UNIVERSIDADE
BEIRA INTERIOR

Portable, multi-task, on-the-edge and low-cost computer vision framework based on deep learning: Precision agriculture application

Eduardo Timóteo de Assunção

Tese para obtenção do Grau de Doutor em
Engenharia Informática
(3º ciclo de estudos)

Orientador: Prof. Doutor Hugo Pedro Martins Carriço Proença
Coorientador: Prof. Pedro Miguel de Figueiredo Dinis Oliveira Gaspar

Júri:
Prof. Doutor Luís Filipe Barbosa de Almeida Alexandre
Prof. Doutora Bernardete Martins Ribeiro
Prof. Doutor Carlos Manuel Antunes Lopes
Prof. Doutora Isabel Maria do Nascimento Lopes Nunes
Prof. Doutor António José Ribeiro Neves
Prof. Doutor Pedro Miguel de Figueiredo Dinis Oliveira Gaspar
Prof. Doutor João Carlos Raposo Neves

29 de maio de 2025

Declaração de Integridade

Eu, Eduardo Timóteo de Assunção, que abaixo assino, estudante com o número de inscrição D2385 do 3º Ciclo em Engenharia Informática, 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, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 2025 /06 /26

Assinado por: **Eduardo Timoteo de Assunção**
Num. de Identificação: TR:PT-74Q69R345
Data: 2025.06.26 10:52:05+01'00'

To my parents Timóteo Assunção and Lídia Assunção

Acknowledgments

From the bottom of my heart, I would like to thank my thesis advisor Prof. Hugo Proença and my co-advisor Prof. Pedro D. Gaspar for their consistent support and guidance in the completion of this thesis. I would also like to thank my friend Khadijeh Alibabaei, who has been a great support and given me confidence in my abilities. Finally, my greatest thanks go to my family (my son Marcos Assunção, my daughter Mª Helena Assunção, and my wife Lúcia Assunção) for their patience and encouragement.

This thesis was prepared at the University of Beira Interior, IT - Instituto de Telecomunicações, PIA-Cv Group, Covilhã Delegation, and was submitted to the University of Beira Interior for defense in a public examination session. Also, the work was funded by FCT/MEC through national funds and co-funded by FEDER -PT2020 partnership agreement under the project UIDB/50008/2020.

This work was funded in part by the PrunusBot project—Autonomous controlled spraying aerial robotic system and fruit production forecast, Operation No. PDR2020-101-031358 (leader), Consortium No. 340, Initiative No. 140, promoted by PDR2020 and co-financed by the EAFRD and the European Union under the Portugal 2020 program. This work was also supported by C-MAST (Centre for Mechanical and Aerospace Science and Technologies) under project UIDB/00151/2020.

This thesis was supported in part by the FCT/MEC through National Funds and Co-Funded by the FEDER-PT2020 Partnership Agreement under Project UIDB/50008/2020, Project POCI-01-0247-FEDER-033395; in part by operation Centro-01-0145-FEDER-000019-C4-Centro de Competências em Cloud Computing, co-funded by the Europen Regional Development Fund (ERDF)through the Programa Operacional Regional do Centro (Centro 202), in the scope of the Sistema de Apoio à Investigação Científica e Tecnológica- Programas Integrados de IC&DT. This work was also supported by ”IT:Instituto de Telecomunicação” and ”TOMI: City’s Best Friend” under Project UID/EEA/50008/2019.

Resumo

A agricultura de precisão é um novo conceito que foi introduzido em todo o mundo para aumentar a produção, reduzir a mão de obra e garantir o gerenciamento eficiente de fertilizantes e processos de irrigação. A visão artificial é um componente essencial da agricultura de precisão e desempenha um papel importante em muitas tarefas agrícolas. Serve como uma ferramenta de perceção para a interface mecânica entre robôs e ambientes, bem como para muitas outras tarefas, como a previsão da colheita. Outra consideração importante é que algumas aplicações de visão artificial devem ser executadas em dispositivos móveis, que normalmente têm poder de processamento e memória muito limitados. Portanto, os modelos de visão artificial, que serão executados em dispositivos móveis, devem ser otimizados para obter um bom desempenho. Devido ao impacto significativo da aprendizagem profunda e ao desenvolvimento e crescente utlização de dispositivos móveis com aceleradores, houve um aumento da investigação nos últimos anos sobre visão artificial para aplicações de uso geral que têm o potencial de aumentar a eficiência das tarefas de agricultura de precisão. Esta tese explora como os modelos de aprendizagem profunda executados em dispositivos móveis são afetados por otimizações, ou seja, precisão e tempo de inferência. Neste trabalho, modelos de baixa complexidade computacional e requisitos de recursos para segmentação de infestantes, deteção de frutos de pêssegos e classificação de doenças em frutos são casos de estudos. Primeiro, é realizado um caso de estudo de deteção dos frutos de pêssegos com o conhecido detetor de objetos Faster R-CNN usando a Rede Neural Convolucional (CNN – Convolutional Neural Network) AlexNet para extração de caraterísticas de imagem. Uma precisão de deteção de 0,90 AP (AP – Average Precision) foi obtida. A rede AlexNet não é um modelo otimizado para uso em dispositivos móveis. Para explorar um modelo leve, um caso de estudo de classificação de doenças em pêssegos é conduzido usando a rede MobineNet. A rede MobileNet foi treinada em um pequeno conjunto de dados de imagens de pêssegos saudáveis, podres, com bolor, e com feridas e obteve um desempenho de F1 = 0,96. As lições aprendidas com este trabalho levaram ao uso deste modelo como base para outras aplicações de visão artificial (por exemplo, deteção de frutas e segmentação de infestantes. Em seguida, foi realizado um estudo sobre o controle robótico de infestantes usando um pulverizador automático de herbicida. O modelo de segmentação semântica DeepLab com a espinha dorsal MobileNet foi usado para segmentar as infestantes e determinar as coordenadas espaciais para o mecanismo. O modelo foi otimizado e implantado no dispositivo Jetson Nano e integrado ao veículo robótico para avaliar o desempenho em tempo real. Um tempo de inferência de 0,04 s foi alcançado e os resultados obtidos neste trabalho fornecem informações sobre como o desempenho do modelo de segmentação semântica de plantas e infestantes se degrada quando o modelo é adaptado por meio de otimização para operação em dispositivos móveis. Por fim, para ampliar a aplicação de modelos de baixa complexidade computacional e requisitos de recursos de aprendizagem profunda e o uso de dispositivos móveis suportado por acelerador, o modelo de deteção de objetos (SSD - Single Shot Detector) foi treinado para detetar três cultivares diferentes de pêssegos e foi implantado em um dispositivo Raspberry Pi com um acelerador integrado (TPU – Tensor Processing Unity). Algumas variações da

rede MobileNet como espinha dorsal foram exploradas para investigar a compensação entre precisão e tempo de inferência. O MobileNetV1 apresentou o melhor tempo de inferência (21,01 FPS), enquanto o MobileDet obteve a melhor precisão de deteção (88,2% AP). Além disso, um conjunto de dados de imagens de três cultivares de pêssego de Portugal foi desenvolvido e disponibilizado. Esta tese visa contribuir para etapas futuras no desenvolvimento da agricultura de precisão e da robótica agrícola, especialmente quando a visão artificial precisa ser processada em pequenos dispositivos.

Palavras-chave

Agricultura de precisão, Aprendizagem profunda, Visão Artificial, Modelo leve, Dispositivo Móvel, Deteção de frutos, Controlo robótico de infestantes.

Abstract

Precision agriculture is a new concept that has been introduced worldwide to increase production, reduce labor and ensure efficient management of fertilizers and irrigation processes. Computer vision is an essential component of precision agriculture and plays an important role in many agricultural tasks. It serves as a perceptual tool for the mechanical interface between robots and environments or sensed objects, as well as for many other tasks such as crop yield prediction. Another important consideration is that some vision applications must run on edge devices, which typically have very limited processing power and memory. Therefore, the computer vision models that are to run on edge devices must be optimized to achieve good performance. Due to the significant impact of Deep Learning and the advent of mobile devices with accelerators, there has been increased research in recent years on computer vision for general purpose applications that have the potential to increase the efficiency of precision agriculture tasks. This thesis explore how deep learning models running on edge devices are affected by optimizations, i.e., inference accuracy and inference time. Lightweight models for weed segmentation, peach fruit detection, and fruit disease classification are cases of studies. First, a case study of peach fruit detection with the well-known Faster R-CNN object detector using the breakthrough AlexNet Convolutional Neural Network (CNN) as the image feature extractor is performed. A detection accuracy of 0.90 was achieved using metric Average Precision (AP). The breakthrough AlexNet CNN is not an optimized model for use in mobile devices. To explore a lightweight model, a case study of peach fruit disease classification is next conducted using the MobineNet CNN. The MobileNet was trained on a small dataset of images of healthy, rotten, mouldy, and scabby peach fruit and achieved a performance of 0.96 F1. Lessons learned from this work led to using this model as a baseline CNN for other computer vision applications (e.g., fruit detection and weed segmentation). Next, a study was conducted on robotic weed control using an automated herbicide spot sprayer. The DeepLab semantic segmentation model with the MobileNet backbone was used to segment weeds and determine spatial coordinates for the mechanism. The model was optimized and deployed on the Jetson Nano device and integrated with the robotic vehicle to evaluate real-time performance. An inference time of 0.04 s was achieved, and the results obtained in this work provide insight into how the performance of the semantic segmentation model of plants and weeds degrades when the model is adapted through optimization for operation on edge devices. Finally, to extend the application of lightweight deep learning models and the use of edge devices and accelerators, the Single Shot Detector (SSD) was trained to detect peach fruit in three different varieties and was deployed in a Raspberry Pi device with an integrated Tensor Unity Processor (TPU) accelerator. Some variations of MobileNet as a backbone were explored to investigate the tradeoff between accuracy and inference time. MobileNetV1 yielded the best inference time with 21.01 Frame Per Second (FPS), while MobileDet achieved the best detection accuracy (88.2% AP). In addition, an image dataset of three peach cultivars from Portugal was developed and published. This thesis aims to contribute to future steps in the development of precision agriculture and agricultural robotics, especially when computer vision needs to be processed on small devices.

Keywords

Precision agriculture, Deep learning, Computer vision, Lightweight model, Edge device, Fruit detection, Robotic weed control.

