, and this effect is highly non-linear. The Penman-Monteith equation recommended by the Food and Agriculture Organization (FAO) is the most commonly used equation for calculating ET [137].
Soil Water Content (SWC) is the volume of water per unit volume of soil. It can be measured by weighing a soil sample, drying the sample to remove the water, and then weighing the dried soil or by remote sensing methods. Measuring SWC is important for studying the statistics of available water, and scheduling irrigation [136]. ET and SWC can be measured by remote sensing methods. Jimenez et al. [138] lists some advantages and disadvantages of some remote sensing methods. For example, Clulow et al. [139] used Eddy Covariance (EC) Systems in South Africa to determine evapotranspiration in a forested area. The system EC uses sensors (such as an anemometer, Infrared Sensor, quality air sensor, precision flowmeter, or vapor flow meter) to collect data from the field and calculate evapotranspiration. The study points out some disadvantages of using this system in remote areas, such as the relatively high power requirement, which requires careful and frequent attention, and the verification, correction, and analysis of the data for complete records. However, the machine learning algorithm is able to extract the feature from the raw data and calculate ET and SWC without any further effort. In recent years, many studies have been conducted on the use of machine learning methods for irrigation scheduling. Support vector machine, single-layer neural network, and deep neural network are the most commonly used methods for simulating soil moisture, and ET [140, 141, 142, 143, 144, 138]. Achieng [136] used machine learning techniques including three support vector regressions (Radial Basis Function (RBF), linear and polynomial kernel SVM), a single-layer Artificial Neural Network (ANN), and eight multilayer Deep Neural Networks (DNN) to simulate soil water content. The results showed that the RBF-based SVR outperformed the other models to simulate soil water content. Tseng et al. [145] used CNN to predict soil moisture status from images. The experimental result on the test images showed that CNN performed best with a normalized mean absolute error of 3.4% compared to SVM, RF, and two-layer Neural Networks. Song et al. [146] used a Macroscopic Cellular Automata (MCA) model by combining the deep belief network (DBN) to predict the SWC. Compared to the multilayer perceptron, the DBN-MCA model resulted in an 18% reduction in mean squared error. Adamala [147] developed a generalized wavelet neural network (WNN)-based model to estimate the reference evapotranspiration. The pro-

posed models were compared with ANN, linear regression (LR), wavelet regression (WR) and HG methods. The WNN and ANN models performed better compared to the LR, WR, and HG methods. Saggi and Jain [123] used a Deep Learning-Multilayer Perceptron (MLP) to determine the daily ETo for Hoshiarpur and Patiala Districts from Punjab. The performance of the MLP was compared with machine learning algorithms such as Random Forest (RF), Generalized Linear Models (GLM) and Gradient Boosting Machine (GBM). The MLP model had the best performance, with the mean square error ranging from 0.0369 to 0.1215. De Oliveira e Lucas et al. [148] employed three CNNs with distinct structures to make daily predictions for ETo. Comparisons were made between the CNN, the seasonal ARIMA (SARIMA), and Seasonal Naive (SNAIVE). The CNN model performed better in terms of variance, accuracy, and computational cost.

However, traditional machine learning cannot store and learn from the past observations of a time series. Under Deep Learning, LSTM models are designed to model time series data. Adeyemi et al. [142] developed a Neural Network approach to model temporal soil moisture. In the model evaluation, a value of R^2 above 0.94 was obtained at all test sites. Zhang et al. [124] used an LSTM model to predict agricultural water table elevation. They used 14 years of time-series data such as water diversion, precipitation, reference evapotranspiration, temperature, and water table depth. The proposed model achieved higher values of R^2 than the model Feed-Forward Neural Network (FFNN) in predicting water table depth. As indicated earlier, measuring reference evapotranspiration volume and soil water volume is time and labor-intensive.

In this paper, a Long Short-Term Memory (LSTM) was developed to predict these two critical parameters. The model forecasts one day- ahead ET and SWC using eight days of historical data. The various evaluation metrics such as Mean Square Error (MSE), Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Bias Error (MBE), and R^2 were used to demonstrate the effectiveness of the models. Each metric has its advantages and disadvantages, and as the results of this paper will show, using only one metric can lead to misleading results. The Convolutional Neural Network (CNN) is capable of extracting features from raw data [22]. A CNN model is superimposed on the LSTM model to investigate the performance of CNN- LSTM models compared to LSTM. The performance of the model is also compared with CNN, and traditional machine learning algorithms such as Support Vector Regression (SVR) and Random Forest [23, 24]. Finally, the impact of the loss function on the training of the model is investigated. The most commonly used loss functions for regression problems are selected, including MSE, RMSE, MAE, and the Huber function. The Huber function is basically MAE, which becomes MSE when the error is small. Unlike MAE, the Huber function is differentiable at 0 and is less sensitive to outliers in the data than MSE [25]. Each of these functions yields a different error for the same prediction and affects the performance of the model on the test set.

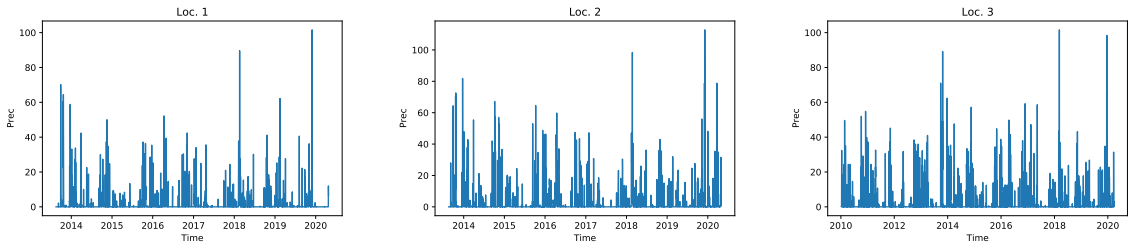


Figure 3.1: Precipitation amount for each site.

3.2 Materials and Methods

3.2.1 Data Collection

Data from three sites in Portugal, collected over a period of 7 to 10 years, were used for the training. Datasets were retrieved from the stations Estação Póvoa de Atalaia, Estação Borralheira and Direção Regional de Agricultura e Pescas do Centro, Portugal. Table 3.1 shows details of these sites. The soils of Póvoa de Atalaia (Loc. 1) and Fadagosa (Loc. 3) are of light texture (sandy) or more often of medium texture (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. Climatically, these are areas with mild winters, with precipitation falling mostly in autumn and winter and occasionally in the spring. Usually, temperatures increase significantly from May onwards, peaking in July and occasionally in August, with the registration of several days of maximum temperatures above $40°$C being common. It has also been noted that registered temperatures reach progressively higher values in September, extending the summer period into October. In the Borralheira (Loc. 2), soils are similar, although soils with medium texture (loam) are more common. Climatically, the regions differ mainly in spring and summer with lower temperature values and higher relative humidity. In terms of precipitation, the values are always higher, and the number of days recorded with precipitation is higher. These plots are covered with fruit trees such as pears, nectarines, cherry, and peach trees.

Datasets Loc. 1 and Loc. 2 were recorded daily, and the Loc. 3 record was recorded every 15 minutes. These datasets included minimum, maximum, and average temperature; minimum, maximum, and average relative humidity; average solar radiation; minimum and average wind speed; and precipitation. Table 3.13 in the Appendix shows the details of these variables. Figure 3.1 shows the amount of precipitation at each site, which translates to the soil water content in the surface layer of the soil.

The ETo calculator is a software developed by the Land and Water Division [137]. The ETo calculator estimates ETo from meteorological data using FAO Penman-Monteith equation [137]. This calculator was used to calculate the daily ETo at these three locations. The amount of soil water content at the end of the day was collected from the ECMWF dataset (ERA5) [149]. ERA5-Land is a reanalysis dataset that provides a consistent view of the evolution of land variables over several decades with improved resolution compared to ERA5. ERA5-Land was created by reproducing the land component of the ECMWF ERA5 climate reanalysis. In particular, ERA5-land runs at a higher resolution (9 km compared to 31 km

in ERA5). The reanalysis combines model data with observations from around the world to produce a globally complete and consistent dataset using the laws of physics. The temporal frequency of the output is hourly. ERA5-Land is generated in a single simulation without coupling to the atmospheric module of ECMWF Integrated Forecasting System (IFS) or to the ocean wave model of IFS. It runs without data assimilation, making it computationally affordable for relatively rapid updates. The core of ERA5-Land is the Tiled ECMWF Scheme for Surface Exchanges over Land, which incorporates land surface hydrology (H- TESSEL). It uses the CY45R1 version of the IFS [150].
Soil moisture values refer to the top 7 cm of soil as modeled using the Hydrology Tiled ECMWF Scheme for Surface Exchanges over Land (H-TESSEL) [151] using Richard's equation for water flow through the unsaturated soil profile [152]. H-TESSEL uses a variety of soil texture classes with specific properties, such as infiltration capacity and wilting point. H-TESSEL has a four-layer representation of the soil (0-7cm, 7-28cm, 28-100cm, 100-289cm). The volumetric soil water is associated with soil texture, soil depth, and underlying water table [149]. Since layer four is very deep and its role in agriculture applications is limited, layer four was discarded from the dataset, and the three layers between 0 and 100cm were considered in this study.
Figure 3.2 shows ETo ($\mathrm{mm\ d^{-1}}$) and SWC ($\mathrm{m^3 m^{-3}}$) at different levels for each site.

Table 3.1: Details of the locations in the datasets.

Location	**Póvoa de Atalaia (Loc. 1)**	**Borralheira (Loc. 2)**	**Fadagosa (Loc. 3)**
Longitude	$40°4'15.43''$N	$40°19'14.14''$N	$40°1'46.55''$N
Latitude	$7°24'26.02''$W	$7°25'23.67''$W	$7°26'36.27''$W
Start date	2013-08-19	2013-08-19	2020-05-16
End date	2020-05-16	2010-01-07	2020-03-23
Temporal resolution	daily	daily	15 minutes

3.2.2 Data Pre-processing

If there are values in the dataset that are missing, invalid, or difficult to process, the algorithm may end up with overfitted, less accurate, or even misleading results. So, before a model can be trained, data preparation procedures such as data transformation, statistical testing, and more must be performed. The missing data were filled in using the moving average method [153].
The other pre-processing in this work involves the removal of collinear parameters and normalization. The collinear parameters contain the same information about the variables and can lead to redundancies in the calculations. For each parameter x and y, the collinear coefficient is calculated by Equation (3.1) [154]:

$$\rho_{\mathrm{x,y}} = \frac{\mathrm{cov(x,y)}}{\sigma_{\mathrm{x}}\sigma_{\mathrm{y}}} \quad (3.1)$$

where $\mathrm{cov(x,y)}$, σ_{x}, and σ_{y} are the covariance of x and y, the variance of x, and the variance

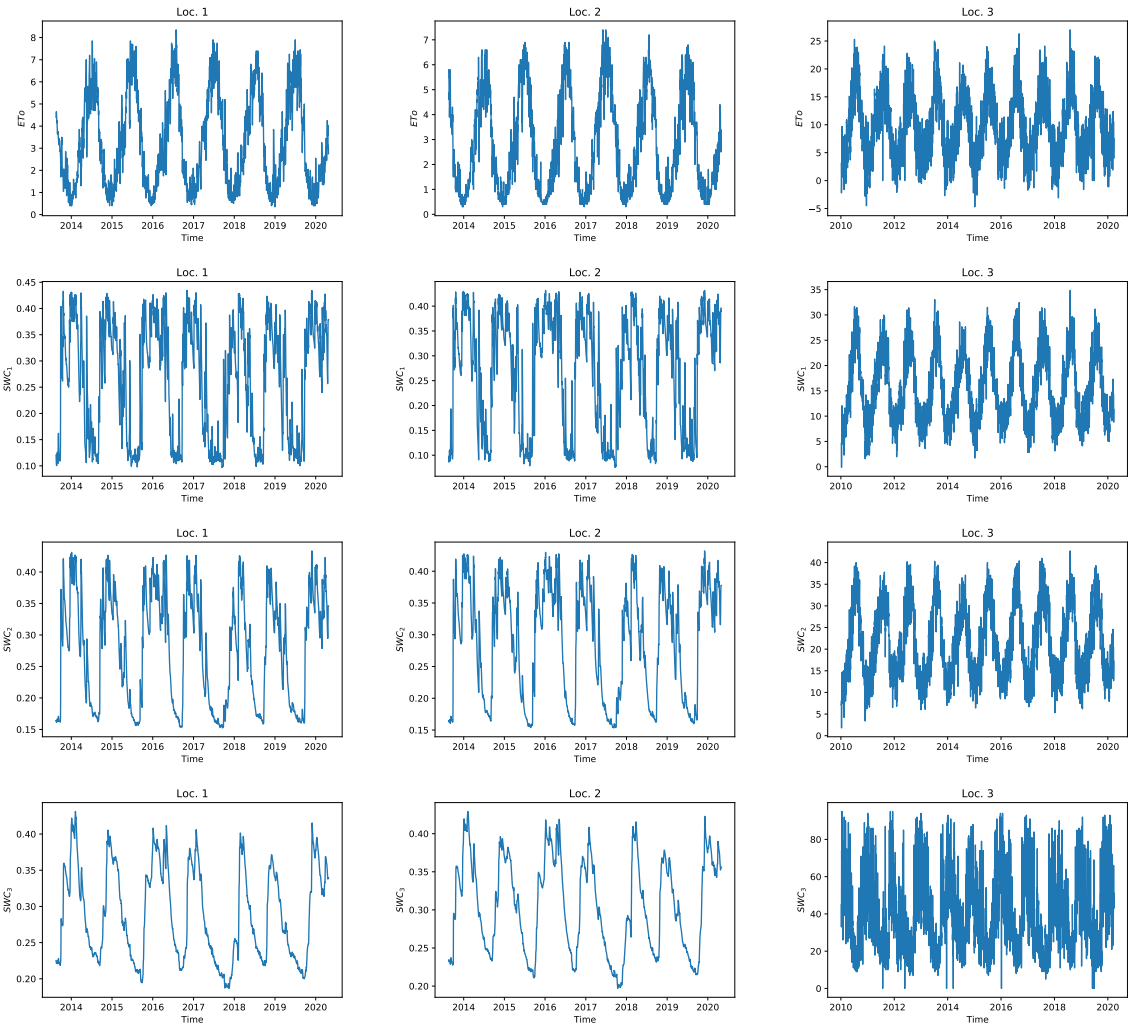


Figure 3.2: ETo and SWC at different levels for each site.

of y, respectively. The values 1 and -1 of the correlation coefficient indicate the linear and inverse linear correlation, respectively, and the value 0 means no correlation. In this work, if $\rho_{x,y}$ exceeded a threshold, then one of the parameters was removed from the dataset. In the end, the parameters temperature (T_{min}, T_{max}), average humidity (HR_{Avg}), average wind speed (WS_{Avg}) and solar radiation (SR_{Avg}), precipitation (Prec), ETo, SWC were used as inputs to the models. Table 3.14 and 3.15 in the Appendix show the collinear coefficients between variables in the different datasets.

The goal of normalization is to bring the values of the dataset to a common scale without distorting the differences in the ranges of values. The standardization formula given by Equation (3.2) was used to normalize the data [155].

$$x_{new} = \frac{x_{old} - \bar{x}}{\sigma} \quad (3.2)$$

where $\bar{x}$ is the mean value of the data.

3.2.3 Deep Learning Methodes

The LSTM, Bidirectional LSTM, and CNN-LSTM models were used in this paper.

3.2.3.1 LSTM Architectures

Recurrent Neural Networks (RNN) are specifically designed for processing sequential data. In the RNN model, the output of the previous step is fed into the current step as input. Mathematically, an RNN model is a nonlinear function that takes t-steps of variables in a time series as input and outputs a vector value at time $t + 1$ [54]. The output of an RNN unit is determined using a tanh function by Equation (3.3):

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_t) \quad (3.3)$$

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 during the training. The main problem with RNN is the vanishing gradient, i.e., the gradient of the loss function approaches zero [54]. LSTM [55] is a special type of RNN, created as a solution to the vanishing problem in RNN. A single-layer LSTM is shown in Figure 3.3, where there is one unit for each time- step. A common LSTM unit consists of a block input z_t with corresponding weight ma-

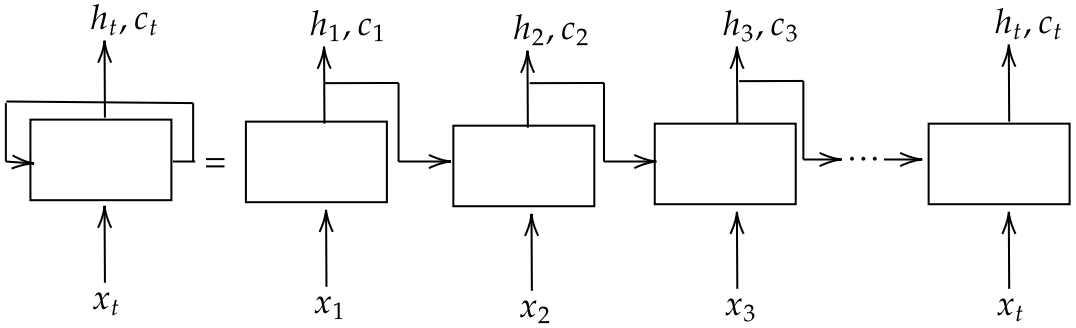


Figure 3.3: LSTM in loops and unfolded in sequence.

trices W_{zx}, W_{zh}, b_z, an input gate i_t with corresponding weight matrices W_{ix}, W_{ih}, b_i, a forget gate f_t with corresponding weight matrices W_{fx}, W_{fh}, b_f, an output gate o_t with corresponding weight matrices W_{ox}, W_{oh}, b_o, and a memory cell c_t (see Figure 3.4).
Forget gate f_t determines what information remains in the cell by outputting a number between 0 and 1 using Equation (3.4). The output 0 means "forget all information completely," and one means "keep all information". The input gate i_t protects the unit from irrelevant inputs and decides which values to update using Equation (3.5). The output gate o_t uses the sigmoid function given by Equation (3.6) to decide which information in the cell is used to calculate the output of the LSTM unit. The block input z_t creates a new vector that could be added to the new state using the tanh function given by Equation (3.7). The memory cell c_t decides whether to update the information obtained from the previous inputs and the current

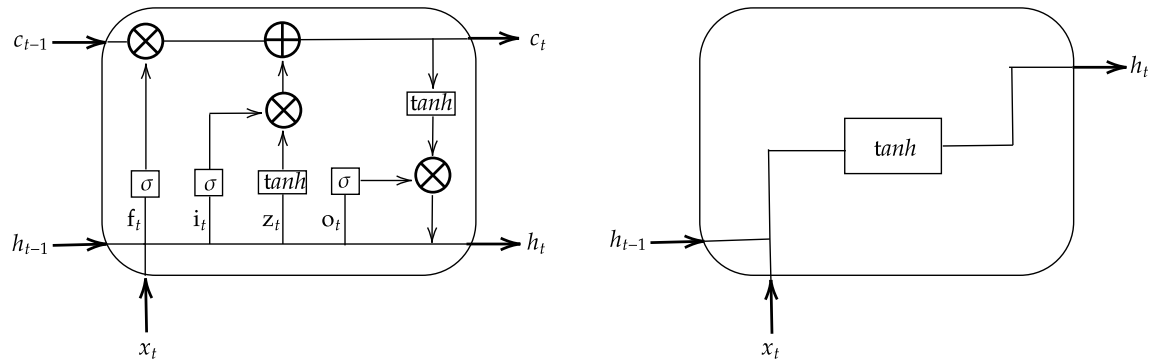


Figure 3.4: A. Diagram of LSTM unit and B. regular RNN unit. Left side shows LSTM cell, right side shows RNN cell.

information provided by Equation (3.8).

$$\mathrm{f_t} = \sigma(\mathrm{W_{fx}x_t} + \mathrm{W_{fh}h_{t-1}} + \mathrm{b_f}) \quad (3.4)$$

$$\mathrm{i_t} = \sigma(\mathrm{W_{ix}x_t} + \mathrm{W_{ih}h_{t-1}} + \mathrm{b_i}) \quad (3.5)$$

$$\mathrm{o_t} = \sigma(\mathrm{W_{ox}x_t} + \mathrm{W_{oh}h_{t-1}} + \mathrm{b_o}) \quad (3.6)$$

$$\mathrm{z_t} = \tanh(\mathrm{W_{zx}x_t} + \mathrm{W_{zh}h_{t-1}} + \mathrm{b_z}) \quad (3.7)$$

$$\mathrm{c_t} = \mathrm{f_t} * \mathrm{c_{t-1}} + \mathrm{i_t} * \mathrm{z_t} \quad (3.8)$$

The output of the current block $\mathrm{h_t}$ is calculated using Equation (3.9).

$$\mathrm{h_t} = \mathrm{o_t} * \tanh(\mathrm{c_t}) \quad (3.9)$$

3.2.3.2 Bidirectional LSTM

Bidirectional LSTM (BLSTM) is an extension of the LSTM model that can improve the results [156]. Each layer of a BLSTM is a stack of two LSTM layers, called the forward and backward layers (see Figure 3.5). The input of the forward layer is the sequence in positive order and computes a sequence of forward hidden states $\overrightarrow{\mathrm{h}_1}, \cdots, \overrightarrow{\mathrm{h}_t}$. The backward layer obtains an inverted copy of the sequence and computes a sequence of the backward hidden states $\overleftarrow{\mathrm{h}_1}, \cdots, \overleftarrow{\mathrm{h}_t}$. The final representation of the observation is obtained by concatenating the forward hidden states with the backward hidden states.

3.2.3.3 CNN-LSTM model

CNN is a supervised learning model specifically designed to handle classification, detection, and segmentation [18]. A CNN model consists of a stack of convolutional layers, nonlinear activation functions (e.g., tanh, ReLU), pooling layers (e.g., maximum pooling, average pooling), and fully connected layers [54]. A convolutional layer contains a set of kernels whose parameters must be learned, and it is used to extract features from data. Each kernel slides

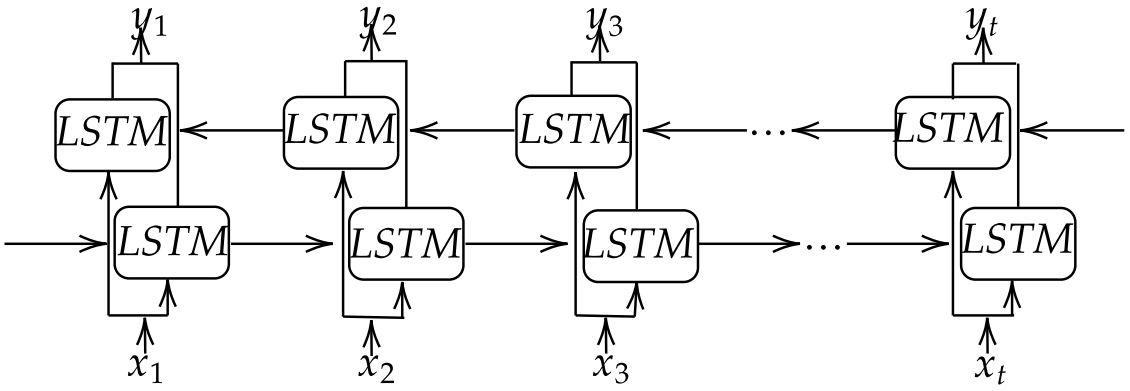


Figure 3.5: One layer BLSTM model.

across the height and width of the input, and the dot product between the entries of the kernel, and each position of the input is computed (see Figure 3.6). A nonlinear activation function is used to introduce nonlinear features into a model. Rectified linear units (ReLU) have become state of the art and are the most commonly used activation function in deep learning models [54]. The output of a ReLU function is $\mathrm{Max}(\mathrm{x}, 0)$.

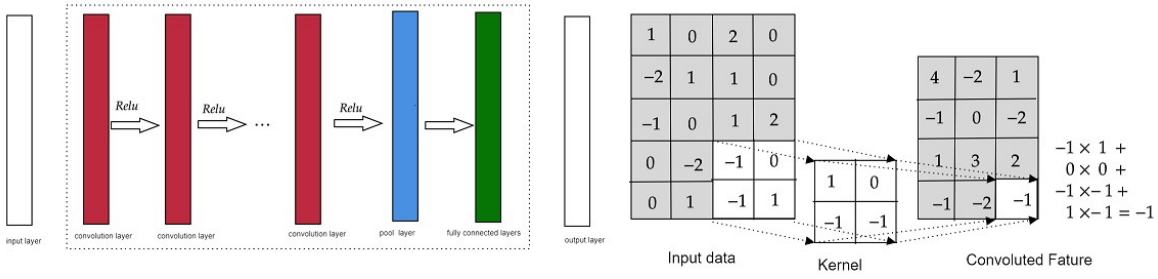


Figure 3.6: The left-hand side is the general CNN architecture, and the right-hand side is the convolutional operation.

A CNN-LSTM model is a participation of the convolutional neural network and an LSTM model. The CNN layers are used to extract features from the input sequence, and an LSTM model is used to support sequence prediction [157].

3.2.3.4 Dropout

Overfitting occurs when the training loss is small but the generalization of the model to the unseen data is unreliable. To prevent overfitting, it is suggested that regularization techniques such as dropout be used [54]. Dropout is a computationally inexpensive way to regularize during model training by removing units from the network. In LSTM models, dropout can be added to the input layer, outputs, and recurrent outputs [158]. Dropout size is a hyperparameter that should be set during training. Table 3.2 shows the effect of dropout based on the lost validation. In this work, the dropout was used on recurrent outputs.

Table 3.2: Validation MSE for different dropout size (Loc. 3).

	Dropout Size		
Dropout on	0.00000	0.10000	0.20000
outputs	0.02050	0.01970	0.01960
inputs	-	0.01996	0.02000
recurrent-outputs	-	0.01950	0.09136

3.2.4 Bayesian Optimization

Bayesian optimization is a strategy for the global optimization of black-box functions. A black-box function is an unknown analytic expression of the function, and the evaluation of the function is restricted to a sample at each point [130]. The core components in Bayesian optimization are the objective function, the surrogate model, and the acquisition function. The objective function is the black box function used to maximize compliance with some parameters. For example, in the classification problem, accuracy can be used as the objective function. The surrogate model is a model to approximate the objective function. The Gaussian processes and Parzen-Tree Estimator are the most popular models for the surrogate model [159]. The acquisition function is used to determine where to evaluate the objective function next. There are many options for acquisition functions, such as the Probability of Improvement (PI), Expected Improvement (EI), and upper confidence bound [159].
The Bayesian algorithm is as follows [160]:

1. Choose a surrogate function to approximate the objective function and define its prior. The choice of this function is optional; the algorithm is convergent, and the choice of the prior function can help increase the speed of convergence.

2. Given a set of observations, Bayes' rule is used to obtain the posterior distribution over the objective function. Bayes rule is given by Equation (3.10):

$$\mathrm{P}(\mathrm{A}|\mathrm{B}) = \frac{\mathrm{P}(\mathrm{B}|\mathrm{A})\mathrm{P}(\mathrm{A})}{\mathrm{P}(\mathrm{B})} \quad (3.10)$$

   where A and B are models, and new observations, respectively; $\mathrm{P}(\mathrm{A})$ and $\mathrm{P}(\mathrm{B})$ are the probabilities of the prior distribution (probability that A is true) and the probability that B is observed, respectively; $\mathrm{P}(\mathrm{A}|\mathrm{B})$ is the posterior distribution, and $\mathrm{P}(\mathrm{B}|\mathrm{A})$ is the likelihood of the event B occurs if model A is true.

3. Use an acquisition function α which is a function of the posterior distribution, to determine the next sample point as $\mathrm{x_t} = \underset{\mathrm{x}}{\operatorname{argmax}}\, \alpha(\mathrm{x})$.

4. add the new sample to the observation set and go back to step 2 until convergence or budget has elapsed.

In this paper, Bayesian optimization [161] is used to minimize the loss function in terms of batch size, dropout size, learning rate, and decay. Figure 3.7 shows the validation loss during Bayesian optimization for Loc. 3.

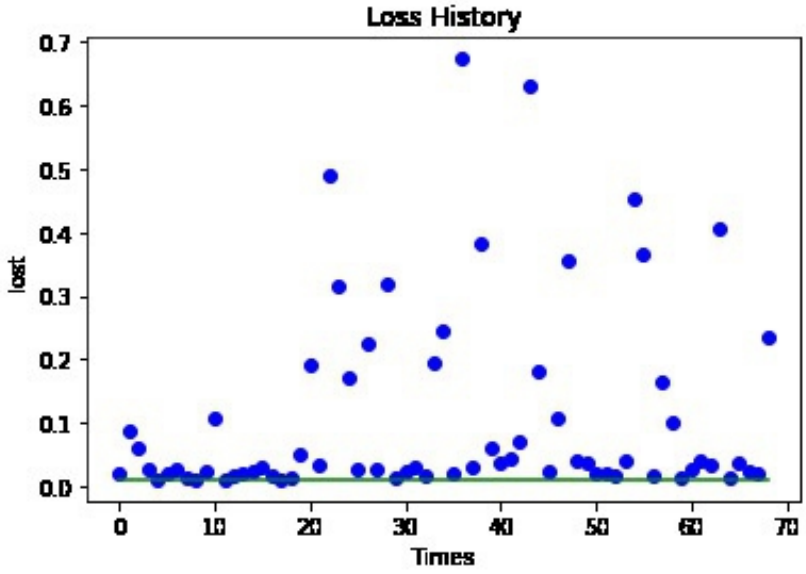


Figure 3.7: Validation loss vs time using Bayesian optimization.

3.2.5 Evaluating the models

Once the model is trained, the critical question is how good the model is. The following metric is used to evaluate the model:

- Mean Square Error (MSE): is the average of the squared differences between prediction and actual observation. In other words, the MSE is the variance of the error [162]. It is calculated by Equation (3.11):

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad (3.11)$$

- Root Mean Square Error (RMSE): is the square root of the MSE. In other words, the RMSE is the standard deviation of the error [162]. In the MSE metric, the errors are first squared before averaging, resulting in a high penalty for large errors. Therefore, the RMSE is useful when large errors are undesirable.

- Mean Absolute Error: is the average of the absolute differences between predictions and actual observations and is calculated by Equation (3.12) [162]:

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (3.12)$$

- Mean Bias Error (MBE): captures the average bias in the prediction. A positive bias in a variable means that the data from the datasets are underestimated, while a negative bias means that the model overestimates the observed data. It is calculated by Equation (3.13):

$$\mathrm{MBE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i) \quad (3.13)$$

- R^2: is a statistical measure that calculates the variance explained by the model over the total variance. The higher R^2 is, the smaller the differences between the actual observations and the predicted values. It is calculated by Equation(3.14) [124]:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2} \quad (3.14)$$

where y_i's, y_i's, and $\overline{y}$ are the actual observations, the values predicted by the model, and the mean value of the actual observations, respectively.

3.3 Results and Discussions

The experimental environment configuration consists of an Intel Core i5-4200H CPU, an NVIDIA GEFORCE GTX 950M graphics card, and 6.144GB RAM. The models were implemented using the Tensorflow framework, and the Keras library [163, 164].
The dataset was divided into training, validation, and test set (70%, 20%, and 10%, respectively). The model predicted SWC and ETo for one day ahead, considering eight days of historical data, including minimum temperature (T_{min}), maximum temperature (T_{max}), average humidity (HR_{Avg}), average wind speed (WS_{Avg}) and solar radiation (SR_{Avg}), daily ETo and SWC . Mean square error and Adam optimization method [165] were used for the loss function and optimization method.
Various machine learning algorithms have certain values that should be set before starting the learning process. These values are called hyperparameters and are used to configure the training process. The validation set is used to set the hyperparameters. The Hyperparameters of the proposed model are learning rate, batch size, dropout size, number of layers in the model, and number of nodes per layer. The number of hidden layers and the number of nodes per layer were selected manually, and then Bayesian optimization was used to set the remaining hyperparameters. Unlike neural networks, LSTM does not require a deep network to obtain accurate results [22]. Changing the number of nodes per layer has been shown to have a greater impact on results than changing the number of layers [22]. In this work, the number of LSTM layers stayed between 2 and 4, for BLSTM less than 3, and kept the number of nodes per layer below 513. Finally, the network architectures were tested with two, three, and four LSTM layers, one and two BLSTM layers, 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, and the results for the different sites are shown in Table 3.3. Increasing the number of LSTM layers to four and the number of BLSTM layers to two increased the MSE on the validation dataset, showing the overfitting of the models. Keeping

the LSTM layers of two and three for Loc. 1 and Loc. 2 and increasing the number of LSTM nodes per layer, the network accuracy on the validation dataset first increases and then decreases, and the best network accuracy is achieved with 256 nodes. The best accuracy for Loc. 3 was achieved with a single-layer BLSTM with 512 nodes. In the end, the mean error for each architecture was calculated on three datasets, and the single-layer BLSTM with 512 nodes had the best performance with a mean of 0.01536. Among the simple LSTM models, the double-layer LSTM with 256 nodes performed better than the other models.

Table 3.3: Validation MSE for different number of LSTM layers and LSTM nodes per layer.

model architecture	**locations**	**Number of nodes per layer**			
		64	128	256	512
LSTM-2 layers	Loc. 1	0.01713	0.01197	0.01135	0.01222
	Loc. 2	0.01906	0.01591	0.01446	0.01466
	Loc. 3	0.02064	0.02059	0.02065	0.02035
	Mean	0.018943	0.016156	0.015486	0.015743
LSTM-3 layers	Loc. 1	0.01973	0.01221	0.01114	0.01128
	Loc. 2	0.01923	0.01566	0.01463	0.01460
	Loc. 3	0.02168	0.02089	0.02106	0.02134
	Mean	0.020213	0.016253	0.01561	0.01574
LSTM-4 layers	Loc. 1	0.01911	0.01401	0.01317	0.01156
	Loc. 2	0.02280	0.01623	0.01490	0.01490
	Loc. 3	0.21685	0.15875	0.12353	0.11088
	Mean	0.086253	0.0629	0.05053	0.045886
BLSTM-1 layer	Loc. 1	0.01527	0.01207	0.01104	0.0115
	Loc. 2	0.01870	0.01715	0.01556	0.01522
	Loc. 3	0.02001	0.02023	0.01949	0.01936
	Mean	0.017993	0.016483	0.015363	0.01536
BLSTM-2 layers	Loc. 1	0.01574	0.01374	0.01268	0.01289
	Loc. 2	0.01893	0.01689	0.01583	0.01583
	Loc. 3	0.02017	0.01959	0.01945	0.01982
	Mean	0.01828	0.01674	0.015986	0.01618

Optimizing the hyperparameters of a machine learning model is simply a minimization problem that seeks the hyperparameters with the least validation loss. Tuning hyperparameters using Bayesian optimization can reduce the time required to find the optimal set of hyperparameters. In this paper, the validation loss in terms of the learning rate, batch size, decay, and dropout size was minimized using Bayesian optimization [161]. Ten steps of random exploration and 60 iterations of Bayesian optimization were performed. Random exploration can help by diversifying the exploration space [161]. The learning rate and decay were kept between 10^{-7} and 10^{-2}, and the batch size and dropout size were kept in the range of $(32, 128)$ and $(0, 0.5)$, respectively. For each iteration of the Bayesian optimization algorithm, the network was run for 50 epochs. In the first ten iterations, the algorithm randomly selected the hyperparameters, calculated the validation loss, and then attempted to minimize the validation loss with respect to the hyperparameters (see Figure 3.7). The validation loss and the selected value for each hyperparameter were stored after each iteration. At the end of 70 iterations, the set of hyperparameters with minimal validation loss was selected. These selected values are shown in Table 3.4. For simplicity, the decay and learning rates were modified to a power of 10.

Table 3.4: Hyperparameters obtained from Bayesian optimization for each location.

Model	learning rate	decay	Batch size	Dropout size
BLSTM	10^{-4}	10^{-5}	124	0.2

The model was trained on different datasets. Once the model was trained on all datasets (70% of each dataset), the other time, the model was trained on two datasets and tested on the third dataset. Each model was trained for a maximum number of 500 iterations (epoch=500). Early stopping was used to avoid overfitting. Early stopping is a regularization strategy that determines how many epochs can be run before overfitting begins [54]. In our implementation, the validation error was monitored during training, and if it did not improve after a maximum of ten epochs, training was stopped. Figure 3.8 shows the R^2 and RMSE during training on the training set and the validation set. The error improved after 194 epochs on the training set but did not improve again on the validation set.

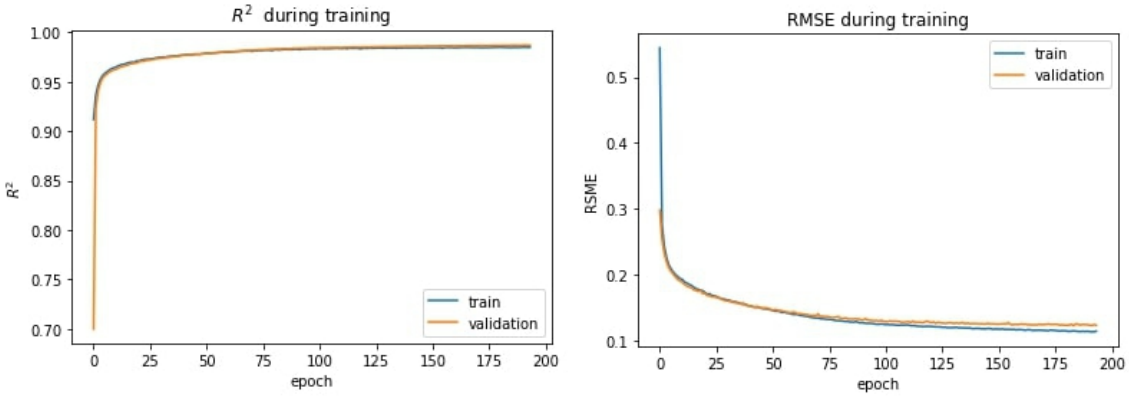


Figure 3.8: R^2 and RMSE during training. The left side shows the R^2-score and the right side shows the RMSE.

Table 3.5, 3.6 and Figure 3.9 show the evaluation of the model on the test set and the predicted value compared to the true values of the SWC and ETo, respectively. The results indicated that although the MSE (the variance of the error) at Loc. 2 was lower than the MSE at Loc. 3, the R^2- score (the variance explained by the model over the total variance) improved at Loc. 3. Therefore, it is not sufficient to use only one metric to evaluate a model.
As Figure 3.9 and Table 3.5 show the best accuracy in predicting ETo, and SWC was achieved with the data set of Loc.2 and Loc. 3, respectively. ET was underestimated with the model in Loc. 2 and Loc. 3 but overestimated in Loc. 1. The results also show that as the depth of the soil increases, the accuracy of the model for predicting SWC increases. This could be due to the fact that the range and standard deviation of SWC decrease as the soil depth increases. Adding some other variables to the input of the model, such as irrigation amount, and adding the real data to the simulated dataset may improve the accuracy of the predicted value of SWC in the upper layer (see Table 3.13).
As the results show, the models trained with two datasets showed similar performance to the model trained with all datasets. The results also show that the model trained with datasets from different locations can be generalized to other locations.
CNN models are capable of extracting features from raw data. The convolutional layers were

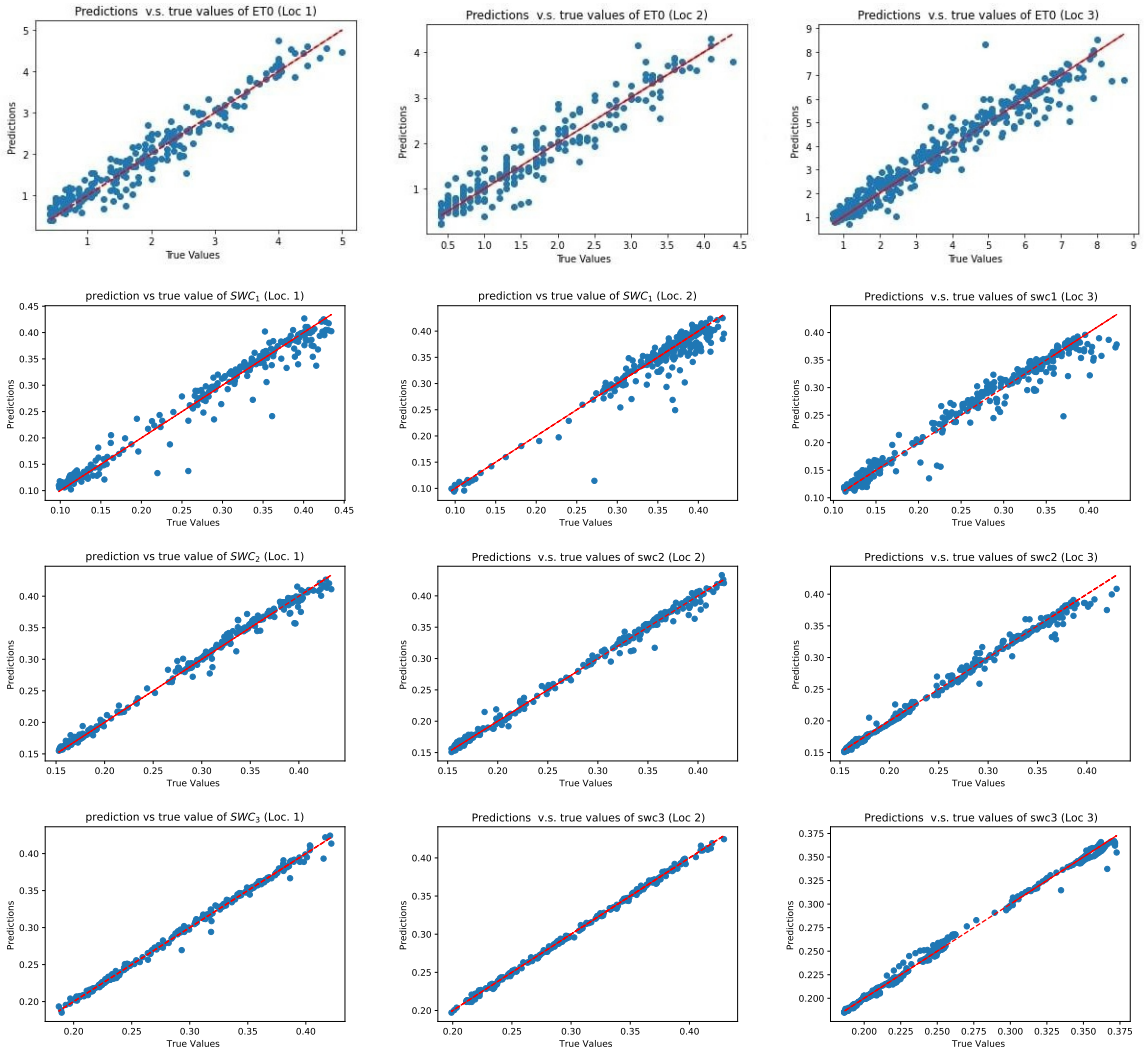


Figure 3.9: True values SWC (m^3m^{-3}) vs. Predicted values. The x-axis indicates true values, and the y-axis indicates prediction values by models.

added to the proposed models to investigate the performance of the CNN- LSTM model for predicting ETo and SWC. Table 3.7 shows the CNN architectures that were used to stack the LSTM models. First, the input sequence was divided into two subsequences, each with four-time steps. The CNN can interpret each subsequence with four-time steps, and the LSTM used a time series of the interpretations from the subsequences as input. The time-distributed layer wrapped the CNN model and allowed each layer to be applied to each subsequence [164]. The CNN model contained two convolutional layers with kernel sizes of 3 and 2, respectively, followed by an average pooling layer with a pooling size of 2. The number of filters remained the same as the number of nodes in the proposed BLSTM model, i.e., 512 filters in each convolutional layer.

In this paper, the performance of the models was improved by using the function tanh and the Average Pooling layer instead of the layer Relu or Max Pooling (see Table 3.8).

The performance and computational efficiency of the CNN-BLSTM models are shown in Table 3.8 and Table 3.9. The BLSTM model outperformed the CNN-BLSTM model in terms of model performance and computation time. As shown in Table 3.9, the number of trainable

Table 3.5: Evaluation of the LSTM model on the test set for each location. The model was trained on all datasets. SWC$i represents the soil water content at level i, and the unit for SWC is "$\mathrm{m}^3\mathrm{m}^{-3}$" (Volumetric Water Content).

Locations	Metrics	ETo	$SWCl_1$	$SWCl_2$	$SWCl_3$	In-general
Loc. 1	MSE	0.07	0.00074	0.00027	4.94×10^{-5}	0.0144
	RMSE	0.29	0.021	0.01	0.004	0.13
	MAE	0.115	0.02	0.013	0.0072	0.12
	R^2	0.94	0.9	0.94	0.98	0.98
	MBE	0.032	-0.0013	0.0016	0.003	0.0072
Loc. 2	MSE	0.093	0.0004	7.48×10^{-5}	1.86×10^{-5}	0.018
	RMSE	0.3	0.02	0.0086	0.0043	0.137
	MAE	0.23	0.012	0.005	0.002	0.05
	R^2	0.91	0.92	0.98	0.99	0.96
	MBE	- 0.024	0.0034	0.0006	0.0004	-0.004
Loc. 3	MSE	0.27	0.0002	3.8×10^{-5}	7.9×10^{-6}	0.056
	RMSE	0.51	0.016	0.006	0.002	0.23
	MAE	0.36	0.01	0.003	0.0016	0.69
	R^2	0.93	0.97	0.99	0.99	0.98
	MBE	-0.06	-0.0007	-0.0007	-0.00027	-0.013

parameters tripled in the CNN-BLSTM model compared to the BLSTM. Therefore, the model began to overfit due to the complexity of model. Figure 3.10 shows the RMSE and R^2- score during the training of the CNN-LSTM models.

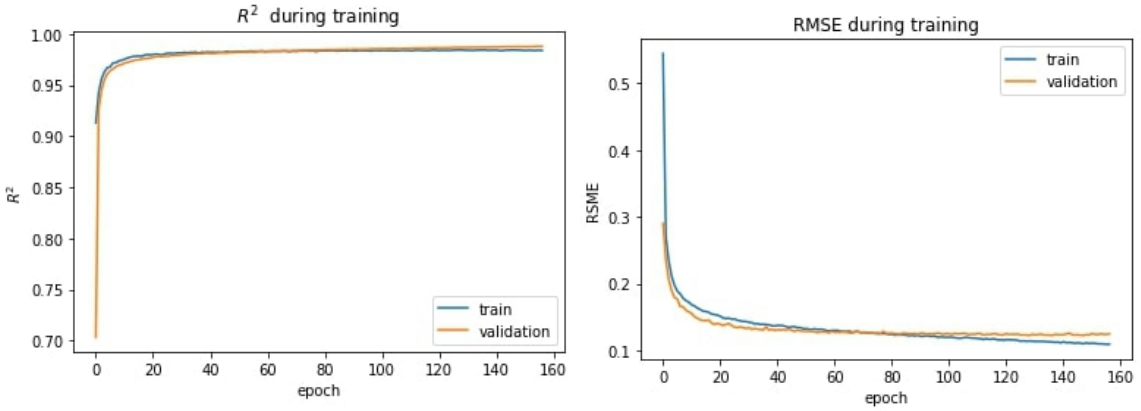


Figure 3.10: The RMSE and R^2-score of CNN-BLSTM model during training. The left side shows the R^2-score and the right side shows the RMSE.

The performance of the LSTM model was compared with the CNN model and two traditional machine learning models Support Vector Regression (SVR) and Random Forest (RF). The Convolutional Neural Network (CNN) architecture used for the comparison was the same as the CNN architecture built on the LSTM architecture, with the expectation that the time-distributed layers were removed and a dense layer was added in place of the LSTM layers to make the prediction (see Table 3.10).

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 checked, and if it is satisfied, the flow goes through one branch,

Table 3.6: Evaluation of the LSTM model on the test set for each site. The model was trained on two datasets and tested on the third dataset. SWC$i represents the soil water content at level i, and the unit for SWC is "m^3m^{-3}" (Volumetric Water Content).

Locations	Metrics	ETo	$SWCl_1$	$SWCl_2$	$SWCl_3$	In-general
Loc. 1	MSE	0.08	0.0004	9.4×10^{-5}	2.3×10^{-5}	0.016
	RMSE	0.28	0.021	0.009	0.004	0.129
	MAE	0.21	0.013	0.0061	0.0029	0.12
	R^2	0.94	0.9	0.94	0.98	0.98
	MBE	0.032	-0.0013	0.0016	0.003	0.0072
Loc. 2	MSE	0.1	0.00047	7.8×10^{-5}	1.89×10^{-5}	0.02
	RMSE	0.33	0.021	0.0088	0.0043	0.15
	MAE	0.25	0.014	0.006	0.0029	0.055
	R^2	0.89	0.91	0.98	0.99	0.95
	MBE	- 0.024	0.0034	0.0006	0.0004	-0.004
Loc. 3	MSE	0.35	0.0004	7.9×10^{-5}	1.2×10^{-5}	0.075
	RMSE	0.59	0.02	0.008	0.003	0.28
	MAE	0.46	0.014	0.005	0.002	0.9
	R^2	0.9	0.95	0.98	0.99	0.97
	MBE	-0.26	0.0012	0.0002	0.0005	-0.004

Table 3.7: CNN used to stake to the LSTM model.

Layer (type)	Output Shape
Input	(Batch Size, 2, 4, 11)
TimeDistributed(Conv1D) (kernel-size=3, padding=same)	(Batch Size, 2, 4, N. Filters)
Dropout 1	(Batch Size, 2, 4, N. Filters)
TimeDistributed(Conv1D) (kernel-size=2, padding=same)	(Batch Size, 2, 4, N. Filters)
Dropout 1	(Batch Size, 2, 4, N. Filters)
TimeDistributed(Averagepooling) (pool-size=2)	(Batch Size, 2, 2, N. Filters)
TimeDistributed(Flatten)	(Batch Size, 2, $2 \times$ N. Filters)

Table 3.8: Performance of the CNN-LSTM model on the test set.

Models	Locations	MSE	RMSE	MAE	R^2	MBE
CNN-LSTM (AveragePooling+tanh)	Loc. 1	0.161	0.127	0.047	0.974	-0.006
	Loc. 2	0.019	0.138	0.051	0.96	-0.003
	Loc. 3	0.063	0.25	0.083	0.976	-0.016
CNN-LSTM (MaxPooling, Relu)	Loc. 1	0.016	0.12	0.05	0.975	-0.014
	Loc. 2	0.021	0.14	0.055	0.95	-0.017
	Loc. 3	0.61	0.24	0.082	0.977	-0.012

otherwise through the other, always to the next node until the tree is finished [24].

Support Vector Regression (SVR) is a supervised learning algorithm used to predict discrete values. The basic idea behind SVR is to find the best fitting line. In SVR, the best fitting line is the hyperplane that has the maximum number of input points [23]. A kernel in the SVR model is a set of mathematical functions that take data as input and transform it into the desired shape. They are generally used to find a hyperplane in a higher-dimensional

Table 3.9: Computation efficiently of the LSTM and CNN-LSTM models.

Models	T(E/S)	N. Param.	N. E	N. P	Tr. S	val. S	Te. S	TT
CNN-LSTM	2	6,842,885	160	20	6197	1550	365	<1s
LSTM	1	2,151,429	250	20	6197	1550	365	<1s

Abbreviations represent the following: T: Computational time; E/S: Epoch per second; N.: Number; Param: Parameters; E: Epoch; P: number of patience was used in early stopping method; Tr. S: number of training samples; val. S: number of validation sample; Te. S number of test samples, TT: Computation time for one round of prediction.

Table 3.10: CNN architecture used to compare to the LSTM model.

Layer (type)	Output Shape	param
Input	(Batch size, 8, 11)	0
(Conv1D)(kernel-size=3)	(Batch size, 6, N. Filters)	17408
Dropout 1	(Batch size, 6, 512)	0
(Conv1D)(kernel-size=2)	(Batch size, 5, N. Filters)	524800
Dropout 1	(Batch size, 5, N. Filters)	0
(Averagepooling)(pool-size=2)	(Batch size, 2, N. Filters)	0
Flatten	(Batch size, $2 \times$ N. Filters)	0
Dense	(Batch size, N. Filters)	524800
Dense	(Batch size, 5)	2565

space. The most commonly used kernels include Linear, Nonlinear, Polynomial, Radial Base Function (RBF), and Sigmoid. Each of these kernels is used depending on the dataset. In this work, the RBF kernel has been used [166].

Table 3.11 shows the performance of each model on the test set. As shown, the LSTM models show better performance on the datasets. The CNN model has achieved the second-best performance.

Table 3.11: Performance of the traditional machine learning algorithms and CNN model on the dataset (MSE). The abbreviations are RF: Random Forest, SVR-RBF: Support Vector Regression with Radial Basis Function (RBF) kernel and CNN for Convolutional Neural Network

Locations	Metrics	SVR-RBF	RF	CNN
Loc. 1	MSE	0.2	0.085	0.033
	RMSE	0.45	0.29	0.18
	MAE	0.31	0.21	0.7
	R^2	0.81	0.92	0.94
	MBE	0.14	0.002	0.03
Loc. 2	MSE	0.17	0.11	0.027
	RMSE	0.42	0.34	0.16
	MAE	0.31	0.26	0.06
	R^2	0.83	0.88	0.94
	MBE	0.14	0.02	-0.03
Loc. 3	MSE	0.33	0.32	0.2
	RMSE	0.58	0.56	0.45
	MAE	0.39	0.39	0.16
	R^2	0.92	0.92	0.93
	MBE	-0.023	-0.02	-0.016

One of the applications of this model is a decision system that decides when and how much

to irrigate in order to avoid water waste without compromising productivity. In future work, a reinforcement learning agent can be trained to select the best amount of irrigation. In deep reinforcement learning algorithms, there is an environment that interacts with an agent. During training, the agent chooses an action based on the current state of the environment, and the environment returns the reward and the next state based on the previous state to the agent. The agent tries to choose the action that maximizes the reward [167]. In the agricultural domain, the state of the environment can be defined as the climate data and the water content of the soil and ETo; the action is the amount of irrigation, and the reward is the net yield. An agent can be trained to choose the irrigation amount based on the state of the field, and the SWC and ET prediction model can be used as part of the environment of Deep Reinforcement Learning to calculate the next state of the environment. By using the trained model in this paper, the training of the agent can be end-to-end and does not require manual processing.

In the end, the importance of the loss function to train the LSTM models was investigated for Loc. 3. The model was trained using MSE, RMSE, MAE, and Huber loss function. The Huber loss function (H_δ) is the composition of the MAE and MSE and can be calculated by Equation (3.15):

$$\mathrm{H}_\delta = \begin{cases} \frac{1}{2}(\mathrm{y}-\hat{\mathrm{y}})^2 & \text{if } |\mathrm{y}-\hat{\mathrm{y}}| \leq \delta \\ \frac{1}{2}\delta^2 + \delta|\mathrm{y}-\hat{\mathrm{y}}| & \text{otherwise} \end{cases} \tag{3.15}$$

where δ is a hyperparameter that should be set. The Huber function is basically MAE, which becomes MSE when the error is small. Unlike the MAE, the Huber function is differentiable at 0. Table 3.12 shows the performance of the model on the validation set using Huber, MAE, RMSE, and MSE as loss function for Loc. 3. The model with MSE as a loss function outperformed the others.

Table 3.12: Performance of the models using different loss functions for Loc. 3. The table showcases the corresponding values for each loss function, highlighting their respective performance metrics.

	Metrics			
Loss Functions	**MSE**	**RMSE**	**MAE**	R^2
MSE	0.056	0.23	0.69	0.98
RMSE	0.067	0.25	0.8	0.975
MAE	0.07	0.07	0.27	0.97
$\mathrm{H}_{\delta=1}$	0.063	0.25	0.79	0.976
$\mathrm{H}_{\delta=0.1}$	0.068	0.26	0.77	0.974
$\mathrm{H}_{\delta=0.01}$	0.074	0.27	0.8	0.972
$\mathrm{H}_{\delta=0.001}$	0.062	0.27	0.07	0.977
$\mathrm{H}_{\delta=0.0001}$	0.09	0.3	0.1	0.965

3.4 Conclusions

The advantage of the recurrent neural network is its ability to process time-series data like agricultural datasets. In this paper, the ability of recurrent neural networks, including LSTM

and its extension BLSTM, to model ETo and SWC was investigated. A drawback of deep learning methods is that they require a large amount of data to train. To overcome this problem, the simulated dataset was used for ET and SWC. The BLSTM model outperformed the LSTM model on the validation set, and hence a single layer BLSTM model with 512 nodes was trained. The proposed model achieved MSE in the range of 0.014 to 0.056, and the results show that the BLSTM model has good potential for modeling ETo and SWC. The CNN models are able to extract features from the data. A CNN model was added to the BLSTM model to extract features, and then BLSTM used these features to predict ET and SWC. When a CNN model was added to the LSTM model, and the number of parameters in the CNN-BLSTM was increased by three times, the model began to overfit after 100 epochs, and BLSTM showed better results than the CNN-BLSTM. Therefore, increasing the number of trainable parameters in Deep Learning algorithms may cause the model to overfit.
The performance of BLSTM was also compared with Random Forest, SVR, and CNN. The best performance was obtained with BLSTM. The second-best performance was obtained with the CNN model. Among the machine learning methods, RF outperformed the SVR model. One disadvantage of BLSTM compared to CNN, SVR and RF was that the training time of BLSTM was higher than the other models, but when trained, it was more accurate, and the computation time for prediction was almost the same as the other models.
Finally, the importance of choosing a loss function was investigated. The model with MSE as the loss function performed better than the other models with an MSE of 0.056, and the model with $\mathrm{H}_{\delta=0.0001}$ as the loss function had the worst performance. These results show that the choice of a loss function depends on the problem and must be chosen carefully.
In future work, the variables such as irrigation amount and groundwater level will be added to the input of the model to make the prediction more accurate. As mentioned earlier, one of the applications of the SWC and ET 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. Deep reinforcement learning algorithms require an environment that interacts with the agent and tells the agent the next state and reward. The SWC and ET prediction model is used as part of the algorithms' environment and determines the next state, which is the SWC and ET a day per head.