Contents

Acronym List

AI	Artificial Intelligence
AutoML	Auto-Machine Learning
AP	Average Pecision
CSI	Camera Serial Interface
CV	Computer vision
CNN	Convolutional Neural Network
DCNNs	Deep Convolutional Neural Networks
DNNs	Deep Neural Networks
DM	Depth Multiplier
FN	False Negative
FP	False Positive
Faster R-CNN	Faster Region-based Convolutional Neural Network
CWFID	Field Image Dataset
FLOPs	Floating-Point Operations
FPS	Frames Per Second
FCN	Fully Convolutional Network
GFLOPs	Giga Floating-Point Operations
GPU	Graphics Processing Unit
GHG	Greenhouse Gas
HSV	Hue, Saturation, and Value
IoT	Internet of Things
IOU	Intersection Over Union
mIOU	Mean Intersection Over Union
HOG	Histogram of Oriented Gradients
OS	Output Stride
PTQ	Post-Training Quantization
P	Precision
API	Application Programming Interface
QAT	Quantization-Aware Training
R	Recall
RPN	Region Proposal Network
ROIs	Regions of Interest
SSD	Single Shot Detector
TPU	Tensor Unity Processor
TMG	Tensorflow Model Garden
TN	True Negative
TP	True Positive
YOLO	You Only Look Once

Chapter 1

Introduction

Agriculture is one of the most traditional and important activities of man. As the world population grows, agricultural production must increase to meet the demand for food. On the other hand, agriculture has a direct impact on climate change, as it is responsible for some of the world's greenhouse gas emissions. For this reason, all agricultural activity must be directed towards the application of sustainable agricultural practices [1]. Precision agriculture is a new concept that has been introduced worldwide and can help achieve this goal. Computer vision (CV) is widely used in precision agriculture. It serves as a perceptual tool for the mechanical interface between robots and environments, such as vision systems for fruit harvesting, yield prediction, plant disease detection, and fruit detection. In the past, CV models were dominated by the traditional (manual) methods [2, 3]. Nowadays, traditional models have been replaced by deep neural networks (DNN). With this new approach, the parameters of the model are no longer created manually, but are learned. DNN are a branch of Artificial Intelligence (AI) and are proving to be powerful tools in precision agriculture because they can provide accurate predictions that lead to improved resource management (e.g., irrigation water, crop yield prediction, plant disease identification, herbicides, and pesticides) and help reduce food loss and waste through better planning of agricultural activities [4, 5, 6]. However, to achieve good performance, a large amount of data is required to train the model. However, there are some techniques to overcome this challenge. Another challenge of using DNN is the enormous amount of processing (mathematical operations) required. For applications that require processing at the edge (local) and require low latency, processing the data in a CPU is impractical. In this scenario, the solution is to process the data using a Graphics Processing Unit (GPU). GPUs are able to split complex problems into thousands or millions of individual tasks and process them simultaneously (parallel processing). To summarise, CPUs are good for serial processing and can perform a handful of operations at once. GPUs, on the other hand, are good for parallel processing and can perform thousands of operations at once [7].
The research questions are:

- Question 1: How does the deep learning based object detection model handle the detection of peach fruits with different colors, overlaps and occlusions?
- Question 2: Is a deep learning model suitable for fruit disease classification when the amount of data to be trained is small and unbalanced?
- Question 3.1: How does optimization of the model affect the inference time and accuracy?
- Question 3.2: Is it possible to use the optimized model in a robotic application for precision agriculture?

- Question 4: Is there an advantage to using TPU accelerators instead of GPUs to perform inference in precision agriculture?

1.1 Motivation

Model optimization is essential to enable the use of models on portable devices. However, information on how models are influenced by optimization is rarely mentioned in the literature.
Furthermore, there is a lack of practical applications in which edge devices (end-to-end) and computer vision are used in precision agriculture.

1.2 Objectives

The aim of this work is to:

- Explore the consolidated computer vision models based on deep learning by applying the necessary optimizations to run on edge devices.
- Explore the performance of the optimized models in precision agriculture tasks: Image classification, image segmentation, and object detection.
- Perform a practical application in precision agriculture with an optimized model deployed on an Edge device.
- Create an image dataset for precision agriculture.

1.3 Contributions

The most important research contributions of this work include:

- Critically evaluating the optimization of existing deep learning models for the use of edge devices.
- Making a significant contribution to the practical application of computer vision based on deep learning in the field of precision agriculture.
- Creation of a novel dataset for peach varieties that helps to close the gap in this field.

1.4 Outline of the Study

In this work, the study with application in precision agriculture was divided into exploration and optimization of models with four projects (Figure 1.1).

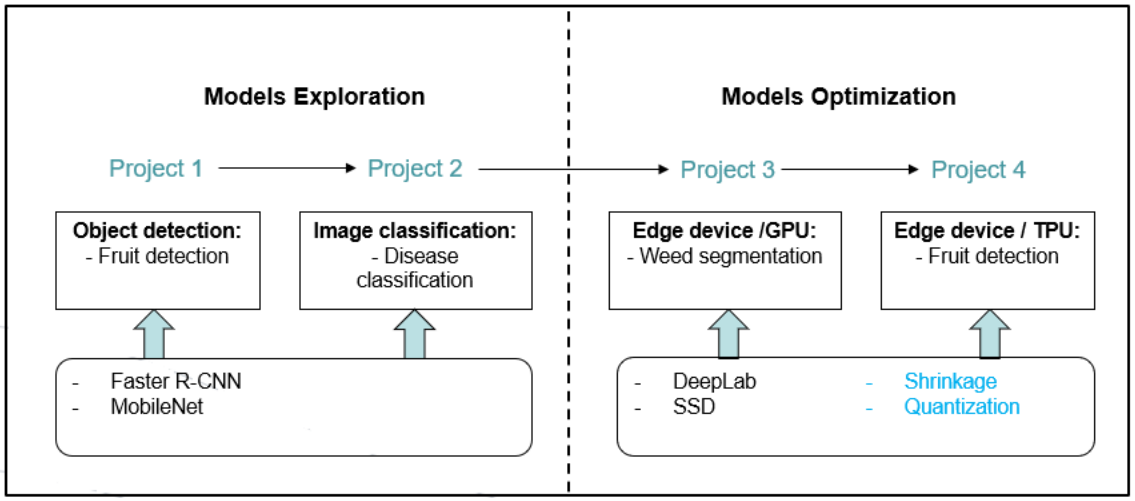


Figure 1.1: Outline of the study.

In project 1, the Faster R-CNN object detection model was implemented to detect peaches. In project 2, the MobileNet image classification model was implemented to classify peach fruit diseases.
In project 3, which aimed to implement the models in an edge device with a GPU accelerator, an image segmentation model was implemented to segment weeds.
In project 4, the SSD object detector was implemented to detect peaches. The model was embedded in an edge device with a TPU accelerator.
Optimization methods (Figure 1.2) were used in projects 3 and 4, namely pruning and quantization.

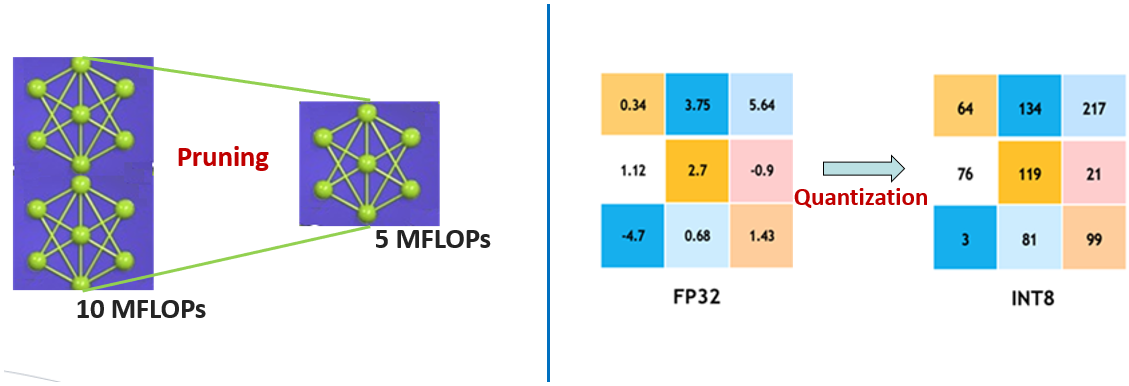


Figure 1.2: Model optimization methods.

1.5 Structure of the Thesis

This thesis is divided into seven chapters: Chapter 3 discusses fruit detection by applying the Faster R-CNN two-stage detector to peach fruit detection. Chapter 4 explores the lightweight peach fruit disease classification model. Chapter 5 extends the concept of the lightweight model in a weed segmentation case study through hyperparameter selection, model optimization, and deployment in an edge device with a GPU accelerator. A proof of concept is also developed with a prototype for robotic herbicide spraying. In Chapter 6, a dataset of

three peach cultivars was created and work was done on real-time peach fruit detection using a Raspberry Pi and a TPU accelerator. Chapter 7 discusses the results of all chapters and answers the research questions. Chapter 8 provides concluding remarks on this work. It also discusses future research directions in deep learning and edge device technology for application in precision agriculture.

Chapter 2

Contextualization

Various deep learning-based models for detection in precision agriculture exist in the literature. Wu et al. [8] proposed a real-time method for apple blossom detection (in natural environments) based on YOLO v4 combined with the channel pruning algorithm. The precision and recall index of the proposed method were 89.43% and 98.15%, respectively; the average IOU and mAP were 84.41% and 97.31%, respectively; the detection speed reached 72.33 f/s. Jia et al. [9] propose an improved mask R-CNN by using the ResNet in combination with the DenseNet network as a backbone, and apply it to the vision system of the apple picking robot. The method has been tested with a random test set of 120 images, and the precision rate has reached 97.31%, and the recall rate has reached 95.70%. Santos et al. [10] evaluated three deep learning architectures for grape detection: Mask R-CNN, YOLOv2, and YOLOv3. Using IoU of 0.5 and AP metrics, the performance for grape detection for the Mask R-CNN, YOLOv2, and YOLOv3 models was 71%, 48% and 39%, respectively. Fangfang et al. [11] proposed to use the Faster R-CNN model for detecting apples under different conditions, such as non-occluded, leaf-occluded, branch/wire-occluded, and fruit-occluded fruit. AP of non-occluded, leaf-occluded, branch/wire-occluded, and fruit-occluded were 90%, 89%, 85%, and 84%, respectively. Overall, the mAP of the four classes was 87%, and an average of 241 ms was required to process an image. Hanwen et al. [12] proposed a framework for a deep-learning-based fruit detector for apple harvesting using the LedNet network. The model achieved 82% and 85% in recall and accuracy in detecting apples in orchards, and its inference time was 28 ms.

Reducing pesticide use through selective spraying is an important building block for more sustainable computer-assisted agriculture. Identifying weeds at an early growth stage contributes to reduced herbicide use. Borja et al. [13] presented a novel system for crop and weed identification based on a combination of convolutional networks (Xception, Inception-Resnet, VGNets, Mobilenet, and Densenet) with the combination of ”traditional” machine learning classifiers (Support Vector Machines, XGBoost, and logistic regression). They evaluated the models on two crop datasets (tomato and cotton) and two weed species (black nightshade and velvetleaf). The best crop/weed identifier (Densenet and Support Vector Machine) performed 99.29% F1 score. Wang et al. [14] proposed to use an encoder-decoder deep learning network to study a segmentation of crops (sugar beets, oil seeds) and weeds. The best mIoU value for pixel-wise segmentation was 88.91% and the best mean accuracy of object-wise segmentation was 96.12%. The overall inference time is about 100 ms for an image with a resolution of 1296×966 pixels using a GPU NVIDIA GTX1080Ti. Le et al. [15] compared the performances of the Filtered Local Binary Patterns with contour masks and coefficient k (k-FLBPCM) algorithm with CNN models (VGG-16, VGG-19, ResNet-50, and InceptionV3) for detecting crops and weeds with similar morphology. The experimental results for the ”bccr-segset” dataset show that the accuracy of the CNN models is slightly

higher than that of the k-FLBPCM method, while the accuracy of the k-FLBPCM algorithm is higher than that of the CNN models (except VGG-16) for the ”fieldtrip_can_weeds” dataset collected from real agricultural fields. With an image size of 228×228 pixels and using the GTX1080Ti GPU, the test time of the k-FLBPCM method was 0.223 ms per image, while it was 2.667 ms, 3.033 ms, 2.333 ms, and 3.5 ms for VGG-16, VGG-19, ResNet-50, and InceptionV3, respectively. Czymmek et al. [16] presented a method for detecting weeds in carrot fields in real time. The method uses a convolutional neural network to simultaneously locate and classify plants. An accuracy of 89% was achieved with an inference time of 18.56 FPS. Accepting lower accuracy in favor of higher processing, the inference time was about 56 FPS. Partel et al. [17] presented a study on weed control using the YOLO detector. This is one of the few works targeting edge devices. Using the NVIDIA Jetson TX2 device, an overall precision and recall of 77% and 58% was achieved with real plants. At an image resolution of 1024×256, the inference time was 22 FPS. Kounalakis et al. [18] proposed a weed detection system combining traditional computer vision techniques (sliding windows and binary classifier) and newer approaches (CNN for feature extraction). The combination that performed the best was Inception_v1 + L2regLogReg with accuracy, precision, and recall of 96.13, 15.48, and 69.48, respectively. Rahman et al. [19], using a three-class weed dataset, evaluated 13 weed detections with DL -based one-stage and two-stage object detectors, including YOLOv5, RetinaNet, EfficientDet, Fast RCNN, and Faster RCNN. RetinaNet achieved the highest overall detection accuracy with an average precision of 79.98%. YOLOv5 achieved the fastest inference time (17 ms on the Google Colab). Moazzam et at. [20] presented a method for classifying crop and weed pixels (semantic segmentation in two stages). The proposed method was evaluated on seven different images of tobacco fields and achieved MIOU ranging from 0.78 to 0.91. The inference time for an image of size 480×352 was 700 ms using a GTX 1050 NVIDIA GPU. It is important to emphasise that most of the literature on computer vision for precision agriculture ignores resource-constrained platforms, such as edge devices.