3.5 Anexos

3.5.1 Dataset Detail

Table 3.13: Dataset details.

Variables	Unit	Data Source	Locations	Temporal Resolution	Max	Min	Mean	SD
T_{Min}	°C	AppiZezere, DRAP-Centro	Loc. 1	daily	26.76	-5.26	9.14	5.86
			Loc. 2	daily	21.89	-7.43	7.30	5.39
			Loc. 3	15-minutes	27	-4.7	9.76	5.63
T_{Avg}	°C	AppiZezere, DRAP-Centro	Loc. 1	daily	35.94	1.96	16.13	7
			Loc. 2	daily	32.2	0	14.66	6.65
			Loc. 3	15-minutes	34.84	-0.12	15.68	6.90
T_{Max}	°C	AppiZezere, DRAP-Centro	Loc. 1	daily	44.8	5.56	23.97	8.68
			Loc. 2	daily	42.9	4.43	22.99	8.42
			Loc. 3	15-minutes	42.7	1.8	21.84	8.42
HR_{Min}	%	AppiZezere, DRAP-Centro	Loc. 1	daily	99.1	0	41.91	20.56
			Loc. 2	daily	99.1	0	42.78	20.17
			Loc. 3	15-minutes	95	0	38.72	20.20
HR_{Avg}	%	AppiZezere, DRAP-Centro	Loc. 1	daily	99.1	21.84	67.16	18.45
			Loc. 2	daily	99.1	0	70.96	16.34
			Loc. 3	15-minutes	95.89	27.75	60.38	18.78
HR_{Max}	%	AppiZezere, DRAP-Centro	Loc. 1	daily	100	33.33	89.55	13.87
			Loc. 2	daily	99.1	48.37	93.93	10.08
			Loc. 3	15-minutes	97	24	81.29	15.05
SR_{Avg}	Wm^{-2}	AppiZezere, DRAP-Centro	Loc. 1	daily	592.08	3.90	214.67	140.11
			Loc. 2	daily	592.08	3.91	215.57	140.87
			Loc. 3	15-minutes	346.66	6.35	172.02	89.25
WS_{Avg}	ms^{-1}	AppiZezere, DRAP-Centro	Loc. 1	daily	7.28	0.014	1.32	0.81
			Loc. 2	daily	5.91	0	1.06	0.82
			Loc. 3	15-minutes	28.85	0.031	4.62	3.80
WS_{Max}	ms^{-1}	AppiZezere, DRAP-Centro	Loc. 1	daily	74.04	1.06	4.82	2.16
			Loc. 2	daily	14.65	0	4.87	1.92
			Loc. 3	15-minutes	86.5	3.5	24.67	10.61
Prec	$mm\ d^{-1}$	AppiZezere, DRAP-Centro	Loc. 1	daily	101.59	0	2	6.84
			Loc. 2	daily	112.80	0	2.92	8.95
			Loc. 3	15-minutes	101.6	0	2.28	7.20
ETo	$mm\ d^{-1}$	Penman-Monteith equation	Loc. 1	daily	8.5	0.3	3.260	2.09
			Loc. 2	daily	7.4	0.3	2.83	1.964
			Loc. 3	daily	9.8	0.2	3.68	2.088
$SWCl_1$	m^3m^{-3}	ECMWF	Loc. 1	daily at 23:00	0.43	0.097	0.26	0.11
			Loc. 2	daily at 23:00	0.43	0.07	0.27	0.11
			Loc. 3	daily at 23:00	0.43	0.11	0.26	0.10
$SWCl_2$	m^3m^{-3}	ECMWF	Loc. 1	daily at 23:00	0.43	0.15	0.28	0.086
			Loc. 2	daily at 23:00	0.43	0.15	0.29	0.08
			Loc. 3	daily at 23:00	0.43	0.15	0.27	0.086
$SWCl_3$	m^3m^{-3}	ECMWF	Loc. 1	daily at 23:00	0.43	0.18	0.29	0.06
			Loc. 2	daily at 23:00	0.42	0.2	0.30	0.061
			Loc. 3	daily at 23:00	0.43	0.17	0.28	0.064

Abbreviations represent the following: Max: maximum, Min: minimum, SD: standard deviation, Avg: Average, T: Temperature, HR: relative Humidity, SR: Solar Radiation, WS: Wind Speed, Prec: Precipitation, $SWCL_i$: Volumetric Soil Water Level i ($i \in \{1, 2, 3, 4\}$).

Table 3.14: Collinear coefficient between parameters in dataset (Loc. 1 and Loc. 2).

Variables	T_{Min} (°C)	T_{Avg} (°C)	T_{Max} (°C)	HR_{Min} (%)	HR_{Avg} (%)	HR_{Max} (%)	SR_{Avg} Wm^{-2}	WS_{Avg} ms^{-1}	WS_{Max} ms^{-1}	Prec $mm\ d^{-1}$	ETo $mm\ d^{-1}$	$SWCl_1$ m^3m^{-3}	$SWCl_2$ m^3m^{-3}	$SWCl_3$ m^3m^{-3}
Loc. 1														
T_{Min}	1	0.93	0.82	-0.44	-0.56	-0.56	0.24	0.01	-0.02	-0.04	0.77	-0.66	-0.64	-0.56
T_{Avg}	0.93	1	0.97	-0.66	-0.72	-0.58	0.37	-0.11	-0.09	-0.18	0.82	-0.82	-0.78	-0.63
T_{Max}	0.82	0.97	1	-0.78	-0.77	-0.56	0.40	-0.20	-0.15	-0.27	0.89	-0.86	-0.81	-0.64
HR_{Min}	-0.44	-0.66	-0.78	1	0.91	0.62	-0.45	-0.01	0.03	0.46	-0.75	0.79	0.73	0.50
HR_{Avg}	-0.56	-0.72	-0.77	0.91	1	0.84	-0.41	-0.15	-0.04	-79	-0.79	0.81	0.73	0.51
HR_{Max}	-0.56	-0.58	-0.56	0.62	0.84	1	-0.22	-0.28	-0.08	0.21	-0.62	0.61	0.54	0.38
SR_{Avg}	0.24	0.37	0.40	-0.45	-0.41	-0.22	1	0.02	0.10	-0.20	0.027	-0.39	-0.34	-0.17
WS_{Avg}	0.01	-0.11	-0.20	-0.01	-0.15	-0.28	0.02	1	0.57	0.10	0.069	0.09	0.13	0.18
WS_{Max}	-0.02	-0.09	-0.15	0.03	-0.04	-0.08	0.10	0.57	1	0.14	-0.007	0.10	0.10	0.12
Prec	-0.04	-0.18	-0.27	0.46	0.37	0.21	-0.20	0.10	0.14	1	-0.24	0.31	0.27	0.13
ETo	0.77	0.89	0.89	-0.75	-0.79	-0.62	0.027	0.069	-0.007	-0.24	1	-0.85	-0.79	-0.57
$SWCl_1$	-0.66	-0.82	-0.86	0.79	0.81	0.61	-0.39	0.09	0.10	0.31	-0.85	1	0.95	0.72
$SWCl_2$	-0.64	-0.78	-0.81	0.73	0.73	0.54	-0.34	0.13	0.10	0.27	-0.79	0.95	1	0.83
$SWCl_3$	-0.56	-0.63	-0.64	0.50	0.51	0.38	-0.17	0.18	0.12	0.13	-0.57	0.72	0.83	1
Loc. 2														
T_{Min}	1	0.90	0.73	-0.30	-0.44	-0.39	0.23	0.04	0.05	0.03	0.49	-0.60	-0.60	-0.54
T_{Avg}	0.90	1	0.94	-0.60	-0.66	-0.43	0.37	-0.08	-0.04	-0.17	0.60	-0.81	-0.77	-0.63
T_{Max}	0.73	0.94	1	-0.76	-0.72	-0.40	0.39	-0.20	-0.16	-0.30	0.58	-0.86	-0.81	-0.63
HR_{Min}	-0.30	-0.60	-0.76	1	0.88	0.49	-0.42	-0.04	-0.02	0.50	-0.47	0.75	0.68	0.45
HR_{Avg}	-0.44	-0.66	-0.72	0.88	1	0.75	-0.43	-0.23	-0.15	0.40	-0.44	0.78	0.70	0.47
HR_{Max}	-0.39	-0.43	-0.40	0.49	0.75	1	-0.17	-0.34	-0.15	0.19	-0.18	0.51	0.45	0.32
SR_{Avg}	0.23	0.37	0.39	-0.42	-0.43	-0.17	1	-0.0006	0.13	-0.19	0.45	-0.35	-0.30	-0.12
WS_{Avg}	0.04	-0.08	-0.20	-0.04	-0.23	-0.34	-0.001	1	0.78	0.10	-0.001	0.06	0.10	0.13
WS_{Max}	0.05	-0.04	-0.16	-0.02	-0.15	-0.15	0.13	0.78	1	0.19	-0.01	0.09	0.11	0.13
Prec	0.03	-0.17	-0.30	0.50	0.40	0.19	-0.19	0.10	0.19	1	-0.16	0.34	0.30	0.16
ETo	0.49	0.60	0.58	-0.47	-0.44	-0.18	0.45	-0.0006	-0.01	-0.16	1	-0.46	-0.38	-0.17
$SWCl_1$	-0.60	-0.81	-0.86	0.75	0.78	0.51	-0.35	0.06	0.09	0.34	-0.46	1	0.95	0.74
$SWCl_2$	-0.60	-0.77	-0.81	0.68	0.70	0.45	-0.30	0.10	0.11	0.30	-0.38	0.95	1	0.85
$SWCl_3$	-0.54	-0.63	-0.63	0.45	0.47	0.32	-0.12	0.13	0.13	0.16	-0.17	0.74	0.85	1

Table 3.15: Collinear coefficient between parameters in dataset (Loc. 3).

Variables	Loc. 3													
	T_{Min} (°C)	**T_{Avg}** (°C)	**T_{Max}** (°C)	**HR_{Min}** (%)	**HR_{Avg}** (%)	**HR_{Max}** (%)	**SR_{Avg}** Wm^{-2}	**WS_{Avg}** ms^{-1}	**WS_{Max}** ms^{-1}	**Prec** $mm\ d^{-1}$	**ETo** $mm\ d^{-1}$	**$SWCl_1$** m^3m^{-3}	**$SWCl_2$** m^3m^{-3}	**$SWCl_3$** m^3m^{-3}
T_{Min}	1	0.94	0.86	-0.49	-0.55	-0.52	-0.07	0.57	-0.09	-0.08	0.73	-0.66	-0.64	-0.50
T_{Avg}	0.94	1	0.98	-0.68	-0.70	-0.56	-0.21	0.74	-0.18	-0.14	0.833	-0.79	-0.74	-0.55
T_{Max}	0.86	0.98	1	-0.77	-0.75	-0.56	-0.28	0.79	-0.22	-0.18	0.84	-0.82	-0.77	-0.55
HR_{Min}	-0.49	-0.68	-0.77	1	0.93	0.68	0.49	-0.81	-0.05	-0.04	-0.77	0.77	0.68	0.42
HR_{Avg}	-0.55	-0.70	-0.75	0.93	1	0.87	0.44	-0.76	-0.17	-0.12	-0.80	0.77	0.66	0.40
HR_{Max}	-0.52	-0.56	-0.56	0.68	0.87	1	0.28	-0.51	-0.28	-0.17	-0.67	0.59	70.49	0.31
SR_{Avg}	0.57	0.74	0.79	-0.81	-0.76	-0.51	-0.40	1	0.02	0.03	0.86	-0.70	-0.60	-0.30
WS_{Avg}	-0.09	-0.18	-0.22	-0.05	-0.17	-0.28	0.05	0.02	1	0.85	0.121	0.11	0.13	0.18
WS_{Max}	-0.08	-0.14	-0.18	-0.04	-0.12	-0.17	0.12	0.03	0.85	1	0.118	0.08	0.09	0.12
Prec	-0.07	-0.21	-0.28	0.49	0.44	0.28	1	-0.40	0.05	0.12	-0.284	0.35	0.25	0.09
ETo	0.73	0.833	0.84	-0.77	-0.80	-0.67	0.86	0.121	0.118	-0.284	1	-0.79	-0.70	-0.40
$SWCl_1$	-0.66	-0.79	-0.82	0.77	0.77	0.59	0.35	-0.70	0.11	0.08	-0.79	1	0.95	0.68
$SWCl_2$	-0.64	-0.74	-0.77	0.68	0.66	0.49	0.25	-0.60	0.13	0.09	-0.70	0.95	1	0.79
$SWCl_3$	-0.50	-0.55	-0.55	0.42	0.40	0.31	0.09	-0.30	0.18	0.12	-0.40	0.68	0.79	1
$SWCl_4$	0.03	0.04	0.03	-0.05	-0.05	-0.03	-0.02	0.23	0.10	0.06	0.154	0.06	0.15	0.431

Chapter 4

Crop yield estimation using deep learning based on Climate big data and irrigation scheduling

Abstract [1], K. Alibabaei, P. D. Gaspar, and T. M. Lima, "Crop yield estimation using deep learning based on climate big data and irrigation scheduling," Energies, vol. 14, no. 11, 2021. https://doi.org/10.3390/en14113004

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 extension 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-term short memory outperformed the Gated Recurrent Units network, the long-term short 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 $(\mathrm{kg/ha})^2$. The performance of the Bidirectional Long-Short Term Memory in the test was compared with the most commonly used deep learning method called 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 a R^2-score between 0.97 and 0.99. The results show that analyzing agricultural data with the Longe-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.

[1] This work was published in [27]

Keywords

Agriculture; Deep learning; LSTM; support decision-making algorithms; yield estimation; irrigation management.

4.1 Introduction

Agriculture is in a state of flux, and obstacles are emerging, such as climate change, environmental impact, 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 [5]. This scenario reinforces the constant effort 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 [5]. 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 this 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 the sustainable use of natural resources [5].

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 [12]. Deep here refers to the number of hidden layers of the neural network. The performance of the model improves as the network becomes deeper [12].

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 [168], and therefore, optimization of irrigation and more efficient irrigation systems are critical. Predicting yield based on different irrigations is one way to optimize.

Machine learning algorithms are already used to estimate yield from images. Bargoti and Underwood [101] 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. [169] developed a Region-based CNN (R-CNN) model with AlexNet as the backbone and detected ripe and unripe strawberries in the greenhouse images. The model achieved an average precision of 82.61%. Kang and Chen [170] imple-

mented 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 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. [98] 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. [100] 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. [171] 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 ($\mathrm{R}^2 \geq 0.90$). Tian et al. [102] 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. [172] used a Faster R-CNN model and a Long Short-Term Memory (LSTM) 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. Yang et al. [76] used AlexNet to estimate rice grain yield at the ripping stage from images taken in the field. Two models were trained, one with RGB and multispectral images and another with RGB images only. The first model achieved R^2 ranging from 0.464 to 0.511, and the second model R^2-score ranging from 0.424 to 0.499. Maimaitijiang et al. [173] 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. [174] 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 VIs-based regression model in estimating rice grain yield at the maturity stage. Chen et al. [175] 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. [103] 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 rate 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 in the images [74].

Ma et al. [65] used climate, remote sensing data, and rice information to estimate rice yield. Stacked Sparse Auto-Encoder (SSAE) was trained and achieved a percent mean square error of 33.09 kg$(10\mathrm{a})^{-1}$ (kilograms per ten acres). Han et al. [176] implicated machine learning methods including Support Vector Machine (SVM), Gaussian Process Regression (GPR), Neural Network (NN), K- Nearest Neighbor Regression, Decision Tree (DT), 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 a $\mathrm{R}^2 > 0.75$ were the three best yield prediction methods, among others. They also found that different agricultural zones and temporal training settings affected prediction accuracy. Kim et al. [177] 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. [178] 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. In irrigated crops 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 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 extension 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 [2]. The performance of the model is 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, 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 collected images from the field.
- Irrigation amount was used as input of 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.

4.2 Materials and Methods

4.2.1 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 [12].
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

[2]The codes are available at the following links:
https://github.com/falibabaei/tomato-yieldestimation/blob/main/main
https://github.com/falibabaei/potato-yield-estimation/tree/main

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 [19]. 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 [17] won the LSVRC competition for classification. Sermanet et al. [18] 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 decide the future.

Unlike CNN, information in recurrent neural networks goes through a loop that allows the network to remember the last previous outputs [54]. 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 4.1).

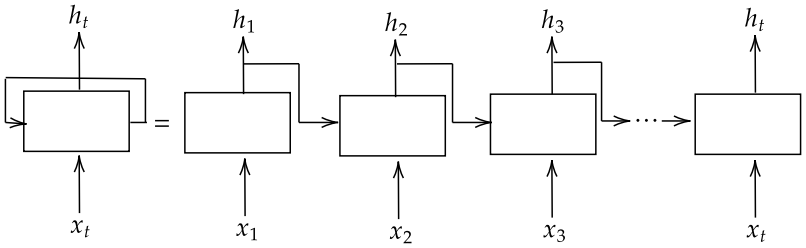


Figure 4.1: RNN looped and unfolded sequentially

The output of an RNN unit is calculated by Equation (4.2.1).

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

where $\mathrm{h_{t-1}}$ is the recurrent output from the previous step, $\mathrm{x_t}$ is the input at the current time, and $\mathrm{W_x}, \mathrm{W_h}, \mathrm{b_t}$ 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 [54].

4.2.1.1 LSTM and GRU Structures

The LSTM [55] and Gated Recurrent Units (GRU) [179] were developed to address RNN problem.

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 $\mathrm{z_t}$, an input gate $\mathrm{i_t}$, a forget gate $\mathrm{f_t}$, an output gate $\mathrm{o_t}$, and a memory cell $\mathrm{c_t}$ (see Figure 4.2). The forget gate $\mathrm{f_t}$ is used to remove information that is no longer useful in the cell state using

Equation (4.1). 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 (4.2). 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 (4.3), which outputs from -1 to +1. The vector values and the controlled values are multiplied to obtain useful information using Equation (4.4). The output gate o_t decides which information in the cell is used to calculate the output of the LSTM unit using Equation (4.5).

$$f_t = \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f), \quad (4.1)$$
$$i_t = \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i), \quad (4.2)$$
$$z_t = \tanh(W_{zx}x_t + W_{zh}h_{t-1} + b_z), \quad (4.3)$$
$$c_t = f_t * c_{t-1} + i_t * z_t. \quad (4.4)$$
$$o_t = \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \quad (4.5)$$

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 (4.6) to be sent as output and input to the next cell.

$$h_t = o_t * \tanh(c_t). \quad (4.6)$$

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 (4.7) to (4.10):

$$z_t = \sigma(W_{ux}x_t + W_{uh}h_{t-1} + b_u), \quad (4.7)$$
$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r), \quad (4.8)$$
$$o_t = \tanh(W_{ox}x_t + W_{oh}(r_t * h_{t-1}) + b_o). \quad (4.9)$$
$$h_t = (1 - z_t) * o_t + z_t * h_{t-1}. \quad (4.10)$$

The functions tanh and σ add nonlinearity to the network. These functions allow the model to capture the nonlinear relationships between inputs and outputs of the model. At the beginning of training, the weights and biases in Equations (2.3) to (2.5), (2.7) and (4.7) to (4.9) are set randomly. During training, the model tries to set the weights and biases in such a

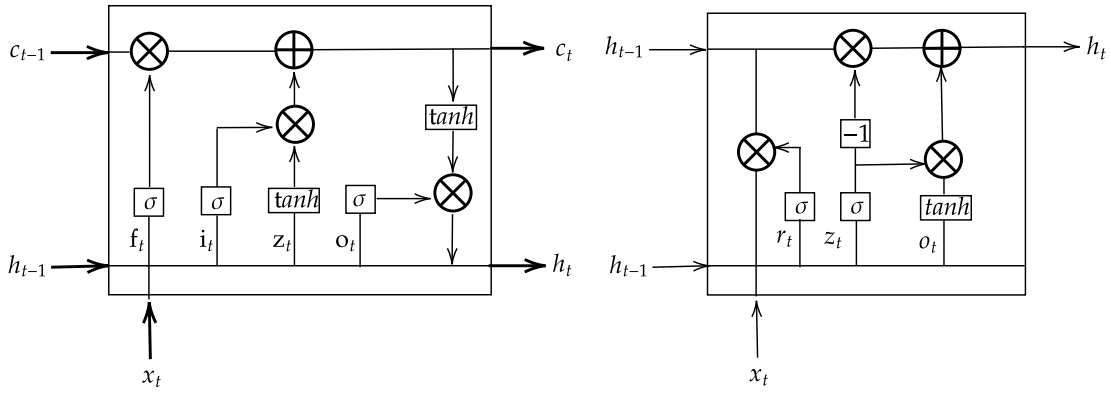


Figure 4.2: LSTM and GRU structures. Left side shows LSTM cell, right side shows GRU cell.

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 [180]. 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 implied to see which model performs better on the yield estimation problem.

4.2.1.2 Bidirectional LSTM Structure

Bidirectional LSTM (BLSTM) is an extension of the LSTM model that can improve the results [156]. 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 context information from both sides of the sequence (see Figure 4.3).

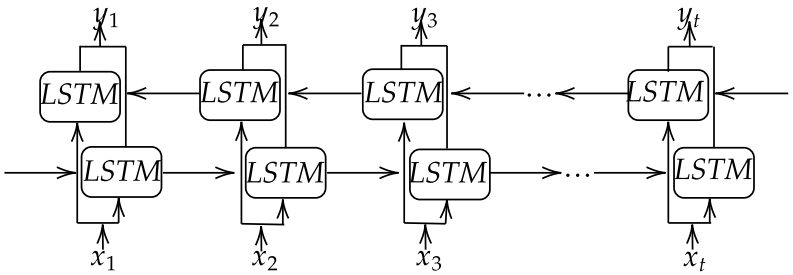


Figure 4.3: BLSTM layer.

4.2.1.3 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 [158]. 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 [158]. The dropout size is a hyperparameter that should be set during training (see Figure 4.4). Table 4.1 shows the effect of the dropout on the validation loss.

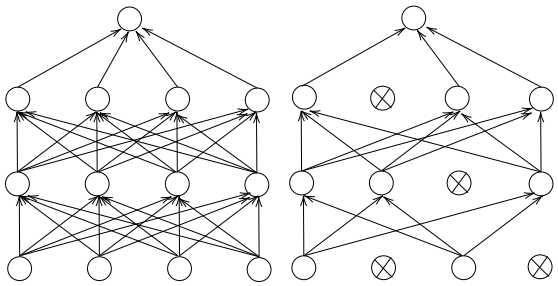


Figure 4.4: left side shows the normal neural network and the right side shows the network with dropout.

Table 4.1: validation lost under different dropout size.

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 [181]. 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.

4.2.2 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). 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 4.5 shows the Fadagosa region from Google Earth and Table 4.2 shows location details.

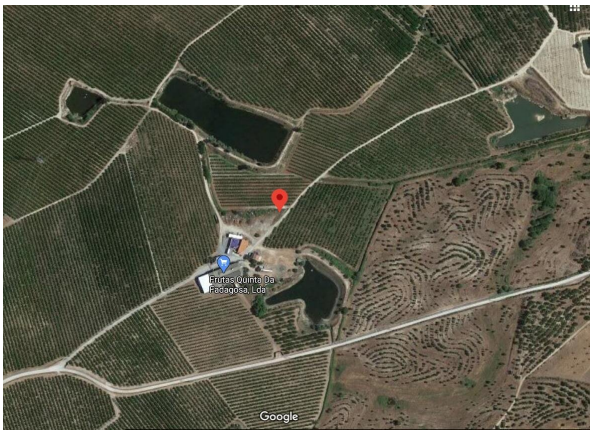


Figure 4.5: Map of Fadagosa region.

Table 4.2: Details of the location.

Location	Longitude	Latitude	Start date	End date	Temporal resolution
Fadagosa	40°1′46.55″N	7°26′36.27″W	2010-01-01	2020-03-23	15-Minutes

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 minutes and converted to the daily variable in the data preprocessing section. Table 4.3 shows the details of these variables. Figure 4.6 shows the daily variables from 2010 to 2019 (10 years of data).

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., 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 [182]. 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 [182]. 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

Table 4.3: 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
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.

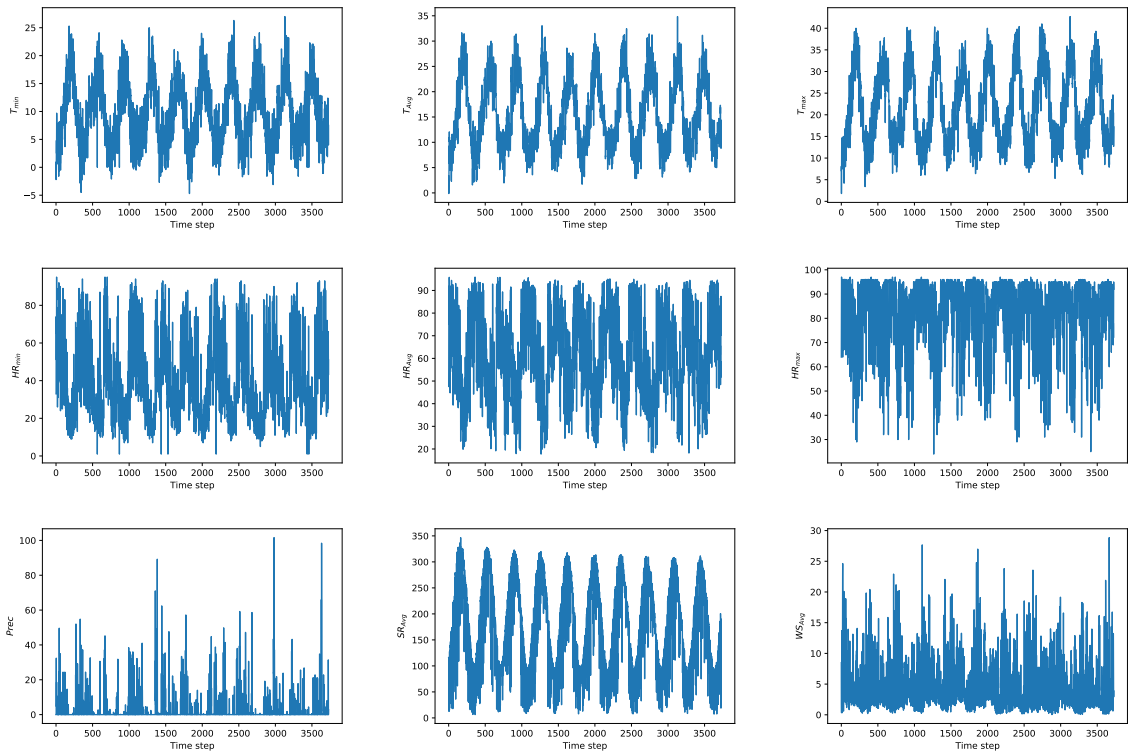


Figure 4.6: Fadagosa dataset. The unit for each variable is the same as shown in Table 4.3

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 4.7 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.

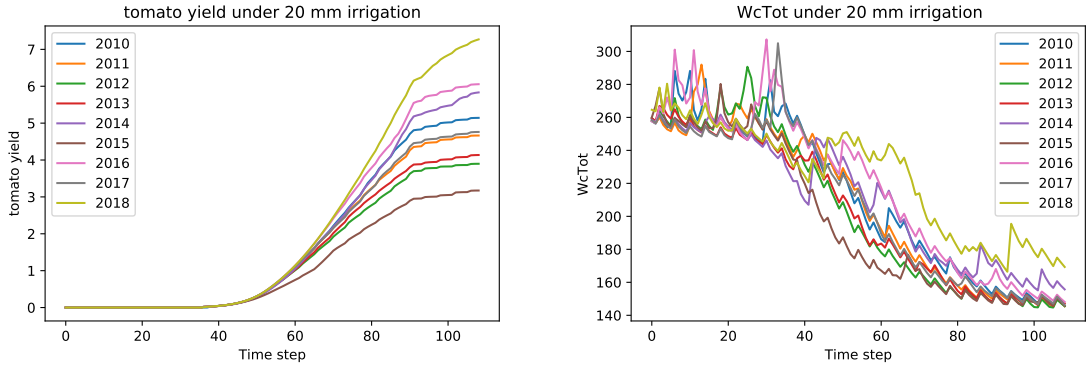


Figure 4.7: Evolution of tomato yield ($\mathrm{kg/ha}$) and total water content profile (mm) under fixed irrigation depth of 20 mm. The left side shows tomato yield and the right side shows WCTot.

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 (4.2.2) [137].

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

where $\mathrm{R_n}$ is the net radiation, G is the soil heat flux, $(\mathrm{e_s} - \mathrm{e_a})$ represents the vapor pressure deficit of the air, ρ is the mean air density at constant pressure, $\mathrm{c_p}$ is the specific heat of the air, Δ represents the slope of the saturation vapour pressure-temperature relationship, γ is the psychrometric constant, and $\mathrm{r_s}$ and $\mathrm{r_a}$ are the (bulk) surface and aerodynamic resistances. Figure 4.8 shows the ETo from 2010 to 2019.

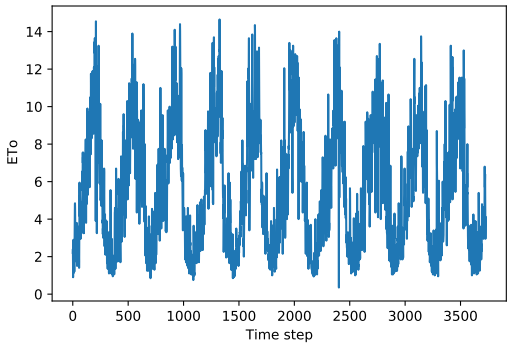


Figure 4.8: ETo ($\mathrm{mm\ d^{-1}}$) calculated by Aquacrop.

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.

4.2.3 Data pre-processing

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 [153].

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 [183]. 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 ($\mathrm{T_{Avg}}$), average humidity ($\mathrm{HR_{Avg}}$), average wind speed ($\mathrm{WS_{Avg}}$), 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 (4.11).

$$\mathrm{x_{new}} = \frac{\mathrm{x_{old}} - \mathrm{x_{min}}}{\mathrm{x_{max}} - \mathrm{x_{min}}} \quad (4.11)$$

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

4.2.4 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 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 [162]. The higher the R^2-score, the smaller the differences between the observed data and the fitted values [124]. MSE and R^2-score can be calculated using Equations (4.12) and (4.13):

$$\mathrm{MSE} = \frac{1}{\mathrm{n}} \sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{y_i} - \hat{\mathrm{y}}_\mathrm{i})^2, \quad (4.12)$$

$$\mathrm{R}^2 = 1 - \frac{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{y_i} - \hat{\mathrm{y}}_\mathrm{i})^2}{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{y_i} - \bar{\mathrm{y}}_\mathrm{i})^2}, \quad (4.13)$$

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

4.3 Results and Discussions

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 [163, 164]. 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 ($\mathrm{T_{Avg}}$), average humidity ($\mathrm{HR_{Avg}}$), average wind speed ($\mathrm{WS_{Avg}}$), reference evapotranspiration (ETo), total soil profile (WCTot), irrigation amount, and precipitation (Prec) and estimated 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 is used to compare different models, select hyperparameters, and prevent overfitting during the training phase. 30% of the data was selected as a test set, and the rest was 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 [79]. 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 [12].

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 [54]. 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 [54]. 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 [22]. Changing the number of nodes per layer has been shown to have a more significant impact on results than changing the number of layers [22]. Since choosing LSTM and GRU with four layers and BLTM with three layers significantly increases the number of trainable parameters, leading to overfitting 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 [184]. 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 [54]. 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 4.4. As can be seen in Table 4.4, 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.

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. Table 4.5 and Table 4.6 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 acti-

Table 4.4: Validation MSE for different number of LSTM layers and LSTM nodes per layer. The best performance is shown in red.

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

vation function after each BLSTM layer to capture the nonlinear relationship between input and output [54].

Table 4.5: BLSTM architecture used to predict tomato yield.

Layer (type)	Output Shape	param
Input	((Batch Size, 110, 7)	0
BLSTM	(Batch Size, 110, 1024)	2134016
Dropout	((Batch Size, 110, 1024)	0
BLSTM	(Batch Size, 1024)	6295552
Dropout	(Batch Size, 1024)	0
Dense	(Batch Size, 512)	524800
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 was 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 4.7 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.

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 [158]. Adding dropout on the recurrent outputs with

Table 4.6: BLSTM architecture used to predict potato yield.

Layer (type)	Output Shape	param
Input	((Batch Size, 121, 7)	0
BLSTM	(Batch Size, 121, 256)	139264
Dropout	((Batch Size, 121, 256)	0
BLSTM	(Batch Size, 512)	394240
Dropout	(Batch Size, 512)	0
Dense	(Batch Size, 512)	131584
Dense	(Batch Size, 1)	513

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 4.1).

Table 4.7: Validation lost under different hyperparameters. The best performance is shown in red.

	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 [181]. 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 4.9, 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.

Since the test dataset was randomly selected, there were different seasons with different amounts of irrigation in each test dataset. Table 4.9 shows the performance of the models on the test dataset, and Figure 4.10 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

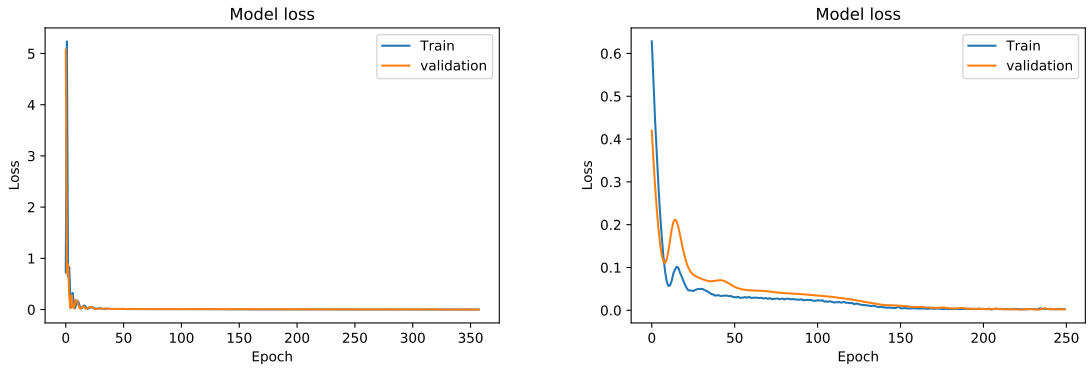


Figure 4.9: Model loss during training. The left side shows the loss of tomato yield prediction model and the right side shows potato yield prediction model.

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 4.3) and also, as mentioned earlier, the model used to estimate tomato yield was trained with a larger training set.

As shown in Figure 4.10, the tomato test dataset included the 2010 season under four different irrigation levels. The irrigation amounts were 0, 10, 20, and 60 mm, 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.

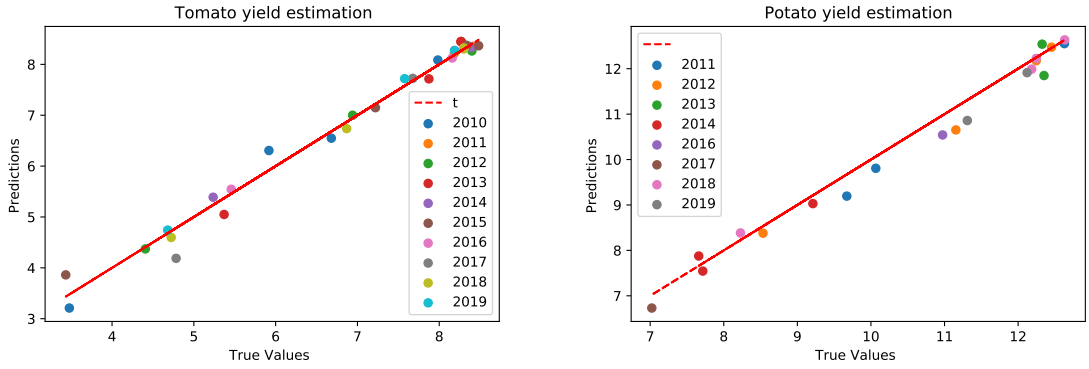


Figure 4.10: Predicted values of tomato and potato yield (kg/ha) vs. true values.

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 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. They are 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 [185]. 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 4.8 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 [54]. 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. 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 [54]. 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 implanted. The CNN model with 512 kernels and the number of layers 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 has 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 4.8 shows the architecture of the CNN model.

The CNN with two layers and three layers and a kernel size of 5 and 11 (tomato and potato, respectively) with 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 4.9 shows, the

Table 4.8: 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) $\times$ 512)
Dense	(Batch size, 512)
Dropout	(Batch size, 512)
Dense	(Batch size, 1)

RF model outperformed the MLP model in predicting yield, with an R^2- score of 0.87 to 0.90.

Table 4.9: 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	-

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. In the given equation (4.14), the net return in agriculture is calculated based on the yield and prices of the crop, as well as the total amount of water used. Note that in this work, the other inputs such as plants, fertilizers, and other resources are assumed to be fixed or constant, which means that their quantities or costs remain unchanged throughout the

calculation of the net return.

$$R = Y * P_y - W * P_w \quad (4.14)$$

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. The 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 [186], and tomato prices were assumed to be nearly 728.20 USD/ton (www.tridge.com).

Algorithm 1 Pseudocódigo para o semáforo

```
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 (4.14)
else
   done=False
   reward=0
for i in range steps to  do
   if k= steps then
      action=random_action(0,60)
   else

loop
   accionar verde no semáforo principal
   aguardar por sinal dos sensores de posição
   if carro no sensor then
      mudar para vermelho semáforo principal
until interruptor de manutenção activado
```

Table 4.10 and 4.11 show the net returns and the parameters used in Algorithm 1.

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

Table 4.10: The net return of random irrigation. The net return is measured in dollars per hectare (ha) and represents the financial outcome or profit obtained from agricultural production.

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

Table 4.11: The parameters used in Algorithm 1.

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 [167]. 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.

4.4 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 reduce energy consumption, increase productivity, 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 like 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 problem.

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 like 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 decide when and how much to irrigate and reduces water consumption without affecting productivity.

Chapter 5

Irrigation Optimization with a Deep Reinforcement Learning model - Case study on a site in Portugal

Abstract [1]

In the field of agriculture, the water used for irrigation should be given special treatment, as it is responsible for a large proportion of total water consumption. Irrigation scheduling is critical to food production because it guarantees producers a consistent harvest and minimizes the risk of losses due to water shortages. Therefore, the creation of an automatic irrigation method using new technologies is essential.
New methods such as deep learning algorithms have attracted a lot of attention in agriculture and are already being used successfully. In this work, a Deep Q-Network was trained for irrigation scheduling. The agent was trained to schedule irrigation for a tomato field in Portugal. Two Long Short Term Memory models were used as the agent environment. One predicts the total water in the soil profile on the next day. The other one was employed to estimate the yield based on the environmental condition during a season and then measure the net return. The agent uses this information to decide the following irrigation amount. An Artificial Neural Network, a Long Short Term Memory, and a Convolutional Neural Network were used to estimate the Q-table during training. Unlike the Long-Short Terms Memory model, the Artificial Neural Network and the Convolutional Neural Network could not estimate the Q-table, and the agent's reward decreased during training. The comparison of the performance of the model was done with fixed-base irrigation and threshold-based irrigation. The trained model increased productivity by 11% and decreased water consumption by 20% to 30% compared to the fixed method.

5.1 Introduction

Water is the most strategic commodity on the planet and can become scarce if not protected. On average, 70% of water is consumed in agriculture [187]. According to FAO, agriculture is also the sector where the greatest need for action to reduce water consumption [134]. Challenges such as increasing population and demand for agricultural communities, food preferences, degradation of soils and water-related ecosystems, pollution, and climate change bring uncertainties regarding the availability of water resources [32, 5, 188]. In the future,

[1] This work was published in [28], K. Alibabaei, P. D. Gaspar, E. Assunção, S. Alirezazadeh, and T. M. Lima, "Irrigation optimization with a deep reinforcement learning model: Case study on a site in portugal," Agricultural Water Management, vol. 263, p. 107480, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378377422000270

these challenges include high uncertainty about the impacts of climate change, rising costs of groundwater and pump irrigation, and the increasing viability of large storage systems [32]. Therefore, how to manage and reduce water consumption without compromising productivity is critical. Appropriate agricultural practices can be achieved if new technologies and innovations are used [5].

Smart farms have adopted modern technologies in agriculture to bring farmers closer to modern digital technologies that revolutionize decision-making and productivity in the field [5, 11]. It uses sensors and communication technology to enable remote monitoring and control of various parameters such as soil, livestock, and others, from a computer or smartphone and provides better control and monitoring [10]. The vast amount of data and information generated by agricultural technology and connectivity will be the basis for smart farms [5].

There is a need to analyze and store the huge amount of data collected by sensors per second and to consider and study the observations obtained about the variable attributes of the field. Artificial Intelligence algorithms, and in particular Deep Learning algorithms (DL), are a tool to extract features from these data and develop decision systems [10]. In a DL model, a huge amount of data (Big Data) is fed into an artificial neural network, valuable features are extracted from the data, and these features are used for future predictions [189].

DL has already successfully applied in various fields of agriculture [19]. Evapotranspiration and soil moisture are two important variables that should be considered in irrigation scheduling. [145] applied a convolutional neural network (CNN) to estimate soil moisture from images. The CNN model performed better than RF, SVM, and ANN. [146] used a Macroscopic Cellular Automata (MCA) combined with Deep Belief Network (DBN) to predict soil moisture. The proposed model reduced the mean square error by 18% compared to the multilayer perceptron. [147] used a generalized wavelet neural network (WNN) to estimate the reference evapotranspiration. The WNN and Artificial Neural Network (ANN) models outperformed wavelet regression and linear regression methods. In [123], Deep Learning-Multilayer Perceptron (MLP) was used to estimate daily evapotranspiration. The MLP outperformed the Gradient Boosting Machine, Random Forest, and Generalized Linear Models. In [148], three Convolutional Neural Networks (CNN) with unique structures were used to predict evapotranspiration. The CNN model performed better than the seasonal ARIMA (SARIMA) and seasonal naive model (SNAIVE) in terms of computational complexity, accuracy, and variance. In Deep Learning, recurrent neural networks are developed to model time series data. A Neural Network was implemented in [142] to predict soil moisture. The model achieved R^2 above 0.94 in all test areas. A Long-Short Terms Memory (LSTM) model was used in [124] to estimate agricultural groundwater depth. Data from 14- years were used, including precipitation, water runoff, temperature, reference evapotranspiration, and groundwater depth. The LSTM model performed better compared to Feed-Forward Neural Network (FFNN). [26] used a single layer bidirectional LSTM model to predict evapotranspiration and soil water content one day in advance. During the test period, an R^2 score between 0.96 and 0.98 was obtained. The CNN model was added to the LSTM to investigate possible performance improvement. Performance decreased in all data sets due to the complexity of the model.

However, farmers do not have the knowledge to work with soil water content or evapotranspiration and need an expert to analyze the data, which makes it impractical for small farms. In this work, we trained a deep Q(LSTM)-Networks network model that uses this knowledge and simply tells farmers when and how much to irrigate to achieve the best productivity without wasting water for a tomato field.

5.2 Materials and Methods

A computer system comprising an Intel Core i7-9700 CPU, 32.0 GB RAM, and an NVIDIA GEFORCE RTX 2080 graphics card was used for the work. The Tensorflow 2.1.0 framework and Keras library 2.2.4-tf were used to implement the models.
Figure 5.1 shows the general framework of this work, which consists of four parts. First, historical data are collected from various sources and prepared for use as input to the models (Figure 5.1. A). Then, two LSTM models are trained on the obtained historical data to predict SWTD for the next day and tomato yield at the end of a season, respectively (Figure 5.1. B). Training the LSTM models is a unique process and after training, they use as a feature in the DRL training environment, which takes the current state s (historical climate data) and action a (amount of irrigation), and then returns the next state s' and reward r (Figure 5.1. C). During the agent's training, it selects an action for the next irrigation based on the current state of the field and evaluates that action using a function called Q-value. The environment receives the current state and the action chosen by the agent and indicates the next state and the reward. This interaction between the DRL agent and training environment is repeated until the DRL agent converges to an optimal strategy for choosing the next day's irrigation amount (Figure 5.1. C and D). [2]
In the following section, each part of the framework is explained in detail.

5.2.1 Data Collection

In this paper, the climate dataset for Fadagosa (Portugal) of [27] was used, which comes 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`). The soil type in Fadagosa is sandy or sandy loam, permeable, with low to medium organic matter content, with low reaction from acid to neutral, rich in phosphorus and potassium, and without salinity. The climate in Fadagosa is a warm Mediterranean summer climate [27]. The location and details of the dataset were described in Tables 5.1 and 5.2. Figure 5.3 also shows the time series of the dataset.

Location	Longitude	Latitude	Start date	End date
Fadagosa	39.483629	-7.382767	2010-01-01	2020-03-23

Table 5.1: Details of locations in datasets.

[2] The codes can be found in the following link:
`https://github.com/falibabaei/DQN_irrigation-in-agriculture`

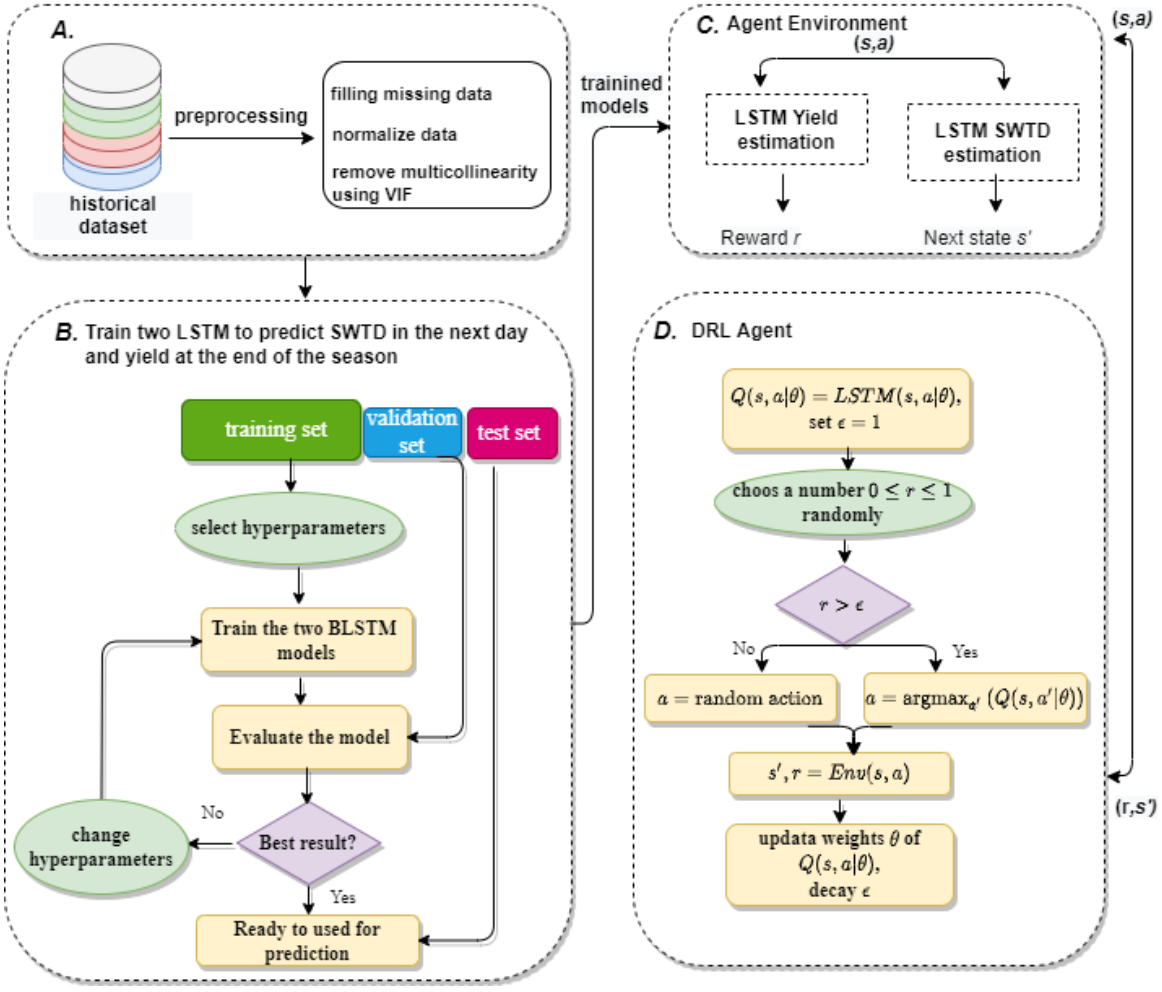


Figure 5.1: The general framework of this paper. The action chosen with agent is the volume of the next irrigation (mm/day).

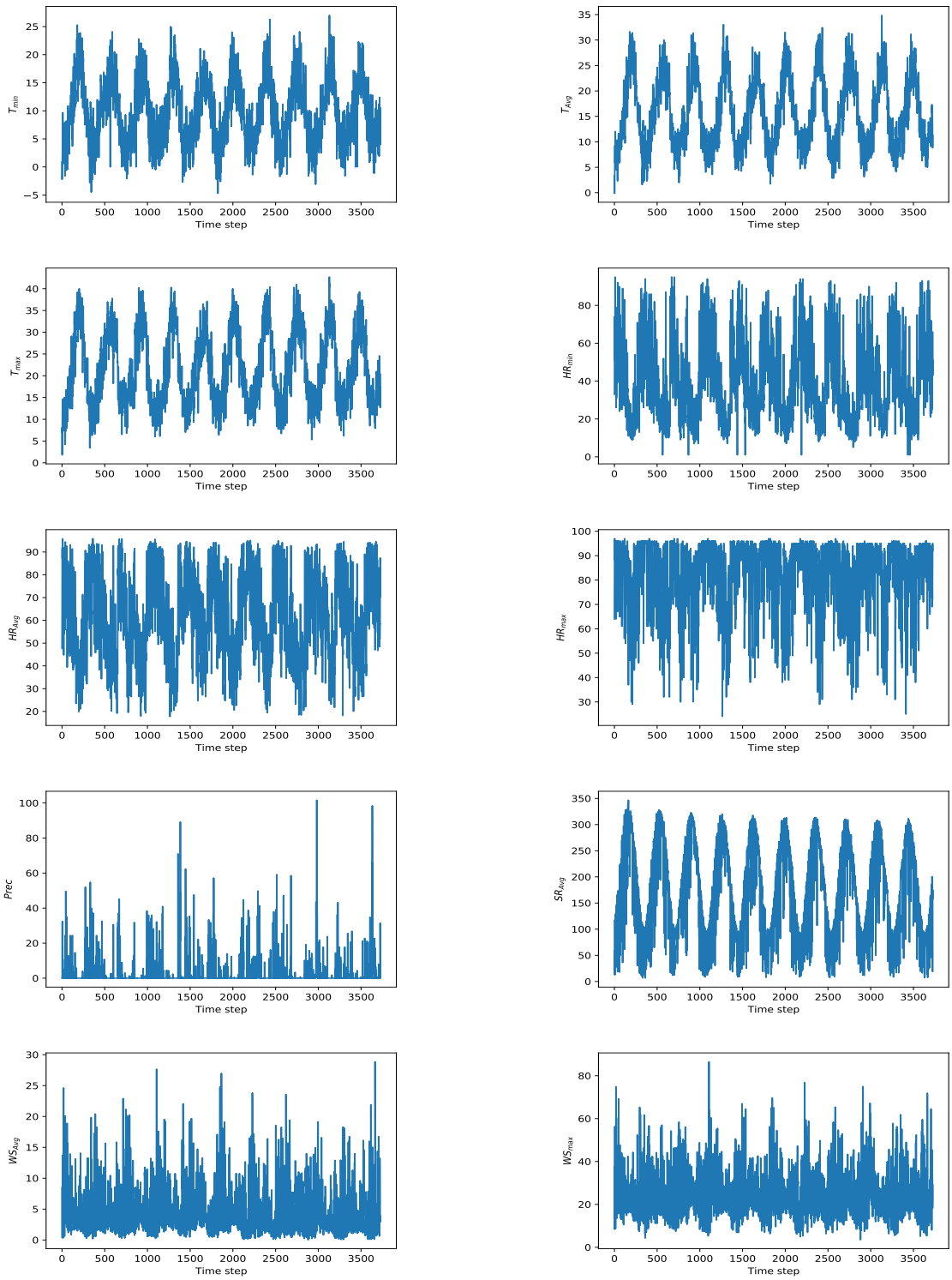


Figure 5.2: Dataset over time [27]. The unit for each variable can be seen in Table 5.2

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

Table 5.2: Dataset details [27]. Abbreviations represent the following: Max: maximum, Min: minimum, SD: standard deviation, Avg: Average, T: Temperature, HR: relative Humidity, SR: Solar Radiation, WS: Wind Speed, Prec: Precipitation.

To create an environment in which a DRL agent can interact during the training period, crop yields and total soil water in profile (SWTD) based on different irrigation were needed in addition to climate data. In the real world, recording crop yields and SWTD with various irrigation is extremely time-consuming and sometimes impractical. Therefore, Decision Support System for Agrotechnology Transfer (DSSAT) was employed to simulate these variables. DSSAT is a software operating program consisting of crop production simulation models for more than 42 crops that evaluate the effects of environmental and management influences on crop production. DSSAT simulates crop yield response to water. Tomato yields were simulated using DSSAT under different irrigation schedules in Fadagosa. The crop simulation model CROPGRO-Tomato from DSSAT was used to conduct the study. The soil type selected was sandy loam, which is typical of Fadagosa. The planting date was fixed to the first of April and the harvest date to the time of maturity. In each season, tomato seedlings were selected as direct seeding in a single row with a planting distance of 60 cm. The initial soil condition was selected with 70% available water. Sprinkler irrigation was selected as an irrigation method, which is adjustable in the DSSAT software. Four days and values between 0-60 were chosen as the time and depth norms for determining the irrigation schedule. The irrigation amount for each experimental year included no irrigation and specified depths of 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 60 mm per four days. ETo calculator is software developed by the Land and Water Division of FAO to calculate reference evapotranspiration using the FAO-56 Penman-Monteith method [137]. In this work, this software was used to calculate ETo (see figure 5.2).

5.2.2 Data Pre-Processing

The same data preprocessing as in [27], including filling missing data with the moving average method, removing multicollinear parameters in the dataset, and normalizing parameters between 0 and 1, was applied in this work. The variance inflation factor (VIF) is a measure of the extent of multicollinearity in a set of multiple regression variables and an instrument for measuring the degree of multicollinearity between variables [183]. In this paper, the VIF was used to remove the variables with a VIF greater than 5. In the end, the variables aver-

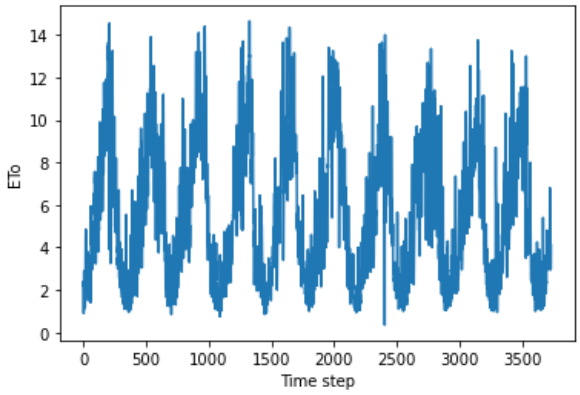


Figure 5.3: ETo $\mathrm{mm\ d^{-1}}$) over time [27].

age temperature ($\mathrm{T_{Avg}}$), average humidity ($\mathrm{HR_{Avg}}$), average wind speed ($\mathrm{WS_{Avg}}$), reference evapotranspiration (ETo), total soil water in profile (SWTD), irrigation, and precipitation (Prec) remain as inputs to the model.
The goal of normalization is to bring the values of the numeric columns in the dataset to a common scale without distorting the differences in the ranges of values. For machine learning, normalization is required only when the resources have different value ranges. The equation (5.1) was used to bring data between 0 and 1.

$$y = \frac{x - x_{min}}{x_{max} - x_{min}} \quad (5.1)$$

where x_{max} and x_{min} are the maximum and minimum values in the dataset.

5.2.3 Deep Learning Algorithm

Machine learning is a data analysis technique that allows computers to extract information from data to detect patterns and decide with minimal human intervention. It is a subfield of artificial intelligence that teaches computers the ability to perform tasks based on examples without explicit programming [12]. [16] presented an overview of the application of ML algorithm in various agricultural activities of farming. DL is a subfield of machine learning. The term ”deep” points to the number of hidden layers in the design of artificial neural networks. Each layer converts the input into actionable information that an adjacent layer can use for a prediction task. Since the models of DL are faster and more efficient compared to the traditional ML models and, unlike them, have the ability to automatically extract features from the input data, they have been increasingly used in recent years. The DL algorithms are successfully used for detection, classification, and segmentation [18].
Three types of DL models were used in this work, which is listed below.

5.2.3.1 Artificial Neural Networks

Artificial Neural Networks (ANN) is part of the Machine Learning domain, and their use is broad in Supervised Learning. ANNs are applicable to both regression and classification problems in ML [185].

An Artificial Neural Network performs its processing through neurons distributed in layers, aiming for a structure analogous to neurons in the human brain [185].
The layers of ANN models consist of an input layer, hidden layers (fully connected layers), and the output. The input layer processes the data input. The hidden layers are linear functions that add weights and biases to the input of the layer and extract features from the input data and do the processing. Finally, the output layer is the final result of neural network processing. Activation functions are used after a convolution layer to introduce nonlinearities into the model. There are many functions, such as sigmoid, tanh, and softmax. However, the most suitable for convolutional networks is relu, as it is computationally more efficient without much difference in accuracy compared to other functions. This function resets all negative values from the output of the previous layer [185].