Edge computing is about placing computing resources as close as possible to the data source and the location of the required actions. Edge devices are small, stand-alone machines (computers) that often have sensors and actuators directly connected to them, allowing them to respond to real-world events with a latency of less than a millisecond. Because an edge device can process data locally, its sensors (e.g., cameras) can capture samples at higher resolution and higher frequency than would be possible if the data had to be sent to the cloud for processing [21]. An example of the current generation of small edge devices is the NVIDIA Jetson Nano series, a full-featured Linux-based computer. The Jetson Orin Nano (Figure 2.1) features a 6-core 64-bit processor CPU, 8 GB RAM, and a 1024-core NVIDIA Ampere GPU capable of performing 40 trillion operations per second (40 TOPs) [22].

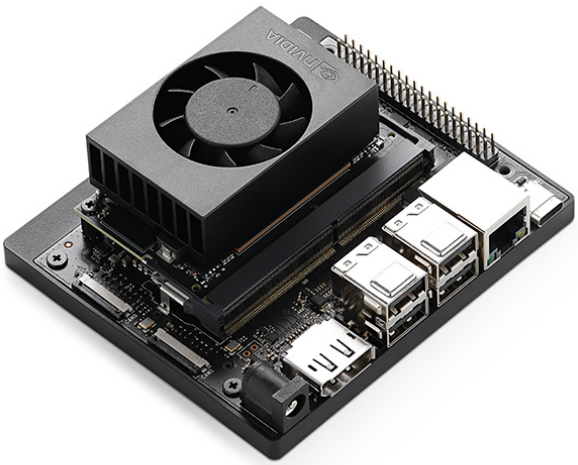


Figure 2.1: NVIDIA Edge Device: Jetson Orin Nano [22].

Another example of a well-known edge device is the Raspberry Pi (Figure 2.2). This device, despite its small size, can function as a full desktop computer like the Jetson Nano by simply connecting it to an HDMI screen, keyboard, and mouse. An important aspect of edge devices is the purchase price. For example, the Raspberry Pi 4 with 1 GB costs about €44,00.

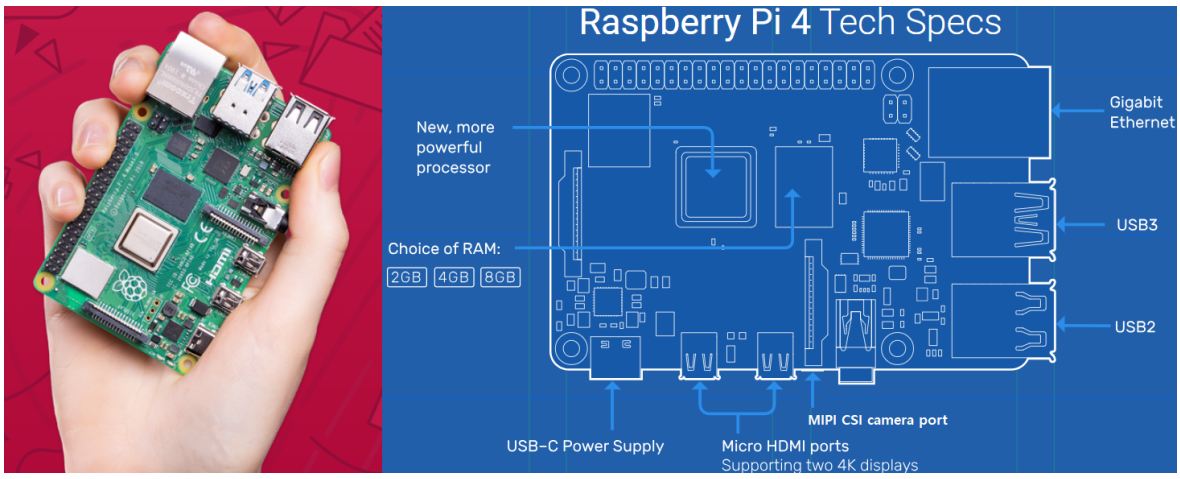


Figure 2.2: Raspberry Pi device [23].

The Raspberry Pi company has a variety of add-on products, such as cameras. Some of the Raspberry Pi camera types are RGB, Infrared, Global Shutter, and High Quality (Figure 2.3). These cameras can be connected directly to the CSI port available on the Edge device.

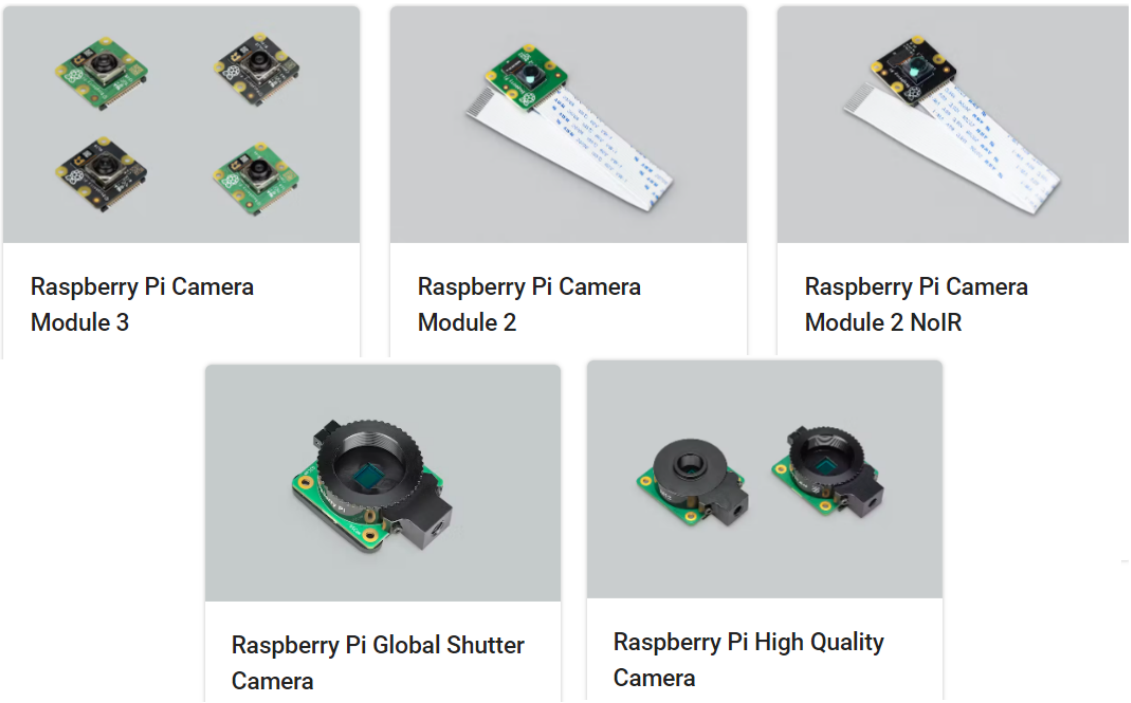


Figure 2.3: Cameras for edge devices [23].

Although edge devices are available on the market, there are very little works using them for CV in precision agriculture. Thus, most of the methods use a desktop-based approach with conventional GPU cards [8, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20]. However, all these methods are not optimized to run on edge devices. Moreover, the few existing edge device approaches only cover object detection models (i.e., no methods for semantic segmentation tasks [17, 24]), and not show details of model optimization for use on edge devices, such as quantization and its impact on model performance.

As already mentioned, edge computing offers an alternative for computing resources close to the data sources. This technology moves calculations from the cloud to the edge to avoid unnecessary delays and additional costs. Edge Artificial Intelligence (AI) deals with the integration of edge computing and AI techniques, not just as a simple combination of the two, but rather as their sophisticated intertwining to address the challenges [25]. Figure 2.4 shows how an edge device can be used together with AI.

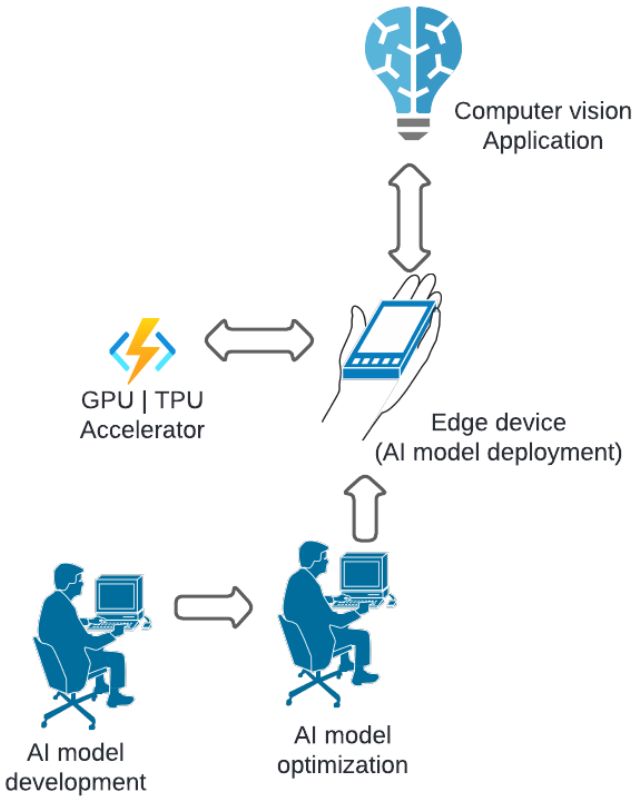


Figure 2.4: Edge device and AI integration.

Chapter 3

Peaches Detection Using a Deep Learning Technique—A Contribution to Yield Estimation, Resources Management, and Circular Economy

The following work was developed by myself, P. Gaspar, M. Ricardo, M. Simões, A. Ramos, H. Proença and P. Inácio between April and December 2021. The aim was to develop a peach fruit detection by applying a deep learning based object detection framework using data recorded in an outdoor orchard.