5.2.3.2 Convolutional Neural Networks

The convolutional layer applies filters to extract features from the input of the layer. These filters are matrices of weights and biases applied to the previous output layer. A filter applies to a region of the image, and a dot product between the input and the filter is calculated, and a matrix is an output. Then the filter is shifted, and this process is repeated until the filter has swept all the inputs. The final output of the layer is called feature maps.
Similar to the ANN model, the activation function is used after each convolution layer to add nonlinearity to the model.
A pooling layer is applied to simplify the information from the previous layer. The most commonly used method is max pooling, where only the larger unit number is passed to the output. This data pooling is used to reduce the number of weights to be learned and to avoid overfitting.

5.2.3.3 Recurrent Neural Network and Long-Short Terms Memory Structure

Recurrent Neural Network (RNN) is constructed to deal with sequential data with a temporal dataset. Unlike CNN and AAN, its purpose is to use data from a previous input to understand the current data [54]. RNN can be considered as several copies of a network, each of which transmits information to another network.
The drawback of the RNN model is that for large inputs, the gradient of the loss function may approach zero during the training of the model, eventually causing the failure to update the weights of the first layer [54]. To overcome this drawback, the Long-Short Terms Memory (LSTM) was designed.
A typical LSTM unit comprises of several gates, including an input gate $\mathrm{i_t}$, an output gate $\mathrm{o_t}$, a block input $\mathrm{z_t}$, a forget gate $\mathrm{f_t}$, and a memory cell $\mathrm{c_t}$ (see Figure 5.4).
The first step of the LSTM unit is to determine what information to discard from the previous cell state. The forget gate makes this decision using a sigmoid function. First, a linear combination of the input $\mathrm{x_t}$ at time t and the output of the previous unit $\mathrm{h_{t-1}}$ is computed by multiplying with weight and bias matrices. Then, a sigmoid function inputs the combination and outputs a number in the range of 0 and 1. Zero means that all knowledge has

been forgotten, and the output of one means that all information from the previous unit has been retained. Similar to forget gate, the input gate uses a sigmoid function to control the information from the current input of the cell and the output of the previous unit and adds valuable information to $\mathrm{c_t}$. The block gate generates a new vector value of $\mathrm{x_t}$ and $\mathrm{h_{t-1}}$ using the tanh function. The vector values and the controlled values generated by the input gate are multiplied together to produce useful information. The output gate $\mathrm{o_t}$ uses a sigmoid function and determines the information of the cell that computes the output of the LSTM unit. First, the tanh function is applied to the cell and a vector is generated. Then a sigmoid function controls the information and derives the values to be obtained using $\mathrm{x_t}$ and $\mathrm{h_{t-1}}$. The controlled values and the vector values are multiplied to be used as input for the next unit and output.

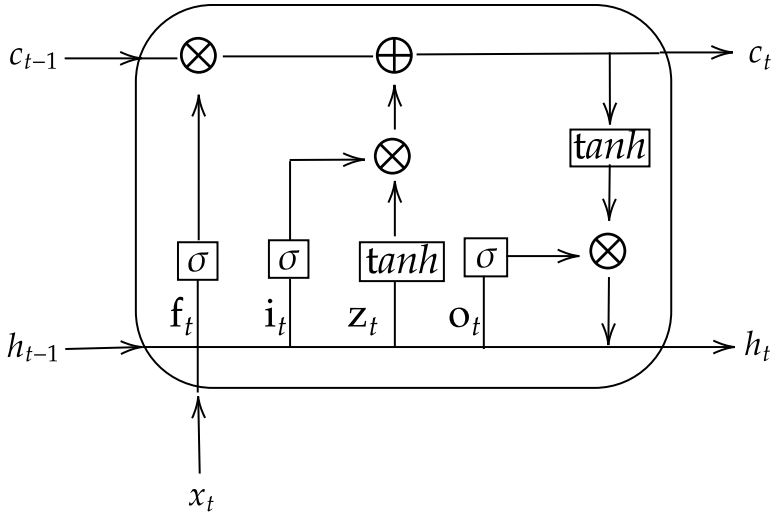


Figure 5.4: The LSTM cell.

5.2.3.4 Deep Q-Network

Before talking about reinforcement learning, it is necessary to explain a Markov Decision Process (MDP). MDP is the mathematical model used to describe the learning problem [167]. There is an agent that makes the decisions and learns within an environment that interacts with the agent at each time t. The agent chooses an action a from a set of actions A that can be performed depending on the current state s. The environment responds at the next time step by transitioning to a new state $\mathrm{s'}$ and gives the agent a corresponding reward $\mathrm{R_a(s, s')}$. The probability that the agent transitions to the new state $\mathrm{s'}$ is a direct consequence of the previous state and the chosen action. More precisely, it is given by the state transition function $\mathrm{T_a(s, s')}$. Therefore, an MDP consists of a tuple $\mathrm{(S, A, T, R)}$ where S is the set of all environmental states, A is all possible moves the agent can make, $\mathrm{R : S \times A \times S \longrightarrow \mathbb{R}}$ is the reward function , which $\mathrm{R(s, a, s')}$ is immediate feedback from the environment evaluating the action a, and $\mathrm{T : S \times A \times S \longrightarrow [0, 1]}$ indicates the probability function that $\mathrm{T(s, a, s')}$ is the probability that action a in state s at time t leads to state $\mathrm{s'}$ at time $\mathrm{t + 1}$ [167].
Policy ($\pi : \mathrm{S} \longrightarrow \mathrm{A}$) is a strategy used by the agent to determine the next action based on the

current state. Reinforcement learning is an MDP problem in which an agent interacts with an environment (see Figure 5.5). The goal of reinforcement learning is to find the optimal

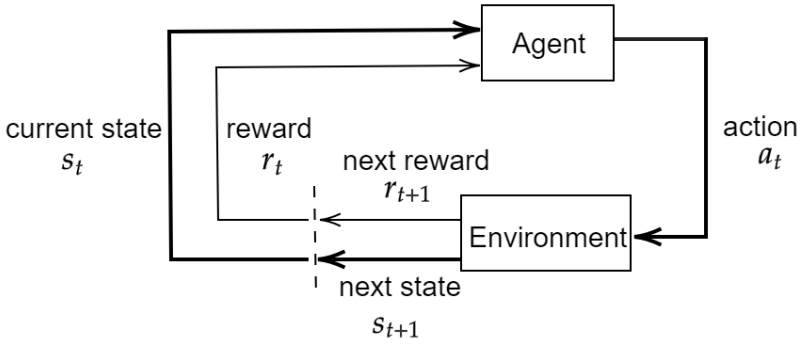


Figure 5.5: interaction of agent and environment.

strategy to maximize the sum of long-term rewards. The long-term reward at time t is defined by Equation (5.3) [167]:

$$\mathrm{G_t} = \mathrm{R_t} + \gamma \mathrm{R_{t+1}} + \gamma^2 \mathrm{R_{t+2}} + \gamma^3 \mathrm{R_{t+3}} \cdots + \gamma^{\mathrm{T}-1} \mathrm{R_T} \quad (5.2)$$

$$= \mathrm{R_t} + \sum_{\mathrm{n}=1}^{\mathrm{T}} \gamma \mathrm{G_{t+1}} \quad (5.3)$$

Where $\mathrm{R_T}$ is the reward of the terminal state and γ is the discount factor satisfying $0 \leq \gamma \leq 1$. It is necessary to add the discount factor because future states are uncertain.

The action-value function , or Q-function, is a function that can measure the value of a state-action pair, that is, for a given state s it can evaluate how good it would be to perform an action.The Q-function for a policy π is defined by Equation (2.2) [167]:

$$\mathrm{Q}_\pi(\mathrm{s}, \mathrm{a}) = \mathbb{E}_\pi[\mathrm{G_t} | \mathrm{S_t} = \mathrm{s}, \mathrm{A_t} = \mathrm{a}] \quad \forall \mathrm{s} \in \mathrm{S}, \quad (5.4)$$

The optimal policy (π^*) is defined as one that maximizes the expected value of rewards. To compute π^*, a Q-value iteration algorithm is used to compute the expected reward of the current state: Given an arbitrary state s and action a such that its expected Q-value is $\mathrm{Q}(\mathrm{s}, \mathrm{a})$, it applies the Bellman equation (5.5) until the convergence of $\mathrm{Q}(\mathrm{s}, \mathrm{a})$, denoted by $\mathrm{Q}^*(\mathrm{s}, \mathrm{a})$, is reached, which is used to compute the optimal policy.

$$\mathrm{Q}^{(\mathrm{i})}(\mathrm{s}, \mathrm{a}) = \sum_{\mathrm{s}'} \mathrm{T}(\mathrm{s}' | \mathrm{s}; \mathrm{a}) \left[\mathrm{R}(\mathrm{s}; \mathrm{a}; \mathrm{s}') + \gamma \max_{\mathrm{a}'} \mathrm{Q}^{(\mathrm{i}-1)}(\mathrm{s}', \mathrm{a}') \right] \quad (5.5)$$

$$\lim_{\mathrm{i} \to \infty} \mathrm{Q}^{(\mathrm{i})}(\mathrm{s}, \mathrm{a}) = \mathrm{Q}^*(\mathrm{s}, \mathrm{a}) \quad (5.6)$$

The optimal policy can be defined using the optimal Q-function with Equation (5.7):

$$\pi^*(s) = \arg\max_a Q^*(s, a) \quad (5.7)$$

A Markov decision process whose transition probability function or reward is unknown becomes a reinforcement learning problem. Deep Q-learning [190] is a form of reinforcement learning that uses deep learning algorithms to approximate the Q-function. The model is given the current state of the environment and outputs the Q-value of the current state for each possible action. The loss function of the model to predict the Q-function is defined by Equation (5.8):

$$L(\theta) = \left(\left(r(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta) \right) - Q(s_t, a_t; \theta) \right)^2 \quad (5.8)$$

where θ are the weights of the network. The problem with the vanilla DQN model is that the training of the model is unstable. In the error calculation, the objective function (Q-value) is changed frequently, and this unstable objective function makes training difficult. To overcome this problem, the target network was defined. The target network $\hat{Q}$ is a copy of the Q-network with the same input-output, but it updates its weights ($\hat{\theta}$) differently. In this paper, we used the soft update given by Equation (5.9) [191]:

$$\hat{\theta} = \tau\theta + (1 - \tau)\,\hat{\theta} \quad (5.9)$$

where τ is a number between 0 and 1 that manages the scope of updates for the target network. Using the target network, the loss function can be modified by Equation (5.10):

$$L(\theta) = \left(\left(r(s_t, a_t, s_{t+1}) + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1}; \hat{\theta}) \right) - Q(s_t, a_t; \theta) \right)^2 \quad (5.10)$$

Another solution to make the training of the DQN more robust is to use Experience-Replay memory , which removes correlations between observations and avoids overfitting the Q-network [192]. At the time t, the experience of the agent is defined as the tuple $e_t = (s_t, a_t, r_t, s_{t+1})$, where s_t is the current state, a_t is the action performed in the current state, r_t is the reward from the environment upon execution of action a_t, and s_{t+1} is the next state. The experience replay memory stores all the experiences the agent has had at each time step over all the episodes it has played. During the agent's training, a random batch is selected from the replay memory and the agent trains with that batch. In this paper, a batch size of 64 was chosen.

5.2.4 Creating environment for the DRL agent

An environment is required to collaborate with the agent during training and aid the agent find the optimal policy. The first step to constructing this interactive training environment is to design the states, actions, and rewards. In our design, the agent controls irrigation by

observing the climate data, irrigation amount (irr), and SWTD. The state at time t is defined by a combination of these parameters. Based on the agent states, the agent selects an action every four days. Table 5.3 shows the states and actions used in this work.

States	T_{Avg}, HR_{Avg}, WS_{Avg}, Prec, ETo, SWTD, irr
actions ($mm - ha/ha$)	0, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60

Table 5.3: State and action parameters.

For agricultural irrigation, the long-term reward is defined as the net return given by Equation (5.11):

$$r = y *_p y - w * p_w \quad (5.11)$$

where w is the overall water consumption during a season, p_w is the cost of 1mm-ha/ha of water, y is the total amount of yield (kg), and p_y is the price of 1kg of yield.
An episode contains a loop in which the agent interacts with the training environment by selecting various actions. Since the reward depends on the crop yield, in our implementation, an episode is defined as the length of a season. During the season, the reward is zero, and at the end of the season, the reward is determined by Equation (5.11).
Bidirectional LSTM (BLSTM) extends the LSTM model. The first recurrent layer in the network is duplicated so that two layers are now adjacent, then the input sequence in its original form is provided as input to the first layer, and an inverted copy of the input sequence is provided to the second [156].
In [27], a two-layer bidirectional LSTM model consists of 512 nodes in each layer developed for tomato yield prediction using irrigation scheduling and climate Big Data. The model receives daily climate data, irrigation scheduling, and SWTD during a season to predict yield. The same model was trained with the new dataset from DSSAT. Since the harvest date was set to the ripening date in the DSSAT, the season length was changed from 110 in [27] to 172 days, and the same change was made in the model. Root Mean Square Error (RMSE) and the R^2-score were used to evaluate the model [162]. RMSE and R^2-score are calculated by Equations(5.12) and (5.13), respectively:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - y'_i)^2}{n}} \quad (5.12)$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - y'_i)^2}{\sum_{i=1}^{n} (y_i - \hat{y})^2} \quad (5.13)$$

where y_i's, y'_i's, and $\hat{y}$ are the true value, the predicted value, and the mean of the true values, respectively. In this work, the trained model was used to estimate the tomato yield and then calculate the agent's reward at the end of the season.

Another LSTM model was trained to estimate the daily SWTD one day in advance and complete the agent environment. This model is used to predict the next state of the environment. In [26] [3], a single-layer bidirectional LSTM model consisting of 512 nodes was used to predict the soil water content and reference evapotranspiration one day ahead. The same structure with the same hyperparameters is trained in this work to predict SWTD one day ahead. As mentioned earlier, LSTMs can look back many time steps and use this information to make predictions about what will happen next. In our implementation, irrigation occurs every four days. Therefore, the lookback of the BLSTM was changed from eight in [26] to four days of historical data representing the current state of the environment. The output of the model was also changed to predict only SWTD and the prediction of ETo was removed from it. The same hyperparameters and strategy as in [26] were used to train the model.

After defining the state, reward, and environment design, Algorithm 2 was created. A boolean value Done is defined to represent whether a training episode is complete for the DRL agent.

Algorithm 2 Environment of the agent

```
function ENV(step, action)
    season_state = List()
    next_WcTot = RF_WcTot.predict(current_state, action)
    season_state.append(current_state, action)
    if time_passed = end_of_season
        done=True
        Y=BLSTM_yield.predict(season_state)
        calculate reward from Equation (5.11)
    else
        done=False
        reward=0
    end if
    return next_SWTD, net_return, done
end function
```

5.2.5 Training configuration of the DRL agent

A two-layer ANN, CNN, and LSTM model with 256 nodes were used to estimate the Q- value. Since the number of time steps was only four, a filter size of three was chosen for each convolutional layer. The models receive the current state of the environment, and the output is the value for each action.

Exploitation versus exploration is a critical issue in Reinforcement Learning. The RL agent should find the best solution as quickly as possible. However, if it commits itself too quickly to a solution without sufficient exploration, it may lead to local minima or outright failure [167]. To explore the environment, the method Epsilon-Greedy was used, i.e., the agent chooses a random action with probability ε and takes the optimal action with probability $1 - \varepsilon$ [167]. The ε was set to 1, and at each episode, ε was decremented by a factor of 0.9997 and stopped

[3] The code is available at the following link:
`https://github.com/falibabaei/-Soil-Water-Content-and-Reference-Evapotranspiration`

when it reached 0.001. The duration of a season was divided into four, and the agent chose the irrigation action 42 times.

In training time, the model randomly selects a batch of experiences from memory, and training is performed on that batch. The batch size is a hyperparameter that must be set before training. A batch size of 64 was chosen to speed up training and avoid overfitting or underfitting the model. The agent was trained on a dataset from 2010 to 2017 for 10000 episodes and tested on the last two years (2018-2019). The rewards received by the DRL agent were calculated using Equation (5.11). Since the net return was very high, the agent could not converge. To avoid this problem, the logarithms of the reward were used to calculate the agent's reward. As the net return during a season was zero, it was truncated by one so that the reward would not be infinite. Irrigation costs for 1 mm/ha were set at 0.5 USD [186] and tomato prices for one hectare at 728.20 USD/ton (`www.tridge.com`). These parameters are tunable. The average reward was calculated every 50 episodes, and when the average reward was improved, the weights of the Q-network were saved. After configuring the training of the agent, Algorithm 3 was created to train the agent.

Algorithm 3 DQN Algorithm

initialize reply memory **D**
initialize training environment env=LSTM()
randomly initialize Action-Value function Q with weight θ
randomly initialize target Action-Value function $\hat{Q}$ with weight $\hat{\theta} = \theta$
for episode=1 to max_episode **do**
  Done=False
  while Done=False **do**
    $\text{random} = \text{random\_uniform}(0, 1)$
    if $\text{epsilon} < \text{random}$ **then**
      $\text{action} = \text{random.choice}(0, \text{nubmber\_of\_actiones})$
    else
      $\text{action} = \text{argmax}_a(Q(\text{current\_state}, a; \theta))$
    next_SWTD, reward, Done=env(current_state, action)
    next_state=Concatenate(next_SWTD, next_climate_data)
    store transition (current_state, action, reward, next_state, Done) in **D**
    sample random batch of transitions from **D**
    Set:

$$y_j = \begin{cases} \text{reward} & \text{if Done} = \text{True}, \\ \text{reward} + \gamma \text{max}_a \hat{Q}(\text{next\_state}, a; \hat{\theta}) & \text{Otherwise}; \end{cases}$$

    do a single gradient step on $(y_j - Q(s_j, a_j; \theta))^2$ w.r.t θ
    update the weights of $\hat{Q}$ using equation (5.9)
    current_state=new_state

5.3 Results and Discussions

5.3.0.1 Evaluation of the LSTM models

The BLSTM model for yield estimation was trained for a maximum of 500 epochs and it achieved an R^2- score of 0.97 and an RMSE of 366 (kg/ha) on the tomato yield test data set. The model predicting SWTD was also trained for a maximum of 500 epochs and it obtained an RMSE of 6.841 mm and an R^2 score of 0.98. Figure 5.6 shows the actual values of SWTD and of tomato yield compared to the predicted value by the models.

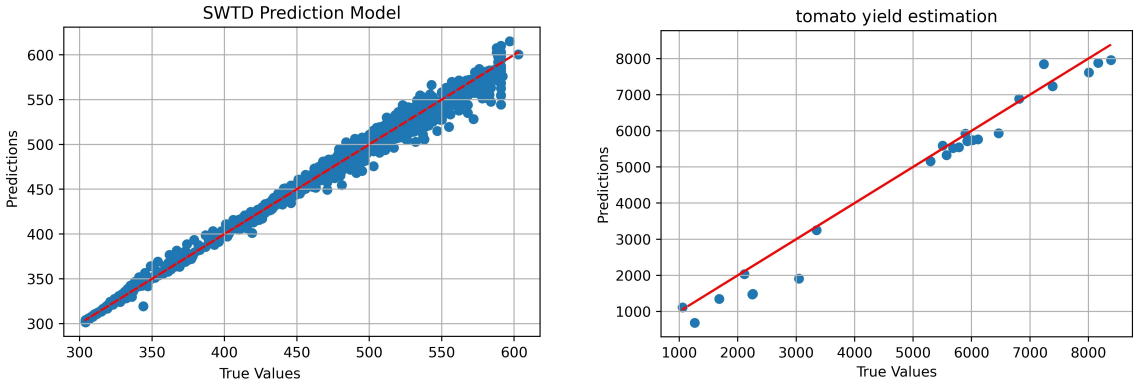


Figure 5.6: True values STWD (mm/ha) vs. predicted values by models.

Since the result of these two models has already been compared with other models in [27] and [26], no comparison with other models was made in this paper. The trained LSTM models were replaced in Algorithms 2 to complete the agent environment.

5.3.1 Evaluation of the DRL agent

Figure 5.7 shows the average rewards received by the agent during the training of the Q(ANN)-network and the Q(CNN)-network. As Figure 5.7 shows, the agent did not learn, and the average rewards during training not only did not improve but actually decreased. Therefore, in the irrigation scheduling problem where the state of the environment is time series, the Q(ANN)-network and the Q(CNN)-network do not converge to an optimal policy.

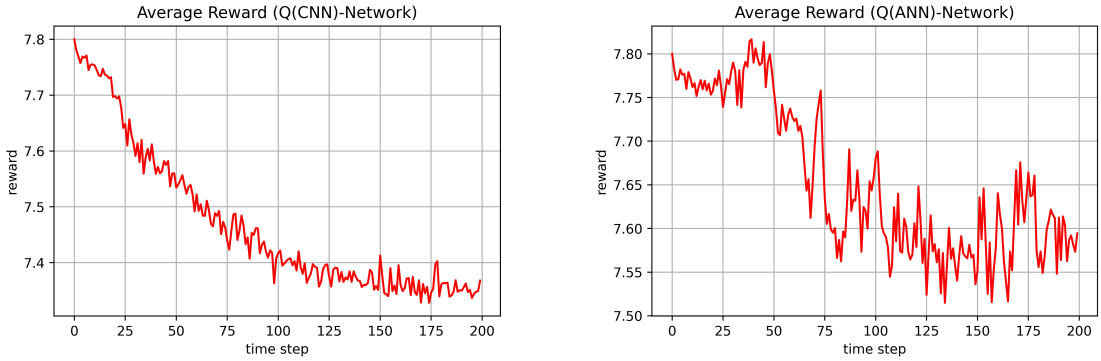


Figure 5.7: Agent average rewades of Q(ANN)-network and Q(CNN)-network during training.

Figure 5.8 shows the agent's average rewards during the training of the Q(LSTM)-network. The Q(LSTM)-network showed the best performance in predicting the Q-table, and the reward improved during training compared to ANN and the CNN model. At the beginning of training, the model starts exploring the environment and converges to an optimal policy after a while. The reason for the better performance of the LSTM is its ability to achieve higher accuracy in time-series predictions, as it can use information from previous inputs and loops within the LSTM layers [193].

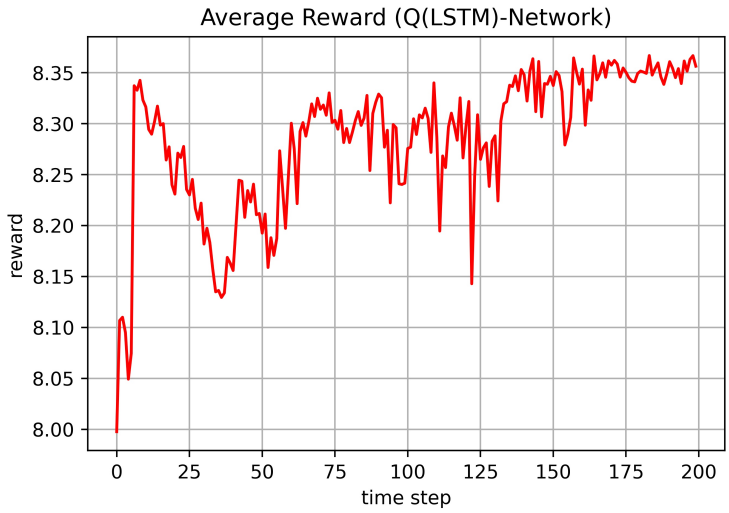


Figure 5.8: Agent average rewards during training.

Figure 5.9 shows the loss of LSTM prediction of Q-table and epsilon during training of Q(LSTM)-networks. As shown in Figure 5.8 and 5.9, when epsilon is high, the model explores the environment, and when epsilon becomes lower, the model starts exploiting. The loss of the LSTM predicting Q-table tends close to zero very quickly, but the reward coverage after almost 8000 episodes.

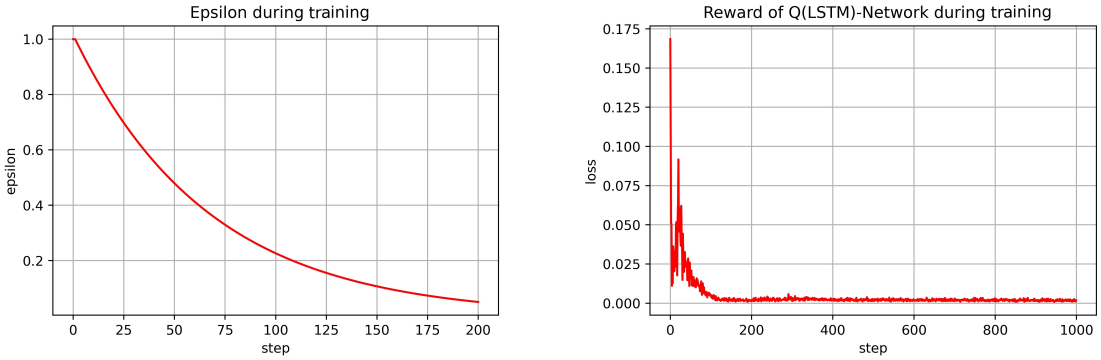


Figure 5.9: Epsilon and loss of LSTM model predicting Q-value.

The agent was tested with the 2018 and 2019 datasets. Figure 5.10 shows the SWTD predicted by the model and the amount of irrigation chosen by the agent in 2018 and 2019. As the figure shows, the agent automatically learned how to avoid overwatering the plants. In the

beginning, it adopts the measure of lower water supply, but in the later stages, it increases the water supply. This prevents wasting water at the beginning of the season and stresses the plants at the end of the season. Moreover, the total precipitation in the 2018 and 2019 seasons was 575.19 mm and 157 mm, and the total irrigation amounts were 965 and 1165 mm/ha, respectively. Therefore, the trained model automatically learned to irrigate more when there is less rainfall, and it adjusts the irrigation based on climate change.

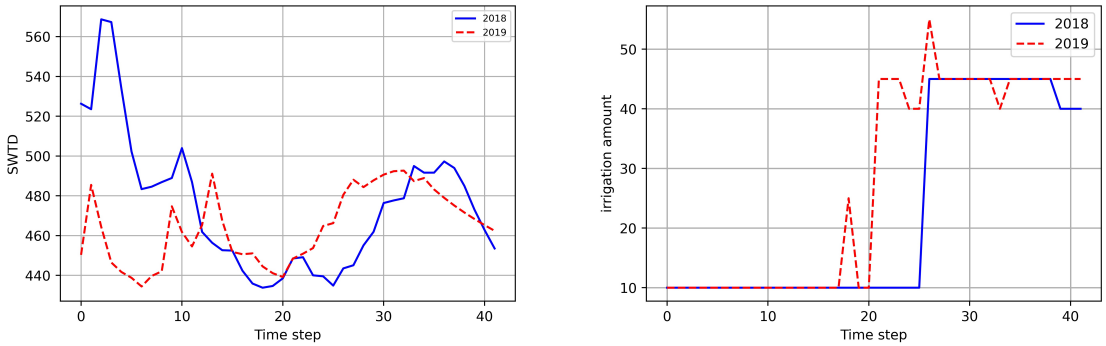


Figure 5.10: The right side shows the SWTD (mm/ha) predicted by the model and the right side shows the amount of irrigation.

Table 5.4 shows the comparison of net return under different irrigation rates. As can be seen from the table, the trained model obtained the highest net return compared to other fixed irrigation rates. The net returns of the trained model improved by 11% in 2018 and 2019 relative to the best net return of the fixed irrigation volume, while water consumption was reduced by 20% to 30%. This amount of reduction is significant.

The analysis of tomato yield shows that the only cases where yield increased compared to the trained model are fixed irrigation at 40 and 50 mm. In these cases, yield increased by 2-10%, but water use increased by 18-55%, resulting in a reduction in net return of at least 7%.

Irrigation	Yield (Kg/ha)	Total Irrigation (mm/ha)	Net Return ($/ha)
trained Agent (2018)	4675	965	3270
Fixed 10 mm (2018)	1065	420	645
Fixed 15 mm (2018)	1548	630	928
Fixed 20 mm (2018)	2562	840	1636
Fixed 25 mm (2018)	3508	1050	2291
Fixed 30 mm (2018)	4071	1260	2638
Fixed 40 mm (2018)	4725 (2%↑)	1165 (18%↑)	2952
Fixed 50 mm (2018)	5157 (10%↑)	2100 (55%↑)	3089
trained Agent (2019)	4046.6	1165	2666
Fixed 10 mm (2019)	889	420	503
Fixed 15 mm (2019)	995	630	484
Fixed 20 mm (2019)	1227	840	840
Fixed 25 mm (2019)	1841	1050	953
Fixed 30 mm (2019)	2681	1260	1522
Fixed 40 mm (2019)	3793	1680	2205
Fixed 50 mm (2019)	4399 (8%↑)	2100 (47%↑)	2481

Table 5.4: comparison of net return under different amounts of irrigation.

Figure 5.11 shows the SWTD under different irrigation rates. Under the fixed method, the SWTD is high at the beginning of the season and decreases at the end of the season, but the trained model keeps the SWTD above a value of 440 and does not stress the plant at the end of the season, and does not allow the water to be wasted at the beginning of the season.

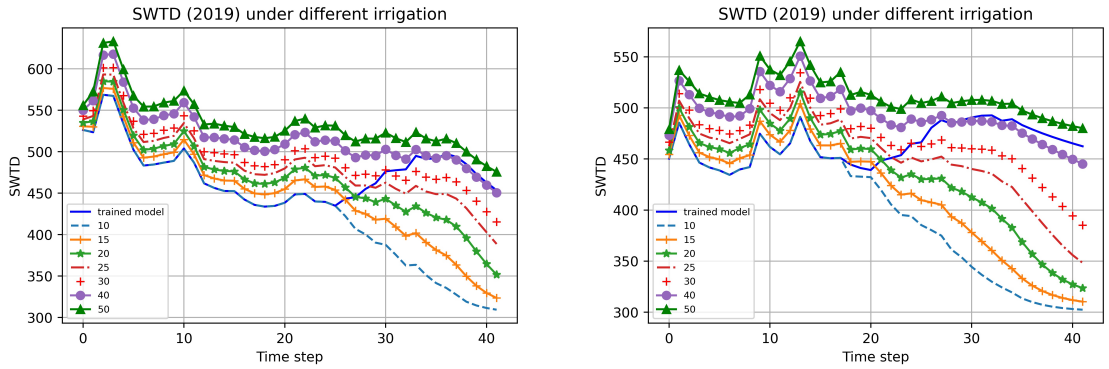


Figure 5.11: SWTD ($\mathrm{mm/ha}$) under different irrigation.

Since the model kept SWTD above the threshold of 440, the model was compared to a threshold-based irrigation method with thresholds of 440, 460, and 480, which irrigates a fixed amount of water in four days when SWTD is below a threshold. The result is shown in Tables 5.5, 5.6, and 5.7. The percentages next to the net return show the reduction in the net return of the threshold methods compared to the net return of the trained model. As can be seen in the Tables 5.5 to 5.7, the best net return of the threshold-based irrigation method still cannot keep up with the trained model, and when the net return of the threshold method approaches the net return of the trained model, the amount of water used for irrigation increases significantly. For example, in 2018, the net return of threshold-based irrigation with a threshold of 480 and a fixed water volume of 50 is 1% higher than the net return of the trained model, but the amount of water used increased by 9%. Therefore, the trained model performed better than the threshold-based irrigation method. In terms of net return, among the threshold irrigators, the threshold of 460 had the best performance but still could not match the performance of the trained model.
In threshold irrigation 480 with irrigation rates of 40 and 50 mm, tomato yield increased between one and 8%, but irrigation volume increased by 20-35% compared with the trained model, which is a considerable amount of water. In other cases of threshold irrigation, tomato yield decreased compared with the trained model.

5.4 Conclusions

In this paper, the ability of reinforcement learning to schedule irrigation of a tomato field was investigated. A DQN model was trained with seven years of data and tested with two years of data. The results show that the LSTM model can predict the Q- table better than CNN and ANN models when the states of the environment are time series. The trained Q(LSTM)-network automatically learns to avoid water wastage at the beginning of the season and stress

Irrigation	Yield (Kg/ha)	Total Irrigation (mm/ha)	Net Return ($/ha)
trained Agent (2018)	4675	965	3270
Fixed 25 mm (2018)	2874	600	2007 (42% ☐)
Fixed 30 mm (2018)	3733	690	2651 (19% ☐)
Fixed 40 mm (2018)	4164	800	2942 (10% ☐)
Fixed 50 mm (2018)	4316	849	3039 (8% ☐)
trained Agent (2019)	4046.6	1165	2666
Fixed 25 mm (2019)	1541	725	874 (67% ☐)
Fixed 30 mm (2019)	2341	870	1444 (46% ☐)
Fixed 40 mm (2019)	3469	1000	2284 (15% ☐)
Fixed 50 mm (2019)	3646	1150	2412 (10% ☐)

Table 5.5: comparison of net return of trained model with irrigation with the threshold of 440.

Irrigation	Yield (Kg/ha)	Total Irrigation (mm/ha)	Net Return ($/ha)
trained Agent (2018)	4675	965	3270
Fixed 25 mm (2018)	3278	675	2293 (20% ☐)
Fixed 30 mm (2018)	3980	810	2790 (15% ☐)
Fixed 40 mm (2018)	4498	960(1% ☐)	3130 (5% ☐)
Fixed 50 mm (2018)	4769	1050 (9% ↑)	3303 (1% ↑)
trained Agent (2019)	4046.6	1165	2666
Fixed 25 mm (2018)	1712	875	937 (99% ☐)
Fixed 30 mm (2019)	2581	1020	1562 (42% ☐)
Fixed 40 mm (2019)	3755	1279 (9% ↑)	2374 (10% ☐)
Fixed 50 mm (2019)	4190	1450 (20% ↑)	2638(2%☐)

Table 5.6: comparison of net return of trained model with irrigation with the threshold of 460.

Irrigation	Yield (Kg/ha)	Total Irrigation (mm/ha)	Net Return ($/ha)
trained Agent (2018)	4675	965	3270
Fixed 25 mm (2018)	3506	825	2401 (26% ☐)
Fixed 30 mm (2018)	4027	960	2752 (15% ☐)
Fixed 40 mm (2018)	4684 (1% ↑)	1200 (20% ↑)	3160 (4% ☐)
Fixed 50 mm (2018)	5000 (7% ↑)	1400 (31% ↑)	3306 (1% ☐)
trained Agent (2019)	4046.6	1165	2666
Fixed 25 mm (2019)	1831	975	982 (63% ☐)
Fixed 30 mm (2019)	2676	1170	1563 (42% ☐)
Fixed 40 mm (2019)	3791	1519	2283 (15% ☐)
Fixed 50 mm (2019)	4395 (8% ↑)	1800 (35% ↑)	2628 (2% ☐)

Table 5.7: comparison of net return of trained model with irrigation with the threshold of 480.

to plants at the end of the season. Moreover, the trained model adjusts the irrigation amount according to climatic changes and rainfall during the season.

The comparison of the performance of the model was done with fixed base irrigation and threshold based irrigation. The comparison result shows that the trained model increased the productivity in the test field and reduced the water consumption (18%-30%) compared to fixed and threshold based irrigation methods. Although the net return of threshold-based irrigation reached the net return of the trained model at some thresholds, water consumption increased significantly. Therefore, reinforcement learning can be used in our case study for irrigation scheduling to avoid water wastage in agriculture without affecting productivity.

Chapter 6

Comparison of on-policy deep reinforcement learning A2C with off-policy DQN in irrigation optimization: a case study at a site in Portugal

Abstract[1]

Precision irrigation and optimization of water use have become essential issues 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 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 crops 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.

6.1 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 [194, 195]. 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 [194, 195].

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 [134]. The same studies indicate that reducing this loss by 10% would be enough to supply twice the current world population, based on statistical averages [134]. Hence, an efficient water management system is essential.

[1]This work was published in [30], K. Alibabaei, P. D. Gaspar, E. Assunção, S. Alirezazadeh, T. M. Lima, V. N. G. J. Soares, and J. M. L. P. Caldeira, "Comparison of on-policy deep reinforcement learning a2c with off-policy DQN in irrigation optimization: A case study at a site in Portugal," Computers, vol. 11, no. 7, 2022. [Online]. Available: https://www.mdpi.com/2073-431X/11/7/104

With the use of the Internet of Things (IoT) [119] in agriculture, systems are being developed to effectively manage fields [10, 11]. The IoT sensors enable monitoring of light, humidity, temperature, soil moisture, and analysis of water among other parameters [10, 11, 196, 197, 198]. 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 [16, 19, 21]. 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 [12, 59]. 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 [12, 59]. After this training, the model has the potential to make high-quality predictions in future situations related to historical patterns [16, 19, 21, 12, 59].

Zia et al. [199] 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. [145] estimated the soil moisture from the synthetic aerial images of a vineyard with known moisture conditions 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. [146] 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 [123] estimated daily reference evapotranspiration using a deep-learning multilayer 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 $\mathrm{mm\,d^{-1}}$.

De Oliveira and Lucas et al. [148] 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. [200] 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 $\mathrm{m^{-2}}$ for the first, fifth, and seventh-day predictions.

Adab et al. [201] 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 per-

formed 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. [202] 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 [60] 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 [12, 59]. 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 [12, 59]. 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 [60].

Chen et al. [61] 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. [28], 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 [29].

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 irriga-

tion. 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 6.2, 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, 6.2.1, data processing, 6.2.2, an overview of the models used in the work, 6.2.3, and experimental setup, 6.2.4. In Section 6.3, the results are described and discussed. The summary of the work is included in Section 6.4.

6.2 Materials and Methods

The framework of this paper is shown in Figure 6.1. The first two steps are the same as in [28], 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 [203, 204]. 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.

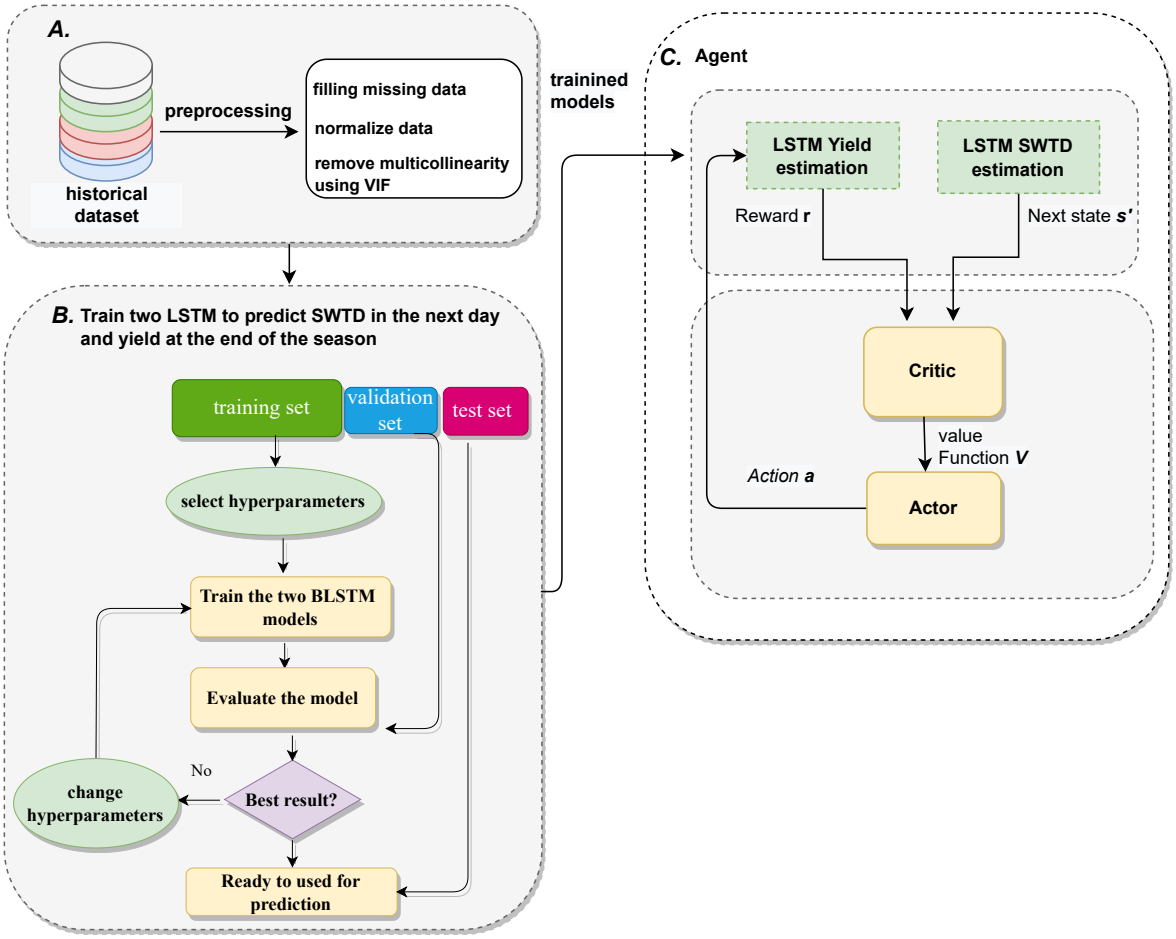


Figure 6.1: Framework of this paper. Modified version of [28].

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

6.2.1 Data Collection

To compare the potential of DQN and A2C models in irrigation scheduling, the same data from [28] 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 03, 04, 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 6.2 shows the Fadagosa region from Google Earth.

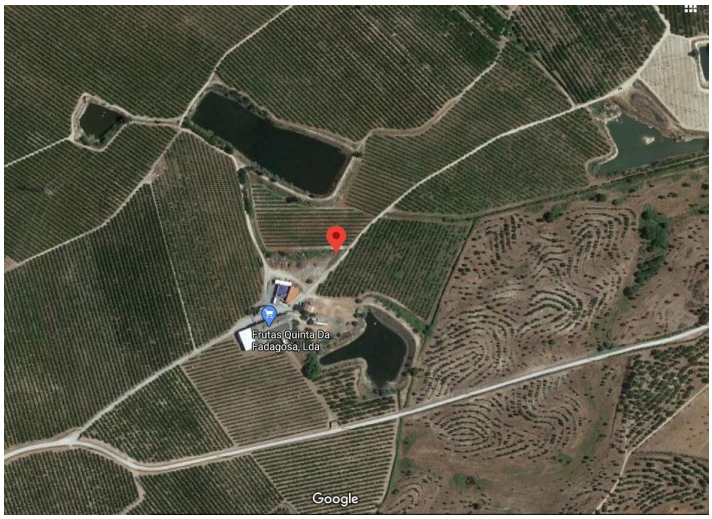


Figure 6.2: Map of the Fadagosa region.

Table 6.1 shows the details of the climate variables retrieved from the weather station, and Figure 6.3 shows the daily climate variables from 2010 to 2019.

Table 6.1: 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 [205].

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) [203, 204] 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 [28]. 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) [137].

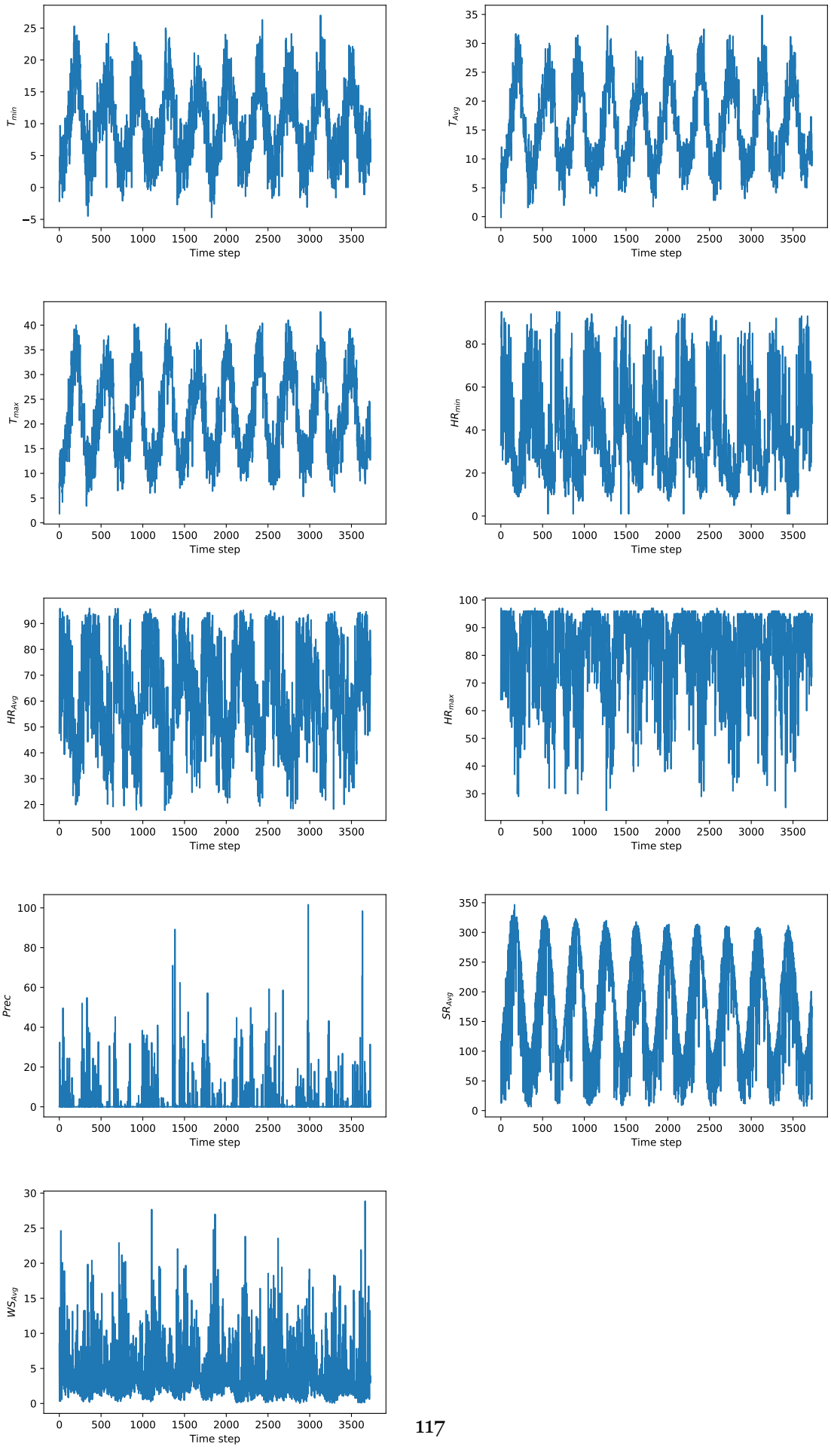


Figure 6.3: Fadagosa daily dataset [205]. The abbreviations and units of the variables are the same as in Table 6.1.

6.2.2 Data Pre-Processing

As in [28], the moving average was used to fill in the missing big data [153]. Then, the multicollinear parameters were removed from the data set using the variance inflation factor (VIF) [183]. 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 (6.1).

$$x_{new} = \frac{x_{old} - x_{min}}{x_{max} - x_{min}} \quad (6.1)$$

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

6.2.3 Model Used

6.2.3.1 Bidirectional LSTM Structure

Recurrent Neural Networks (RNNs) are a family of neural networks designed to process sequential data [206, 12]. A variant of conventional RNNs is Long Short-Term Memory (LSTM), which solves a well-known problem called a vanishing gradient [206, 12]. 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 [206, 12]. 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 6.4 shows the cell of an LSTM.

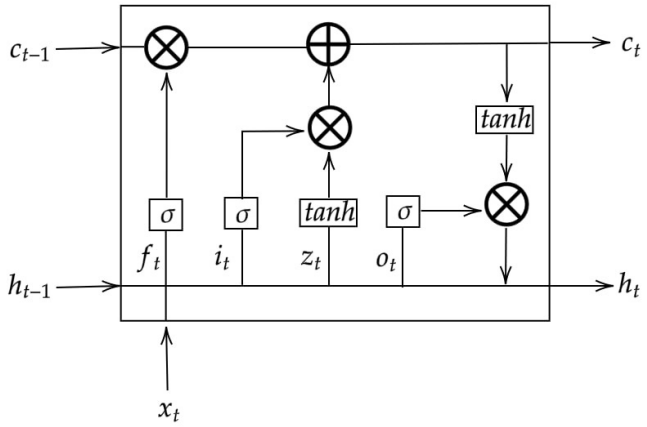


Figure 6.4: LSTM cell [205].

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 $\mathrm{h_{t-1}}$ and the input of the current cell $\mathrm{x_t}$ and produces an output of numbers between 0 and 1, where 1 represents the complete retention of that information (Equation (6.2)).

$$\mathrm{f_t} = \sigma(\mathrm{W_f}[\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_f}) \quad (6.2)$$

It then decides what new information it will retain. To this end, it performs two processes: First, the input gate $\mathrm{i_t}$ formed by the sigmoid function decides which value will be updated, and then the activation function tanh generates a vector of new candidates $\tilde{\mathrm{C}}_\mathrm{t}$ (Equation (6.3)) that can be added to the state.

$$\tilde{\mathrm{C}}_\mathrm{t} = \tanh((\mathrm{W_C}[\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_C}) \quad (6.3)$$

Then, the forget gate $\mathrm{f_t}$ is multiplied by $\mathrm{C_{t-1}}$ so that the information deemed unnecessary is forgotten, and $\mathrm{i_t}$ is multiplied by $\mathrm{C_t}$ to retain the new information that is useful. Then, add the result of these two multiplications. Finally, the output is determined. To do this, the sigmoid layer (Equation (6.4)) decides which parts of the cell state to output, and then multiplies this by the cell state $\tanh(\mathrm{C_t})$ so that only the information that the network has learned is important (Equation (6.5)).

$$\mathrm{o_t} = \sigma(\mathrm{W_o}[\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_o}) \quad (6.4)$$

$$\mathrm{h_t} = \mathrm{o_t} \odot \tanh(\mathrm{C_t}) \quad (6.5)$$

W and b are the weights and biases of the specific gate in Equations (6.2)–(6.5), which should be adjusted when training the model to minimize the loss function.
Bidirectional LSTMs (BLSTM) [162] 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.

6.2.3.2 Advantage Actor–Critic Network

A Markov decision process (MDP) contains a tuple (S, A, R, P), where [167]

- 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 [167, 60]. 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 [167, 60]. 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 (6.6) [167, 60]:

$$\begin{aligned} G_t &= R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} \cdots + \gamma^{T-1} R_T \\ &= R_t + \gamma \left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \cdots + \gamma^{T-2} R_T \right) \end{aligned} \quad (6.6)$$

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 (6.7) results from the definition of G_t and Equation (6.6).

$$G_t = R_t + \sum_{n=1}^{T} \gamma G_{t+1} \quad (6.7)$$

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 (6.8) [167]:

$$V_\pi(s) = \mathbb{E}_\pi[G_t | s_t = s] \quad \forall s \in S \quad (6.8)$$

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:

$$V^*(s) = \max_\pi V_\pi(s) \quad (6.9)$$

The optimal policy π^* is computed using an iteration algorithm over the V function. The Bellman Equation (6.10) is applied to $V(s)$ for any state s until $V(s)$ reaches the maximum value, denoted as $V^*(s)$.

$$V^{(i)}(s) = \sum_{s' \in S, a \in A} T(s'|s;a) \left[R(s;a;s') + \gamma \max_{s'} V^{(i-1)}(s') \right] \quad (6.10)$$

$$\lim_{i \to \infty} V^{(i)}(s) = V^*(s) \quad (6.11)$$

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) [29] is a DRL method that uses two different deep-learning models to perform the learning (see Figure 6.5). The first model is the actor, which is used to define the policy of the applied actions.

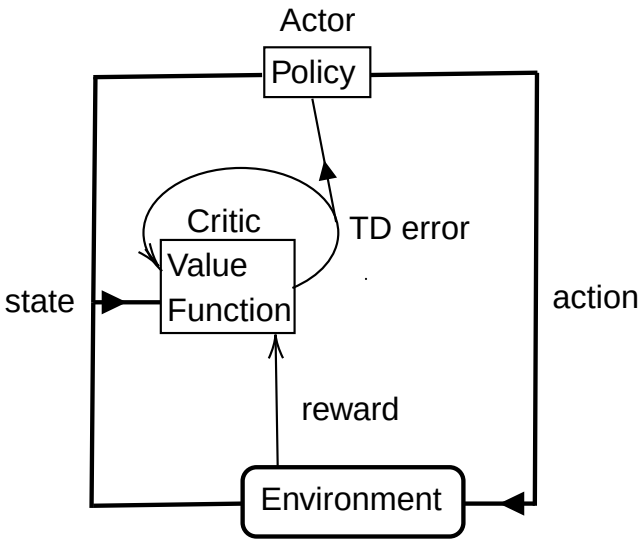


Figure 6.5: The A2C model interacting with environment.

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 [207, 29].

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 [29]. The TD error can be calculated using Equation (6.12):

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \quad (6.12)$$

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 [29]. 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 [208].

The critic model uses the TD error, Equation (6.12), 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 (6.13). 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 [209].

$$L_{\pi_\theta} = -\sum_{t=0}^{T} (\log \pi_\theta(a_t|s_t))\, \delta_t \quad (6.13)$$

where θ is the weight 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 [167]. Exploration without exploitation leads to a suboptimal solution. In the A2C model, the entropy bonus is usually added to the loss to improve exploration [210]. 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 (6.14).

$$\mathrm{E} = -\beta \sum_{\mathrm{t}=0}^{\mathrm{T}} \pi_\theta(\mathrm{a_t}|\mathrm{s_t}) \log \pi_\theta(\mathrm{a_t}|\mathrm{s_t}) \quad (6.14)$$

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 [211, 29].
- Unlike DQN, A2C does not use the Replay Buffer but learns the model using state, action, reward, and next state obtained at each step [211, 29].
- In DQN, the function Q is estimated; however, in A2C, the value function V and policy π are estimated [211, 29].

6.2.4 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 [163] and Keras [164] 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.

6.2.4.1 States and Actions Setup

The state and actions were selected as in [28] to ensure the comparability of the models. Table 6.2 shows the action and state sets.

Table 6.2: State and action sets.

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

6.2.4.2 Environmental Setup

Two BLSTM models were implemented in [205, 26], 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 [54]. 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 [54]. Table 6.3 shows the parameters used for each BLSTM model.

Table 6.3: 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 [28], 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 (6.15):

$$\mathrm{r} = (\mathrm{yield}) * \mathrm{p}_{\mathrm{yield}} - (\mathrm{water}) * \mathrm{p}_{\mathrm{water}} \quad (6.15)$$

where $\mathrm{p}_{\mathrm{water}}$ is the price of 1 mm over 1 ha of water and $\mathrm{p}_{\mathrm{yield}}$ 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 $728.2\$/\mathrm{tonne}$, respectively, ([186] 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 4 shows the environment created for the A2C agent, and Table 6.4 explains the parameters used in Algorithm 4.

Table 6.4: The parameters used in Algorithm 4.

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

6.2.4.3 Training Configuration of Agent

An A2C agent was implied as shown in Algorithm 5. The states are four days of historical climate data (see Table 6.2) 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.