1

[1]This work was published in [26], E. Assunção, P. D. Gaspar, M. Ricardo, M. P. Simões, A. Ramos, and H. Proença, "Peaches Detection Using a Deep Learning Technique—–A Contribution to Yield Estimation, Resources Management, and Circular Economy," Climate, vol. 10, no. 2, 2022. https://doi.org/10.3390/cli10020011

climate

MDPI

Article

Peaches Detection Using a Deep Learning Technique—A Contribution to Yield Estimation, Resources Management, and Circular Economy

Eduardo T. Assun ção [1,2], Pedro D. Gaspar [1,2,*], Ricardo J. M. Mesquita [1], Maria P. Simões [3], António Ramos [3], Hugo Proença [4] and Pedro R. M. Inacio [4]

1 C-MAST Center for Mechanical and Aerospace Science and Technologies, University of Beira Interior, 6201-001 Covilha, Portugal; eduardo.assuncao@ubi.pt (E.T.A.); ricardo.mesquita@ubi.pt (R.J.M.M.)
2 Deparment of Electromechanical Engineering, University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilha, Portugal
3 School of Agriculture, Polytechnic Institute of Castelo Branco, 6000-084 Castelo Branco, Portugal; mpaulasimoes@ipcb.pt (M.P.S.); aramos@ipcb.pt (A.R.)
4 Instituto de Telecomunicações, Department of Computer Science, University of Beira Interior, 6201-001 Covilha, Portugal; hugomcp@di.ubi.pt (H.P.); prmi@ubi.pt (P.R.M.I.)
* Correspondence: dinis@ubi.pt

Abstract: Fruit detection is crucial for yield estimation and fruit picking system performance. Many state-of-the-art methods for fruit detection use convolutional neural networks (CNNs). This paper presents the results for peach detection by applying a faster R-CNN framework in images captured from an outdoor orchard. Although this method has been used in other studies to detect fruits, there is no research on peaches. Since the fruit colors, sizes, shapes, tree branches, fruit bunches, and distributions in trees are particular, the development of a fruit detection procedure is specific. The results show great potential in using this method to detect this type of fruit. A detection accuracy of 0.90 using the metric average precision (AP) was achieved for fruit detection. Precision agriculture applications, such as deep neural networks (DNNs), as proposed in this paper, can help to mitigate climate change, due to horticultural activities by accurate product prediction, leading to improved resource management (e.g., irrigation water, nutrients, herbicides, pesticides), and helping to reduce food loss and waste via improved agricultural activity scheduling.

Keywords: convolutional neural network; deep learning; fruit detection; precision agriculture; sustainability

Citation: Assunção, E.T.; Gaspar, P.D.; Mesquita, R.J.M.; Simões, M.P.; Ramos, A.; Proença, H.; Inacio, P.R.M. Peaches Detection Using a Deep Learning Technique—A Contribution to Yield Estimation, Resources Management, and Circular Economy. *Climate* **2022**, *10*, 11. https://doi.org/10.3390/cli10020011

Academic Editors: Jong Ahn Chun, Hen-I Lin and Daeha Kim

Received: 21 December 2021
Accepted: 10 January 2022
Published: 18 January 2022

1. Introduction

Sustainable agricultural practices can improve product availability in a supply chain. Some traditional practices can dangerously compromise soil structure and an orchard's abilities to adapt to extreme weather events, climate change, and the stresses of intensive harvest production. All methods, whether new or traditional, must be examined when performing sustainable agriculture practices [1].

Agriculture directly impacts climate change as its activities contribute to a portion of global anthropogenic greenhouse gas (GHG) emissions. The main GHGs produced by agriculture are nitrous oxide (N_2O) and carbon dioxide (CO_2). N_2O is produced by the microbial transformation of nitrogen in soils during applying synthetic fertilizers to the ground, and CO_2 is produced by changes in above- and below-ground carbon stocks caused by land-use changes [2]. Continuous monitoring can provide an up-to-date status of a field in real-time, based on the selected parameters of the plants and fields. This approach is made possible by a computer vision framework on the fields. Fruit detection via computer vision allows monitoring and control of yield, contributing to higher production. Higher yields indicate the possibility of rising production, with less water and without extending

the area under agriculture, which means less deforestation and a reduction of natural resource consumption, and GHG emissions.

The application of precision agriculture techniques can help to mitigate climate change. Deep neural networks emerge as powerful tools in precision agriculture as they can provide accurate product prediction, leading to improved resource management, such as irrigation water, nutrients, herbicides, and pesticides, and help reduce food loss and waste by improved agricultural activity scheduling. Alibabaei et al. [3] investigated the ability of an encoder–decoder long short-term memory model, to model the daily evapotranspiration for threes. Assunção et al. [4] used computer vision, based on convolutional neural networks, to classify peach fruit diseases.

Yield estimation is a branch of precision agriculture. It enables planning for cropping, operations, inventory management, and other ancillary services (e.g., fruit pickup by a robot). There are several published works in the field of fruit yield estimation. For example, Häni et al. [5] presented a methodology for apple production estimation. Dorj et al. [6] developed a system for citrus fruit detection and counting. Bargoti and Underwood [7] presented a work on the detection of mangoes, almonds, and apples in orchards. A fruit yield estimation pipeline begins with detection, followed by tracking and counting. The fruit detection phase is critical to the performance of the yield estimation system. Most state-of-the-art fruit detection systems are based on a convolutional neural network (CNN) [8], such as object detection, segmentation, among others. These methods automatically extract features from the appearance of fruit images (i.e., colors and shapes). In this context, the development of fruit detection is specific (because the fruit colors, sizes, shapes, clusters, and distributions in the trees are particular). Figure 1 illustrates this problem. Considering the lack of work with peaches, this paper presents the results for peach detection using the object detection framework faster region-based convolutional neural network (Faster R-CNN). Another contribution is that the images are from a non-controlled environment (outdoors). That is, they come from a natural orchard.

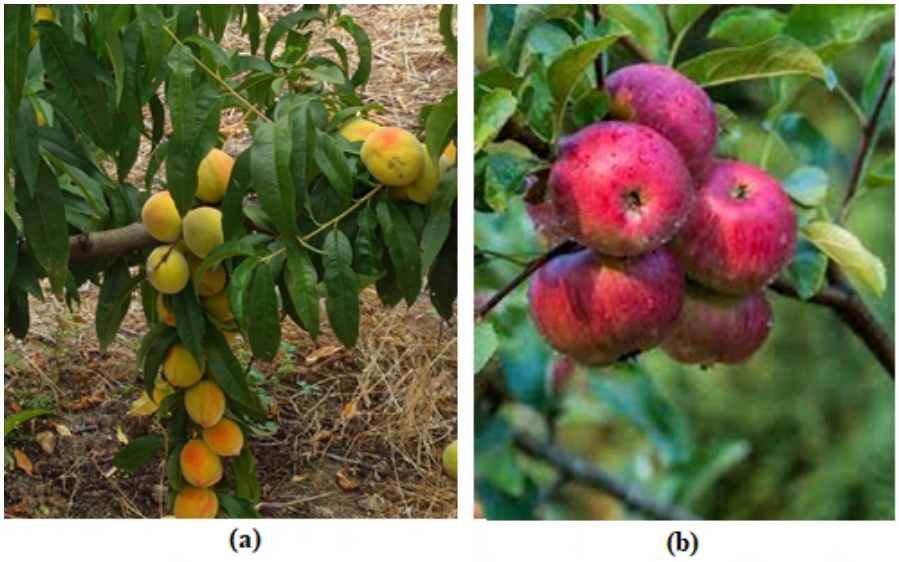


Figure 1. Particularity of fruit type: (**a**) peach. (**b**) apple.

Bargoti and Underwood [9] presented a methodology to obtain an estimate of red and green apples. For the detection phase, they used the traditional computer vision approach based on hue, saturation, and value (HSV) color space for segmentation. With the development of convolutional neural networks (CNNs), many authors use them for fruit detection based on segmentation and object detection approaches. Puttemans et al. [10] proposed a watershed and trinocular stereo triangulation-based segmentation with a cascade of weak classifiers to perform a strawberry and apple detector model. Bargoti

and Underwood [11] trained a CNN, where its output was the probability that each pixel belonged to a fruit. Then, the output result was used to generate a binary mask for segmentation. Häni et al. [12] proposed the CNN segmentation model, named U-NET, for apple segmentation and circular Hough transform for fruit detection. The original purpose for this network is medical image segmentation. The author reports an F1 score of 0.858 for detection.

CNNs have also made massive contributions in improving image classification and object detection. In this regard, the object detection framework R-CNN [13] and its variants fast R-CNN [14], faster R-CNN [15], and mask R-CNN [16] are widely used in the literature. Sa et al. [17] proposed faster R-CNN to detect peppers, apples, avocados, mangoes, strawberries, and oranges. However, only the images of peppers are from the orchard. The rest of the fruit images are from the internet (Google Images). CNNs rely on large amounts of data for training to avoid overfitting, and they have good generalization. To achieve good performance with a relatively small set of training data, a transfer learning technique is often used.

Recently, you only look once (YOLO), another branch of object detection based on CNN, became a framework for fruit detection. Koirala et al. [18] applied this model to detect mangoes in orchard images. They used two metrics for evaluation, F1 and average precision (AP), and reported results of 0.96 and 0.98, respectively. Liu et al. [19] applied the mask R-CNN method to detect cucumber fruits. They chose ResNet-101 as the backbone and proposed a modification in the scales and aspect ratios of the anchor boxes. For evaluation, the authors used an F1 score and obtained a score of 0.894 for the test images.

Other studies were developed to detect fruits. However, fruit detection procedures are specific due to the particular distribution of fruit colors, sizes, shapes, branches, and bundles. This study provides the first application (and, consequently, the annotated dataset) of peach detection that is constrained by (1) peaches with different colors, (2) peaches overlapped, and (3) peaches occluded by leafs. The faster R-CNN computer vision framework is used for fruit detection in peach orchards. Fruit detection can be used to count the number of fruits during the culture, and knowing that value, irrigation and all other cultural practices can be conveniently scheduled, and the correct amount of required inputs for fertilization and plant protection can be acquired. This approach improves the productivity and competitiveness of farmers while ensuring environmental concerns. Thus, it can be considered a precision agriculture application based on a deep neural network to help mitigate climate change, as continuous monitoring provides an up-to-date status of the field culture and allows monitoring and control of yields, which may be used as decision-making support systems that lead to increased production with less consumption of natural resources and GHG emissions.

2. Materials and Methods

2.1. Dataset

The images used to compose the dataset were taken with the Eken H9R camera in a peach orchard in Beira Interior, Portugal. All images that composed the training dataset were from the same orchard and were taken on the same day. A particular feature for this peach dataset is its yellowish color. This feature can be a good way to check the generalization ability of the model when testing images from other orchards with a different peach color. Figure 2 shows a training sample image. The original images have a resolution of 5472 $\times$ 3648 (pixels), RGB channels with 24-bit color depths. The relationship between the object (fruit) size and image size affects the detection performance. This means that a small object is difficult to detect in a large image. This is due to the peculiarity of the CNN polling operation and anchor size, which is part of the object detection algorithm. To mitigate this problem, we crop each original image in for quadrants. Each of them with a resolution of 2736 $\times$ 1824 (pixels). Figure 3 illustrates this process.

Figure 2. Training sample image.

Figure 3. Process of cropping the original image into four parts.

For training, 200 images were used, with 1934 annotated fruits. For testing, 40 images (from the same orchard) with 410 annotated peaches were used.

In addition, a small dataset was created from another orchard (Quinta Nova, also in Beira Interior, Portugal), which has a different color, to check the generalization of the model. Figure 4 shows an image that composes the test dataset. It can be seen that the peaches have a reddish color and the illumination varies a lot compared to the training image. This subset of test images has the same resolution as the training images (2736 × 1824). With seven samples, 91 peaches were annotated.

Figure 4. Testing sample image.