Algorithm 4 A2C training environment [28]

```
function ENV(step, action)
    season_state = List()
    next_WcTot = RF_WcTot.predict(current_state, action)
    season_state.append(current_state, action)
    if time_passed = end_of_season
        done=True
        Y=BLSTM_yield.predict(season_state)
        calculate reward from Equation (6.2)
    else
        done=False
        reward=0
    end if
    return next_SWTD, net_return, done
end function
```

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 5 DQN 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
        random = random_uniform(0, 1)
        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==0 then then
            calculate TD error
            update critic by minimizing δ_t^2
            update actor by minimizing loss from Equation (6.13)
            current_state=new_state
```

6.3 Results and Discussions

6.3.1 BLSTM Models Evaluation

In this work, the trained BLSTM models of [26, 205] 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.

6.3.2 Evaluation of the A2C Agent

Figure 6.6 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.

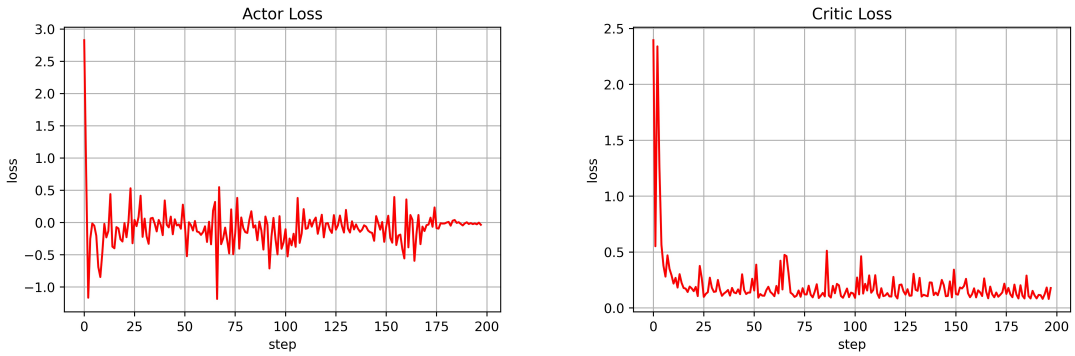


Figure 6.6: Actor–Critic loss during training.

The logarithm of Equation (6.2) was used to accelerate the convergence of the agents. Figure 6.7 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 6.6, 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.

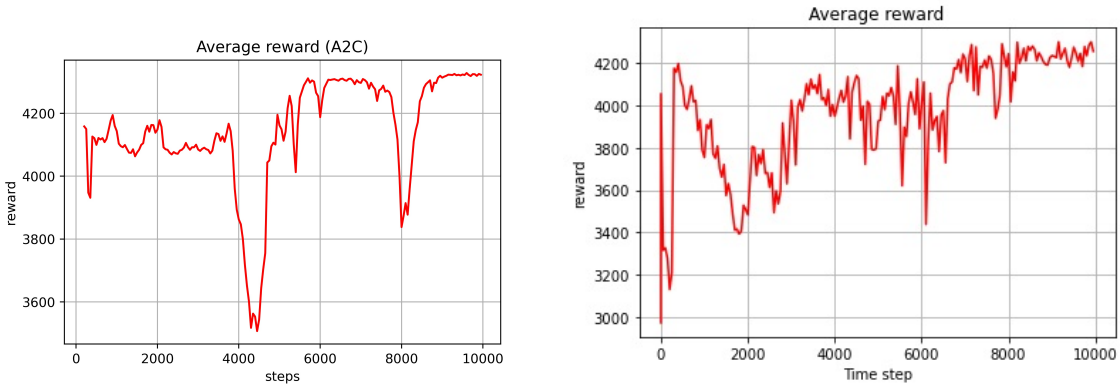


Figure 6.7: From left to right, the A2C and DQN rewards during training.

The agent was tested with the 2018 and 2019 datasets. Figures 6.8 and 6.9 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 6.5 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 [28]. 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 [28] was to minimize water consumption without affecting the net yield of the farmer. The trained model increased the farmer's net yield by 11% and reduced water consumption by 20–30% compared to a fixed and 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 [28], namely to reduce water consumption without affecting the net return to the farmer. As can be seen from Table 6.5, 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.

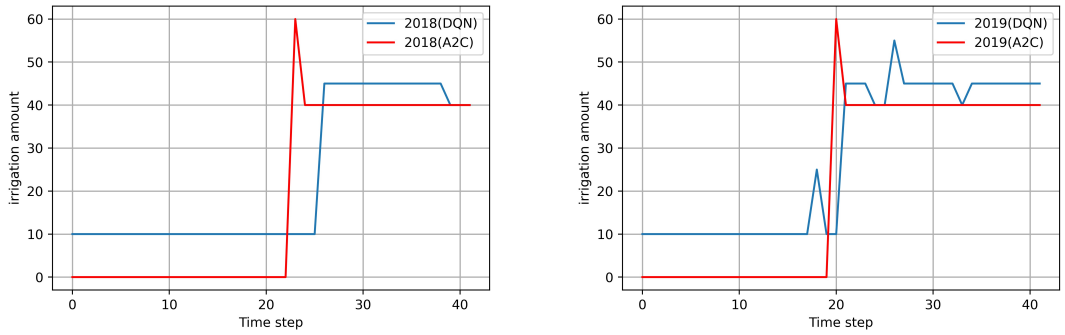


Figure 6.8: Comparison of the irrigation amount (mm/ha) of the trained DQN and A2C models. The time step starts from the beginning of the season to the end of the season every four days.

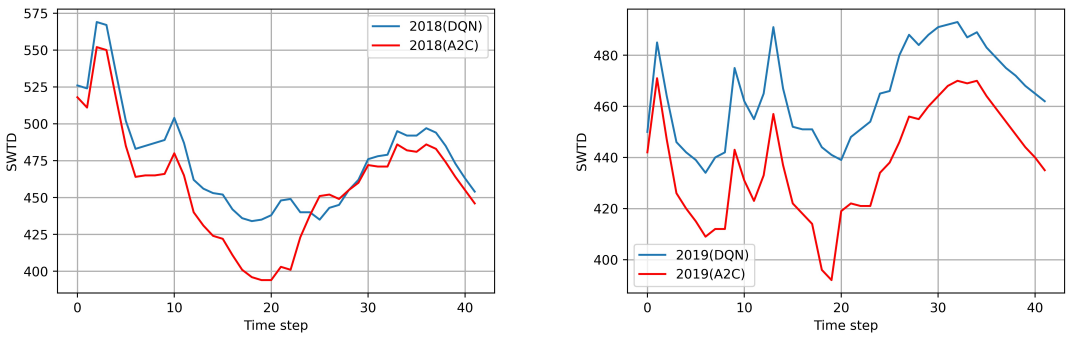


Figure 6.9: Comparison of SWTD (mm/ha) of the trained DQN and A2C models.

Table 6.5: 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 (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% ↑)

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 [28], 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 the 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 func-

tional 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.

Chapter 7

Conclusions

7.1 General Conclusions

Reduced use of agrochemicals, minimal use of pesticides, organic farming, appropriate crop rotations, small-scale fields, and preservation of natural areas between agroecosystems can contribute to more sustainable agriculture and promote biodiversity in agricultural systems, according to [133]. This can be achieved by combining new IoT technologies with artificial intelligence models and novel algorithms to develop decision-making systems that can be used to identify and manage species, improve ecosystem health and productivity without polluting activities, reduce cultivation costs, reduce pollution by determine when and how much fertilizer to use, to increase labor efficiency, to improve fuel efficiency, to complement the worker with the robot, and to save electricity costs by using solar-powered and mobile pumps. By providing farmers with more accurate warnings, improving their understanding of why phenomena occur in agricultural systems to help them manage them, and enabling better predictions based on better data and understanding of those data, AI increased the value of agriculture. This, in turn, encouraged the development of decision support and recommendation systems. Robotics and automated technologies are increasing farm productivity and health and largely eliminating the need for human decision-making [212].
Among the drawbacks of smart farming and the application of AI in agriculture is that farmers must acquire the necessary knowledge and skills to operate smart devices. However, many farmers lack these skills. In addition, unlimited or constant Internet connectivity may be required for success, and installation and maintenance of the devices is not cheap and requires an expert [212]. In addition to these disadvantages, it should be emphasised that these devices require electrical power, which is often not available in the agricultural field, thus requiring batteries that provide limited energy autonomy. Therefore, it is very important to create an AI-based decision support system that is user-friendly and does not require internet or special equipment on site, besides low power consumption.

7.2 Specific Conclusions

Chapter 2 discussed a recent study on the use of deep learning (DL) in agriculture and biodiversity, indicating the need for further research in this area. It was found that convolutional neural networks (CNN) are the most commonly used DL model in agriculture. The application of novel techniques, such as attention mechanisms, lightweight models, and single-stage detection models, can improve the performance of DL models, with even minor improvements in accuracy and runtime having a significant impact on results.

Chapter 3 examined the potential of recurrent neural networks (RNN), specifically long short-term memory (LSTM) and bidirectional LSTM (BLSTM), in modeling reference evapotranspiration (ETo) and soil water content (SWC). The BLSTM model showed promising results, achieving mean square error (MSE) values of 0.07 to 0.27 $(\mathrm{mm\ d^{-1}})^2$ for ETo and 0.014 to 0.056 $(\mathrm{m^3 m^{-3}})^2$ for SWC, with R^2 values of 0.96 to 0.98. Combining a CNN model with a BLSTM for feature extraction showed overfitting, indicating the potential challenges in increasing the trainable parameters in deep learning algorithms.
In Chapter 4, RNN models were used to predict tomato and potato yields based on climate data and irrigation amounts. The BLSTM models outperformed simple LSTM, GRU, and BGRU models and achieved R^2 values between 0.97 and 0.99 in the test group. These models have shown the ability to automatically extract features from agricultural data and capture the relationship between climate and irrigation to predict future yields. A sufficient amount of clean data is critical for training these models, as more data can lead to better predictions.
Chapter 5 focused on the potential of reinforcement learning, specifically Deep Q-Network (DQN), for irrigation scheduling of a tomato field. The trained DQN model effectively prevented crop stress and water waste by adjusting irrigation rates based on seasonal rainfall and climate changes. The model outperformed irrigation methods based on fixed rates and thresholds and achieved higher net yields with lower water use (18%-30%).
In chapter 6, the advantage of on-policy models, such as the Advantage-Actor-Critic model (A2C), over off-policy models, such as the DQN, in irrigation scheduling. The objective of the A2C model was to achieve high crop productivity with efficient use of water resources. Results showed that the A2C model reduced water use by up to 20% compared to the DQN model, albeit with a slight decrease in productivity. This finding suggests that on-policy models are more suitable for regions with limited water resources because they prioritize water conservation while adapting irrigation strategies to changing climatic conditions.
In summary, the models trained in this study demonstrate their potential to optimize irrigation in agricultural areas and reduce water use. These models, such as the BLSTM and reinforcement learning-based approaches, provide valuable insights into determining optimal irrigation strategies based on climate data, crop yield forecasts, and water availability. However, before these models can be used in real-world scenarios, it is essential that they be adapted and re-trained with field-specific data.
The adaptation and retraining process involve fine-tuning the models using data collected directly from the agricultural fields where they will be implemented. This step ensures that the models take into account the unique characteristics of each region, including soil properties, crop varieties, and local climate patterns. Incorporating field data into the training process allows the models to learn and adapt to specific conditions, resulting in more accurate predictions and tailored irrigation recommendations.

7.3 Future work

The models in this thesis were trained with simulated data, but the model becomes more reliable when trained with both simulations and real data sets. For a real-world application,

real data should be collected to re-train the models. In addition, the evaluation of net return is based on water prices and yields in those years. Also, to use the models for other cultivars or other weather conditions, the data should be collected and the model trained with the new data. Training the models for such conditions will increase their reliability and accuracy and lead to more general predictions that can then be applied to other environmental, soil, and cultural conditions. In this way, the models will be useful to a wider range of farmers.
The results of this work will be incorporated into the BioD'Agro project in the future. (`https://biodagro.com`. The main goal of the BioD'Agro project is to develop an information system for the remote monitoring of a vineyard and to assist producers in making decisions that promote agrobiodiversity.
In order to accomplish this, BioD'Agro will combine Earth observation imagery with data from in situ sensors that integrate parameters of the vine (health and water status), the environment (climate and soil), and functional biodiversity (flora, arthropods, and bats), and then use machine learning and artificial intelligence to create a web platform that will allow growers to monitor the water status of their vines or the presence of pests in real time while evaluating and deciding how to treat them. The information gathered from the IoT sensors in the vineyard will therefore be utilized to train the model created in this study and enhance irrigation scheduling and using robots in the field.

Bibliography

[1] M. Clark and D. Tilman, "Comparative analysis of environmental impacts of agricultural production systems, agricultural input efficiency, and food choice," *Environmental Research Letters*, vol. 12, no. 6, p. 064016, jun 2017. [Online]. Available: https://doi.org/10.1088/1748-9326/aa6cd5 1

[2] A. Baiano, "An overview on sustainability in the wine production chain," *Beverages*, vol. 7, no. 1, 2021. [Online]. Available: https://www.mdpi.com/2306-5710/7/1/15 1

[3] J. Rodrigo-Comino, "Five decades of soil erosion research in "terroir". the state-of-the-art," *Earth-Science Reviews*, vol. 179, pp. 436–447, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0012825217305433 1

[4] D. o. E. United Nations and P. D. . Social Affairs, "World urbanization prospects: The 2018 revision (ST/ESA/SER.A/420)," 2019, new York: United Nations. 1, 46

[5] H. Sundmaeker, C. N. Verdouw, J. Wolfert, and L. P. Freire, "Internet of food and farm 2020," in *Digitising the Industry*, O. Vermesan and P. Friess, Eds. River Publishers, 2016, pp. 129–150. 1, 6, 46, 69, 91, 92

[6] M. Jastrzębska, M. Kostrzewska, and A. Saeid, "Chapter 2 - sustainable agriculture: A challenge for the future," in *Smart Agrochemicals for Sustainable Agriculture*, K. Chojnacka and A. Saeid, Eds. Academic Press, 2022, pp. 29–56. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128170366000029 1

[7] J. W. Hopmans, A. Qureshi, I. Kisekka, R. Munns, S. Grattan, P. Rengasamy, A. Ben-Gal, S. Assouline, M. Javaux, P. Minhas, P. Raats, T. Skaggs, G. Wang, Q. De Jong van Lier, H. Jiao, R. Lavado, N. Lazarovitch, B. Li, and E. Taleisnik, "Chapter one - critical knowledge gaps and research priorities in global soil salinity," ser. Advances in Agronomy, D. L. Sparks, Ed. Academic Press, 2021, vol. 169, pp. 1–191. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065211321000341 1

[8] L. Loures, A. Chamizo, P. Ferreira, A. Loures, R. Castanho, and T. Panagopoulos, "Assessing the effectiveness of precision agriculture management systems in mediterranean small farms," *Sustainability*, vol. 12, no. 9, 2020. [Online]. Available: https://www.mdpi.com/2071-1050/12/9/3765 1

[9] K. Nalla, S. V. Pothabathula, and S. Kumar, "Chapter 21 - applications of computational methods in plant pathology," in *Natural Remedies for Pest, Disease and Weed Control*, C. Egbuna and B. Sawicka, Eds. Academic Press, 2020, pp. 243–250. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B978012819304400021X 1

[10] V. Lohchab, M. Kumar, G. Suryan, V. Gautam, and R. K. Das, "A review of iot based smart farm monitoring," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2018, pp. 1620–1625. 1, 6, 92, 111

[11] J. Doshi, T. Patel, and S. kumar Bharti, "Smart farming using iot, a solution for optimally monitoring farming conditions," *Procedia Computer Science*, vol. 160, pp. 746–751, 2019, the 10th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2019) / The 9th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2019) / Affiliated Workshops. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919317168 1, 6, 92, 111

[12] G. Nguyen, S. Dlugolinsky, M. Bobak, V. Tran, A. L. Garcia, I. Heredia, P. Malik, and L. Hluchy, "Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 77–124, JUN 2019. 1, 2, 6, 8, 9, 12, 17, 69, 72, 82, 97, 111, 112, 118

[13] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: A new paradigm to machine learning," *Archives of Computational Methods in Engineering*, pp. 1–22, 2019. 1, 6, 9, 43

[14] Y. Lu, "Artificial intelligence: a survey on evolution, models, applications and future trends," *Journal of Management Analytics*, vol. 6, no. 1, pp. 1–29, 2019. [Online]. Available: https://doi.org/10.1080/23270012.2019.1570365 1

[15] E. Commission, J. R. Centre, B. Delipetrev, C. Tsinaraki, and U. Kostić, *AI watch, historical evolution of artificial intelligence : analysis of the three main paradigm shifts in AI*. Publications Office, 2020. 1

[16] K. Liakos, P. B. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors (Basel)*, vol. 18, pp. 61 – 69, 2018. 2, 6, 97, 111

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. 2, 73

[18] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *CoRR*, vol. abs/1312.6229, 2013. 2, 6, 20, 52, 73, 97

[19] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70 – 90, 2018. 2, 73, 92, 111

[20] A. Ramcharan, P. McCloskey, K. Baranowski, N. Mbilinyi, L. Mrisho, M. Ndalahwa, J. Legg, and D. P. Hughes, "A Mobile-Based Deep Learning Model for Cassava Disease

Diagnosis," *FRONTIERS IN PLANT SCIENCE*, vol. 10, MAR 20 2019. 2, 7, 15, 22, 24, 43

[21] K. Alibabaei, P. D. Gaspar, T. M. Lima, R. M. Campos, I. Girão, J. Monteiro, and C. M. Lopes, "A review of the challenges of using deep learning algorithms to support decision-making in agricultural activities," *Remote Sensing*, vol. 14, no. 3, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/3/638 2, 5, 111

[22] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015. 2, 47, 56, 82

[23] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support vector regression machines," in *Proceedings of the 9th International Conference on Neural Information Processing Systems*, ser. NIPS'96. Cambridge, MA, USA: MIT Press, 1996, p. 155–161. 3, 47, 61

[24] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. 3, 47, 61

[25] P. J. Huber, "Robust estimation of a location parameter," *Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, mar 1964. 3, 47

[26] K. Alibabaei, P. D. Gaspar, and T. M. Lima, "Modeling soil water content and reference evapotranspiration from climate data using deep learning method," *Applied Sciences*, vol. 11, no. 11, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/11/5029 3, 35, 36, 43, 45, 92, 103, 105, 123, 125

[27] ——, "Crop yield estimation using deep learning based on climate big data and irrigation scheduling," *Energies*, vol. 14, no. 11, 2021. xiii, xvi, 3, 15, 68, 93, 95, 96, 97, 102, 105

[28] K. Alibabaei, P. D. Gaspar, E. Assunção, S. Alirezazadeh, and T. M. Lima, "Irrigation optimization with a deep reinforcement learning model: Case study on a site in portugal," *Agricultural Water Management*, vol. 263, p. 107480, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378377422000270 xii, xiv, 3, 13, 37, 38, 43, 91, 112, 113, 114, 116, 118, 123, 124, 125, 127, 128

[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Harley, T. P. Lillicrap, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, p. 1928–1937. 3, 112, 121, 122, 123

[30] K. Alibabaei, P. D. Gaspar, E. Assunção, S. Alirezazadeh, T. M. Lima, V. N. G. J. Soares, and J. M. L. P. Caldeira, "Comparison of on-policy deep reinforcement learning a2c with off-policy dqn in irrigation optimization: A case study at a site in portugal," *Computers*, vol. 11, no. 7, 2022. [Online]. Available: https://www.mdpi.com/2073-431X/11/7/104 4, 110

[31] R. Oppermann and M. Paracchini, *HNV farming - central to European cultural landscapes and biodiversity. High Nature Value Farming in Europe: 35 European Countries—Experiences and Perspectives*, R. Oppermann, G. Beaufoy, , and G. Jones., Eds. Ubstadt-Weiher, Germany: Verlag Regionalkultur, 2012. 6

[32] M. W. Rosegrant, C. Ringler, and T. Zhu, "Water for agriculture: Maintaining food security under growing scarcity," *Annual Review of Environment and Resources*, vol. 34, no. 1, pp. 205–222, 2009. [Online]. Available: https://doi.org/10.1146/annurev.environ.030308.090351 6, 91, 92

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, May 2017. [Online]. Available: https://doi.org/10.1145/3065386 6, 9

[34] J. Liu, J. Xiang, Y. Jin, R. Liu, J. Yan, and L. Wang, "Boost precision agriculture with unmanned aerial vehicle remote sensing and edge intelligence: A survey," *Remote Sensing*, vol. 13, no. 21, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/21/4387 7, 10, 14

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, JUNE 2015, pp. 1–9. 9

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015. 9

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. 9

[38] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, cite arxiv:1704.04861. [Online]. Available: http://arxiv.org/abs/1704.04861 9

[39] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020. 9

[40] A. S. Aguiar, N. N. Monteiro, F. N. d. Santos, E. J. Solteiro Pires, D. Silva, A. J. Sousa, and J. Boaventura-Cunha, "Bringing semantics to the vineyard: An approach on deep learning-based vine trunk detection," *Agriculture*, vol. 11, no. 2, 2021. [Online]. Available: https://www.mdpi.com/2077-0472/11/2/131 10, 24, 40, 41

[41] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. USA: IEEE Computer Society, 2014, pp. 580–587. 10

[42] R. Girshick, "Fast r-cnn," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*. USA: IEEE Computer Society, 2015, pp. 1440–1448. 10

[43] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 39, no. 06, pp. 1137–1149, JUNE 2017. 10

[44] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018. 10, 27

[45] W. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Computer Vision - ECCV 2016*. Cham: Springer International Publishing, 2016, pp. 21–37. 10

[46] Y. Wang, Q. Zhou, J. Liu, J. Xiong, G. Gao, X. Wu, and L. J. Latecki, "Lednet: A lightweight encoder-decoder network for real-time semantic segmentation," in *2019 IEEE International Conference on Image Processing (ICIP)*, 2019, pp. 1860–1864. 10

[47] E. T. Assunção, P. D. Gaspar, R. J. M. Mesquita, M. P. Simões, A. Ramos, H. Proença, and P. R. M. Inacio, "Peaches detection using a deep learning technique —;a contribution to yield estimation, resources management, and circular economy," *Climate*, vol. 10, no. 2, 2022. [Online]. Available: https://www.mdpi.com/2225-1154/10/2/11 xii, 11

[48] S. Haug and J. Ostermann, "A crop/weed field image dataset for the evaluation of computer vision based precision agriculture tasks," in *Computer Vision - ECCV 2014 Workshops*, L. Agapito, M. M. Bronstein, and C. Rother, Eds. Cham: Springer International Publishing, 2015, pp. 105–116. xii, 12

[49] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440. 10

[50] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2017, cite arxiv:1703.06870Comment: open source; appendix on more results. [Online]. Available: http://arxiv.org/abs/1703.06870 10

[51] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," 2017. 10

[52] P. Lottes, J. Behley, A. Milioto, and C. Stachniss, "Fully Convolutional Networks With Sequential Information for Robust Crop and Weed Detection in Precision Farming," *IEEE ROBOTICS AND AUTOMATION LETTERS*, vol. 3, no. 4, pp. 2870–2877, OCT 2018. 10, 31, 32, 42, 43

[53] L. Ghiani, A. Sassu, F. Palumbo, L. Mercenaro, and F. Gambella, "In-field automatic detection of grape bunches under a totally uncontrolled environment," *Sensors*,

vol. 21, no. 11, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/11/3908 10, 24, 27, 42

[54] J. Patterson and A. Gibson, *Deep Learning: A Practitioner's Approach*. O'Reilly, Beijing, 2017. 12, 16, 51, 52, 53, 58, 73, 82, 83, 86, 98, 124

[55] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. 12, 51, 73

[56] H. Kang and C. Chen, "Fruit detection, segmentation and 3d visualisation of environments in apple orchards," *Computers and Electronics in Agriculture*, vol. 171, p. 105302, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919323889 13, 26, 29, 42, 43

[57] J. Tang, D. Wang, Z. Zhang, L. He, J. Xin, and Y. Xu, "Weed identification based on k-means feature learning combined with convolutional neural network," *Computers and Electronics in Agriculture*, vol. 135, pp. 63 – 70, 2017. 13, 42

[58] A. D. S. Ferreira, D. M. Freitas, G. G. da Silva, H. Pistori, and M. T. Folhes, "Unsupervised deep learning and semi-automatic data labeling in weed discrimination," *Computers and Electronics in Agriculture*, vol. 165, p. 104963, 2019. 13, 30, 32, 42

[59] E. Min, X. Guo, Q. Liu, G. Zhang, J. Cui, and J. Long, "A survey of clustering with deep learning: From the perspective of network architecture," *IEEE Access*, vol. 6, pp. 39 501–39 514, 2018. 13, 111, 112

[60] F. Bu and X. Wang, "A smart agriculture iot system based on deep reinforcement learning," *Future Generation Computer Systems*, vol. 99, pp. 500 – 507, 2019. 13, 43, 112, 120

[61] M. Chen, Y. Cui, X. Wang, H. Xie, F. Liu, T. Luo, S. Zheng, and Y. Luo, "A reinforcement learning approach to irrigation decision-making for rice using weather forecasts," *Agricultural Water Management*, vol. 250, p. 106838, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378377421001037 13, 37, 38, 43, 112

[62] D. Loukatos, C. Templalexis, D. Lentzou, G. Xanthopoulos, and K. G. Arvanitis, "Enhancing a flexible robotic spraying platform for distant plant inspection via high-quality thermal imagery data," *Computers and Electronics in Agriculture*, vol. 190, p. 106462, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169921004798 13, 14, 15

[63] B. Rey, N. Aleixos, S. Cubero, and J. Blasco, "Xf-rovim. a field robot to detect olive trees infected by xylella fastidiosa using proximal sensing," *Remote Sensing*, vol. 11, no. 3, 2019. [Online]. Available: https://www.mdpi.com/2072-4292/11/3/221 13, 14

[64] M. Rußwurm and M. Körner, "Multi-temporal land cover classification with sequential recurrent encoders," *ISPRS International Journal of Geo-Information*, vol. 7, no. 4, 2018. 13

[65] J. W. Ma, C. H. Nguyen, K. Lee, and J. Heo, "Regional-scale rice-yield estimation using stacked auto-encoder with climatic and modis data: a case study of south korea," *International Journal of Remote Sensing*, vol. 40, no. 1, pp. 51–71, 2019. 13, 71

[66] S. Bargoti and J. Underwood, "Deep fruit detection in orchards," 05 2017, pp. 3626–3633. 13

[67] M. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep learning for tomato diseases: Classification and symptoms visualization," *Applied Artificial Intelligence*, vol. 31, no. 4, pp. 299–315, 2017. 13, 15

[68] P. Kumar and S. R. N. Reddy, "Wireless sensor networks: a review of motes, wireless technologies, routing algorithms and static deployment strategies for agriculture applications," *CSI Transactions on ICT*, vol. 8, no. 3, pp. 331–345, Sep 2020. [Online]. Available: https://doi.org/10.1007/s40012-020-00289-1 14

[69] R. P. Sishodia, R. L. Ray, and S. K. Singh, "Applications of remote sensing in precision agriculture: A review," *Remote Sensing*, vol. 12, no. 19, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/19/3136 14

[70] J. Rudd, G. Roberson, and J. Classen, "Application of satellite, unmanned aircraft system, and ground-based sensor data for precision agriculture: a review," 01 2017. 14, 15

[71] J. M. Terres, J. Delince, M. van de Steene, and A. Hawkins, "The use of remote sensing and gis capabilities to support the reform of the common agricultural policy of the european community," *Remote Sensing Reviews*, vol. 12, no. 1-2, pp. 53–60, 1995. [Online]. Available: https://doi.org/10.1080/02757259509532275 14

[72] M. Ouhami, A. Hafiane, Y. Es-Saady, M. El Hajji, and R. Canals, "Computer vision, iot and data fusion for crop disease detection using machine learning: A survey and ongoing research," *Remote Sensing*, vol. 13, no. 13, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/13/2486 14

[73] D. Li, X. Sun, H. Elkhouchlaa, Y. Jia, Z. Yao, P. Lin, J. Li, and H. Lu, "Fast detection and location of longan fruits using uav images," *Computers and Electronics in Agriculture*, vol. 190, p. 106465, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169921004828 15, 37, 41, 43

[74] M. Rahnemoonfar and C. Sheppard, "Deep count: Fruit counting based on deep simulated learning," *Sensors*, vol. 17, no. 4, 2017. 15, 71

[75] J. G. A. Barbedo, "Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification," *Computers and Electronics in Agriculture*, vol. 153, pp. 46 – 53, 2018. 15, 19, 21, 41, 42

[76] Q. Yang, L. Shi, J. Han, Y. Zha, and P. Zhu, "Deep convolutional neural networks for rice grain yield estimation at the ripening stage using UAV-based remotely sensed images," *Field Crops Research*, vol. 235, pp. 142 – 153, 2019. 15, 70

[77] J. S. Pan and Q. Yang, "A survey on transfer learning. ieee trans," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, 2010. 15

[78] M. M. Ghazi, B. Yanikoglu, and E. Aptoula, "Plant identification using deep neural networks via optimization of transfer learning parameters," *Neurocomputing*, vol. 235, pp. 228 – 235, 2017. 15

[79] S. Dreyfus, "The numerical solution of variational problems," *Journal of Mathematical Analysis and Applications*, vol. 5, no. 1, pp. 30 – 45, 1962. 17, 81

[80] X. Zhang, Y. Qiao, F. Meng, C. Fan, and M. Zhang, "Identification of Maize Leaf Diseases Using Improved Deep Convolutional Neural Networks," *IEEE ACCESS*, vol. 6, pp. 30 370–30 377, 2018. 17, 43

[81] M. Kerkech, A. Hafiane, and R. Canals, "Vine disease detection in uav multispectral images using optimized image registration and deep learning segmentation approach," *Computers and Electronics in Agriculture*, vol. 174, p. 105446, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016816991932558X 18, 21, 42

[82] ——, "Deep leaning approach with colorimetric spaces and vegetation indices for vine diseases detection in uav images," *Computers and Electronics in Agriculture*, vol. 155, pp. 237–243, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169918310044 19, 21, 42

[83] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311 – 318, 2018. 20, 21, 41, 42

[84] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," 2014. 20

[85] P. Jiang, Y. Chen, B. Liu, D. He, and C. Liang, "Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks," *IEEE Access*, vol. 7, pp. 59 069–59 080, 2019. 20, 21, 42

[86] R. Karthik, M. Hariharan, S. Anand, P. Mathikshara, A. Johnson, and R. Menaka, "Attention embedded residual cnn for disease detection in tomato leaves," *Applied Soft Computing*, vol. 86, p. 105933, 2020. 21, 24

[87] B. Liu, Y. Zhang, D. He, and Y. Li, "Identification of apple leaf diseases based on deep convolutional neural networks," *Symmetry*, vol. 10, no. 1, 2018. 22, 24, 42, 43

[88] A. Picon, M. Seitz, A. Alvarez-Gila, P. Mohnke, A. Ortiz-Barredo, and J. Echazarra, "Crop conditional convolutional neural networks for massive multi-crop plant disease classification over cell phone acquired images taken on real field conditions," *Computers and Electronics in Agriculture*, vol. 167, p. 105093, 2019. 22, 24, 42

[89] J. Chen, D. Zhang, A. Zeb, and Y. A. Nanehkaran, "Identification of rice plant diseases using lightweight attention networks," *Expert Systems with Applications*, vol. 169, p. 114514, 2021. 23, 24, 43

[90] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. P. Hughes, "Deep Learning for Image-Based Cassava Disease Detection," *FRONTIERS IN PLANT SCIENCE*, vol. 8, OCT 27 2017. 24

[91] D. L. Silver and T. Monga, "In vino veritas: Estimating vineyard grape yield from images using deep learning," in *Advances in Artificial Intelligence*, M.-J. Meurs and F. Rudzicz, Eds. Cham: Springer International Publishing, 2019, pp. 212–224. 23, 27, 42

[92] A. S. Aguiar, S. A. Magalhães, F. N. dos Santos, L. Castro, T. Pinho, J. Valente, R. Martins, and J. Boaventura-Cunha, "Grape bunch detection at different growth stages using deep learning quantized models," *Agronomy*, vol. 11, no. 9, 2021. [Online]. Available: https://www.mdpi.com/2073-4395/11/9/1890 24, 27, 41, 43

[93] K. P. Seng, L.-M. Ang, L. M. Schmidtke, and S. Y. Rogiers, "Computer vision and machine learning for viticulture technology," *IEEE Access*, vol. 6, pp. 67 494–67 510, 2018. 25, 27

[94] A. Milella, R. Marani, A. Petitti, and G. Reina, "In-field high throughput grapevine phenotyping with a consumer-grade depth camera," *Computers and Electronics in Agriculture*, vol. 156, pp. 293–306, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169918307580 25, 27

[95] T. T. Santos, L. L. de Souza, A. A. dos Santos, and S. Avila, "Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association," *Computers and Electronics in Agriculture*, vol. 170, p. 105247, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919315765 25, 27, 42

[96] F. Palacios, G. Bueno, J. Salido, M. P. Diago, I. Hernández, and J. Tardaguila, "Automated grapevine flower detection and quantification method based on computer vision and deep learning from on-the-go imaging using a mobile sensing platform under field conditions," *Computers and Electronics in Agriculture*, vol. 178, p. 105796, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920315428 26, 27

[97] B. Millán, A. Aquino, M. P. Diago, and J. Tardáguila, "Image analysis-based modelling for flower number estimation in grapevine." *Journal of the science of food and agriculture*, vol. 97 3, pp. 784–792, 2017. 26, 27

[98] A. Koirala, K. B. Walsh, Z. Wang, and C. McCarthy, "Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of 'mangoyolo'," *Precision Agriculture*, vol. 20, no. 6, pp. 1107–1135, 2019. 27, 29, 42, 70

[99] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, JULY 2017, pp. 6517–6525. 27

[100] Q. Liang, W. Zhu, J. Long, Y. Wang, W. Sun, and W. Wu, "A real-time detection framework for on-tree mango based on SSD network," in *Intelligent Robotics and Applications*. Cham: Springer International Publishing, 2018, pp. 423–436. 28, 29, 42, 70

[101] S. Bargoti and J. P. Underwood, "Image segmentation for fruit detection and yield estimation in apple orchards," *Journal of Field Robotics*, vol. 34, no. 6, pp. 1039–1060, 2017. 28, 69

[102] Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, "Apple detection during different growth stages in orchards using the improved YOLO-V3 model," *Computers and Electronics in Agriculture*, vol. 157, pp. 417–426, FEB 2019. 28, 29, 42, 70

[103] Z. Zhou, Z. Song, L. Fu, F. Gao, R. Li, and Y. Cui, "Real-time kiwifruit detection in orchard using deep learning on android™ smartphones for yield estimation," *Computers and Electronics in Agriculture*, vol. 179, p. 105856, 2020. 28, 29, 42, 43, 71

[104] M. D. Bah, A. Hafiane, and R. Canals, "Deep learning with unsupervised data labeling for weed detection in line crops in uav images," *Remote Sensing*, vol. 10, no. 11- 1690, 2018. 29, 32, 42

[105] A. D. S. Ferreira, D. M. Freitas, G. G. da Silva, H. Pistori, and M. T. Folhes, "Weed detection in soybean crops using convnets," *Computers and Electronics in Agriculture*, vol. 143, pp. 314 – 324, 2017. 30, 32

[106] A. Olsen, D. A. Konovalov, B. Philippa, P. Ridd, J. J. J. C. Wood, W. Banks, B. Girgenti, O. Kenny, J. Whinney, B. Calvert, M. RahimiAzghadi, and R. D. White, "Deepweeds: A multiclass weed species image dataset for deep learning." *Scientific ReportsSP*, vol. 9, no. 2058, 2019. 30, 32

[107] A. Milioto, P. Lottes, and C. Stachniss, "Real-time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs," in *2018 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA)*, ser. IEEE International Conference on Robotics and Automation ICRA, 2018, pp. 2229–2235. 30, 32, 42, 43

[108] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017. 30

[109] A. Wang, Y. Xu, X. Wei, and B. Cui, "Semantic segmentation of crop and weed using an encoder-decoder network and image enhancement method under uncontrolled outdoor illumination," *IEEE Access*, vol. 8, pp. 81 724–81 734, 2020. 31, 32, 42

[110] N. Chebrolu, P. Lottes, A. Schaefer, W. Winterhalter, W. Burgard, and C. Stachniss, "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields," *The International Journal of Robotics Research*, 2017. 32

[111] M. Rußwurm and M. Körner, "Multi-Temporal Land Cover Classification with Long Short-Term Memory Neural Networks," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 42W1, pp. 551–558, May 2017. 31, 34

[112] S. H. Lee, C. S. Chan, S. J. Mayo, and P. Remagnino, "How deep learning extracts and learns leaf features for plant classification," *Pattern Recognition*, vol. 71, pp. 1 – 13, 2017. 31, 34, 42

[113] B. Ayhan, C. Kwan, B. Budavari, L. Kwan, Y. Lu, D. Perez, J. Li, D. Skarlatos, and M. Vlachos, "Vegetation detection using deep learning and conventional methods," *Remote Sensing*, vol. 12, no. 15, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/15/2502 32, 34, 42

[114] D. Skarlatos and M. Vlachos, "Vegetation removal from uav derived dsms, using combination of rgb and nir imagery," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-2, pp. 255–262, 2018. [Online]. Available: https://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/IV-2/255/2018/ 32

[115] S. Bhusal, U. Bhattarai, and M. Karkee, "Improving pest bird detection in a vineyard environment using super-resolution and deep learning," *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 18–23, 2019, 6th IFAC Conference on Sensing, Control and Automation Technologies for Agriculture AGRICONTROL 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896319323924 33, 34

[116] J. Yamanaka, S. Kuwashima, and T. Kurita, "Fast and accurate image super resolution by deep cnn with skip connection and network in network," in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham: Springer International Publishing, 2017, pp. 217–225. 33

[117] O. Mac Aodha, R. Gibb, K. E. Barlow, E. Browning, M. Firman, R. Freeman, B. Harder, L. Kinsey, G. R. Mead, S. E. Newson, I. Pandourski, S. Parsons, J. Russ, A. Szodoray-Paradi, F. Szodoray-Paradi, E. Tilova, M. Girolami, G. Brostow, and K. E. Jones, "Bat detective—deep learning tools for bat acoustic signal detection," *PLOS Computational Biology*, vol. 14, no. 3, pp. 1–19, 03 2018. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1005995 33, 34, 42

[118] B. Ramalingam, R. E. Mohan, S. Pookkuttath, B. F. Gómez, C. S. C. Sairam Borusu, T. Wee Teng, and Y. K. Tamilselvam, "Remote insects trap monitoring system using deep learning framework and iot," *Sensors*, vol. 20, no. 18, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5280 33, 34, 43

[119] C. Verdouw, S. Wolfert, and B. Tekinerdogan, "Internet of things in agriculture," *CAB Reviews*, vol. 11, pp. 1–12, 12 2016. 33, 111

[120] C. Li, Y. Zhang, and X. Ren, "Modeling hourly soil temperature using deep bilstm neural network," *Algorithms*, vol. 13, no. 7, 2020. [Online]. Available: https://www.mdpi.com/1999-4893/13/7/173 34, 36, 42

[121] F. Yu, H. Hao, and Q. Li, "An ensemble 3d convolutional neural network for spatiotemporal soil temperature forecasting," *Sustainability*, vol. 13, no. 16, 2021. [Online]. Available: https://www.mdpi.com/2071-1050/13/16/9174 35, 36

[122] X. SHI, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. WOO, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28. Curran Associates, Inc., 2015. [Online]. Available: https://proceedings.neurips.cc/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf 35

[123] M. K. Saggi and S. Jain, "Reference evapotranspiration estimation and modeling of the punjab northern india using deep learning," *Computers and Electronics in Agriculture*, vol. 156, pp. 387 – 398, 2019. 36, 38, 47, 92, 111

[124] J. Zhang, Y. Zhu, X. Zhang, M. Ye, and J. Yang, "Developing a long short-term memory (LSTM) based model for predicting water table depth in agricultural areas," *Journal of Hydrology*, vol. 561, pp. 918 – 929, 2018. 36, 38, 47, 56, 80, 92

[125] K. Loggenberg, A. Strever, B. Greyling, and N. Poona, "Modelling water stress in a shiraz vineyard using hyperspectral imaging and machine learning," *Remote Sensing*, vol. 10, no. 2, 2018. [Online]. Available: https://www.mdpi.com/2072-4292/10/2/202 36, 38

[126] D. Aghi, V. Mazzia, and M. Chiaberge, "Local motion planner for autonomous navigation in vineyards with a rgb-d camera-based algorithm and deep learning synergy," *Machines*, vol. 8, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2075-1702/8/2/27 38, 41

[127] E. Badeka, E. Vrochidou, G. A. Papakostas, T. Pachidis, and V. G. Kaburlasos, "Harvest crate detection for grapes harvesting robot based on yolov3 model," in *2020 Fourth International Conference On Intelligent Computing in Data Sciences (ICDS)*, 2020, pp. 1–5. 39, 41

[128] Y. Majeed, M. Karkee, Q. Zhang, L. Fu, and M. D. Whiting, "Determining grapevine cordon shape for automated green shoot thinning using semantic segmentation-based

deep learning networks," *Computers and Electronics in Agriculture*, vol. 171, p. 105308, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919324391 39, 41

[129] A. S. Pinto de Aguiar, F. B. Neves dos Santos, L. C. Feliz dos Santos, V. M. de Jesus Filipe, and A. J. Miranda de Sousa, "Vineyard trunk detection using deep learning – an experimental device benchmark," *Computers and Electronics in Agriculture*, vol. 175, p. 105535, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920304555 40, 41, 43

[130] J. Mockus, *Bayesian approach to global optimization*, ser. Mathematics and its Applications (Soviet Series). Kluwer Academic Publishers Group, Dordrecht, 1989, vol. 37, theory and applications, With a 5.25-inch IBM PC floppy disk. 43, 54

[131] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, cite arxiv:1611.01578. [Online]. Available: http://arxiv.org/abs/1611.01578 43

[132] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, and H. Adam, "Searching for mobilenetv3," 2019. 43

[133] W. Nentwig, T. Frank, and C. Lethmayer, "Sown weed strips: Artificial ecological compensation areas as an important tool in conservation biological control," *Conservation Biological Control*, pp. 133–153, 01 1998. 43, 130

[134] FAO. (2002) World agriculture 2030: Main findings. [Online]. Available: http://www.fao.org/english/newsroom/news/2002/7833-en.html 46, 91, 110

[135] A. Laaboudi, B. Mouhouche, and B. Draoui, "Neural network approach to reference evapotranspiration modeling from limited climatic data in arid regions," *International Journal of Biometeorology*, vol. 56, p. 831–841, 2012. 46

[136] K. O. Achieng, "Modelling of soil moisture retention curve using machine learning techniques: Artificial and deep neural networks vs support vector regression models," *Computers & Geosciences*, vol. 133, p. 104320, 2019. 46

[137] R. G. Allen, L. S. Pereira, and M. S. D. Raes, *Crop evapotranspiration - Guidelines for computing crop water requirements FAO Irrigation and drainage paper 56*. FAO - Food and Agriculture Organization of the United Nations, Rome, 1998. 46, 48, 79, 96, 116

[138] A.-F. Jimenez, P.-F. Cardenas, A. Canales, F. Jimenez, and A. Portacio, "A survey on intelligent agents and multi-agents for irrigation scheduling," *Computers and Electronics in Agriculture*, vol. 176, p. 105474, 2020. 46

[139] A. D. Clulow, C. S. Everson, M. G. Mengistu, J. S. Price, A. Nickless, and G. P. W. Jewitt, "Extending periodic eddy covariance latent heat fluxes through tree sap-flow measurements to estimate long-term total evaporation in a peat swamp forest," *Hydrology and Earth System Sciences*, vol. 19, no. 5, pp. 2513–2534, May 2015. 46

[140] M. Kumar, N. S. Raghuwanshi, and R. Singh, "Artificial neural networks approach in evapotranspiration modeling: a review," *IRRIGATION SCIENCE*, vol. 29, no. 1, pp. 11–25, JAN 2011. 46

[141] F. Karandish and J. Šimůnek, "A comparison of numerical and machine-learning modeling of soil water content with limited input data," *Journal of Hydrology*, vol. 543, pp. 892 – 909, 2016. 46

[142] O. Adeyemi, I. Grove, S. Peets, Y. Domun, and T. Norton, "Dynamic Neural Network Modelling of Soil Moisture Content for Predictive Irrigation Scheduling," *SENSORS*, vol. 18, no. 10, OCT 2018. 46, 47, 92

[143] S. S. Yamac, C. Seker, and H. Negis, "Evaluation of machine learning methods to predict soil moisture constants with different combinations of soil input data for calcareous soils in a semi arid area," *AGRICULTURAL WATER MANAGEMENT*, vol. 234, MAY 2020. 46

[144] A. Fernandez-Lopez, D. Marin-Sanchez, G. Garcia-Mateos, A. Ruiz-Canales, M. Ferrandez-Villena-Garcia, and J. M. Molina-Martinez, "A Machine Learning Method to Estimate Reference Evapotranspiration Using Soil Moisture Sensors," *APPLIED SCIENCES-BASEL*, vol. 10, no. 6, MAR 2020. 46

[145] D. Tseng, D. Wang, C. Chen, L. Miller, W. Song, J. Viers, S. Vougioukas, S. Carpin, J. A. Ojea, and K. Goldberg, "Towards automating precision irrigation: Deep learning to infer local soil moisture conditions from synthetic aerial agricultural images," in *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, 2018, pp. 284–291. 46, 92, 111

[146] X. Song, G. Zhang, F. Liu, D. Li, Y. Zhao, and J. Yang, "Modeling spatio-temporal distribution of soil moisture by deep learning-based cellular automata model," *Journal of Arid Land*, vol. 8, no. 5, pp. 734–748, Oct 2016. 46, 92, 111

[147] S. Adamala, "Temperature based generalized wavelet-neural network models to estimate evapotranspiration in india," *Information Processing in Agriculture*, vol. 5, no. 1, pp. 149 – 155, 2018. 46, 92

[148] P. de Oliveira e Lucas, M. A. Alves, P. C. de Lima e Silva, and F. G. Guimarães, "Reference evapotranspiration time series forecasting with ensemble of convolutional neural networks," *Computers and Electronics in Agriculture*, vol. 177, p. 105700, 2020. 47, 92, 111

[149] J. Muñoz Sabater, "Era5-land hourly data from 1981 to present. copernicus climate change service (c3s) climate data store (cds)," *Data of access: 2020, 10.24381/cds.e2161bac*, 2019. 48, 49

[150] *IFS Documentation CY45R1 - Part IV : Physical processes*, ser. IFS Documentation. ECMWF, 2018, no. 4. [Online]. Available: https://www.ecmwf.int/node/18714 49

[151] G. Balsamo, C. Albergel, A. Beljaars, S. Boussetta, E. Brun, H. Cloke, D. Dee, E. Dutra, J. Muñoz Sabater, F. Pappenberger, P. de Rosnay, T. Stockdale, and F. Vitart, "Era-interim/land: a global land surface reanalysis data set," *Hydrology and Earth System Sciences*, vol. 19, no. 1, pp. 389–407, 2015. [Online]. Available: https://hess.copernicus.org/articles/19/389/2015/ 49

[152] L. A. Richards, "Capillary conduction of liquids through porous mediums," *Physics*, vol. 1, no. 5, pp. 318–333, 1931. [Online]. Available: https://doi.org/10.1063/1.1745010 49

[153] D. C. Montgomery, C. L. Jennings, and M. Kulahci, *Introduction to Time Series Analysis and Forecasting*, ser. Wiley Series in Probability and Statistics. Wiley, 2011. 49, 80, 118

[154] J. Benesty, J. Chen, Y. Huang, and I. Cohen, *Pearson Correlation Coefficient*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–4. 49

[155] E. Kreyszig, H. Kreyszig, and E. J. Norminton, *Advanced Engineering Mathematics*, 10th ed. Hoboken, NJ: Wiley, 2011. 50

[156] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997. 52, 75, 102

[157] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, 2017. 53

[158] Y. Gal and Z. Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, pp. 1027–1035. 53, 76, 83

[159] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, ser. NIPS'11. Red Hook, NY, USA: Curran Associates Inc., 2011, p. 2546–2554. 54

[160] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions with application to active user modeling and hierarchical reinforcement learning," Tech. Rep., 2009. 54

[161] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," 2014. [Online]. Available: https://github.com/fmfn/BayesianOptimization 54, 57

[162] C. J. Willmott, S. G. Ackleson, R. E. Davis, J. J. Feddema, K. M. Klink, D. R. Legates, J. O'Donnell, and C. M. Rowe, "Statistics for the evaluation and comparison of models," *Journal of Geophysical Research: Oceans*, vol. 90, no. C5, pp. 8995–9005, 1985. 55, 80, 102, 120

[163] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. S. andvBenoit Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/ 56, 81, 123

[164] F. Chollet *et al.*, "Keras," https://keras.io, 2015. 56, 59, 81, 123

[165] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015. 56

[166] Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin, "Training and testing low-degree polynomial data mappings via linear svm," *Journal of Machine Learning Research*, vol. 11, no. 48, pp. 1471–1490, 2010. 62

[167] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. The MIT Press, 2018. 63, 89, 99, 100, 103, 120, 122

[168] M. J. Hayes and W. L. Decker, "Using noaa avhrr data to estimate maize production in the united states corn belt," *International Journal of Remote Sensing*, vol. 17, no. 16, pp. 3189–3200, 1996. 69

[169] H. Habaragamuwa, Y. Ogawa, T. Suzuki, T. Shiigi, M. Ono, and N. Kondo, "Detecting greenhouse strawberries (mature and immature), using deep convolutional neural network," *Engineering in Agriculture, Environment and Food*, vol. 11, no. 3, pp. 127 – 138, 2018. 69

[170] H. Kang and C. Chen, "Fast implementation of real-time fruit detection in apple orchards using deep learning," *Computers and Electronics in Agriculture*, vol. 168, p. 105108, 2020. 69

[171] M. Stein, S. Bargoti, and J. Underwood, "Image Based Mango Fruit Detection, Localisation and Yield Estimation Using Multiple View Geometry," *Sensors*, vol. 16, no. 11, NOV 2016. 70

[172] O. Apolo-Apolo, J. Martínez-Guanter, G. Egea, P. Raja, and M. Pérez-Ruiz, "Deep learning techniques for estimation of the yield and size of citrus fruits using a uav," *European Journal of Agronomy*, vol. 115, p. 126030, 2020. 70

[173] M. Maimaitijiang, V. Sagan, P. Sidike, S. Hartling, F. Esposito, and F. B. Fritschi, "Soybean yield prediction from uav using multimodal data fusion and deep learning," *Remote Sensing of Environment*, vol. 237, p. 111599, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0034425719306194 70

[174] Q. Yang, L. Shi, J. Han, Y. Zha, and P. Zhu, "Deep convolutional neural networks for rice grain yield estimation at the ripening stage using uav-based remotely sensed images," *Field Crops Research*, vol. 235, pp. 142–153, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S037842901831390X 70

[175] Y. Chen, W. S. Lee, H. Gan, N. Peres, C. Fraisse, Y. Zhang, and Y. He, "Strawberry yield prediction based on a deep neural network using high-resolution aerial orthoimages," *Remote Sensing*, vol. 11, no. 13, 2019. [Online]. Available: https://www.mdpi.com/2072-4292/11/13/1584 71

[176] J. Han, Z. Zhang, J. Cao, Y. Luo, L. Zhang, Z. Li, and J. Zhang, "Prediction of winter wheat yield based on multi-source data and machine learning in china," *Remote Sensing*, vol. 12, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/2/236 71

[177] N. Kim, K.-J. Ha, N.-W. Park, J. Cho, S. Hong, and Y.-W. Lee, "A comparison between major artificial intelligence models for crop yield prediction: Case study of the midwestern united states, 2006–2015," *ISPRS International Journal of Geo-Information*, vol. 8, no. 5, 2019. [Online]. Available: https://www.mdpi.com/2220-9964/8/5/240 71

[178] F. Abbas, H. Afzaal, A. A. Farooque, and S. Tang, "Crop yield prediction through proximal sensing and machine learning algorithms," *Agronomy*, vol. 10, no. 7, 2020. [Online]. Available: https://www.mdpi.com/2073-4395/10/7/1046 71

[179] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "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: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. 73

[180] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014. 75

[181] L. Prechelt, *Early Stopping — But When?* Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 53–67. 76, 84

[182] D. Raes, P. Steduto, T. C. Hsiao, and E. Fereres, "Aquacrop—the fao crop model to simulate yield response to water: Ii. main algorithms and software description," *Agronomy Journal*, vol. 101, no. 3, pp. 438–447, 2009. 77

[183] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014. 80, 96, 118

[184] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus," in *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011. 82

[185] F. Murtagh, "Multilayer perceptrons for classification and regression," *Neurocomputing*, vol. 2, no. 5, pp. 183–197, 1991. 86, 97, 98

[186] L. C. Rodrigues, "Water resources fee in portugali," 2016, led by the Institute for European Environmental Policy (https://ieep.eu/). 88, 104, 124

[187] H. Wang, C. Liu, and L. Zhang, "Water-saving agriculture in china: An overview," ser. Advances in Agronomy. Academic Press, 2002, vol. 75, pp. 135–171. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0065211302750049 91

[188] M. A. Clark and D. Tilman, "Comparative analysis of environmental impacts of agricultural production systems, agricultural input efficiency, and food choice," *Environmental Research Letters*, vol. 12, pp. 064 016–064 016, 2017. 91

[189] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. A. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, pp. 1–21, 2014. 92

[190] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 12 2013. 101

[191] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, and D. Silver, David andWierstra, "Continuous control with deep reinforcement learning." in *ICLR*, Y. Bengio and Y. LeCun, Eds., 2016. [Online]. Available: http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15 101

[192] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Mach. Learn.*, vol. 8, no. 3–4, p. 293–321, May 1992. 101

[193] Z. Zheng, H. Chen, and X. Luo, "Spatial granularity analysis on electricity consumption prediction using lstm recurrent neural network," *Energy Procedia*, vol. 158, pp. 2713–2718, 2019, innovative Solutions for Energy Transitions. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1876610219311051 106

[194] A. Muimba-Kankolongo, "Chapter 3 - factors important to crop production," in *Food Crop Production by Smallholder Farmers in Southern Africa*, A. Muimba-Kankolongo, Ed. Academic Press, 2018, pp. 15–21. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128143834000037 110

[195] S. Fahad, A. A. Bajwa, U. Nazir, S. A. Anjum, A. Farooq, A. Zohaib, S. Sadia, W. Nasim, S. Adkins, S. Saud, M. Z. Ihsan, H. Alharby, C. Wu, D. Wang, and J. Huang, "Crop production under drought and heat stress: Plant responses and management options," *Frontiers in Plant Science*, vol. 8, jun 2017. [Online]. Available: https://doi.org/10.3389/fpls.2017.01147 110

[196] A. K. Shakya, A. Ramola, A. Kandwal, and A. Vidyarthi, "Soil moisture sensor for agricultural applications inspired from state of art study of surfaces scattering models & semi-empirical soil moisture models," *Journal of the Saudi Society of Agricultural Sciences*, vol. 20, no. 8, pp. 559–572, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1658077X21000771 111

[197] A. Kumar Shakya and S. Singh, "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*, vol. 188, p. 110513, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0263224121013944 111

[198] E. A. Abioye, O. Hensel, T. J. Esau, O. Elijah, M. S. Z. Abidin, A. S. Ayobami, O. Yerima, and A. Nasirahmadi, "Precision irrigation management using machine learning and digital farming solutions," *AgriEngineering*, vol. 4, no. 1, pp. 70–103, 2022. [Online]. Available: https://www.mdpi.com/2624-7402/4/1/6 111

[199] H. Zia, A. Rehman, N. R. Harris, S. Fatima, and M. Khurram, "An experimental comparison of iot-based and traditional irrigation scheduling on a flood-irrigated subtropical lemon farm," *Sensors*, vol. 21, no. 12, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/12/4175 111

[200] A. A. M. Ahmed, R. C. Deo, N. Raj, A. Ghahramani, Q. Feng, Z. Yin, and L. Yang, "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 Sensing*, vol. 13, no. 4, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/4/554 111

[201] H. Adab, R. Morbidelli, C. Saltalippi, M. Moradian, and G. A. F. Ghalhari, "Machine learning to estimate surface soil moisture from remote sensing data," *Water*, vol. 12, no. 11, 2020. [Online]. Available: https://www.mdpi.com/2073-4441/12/11/3223 111

[202] A. F. Jimenez, B. Ortiz, V, L. Bondesan, G. Morata, and D. Damianidis, "Evaluation of two recurrent neural network methods for prediction of irrigation rate and timing," *Transactions of the ASABE*, vol. 63, no. 5, pp. 1327–1348, 2020. 112

[203] G. Hoogenboom, , C. H. Porter, K. J. Boote, V. Shelia, P. W. Wilkens, U. Singh, J. W. White, S. Asseng, J. I. Lizaso, L. P. Moreno, W. Pavan, R. Ogoshi, L. A. Hunt, G. Y. Tsuji, and J. W. Jones, "The DSSAT crop modeling ecosystem," in *Advances in crop modelling for a sustainable agriculture*. Burleigh Dodds

Science Publishing, December 2019, pp. 173–216. [Online]. Available: https://doi.org/10.19103/as.2019.0061.10 113, 116

[204] G. Hoogenboom, C. Porter, V. Shelia, K. Boote, U. Singh, J. White, L. Hunt, R. Ogoshi, J. Lizaso, J. Koo, S. Asseng, A. Singels, L. Moreno, and J. Jones, "Decision support system for agrotechnology transfer (dssat) version 4.7.5," 2019, dSSAT Foundation, Gainesville, Florida, USA. [Online]. Available: https://DSSAT.net 113, 116

[205] K. Alibabaei, P. D. Gaspar, and T. M. Lima, "Crop yield estimation using deep learning based on climate big data and irrigation scheduling," *Energies*, vol. 14, no. 11, 2021. [Online]. Available: https://www.mdpi.com/1996-1073/14/11/3004 xiv, 116, 117, 119, 123, 125

[206] L. C. Jain and L. R. Medsker, *Recurrent Neural Networks: Design and Applications*, 1st ed. USA: CRC Press, Inc., 1999. 118

[207] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," 2000, pp. 1008–1014. [Online]. Available: http://papers.nips.cc/paper/1786-actor-critic-algorithms 122

[208] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Mach. Learn.*, vol. 3, no. 1, p. 9–44, aug 1988. 122

[209] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012. 122

[210] R. J. Williams and J. Peng, "Function optimization using connectionist reinforcement learning algorithms," *Connection Science*, vol. 3, no. 3, pp. 241–268, 1991. [Online]. Available: https://doi.org/10.1080/09540099108946587 122

[211] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013. [Online]. Available: http://arxiv.org/abs/1312.5602 123

[212] M. J. Smith, "Getting value from artificial intelligence in agriculture," *Animal Production Science*, vol. 60, no. 1, p. 46, 2020. [Online]. Available: https://doi.org/10.1071/an18522 130

Glossary

Activation function	We use a nonlinear function for some of the layers of neural networks to help them understand complex decision boundaries.
Backpropagation	An approach called backpropagation is used to quickly calculate gradients in a neural network.
Batch size	The number of training examples in one forward/backward pass.
Convolutional Neural Network (CNN)	It is a kind of deep learning algorithm. It is used for image classification, segmentation and regression.
convolutional layer	A layer of a deep neural network in which a convolutional filter passes along an input matrix.
Deep Learning	It is subfield of Machine learning. In deep learning, computational models with numerous processing layers can learn representations of data at different levels of abstraction.
Dropout	Dropout is a method used for regularization in neural networks.
Epoch	In machine-learning and deep learning, an epoch is a complete pass through a given dataset.
F_1-Score	is the weighted average of Precision and Recall that explains how well the network performed during training.
Gated recurrent unit (GRU)	A GRU is a condensed LSTM. GRUs use gating methods to avoid the vanishing gradient problem and learn long-range relationships.
Long short-term memory (LSTM)	It is a kind of RNN model design to solve the vanishing gradient of RNN model and handle the time series data set.
Loss function	The loss function calculates the difference between the algorithm's current output and its predicted output.

Mean Squared Error (MSE)	is the average of the squared differences between prediction and actual observation.
Pooling Layer	A form of non-linear downsampling layer in CNN moldel.
Recurent Neural Network (RNN)	It is a type of deep learning model developed for processing time series data.
Root Mean Squared Error (RMSE)	Root square of Mean Squared Error (MSE)
R^2-score	is a statistical measure calculated from the variance explained by the model versus the total variance.
Reinforcement Learning	Reinforcement learning is a branch of machine learning that is goal-oriented, meaning that reinforcement learning algorithms aim to maximize a reward, often over the course of many decisions.
Single Shot Multibox Detector (SSD)	is a deep learning model designed for real-time object detection.

Index