2.2. Object Detection Framework

Object detection is a branch of computer vision that aims to find and classify objects in an image. Histograms of oriented gradients for human detection (HOG) [20] is one of the most popular and traditional (i.e., hand-crafted) methods used for object detection. Nowadays, traditional object detections are replaced by modern models based on CNNs (e.g., faster R-CNN [15], single shot multibox detector (SSD) [21], you only look once (YOLO) [22]).

In this paper, the faster R-CNN model was used as the basis. The faster R-CNN is a two-stage model in which a CNN backbone is used for feature extraction in the first stage (e.g., VGG16 [23], ResNet [24], Inception [25], and others). Figure 5 shows a simplified workflow for the faster R-CNN in the context of fruit detection. Here, the outputs of the convolution layers are called feature maps. The last layer is used as an input to the region proposal network (RPN), which generates the regions containing possible objects. These regions of interest (ROIs) are used for object classification and bounding box prediction.

Figure 5 shows the faster R-CNN object detection model in the context of fruit detection.

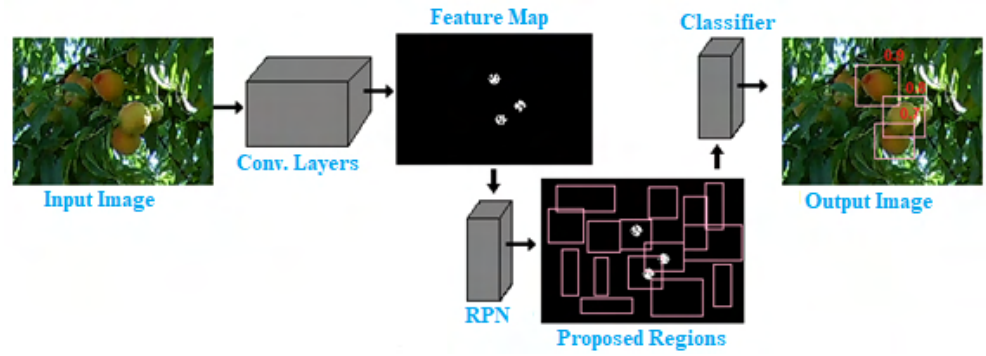


Figure 5. Faster R-CNN in the context of fruit detection.

The TensorFlow model application programming interface (API) was used to perform the experiments [26]. This API implements the object detection frameworks faster R-CNN and SSD. In this work, the faster R-CNN with the Inception v2 backbone was used because it has better detection accuracy compared to the other variants implemented in this API. In addition, transfer learning was performed on the COCO dataset [27]. Training and testing

were performed on a desktop with Intel(R) Core(TM) i7-4790 CPU @ 3.60 GHz, 8 GB RAM, and an NVIDIA RTX 2080 graphics card with 8 G memory.

2.3. Evaluation Metric

Intersection over union (IOU), defined in (1), is a metric used to evaluate object detection tasks [28]. It provides a value for bounding box prediction. The value obtained from IOU can be used to indicate whether the prediction is true positive (TP), false positive (FP), or false negative (FN), depending on the threshold used. For example, if the IoU threshold is 0.5 and the IoU value for a (positive class) prediction is 0.7, then the prediction is classified as true positive (TP). On the other hand, if the IoU value is 0.3, then the prediction is classified as false negative (FN).

$$IOU = \frac{A \cap B}{A \cup B}, \tag{1}$$

were A is the ground truth area and B is the detected area. Figure 6 illustrates the IOU metric.

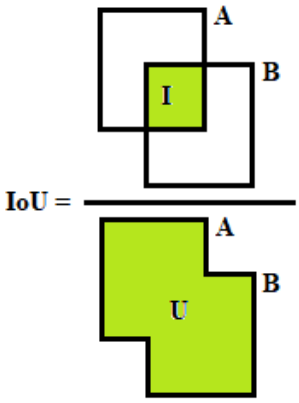


Figure 6. Intersection over union illustration.

Evaluations on the test images were performed using the PASCAL VOC metric average precision (AP) for an IOU threshold of 0.5.

3. Results and Discussions

3.1. Training Result

The training loss curve of faster R-CNN with Inception v2 is shown in Figure 7. The final loss was about 0.1 and the loss curve started to stabilize after 8000 iterations.

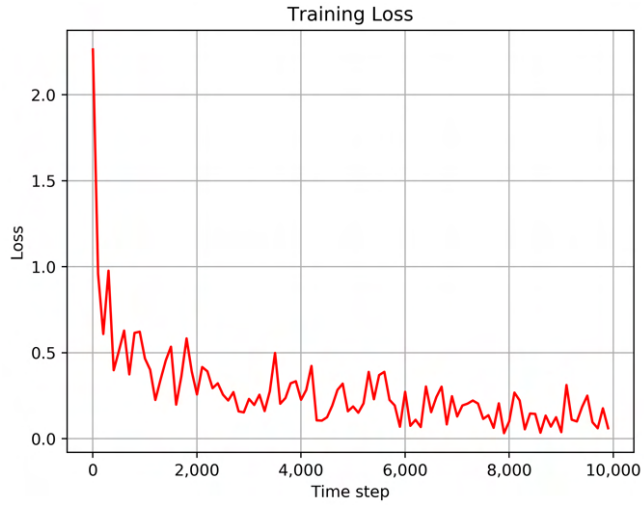


Figure 7. Training loss curve.

3.2. Test Results in the Same Orchard Where the Training Was Performed

Detection performance was assessed by the metric AP with IOU = 0.5 and achieved a value of 0.90. Figures 8 and 9 show the visual results.

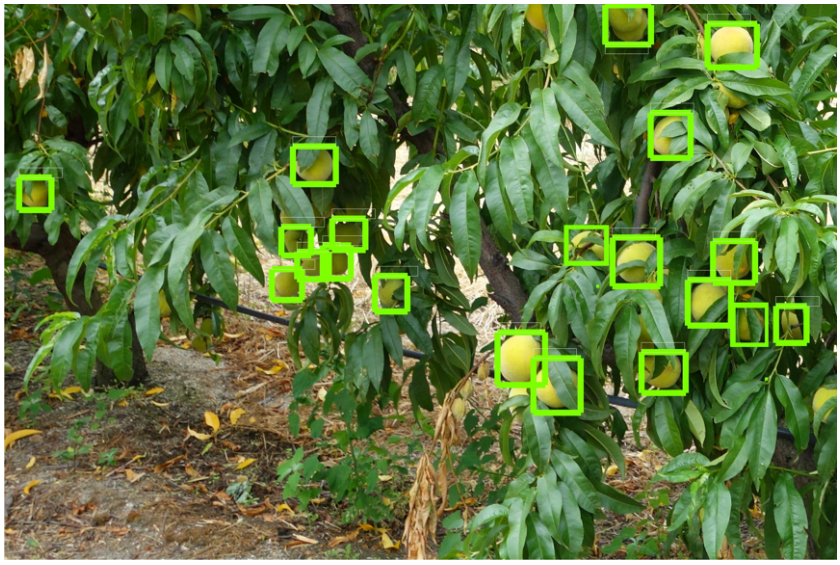


Figure 8. Example (1) of peach detection by faster R-CNN with Inception v2 backbone.

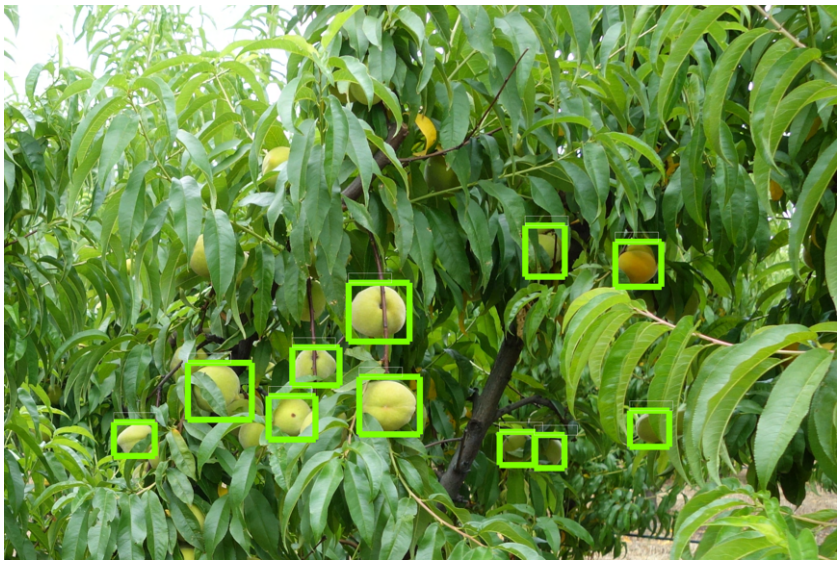


Figure 9. Example (2) of peach detection by faster R-CNN with Inception v2 backbone.

3.3. Test Results in a Different Orchard

The following results and discussion are intended to show the evaluation of the model in a different orchard, with many variations, compared to the orchard in which the model was trained, to evaluate more rigorously the generalization ability of the model. The test was performed on seven images with 91 peach fruits and resulted in an AP of 0.77.

Figure 10 shows the visual result for the first test image. The peculiarity of this image is the illumination, in which the light is too reflected in the leaves. The color of the fruit is also reddish. These features are completely different from the images found in the training dataset. As can be seen in Figure 10a,b, the model predicted non-occluded and clustered fruits well, but failed to detect some occluded fruits. In this image, the model only fails in the most difficult case of detection (very occluded). To improve the accuracy of the model, samples from this orchard need to be included in the training dataset.

Figure 11 shows the result for another test image, where the lighting is not good, as the exposure of the camera did not help to distinguish fruits from leaves. Moreover, there were several small fruits. In Figure 11a,b, it can be seen that the model did not detect many small fruits. This issue is a common problem with object detection models. One way to solve this problem is to take pictures close to the trees.

(**a**)

(**b**)

Figure 10. Test result in a different orchard (sample-1): (**a**) shows a test image and its manually annotated fruits; (**b**) shows the predicted fruits detection by the model. The green rectangular boxes are corrected detections, and the red circles are illustrative missing detections.

(**a**)

(**b**)

Figure 11. Test result in a different orchard (sample-2): (**a**) shows a test image and its manually annotated fruits; (**b**) shows the predicted fruit detection by the model. The green rectangular boxes are corrected detections, and the red circles are illustrative missing detections.

The distinctive feature of the Figure 12 is the many yellow leaves. This feature might confuse the model, since the color of the fruits in the training images is mostly yellow. Figure 12a,b shows the excellent detection results for this particular image. The model copes well with yellow leaves, which could affect the prediction of the model. There is only one false detection for yellow leaves, one for occluded and two for small fruits.

(**a**)

(**b**)

Figure 12. Test result in a different orchard (sample-3): (**a**) shows a test image and its manually annotated fruits; (**b**) shows the predicted fruits detection by the model. The green rectangular boxes are corrected detections, and the red circles are illustrative missing detections.

Figure 13 has two special features: many yellow leaves and reddish fruits. Again, the model shows good generalization. For this image, there was only one miss detection for yellow leaves and one miss detection for occluded fruit. The model did well for clustered and red fruits.

(**a**)

(**b**)

Figure 13. Test results in a different orchard (sample-4): (**a**) shows a test image and its manually annotated fruits; (**b**) shows the predicted fruits detection by the model. The green rectangular boxes are corrected detections, and the red circles are illustrative missing detections.

3.4. Final Discussions

The results show great potential in using the faster R-CNN model for peach detection in an uncontrolled environment. The model can deal well with occluded fruits, bunches of fruits, and light changes, which are features of a non-controlled environment. For a more accurate evaluation of the model, a test was conducted in another orchard. The results showed that the model is robust, as it handles image detections that are completely different from the training data (e.g., the color of the fruit, the color of the leaves, and the lighting). One possible way to improve the detection accuracy is to extend this work by

adding images to the training data from a sequence of frames (from a video). This approach may help the model to better handle the problem of occlusion. Moreover, one may add images from different orchards and take the images closer to the tree.

4. Conclusions

This article contributes toward climate change mitigation through the application of precision agriculture using deep neural networks by providing the field's current status, resulting in higher yields and, consequently, less deforestation, and a reduction in natural resource consumption and greenhouse gas emissions. In this work, we applied the deep learning approach faster R-CNN object detection model to evaluate the detection of peaches in images from an orchard. Peaches have a specific color, size, shape, fruit clustering, and distribution in a tree. The results show that the model handles all of these peculiarities well, in terms of peach fruit detection, and achieves an AP of 0.90 for the test split images belonging to the same orchard of the training, and an AP of 0.77 for the test images belonging to a different orchard (that has a feature discussed earlier). This shows great performance when compared to hand-crafted fruit detection models. For future work, we propose performing the detection in a sequence of frames, to detect hidden fruits more easily. Moreover, one may add images from different orchards and take the images closer to the tree.

Author Contributions: Conceptualization: P.D.G.; data curation: E.T.A. and R.J.M.M.; formal analysis: E.T.A., P.D.G., M.P.S., H.P. and P.R.M.I.; funding acquisition: P.D.G.; investigation: E.T.A. and R.J.M.M.; methodology: E.T.A. and P.D.G.; project administration: P.D.G.; resources: R.J.M.M., M.P.S. and A.R.; software: E.T.A.; supervision: P.D.G.; validation: E.T.A.; visualization: E.T.A.; writing—original draft: E.T.A.; writing—review and editing: P.D.G. and H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research work is funded by the PrunusBot project—autonomous controlled spraying aerial robotic system and fruit production forecast, operation no. PDR2020-101-031358 (leader), consortium no. 340, initiative no. 140, promoted by PDR2020, and co-financed by the EAFRD and the European Union under the Portugal 2020 program.

Acknowledgments: The authors are thankful to Fundação para a Ciência e Tecnologia (FCT) and R&D Unit "Center for Mechanical and Aerospace Science and Technologies" (C-MAST), under project UIDB/00151/2020, for the opportunity and the financial support to carry on this project. The contributions of Hugo Proença and Pedro Inácio in this work were supported by FCT/MEC through FEDER—PT2020 Partnership Agreement under Project UIDB//50008/2021.

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

References

1. Ontario Ministry of Agriculture. Introduction to Sustainable Agriculture. 2016. Available online: http://www.omafra.gov.on.ca/english/busdev/facts/15-023.htm (accessed on 11 October 2021).
2. Balafoutis, A.; Beck, B.; Fountas, S.; Vangeyte, J.; van der Wal, T.; Soto, I.; Gómez-Barbero, M.; Barnes, A.P.; Eory, V. Precision Agriculture Technologies positively contributing to GHG emissions mitigation, farm productivity and economics. *Sustainability* **2017**, *9*, 1339. [CrossRef]
3. Alibabaei, K.; Gaspar, P.; Lima, T.M. Modeling evapotranspiration using Encoder-Decoder Model. In Proceedings of the 2020 International Conference on Decision Aid Sciences and Application (DASA), Sakheer, Bahrain, 8–9 November 2020; pp. 132–136.
4. Assunção, E.; Diniz, C.; Gaspar, P.; Proença, H. Decision-making support system for fruit diseases classification using Deep Learning. In Proceedings of the 2020 International Conference on Decision Aid Sciences and Application (DASA), Sakheer, Bahrain, 8–9 November 2020; pp. 652–656.
5. Häni, N.; Roy, P.; Isler, V. Apple Counting using Convolutional Neural Networks. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 2559–2565. [CrossRef]
6. Dorj, U.O.; Lee, M.; Yun, S.-s. An yield estimation in citrus orchards via fruit detection and counting using image processing. *Comput. Electron. Agric.* **2017**, *140*, 103–112. [CrossRef]
7. Bargoti, S.; Underwood, J. Deep fruit detection in orchards. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3626–3633. [CrossRef]

8. Koirala, A.; Walsh, K.; Wang, Z.; McCarthy, C. Deep learning—Method overview and review of use for fruit detection and yield estimation. *Comput. Electron. Agric.* **2019**, *162*, 219–234. [CrossRef]
9. Wang, Q.; Nuske, S.; Bergerman, M.; Singh, S. Automated Crop Yield Estimation for Apple Orchards. In *Experimental Robotics, Proceedings of the 13th International Symposium on Experimental Robotics, Québec City, QC, Canada, 18–21 June 2012*; Desai, J.P., Dudek, G., Khatib, O., Kumar, V., Eds.; Springer International Publishing: Heidelberg, Germany, 2013; pp. 745–758. [CrossRef]
10. Puttemans, S.; Vanbrabant, Y.; Tits, L.; Goedemé, T. Automated visual fruit detection for harvest estimation and robotic harvesting. In Proceedings of the 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA), Oulu, Finland, 12–15 December 2016; pp. 1–6. [CrossRef]
11. Bargoti, S.; Underwood, J.P. Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards. *J. Field Robot.* **2017**, *34*, 1039–1060. [CrossRef]
12. Häni, N.; Roy, P.; Isler, V. A Comparative Study of Fruit Detection and Counting Methods for Yield Mapping in Apple Orchards. *arXiv* **2020**, arXiv:1810.09499.
13. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 580–587. [CrossRef]
14. Girshick, R. Fast R-CNN. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1440–1448. [CrossRef]
15. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149. [CrossRef] [PubMed]
16. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask R-CNN. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2980–2988.
17. Sa, I.; Ge, Z.; Dayoub, F.; Upcroft, B.; Perez, T.; Mccool, C. DeepFruits: A Fruit Detection System Using Deep Neural Networks. *Sensors* **2016**, *16*, 1222. [CrossRef] [PubMed]
18. Koirala, A.; Walsh, K.B.; Wang, Z.X.; McCarthy, C. Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'MangoYOLO'. *Precis. Agric.* **2019**, *20*, 1107–1135. [CrossRef]
19. Liu, X.; Zhao, D.; Jia, W.; Ji, W.; Ruan, C.; Sun, Y. Cucumber Fruits Detection in Greenhouses Based on Instance Segmentation. *IEEE Access* **2019**, *7*, 139635–139642. [CrossRef]
20. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), San Diego, CA, USA, 20–26 June 2005; Volume 1, pp. 886–893.
21. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.E.; Fu, C.Y.; Berg, A. SSD: Single Shot MultiBox Detector. In Proceedings of the ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016.
22. Redmon, J.; Divvala, S.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
23. Liu, S.; Deng, W. Very deep convolutional neural network based image classification using small training sample size. In Proceedings of the 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, Malaysia, 3–6 November 2015; pp. 730–734. [CrossRef]
24. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778. [CrossRef]
25. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826. [CrossRef]
26. Yu, H.; Chen, C.; Du, X.; Li, Y.; Rashwan, A.; Hou, L.; Jin, P.; Yang, F.; Liu, F.; Kim, J.; et al. TensorFlow Model Garden. 2020. Available online: https://github.com/tensorflow/models (accessed on 12 October 2021).
27. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the ECCV 2014, Zurich, Switzerland, 6–12 September 2014; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 740–755.
28. Rezatofighi, H.; Tsoi, N.; Gwak, J.; Sadeghian, A.; Reid, I.; Savarese, S. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 15–20 June 2019; pp. 658–666. [CrossRef]

Chapter 4

Decision-making support system for fruit diseases classification using Deep Learning

The following work was developed by myself, P. Gaspar, C. Diniz and H. Proença between March and November 2020. The aim was to develop a classification of fruit diseases based on a lightweight deep learning model using a small dataset. The result shows the potential of using lightweight models for fruit disease classification when only a small amount of training datasets are available.

1

[1]This work was published in [27], E. Assunção, C. Diniz, P. D. Gaspar and H. Proença, "Decision-making support system for fruit diseases classification using Deep Learning," 2020 International Conference on Decision Aid Sciences and Application (DASA), 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9317219

Decision-making support system for fruit diseases classification using Deep Learning

Eduardo Assunção
C-MAST - Centre for Mechanical and Aerospace Science and Technologies
Dept. of Electromechanical Engineering
University of Beira Interior
Covilhã, Portugal
eduardo.assuncao@ubi.pt

Catarina Diniz
Dept. of Medical Sciences
University of Beira Interior
Covilhã, Portugal
catarina.8.diniz@gmail.com

Pedro Dinis Gaspar
C-MAST - Centre for Mechanical and Aerospace Science and Technologies
Dept. of Electromechanical Engineering
University of Beira Interior
Covilhã, Portugal
dinis@ubi.pt

Hugo Proença
IT - Instituto de Telecomunicações
Dept. of Computer Science
University of Beira Interior
Covilhã, Portugal
hugomcp@di.ubi.pt

2020 International Conference on Decision Aid Sciences and Application (DASA) | 978-1-7281-9677-0/20/$31.00 ©2020 IEEE | DOI: 10.1109/DASA51403.2020.9317219

***Abstract*—Fruit diseases are a continuous hazard to farmers. By applying computer vision-based techniques, precision agriculture can support the farmers in the decision making for fruit disease control. Features extraction is an essential task for the computer vision pipeline. Nowadays, in general, feature extraction for fruit diseases are handcrafted. However, empirical results in different domains confirm that features learned by Convolutional neural networks (CNNs) provide significant improvements in accuracy over handcrafted features. CNNs have been applied in many computer vision tasks, replacing the hand-engineered models. In general, a large-scale image dataset is necessary for training a CNN. However, there are not many fruit disease images available to compose the dataset. We propose to train a tiny and efficient deep convolutional network developed to run in the mobile devices to classify healthy peach fruits and three peach diseases. Based on transfer learning techniques and data augmentation strategies, the proposed model achieves a Macroaverage F1-score of 0.96. The model does not misclassify any disease class. This achievement shows the potential of using small CNN models for fruit disease classification when having a small quantity of training data.**

***Index Terms*—fruit diseases, deep convolutional network, small dataset, precision agriculture**

I. Introduction

Fruit diseases are a continuous hazard to farmers and responsible for substantial economic losses. The anthracnose is an example of fruit disease, whether left unchecked can cause severe fruit rot infection [1]. In this sense, precision agriculture provides many technologies for supporting farmer's decision making.

In the literature, there are various published works that use computer vision models for fruit disease identification or classification [2], [3], [4], [5], [6], [7], [8]. However, these models tend to use handcrafted feature vectors (i.e., designed by hand). These types of features have a low-level representation [9]. In the other hand, the convolutional neural networks (CNNs) can learn to extract low-level, mid-level, and high-level features [10]. Empirical results in different domains confirm that representations learned by CNNs provide significantly improvements in accuracy over handcrafted features [11].

Because the state-of-art CNNs have millions of parameters, for a good prediction accuracy and generalization, it is necessary to have large-scale training data. Unfortunately, for fruit diseases, there are not many images available for composing the dataset. A common approach to overcome the lack of large-scale dataset for deep learning training is to apply the transfer learning and data augmentation techniques [12]. In this article, we propose to apply the transfer learning using the CNN MobileNetV2 [13], pre-trained in the ImageNet dataset [14], and data augmentation, in order to investigate the diseases classification results in our relatively small peach fruit disease dataset, as illustrated in Fig. 1. Although the CNN approach has already been used in different context (e.g., leaf tree disease in large scale dataset [15]). The novelty of the current work is related to the prediction capabilities considering a small fruit disease dataset. Also, the model is designed to run on mobile devices. This paper is structured as follows. In Section II, summarizes the related work. Section III describes the methodology. Section IV includes the analysis and discussion of results. The conclusions are given in Section V.

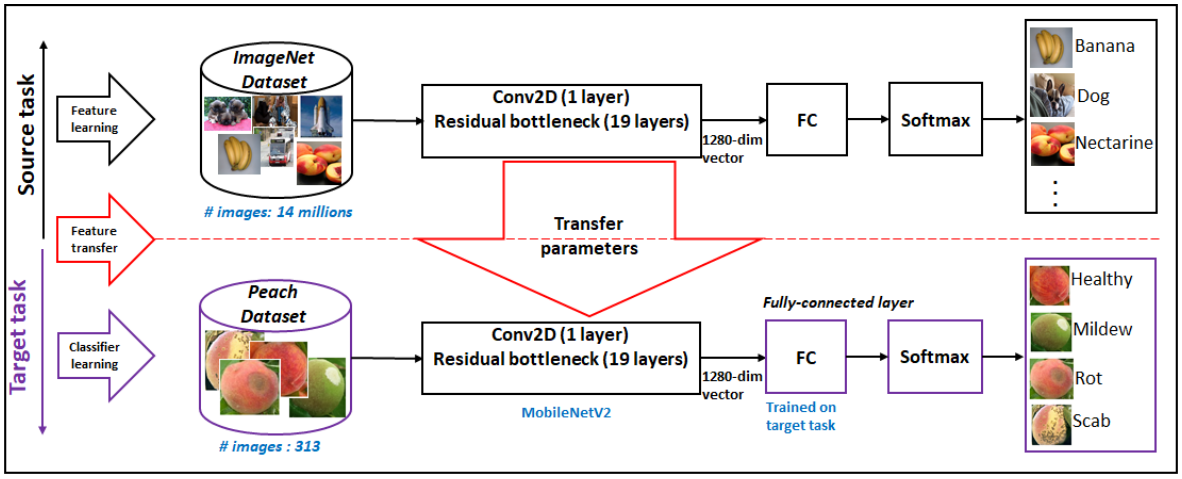


Fig. 1. MobileNetV2 transferring parameters: The model was firstly trained on the ImageNet dataset (source task). This model and its pre-trained parameters are available on the Tensorflow-Hub. Pre-trained parameters are transferred to the target task (fruit disease classification). Furthermore, a fully connected layer is trained on our peach disease dataset to compensate for the difference between the source and target data images.

II. Related Work

Dubey and Jalal [2] propose a method for apple disease (blotch, rot, and scab) classification. The pipeline of the method consists of image fruit segmentation applying K-means, feature extraction from segmented images (global color histogram, color coherence vector, and local binary), and then apply the support vector machine as a classifier. The method achieved an average precision of 0.93, using 50 images per class. This method required finding manually the useful features from the images to discriminate each fruit disease class.

Samajpati and Degadwala [3] fuse color and texture features from apple images and use the random forest to classify scab, rot, and blotch diseases. They used 70 images for training and 10 for testing of each class.This non-automatic method required selecting manually the features (from color and texture) representing each fruit disease class.

Bhange and Hingoliwala [16] suggest a solution for pomegranate fruit disease detection. The method starts with pre-processing by enhancing the images. After that apply K-means clustering algorithm for segmentation. For feature extraction, they construct a feature vector with color, texture, and morphology from the segmented image. As a classifier, they proposed to use K-nearest neighbors.

Similarly to [16], Behera et al. [17] propose an orange disease classification approach with pre-processing, segmentation, and feature extraction. In the pre-processing phase, it is performed image enhancement and L*a*b color transformation. It is used the Gary level co-occurrence matrix for extracting the following features: contrast, correlation, energy, homogeneity, mean, standard deviation, entropy, RMS, variance, smoothness, kurtosis, skewness, and IDM. The authors used the SVM classifier, perform testing in 40 images, and achieved an accuracy of 0.9.

Abirami and Thilagavathi [18] proposed a method for classification of fruit diseases using neural networks (NN) as classifier. First, they perform image pre-processing, then segmentation to produce binary images by thresholding and finally creating feature vectors applying the local binary pattern. They reported test accuracy for 100 images of 0.92 for bacterial disease and 0.86 for fungal diseases with ten hidden layers NN.

Ayyub and Manjramkar [7] performed a method for identifying and classifying rot, scab, and blotch apple fruit diseases. The feature extraction approach combines color, texture, and shape image information. Training a multiclass support vector machine, they report an average accuracy of 0.96.

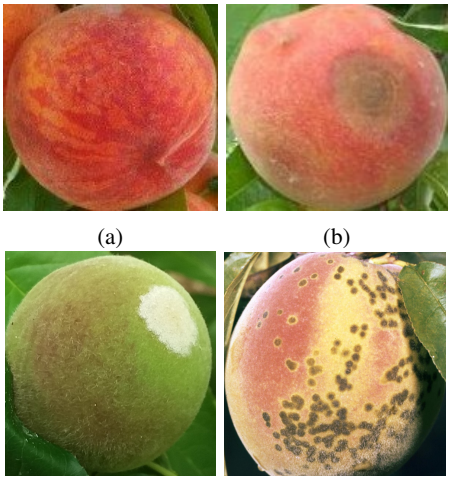
III. Materials and Methods

A. Dataset

Our peach dataset is composed of RGB images collected from the open websites platform Forestry Images [19], PlantVillage [20], Pacific Northwest Pest Management Handbooks [21], Utah State University [22], University of Georgia [23], as well as from the APPIZÊZERE (Associação de Protecção Integrada e Agricultura Sustentável do Zêzere). The dataset has 313 RGB images split into four classes: Healthy, Rot, Mildew, and Scab, as shown in the Table I. Fig. 2 shows the dataset class instances and their visual examples.

TABLE I. Dataset Statistics

Classes	Images	Training	Testing
Healthy	124	99	25
Rot	106	84	22
Mildew	58	46	12
Scab	25	20	5
Total	**313**	**249**	**64**



(a) (b) (c) (d)

Fig. 2. Peach fruit images and their classes: (a) Healthy. (b) Rot. (c) Mildew. (d) Scab.

As mentioned before, CNNs can suffer from poor generalization, whether trained with a small quantity of data. A common approach to overcome this problem is to apply data augmentation. We performed the data augmentation technique in our dataset, according to Table II. We have performed other augmentations, but theses ones shown in the table led to the best results.

TABLE II. Data Augmentation

Transformation	Values
Rotation	40
Width-shift	0.15
Height-shift	0.15
Zoom	0.2
Horizontal-flip	True

B. Model Assessment

To evaluate the model performance, we used the Precision (1), Recall (2), and F1-score (3), which is a function of Precision and Recall metrics. In the field of disease identification and classification, True Negative (TN) samples are dominant. It means that the dataset composition is uneven (i.e., unbalanced), and the TN can largely contribute to the model test results. In this context, F1-score is a suitable metric to assess the proposed method.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Where TP is True Positive (means the number of correctly classified); FP is False Positive (means the number of misclassified as belonging to the class); FN is False Negative (is the number of cases that should be classified as belonging to the class).

$$F1\text{-}score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{3}$$

C. Convolutional Neural Network Architectures

We have mentioned the deep neural networks overfitting problem due to a small quantity of data for training. Unfortunately, the available fruit diseases datasets have a relatively small quantity of images. Taking this restriction into account, we select the MobileNetV2 CNN model to investigate the peach fruit diseases. MobileNetV2 is a general-purpose computer vision neural network designed to support classification, object detection, semantic segmentation, and run in mobile devices and real-time applications.

1) MobileNetV1: The main key to reduce the computational cost and model size is splitting the typical convolution into two layers: a depthwise convolution and a pointwise convolution. The depthwise convolution is performed by applying one filter kernel per input channel and the pointwise convolution by applying a linear combination in the depthwise output layer. In total, the MobileNetV1 has 28 layers [24].

2) MobileNetV2: It has an initial fully convolution layer followed by 19 bottleneck layers, according to Table III. Its building block (bottleneck layer) is based on MobileNetV1 structure with two additional layers that perform expansion in the low-dimensional input to a high dimension and filtered with a lightweight depthwise convolution. In the end, the features are again projected back to a low-dimensional representation with a linear convolution.

TABLE III. MobileNetV2 structure: Each line describes a sequence of 1 or more equal layers, respected to *n*. All layers in the same line have the same number *c* of output channels. The first layer of each sequence has a stride *s* and all others use stride 1. The expansion factor applied to the input is *t*.

Input	Operator	*t*	*c*	*n*	*s*
224^2 x 3	conv2d	-	32	1	2
112^2 x 32	bottleneck	1	16	1	1
112^2 x 16	bottleneck	6	24	2	2
56^2 x 24	bottleneck	6	32	3	2
28^2 x 32	bottleneck	6	64	4	2
14^2 x 64	bottleneck	6	96	3	1
14^2 x 96	bottleneck	6	160	3	2
7^2 x 160	bottleneck	6	320	1	1
7^2 x 320	conv2d 1x1	-	1280	1	1
7^2 x 1280	avgpool 7x7	-	-	1	-
1 x 1 x 1280	conv2d 1x1	-	k	-	-

D. Implementation Details

The experiment was developed in the deep learning frameworks Keras, TensorFlow, and TensorFlow-Hub.

1) Keras & TensorFlow: Keras is a high-level deep learning API that runs on top of the machine learning platform TensorFlow, which efficiently executes low-level tensor operations on CPU, GPU, or TPU. We used the Keras application version 1.0.8 and TensorFlow-GPU 2.0.0. The image data preparation resize, and augmentation also was performed with Keras processing library.

2) TensorFlow Hub: TensorFlow Hub is a repository for publication and consumption of reusable parts of machine learning models across different tasks supporting transfer learning. In the TensorFlow Hub, a module is a piece of TensorFlow graph. In our work, we used a pre-trained MobilenetV2 classifier module. The image input dimension for the model is 224x224x3.

3) Experiment: Using the Keras API, a fully connected layer and a Softmax activation function were added to the model. We compile the model with optimizer adam, loss function sparse categorical crossentropy, and metrics accuracy. We split the dataset according to Table I and performed the training with 20 epochs and batch size of 16.

IV. Results and Discussion

This article proposes the use of a deep learning-based solution for classifying Rot, Mildew, and Scab diseases in peach fruits, which represents a contribution, considering that previous methods were all handcrafted. We used a MobileNetV2 CNN architecture, fine-tuned based in a set of images collected from well known datasets and data augmentation techniques. Fig. 3 and Fig. 4 show the training report for the best result achieved. The error for training and validation is about 0.2, and the difference between each other is very small, indicating that the model does not suffer overfitting for 20 training epochs.

The Table IV presents the classification report for the model assessment. In this table, we can evaluate the model for each class and the total performance of the model. We can see that the best classification performance for an individual class is the Scab disease with F1-score of 1.00. Fig. 2(d) shows that the visual black spots can provide useful features for discriminating the Scab disease. Rot and Mildew diseases both have classification results of 0.96 F1-score. And lastly, the Healthy fruit class with an F1-Score of 0.94. The total performance of the model achieved the Macro-average F1-score of 0.96. One important finding is that the only misclassification was for the Healthy fruit class (Fig. 5 summarizes the classification errors). The model does not misclassify any disease class. This result is an important achievement, because misclassification in the disease investigation for infection and control has a high cost.

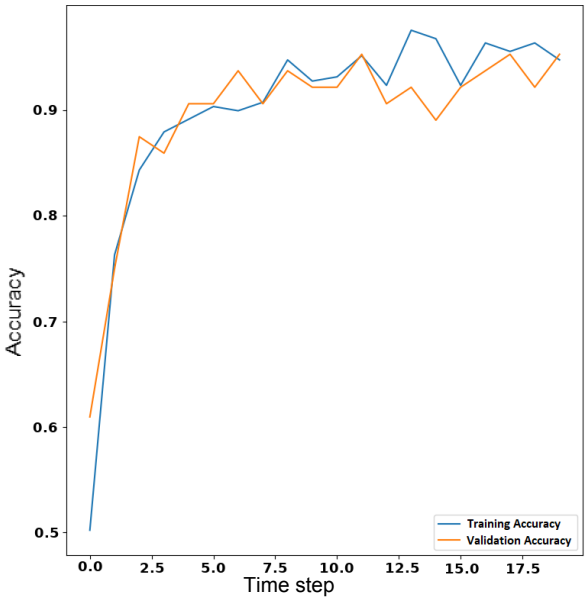


Fig. 3. Training and Validation Accuracy

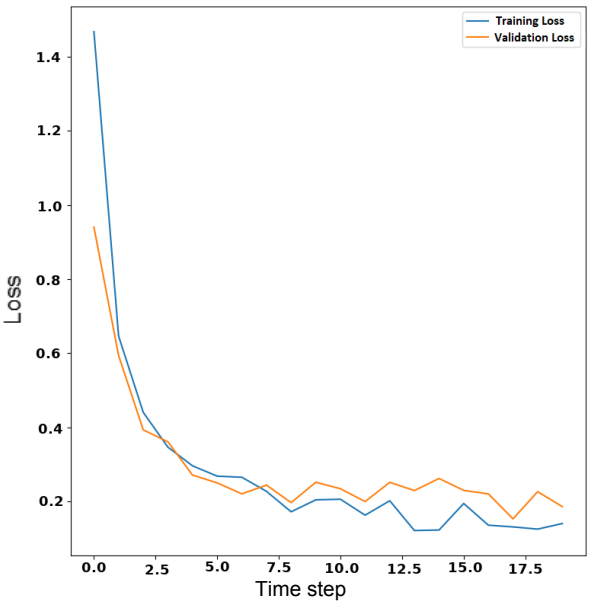


Fig. 4. Training and Validation Loss

TABLE IV. Classification Report

Class	Precision	Recall	F1-score	Support
Healthy	1.00	0.88	0.94	25
Mildew	0.92	1.00	0.96	12
Rot	0.92	1.00	0.96	22
Scab	1.00	1.00	1.00	5
Macro-avg F1-score	0.96	0.97	**0.96**	64

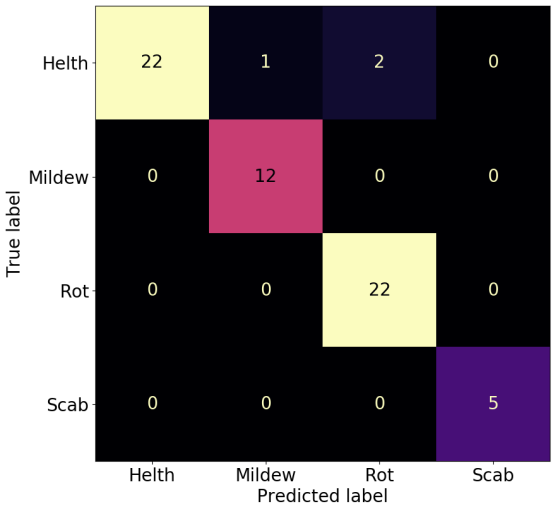


Fig. 5. Confusion Matrix.

V. Conclusions

Scab disease had the best individual classification with F1-score of 1.00, followed by Rot and Mildew classes with both of 0.96 F1-score. The Healthy class had a classification of 0.94 F1-Score. The total performance of the model achieved a Macro-average F1-score of 0.96. The model does not misclassify any disease class, which is very important in disease investigation for infection and control. These achievements show the potential of using CNN for fruit disease classification with a small quantity of training data.

Acknowledgment

This study is within the activities of project PrunusBot - Sistema robótico aéreo autónomo de pulverização controlada e previsão de produção frutícola (autonomous unmanned aerial robotic system for controlled spraying and prediction of fruit production), Operation n.º PDR2020-101-031358 (líder), Consortium n.º 340, Initiative n.º 140 promoted by PDR2020 and co-financed by FEADER under the Portugal 2020 initiative. The authors thank the opportunity and financial support to carry on this project to Fundação para a Ciência e Tecnologia (FCT) and R&D Unit "Centre for Mechanical and Aerospace Science and Technologies" (C-MAST), under project UIDB/00151/2020. The authors thank the support of APPIZÊZERE - Associação de Protecção Integrada e Agricultura Sustentável do Zêzere, and particularly to Eng. Anabela Barateiro, that provided part of the images used in the models.

References

[1] K. Peter, "Peach disease - anthracnose," 2017. [Online]. Available: https://extension.psu.edu/peach-disease-anthracnose

[2] S. Dubey and A. Jalal, "Adapted approach for fruit disease identification using images," *International Journal of Computer Vision and Image Processing*, vol. 2, pp. 51–65, 01 2012.

[3] B. J. Samajpati and S. D. Degadwala, "Hybrid approach for apple fruit diseases detection and classification using random forest classifier," in *2016 International Conference on Communication and Signal Processing (ICCSP)*, 2016, pp. 1015–1019.

[4] K. Kaur and C. Marwaha, "Analysis of diseases in fruits using image processing techniques," in *2017 International Conference on Trends in Electronics and Informatics (ICEI)*, 2017, pp. 183–189.

[5] S. K. Behera, L. Jena, A. K. Rath, and P. K. Sethy, "Disease classification and grading of orange using machine learning and fuzzy logic," in *2018 International Conference on Communication and Signal Processing (ICCSP)*, 2018, pp. 0678–0682.

[6] S. Abirami and M. Thilagavathi, "Classification of fruit diseases using feed forward back propagation neural network," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, 2019, pp. 0765–0768.

[7] S. R. N. M. Ayyub and A. Manjramkar, "Fruit disease classification and identification using image processing," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, 2019, pp. 754–758.

[8] M. T. Vasumathi and M. Kamarasan, "Fruit disease prediction using machine learning over big data," in *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 7, 2019.

[9] M. El-gayar, H. Soliman, and N. meky, "A comparative study of image low level feature extraction algorithms," *Egyptian Informatics Journal*, vol. 14, no. 2, pp. 175 – 181, 2013.

[10] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1717–1724.

[11] "Feedforward neural networks," class notes for Natural Language Processing, Dept. of Computer Science, Columbia University, New York, NY, Spring 2019. [Online]. Available: http://www.cs.columbia.edu/ mcollins/ff.pdf

[12] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," in *Journal of Big Data*, vol. 6, 2019.

[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520.

[14] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[15] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016. [Online]. Available: https://www.frontiersin.org/article/10.3389/fpls.2016.01419

[16] M. A. Bhange and H. A. Hingoliwala, "A review of image processing for pomegranate disease detection," in *International Journal of Computer Science and Information Technologies (IJCSIT)*, vol. 6, 2015, pp. 92–94.

[17] S. K. Behera, L. Jena, A. K. Rath, and P. K. Sethy, "Disease classification and grading of orange using machine learning and fuzzy logic," in *2018 International Conference on Communication and Signal Processing (ICCSP)*, 2018, pp. 0678–0682.

[18] S. Abirami and M. Thilagavathi, "Classification of fruit diseases using feed forward back propagation neural network," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, 2019, pp. 0765–0768.

[19] Image Categories. Forestry Images. Accessed May, 2020. [Online]. Available: https://www.forestryimages.org/index.cfm

[20] PlantVillage. Accessed May, 2020. [Online]. Available: https://plantvillage.psu.edu/topics/peach/infos

[21] Pacific Northwest Pest Management Handbooks. Accessed May, 2020. [Online]. Available: https://pnwhandbooks.org/

[22] Integrated Pest Management. Utah State University. Accessed May, 2020. [Online]. Available: https://utahpests.usu.edu/ipm/notes_ag/fruit-powdery-mildew

[23] College of Agricultural and Environmental Sciences. University of Georgia. Accessed May, 2020. [Online]. Available: https://www.caes.uga.edu/news-events/news/story.html?storyid=5097story=Peach-Scab

[24] 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," *ArXiv*, vol. abs/1704.04861, 2017.

Chapter 5

Real-Time Weed Control Application Using a Jetson Nano Edge Device and a Spray Mechanism

The following work was developed by myself, P. Gaspar, M. Ricardo, M. Simões, K. Alibabaei, A. Veiros and H. Proença between June 2021 and July 2022. The aim was to develop a new mechanism and algorithm for weed control, using data from previous scientific work and new data from the orchard in Portugal. The results obtained were considered very encouraging.

1

[1]This work was published in [28], E. Assunção, P. D. Gaspar, M. Ricardo, M. Simões, K. Alibabaei, A. Veiros and H. Proença, "Real-TimeWeed Control Application Using a Jetson Nano Edge Device and a Spray Mechanism," Remote Sensing, vol. 14, no. 17, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/17/4217

remote sensing

MDPI

Article

Real-Time Weed Control Application Using a Jetson Nano Edge Device and a Spray Mechanism

Eduardo Assunção [1,2,3], Pedro D. Gaspar [1,2,*], Ricardo Mesquita [2], Maria P. Simões [4], Khadijeh Alibabaei [1,2], André Veiros [2] and Hugo Proença [3]

1 C-MAST Center for Mechanical and Aerospace Science and Technologies, University of Beira Interior, 6201-001 Covilhã, Portugal; eduardo.assuncao@ubi.pt (E.A.); k.alibabaei@ubi.pt (K.A.)
2 Department of Electromechanical Engineering, University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal; ricardo.mesquita@ubi.pt (R.M.); andre.veiros@ubi.pt (A.V.)
3 Instituto de Telecomunicações, Department of Computer Science, University of Beira Interior, 6201-001 Covilhã, Portugal; hugomcp@di.ubi.pt
4 School of Agriculture, Polytechnic Institute of Castelo Branco, 6000-084 Castelo Branco, Portugal; mpaulasimoes@ipcb.pt
* Correspondence: dinis@ubi.pt

Abstract: Portable devices play an essential role where edge computing is necessary and mobility is required (e.g., robots in agriculture within remote-sensing applications). With the increasing applications of deep neural networks (DNNs) and accelerators for edge devices, several methods and applications have been proposed for simultaneous crop and weed detection. Although preliminary studies have investigated the performance of inference time for semantic segmentation of crops and weeds in edge devices, performance degradation has not been evaluated in detail when the required optimization is applied to the model for operation in such edge devices. This paper investigates the relationship between model tuning hyperparameters to improve inference time and its effect on segmentation performance. The study was conducted using semantic segmentation model DeeplabV3 with a MobileNet backbone. Different datasets (Cityscapes, PASCAL and ADE20K) were analyzed for a transfer learning strategy. The results show that, when using a model hyperparameter depth multiplier (DM) of 0.5 and the TensorRT framework, segmentation performance mean intersection over union (mIOU) decreased by 14.7% compared to that of a DM of 1.0 and no TensorRT. However, inference time accelerated dramatically by a factor of 14.8. At an image resolution of 1296 × 966, segmentation performance of 64% mIOU and inference of 5.9 frames per second (FPS) was achieved in Jetson Nano's device. With an input image resolution of 513 × 513, and hyperparameters output stride OS = 32 and DM = 0.5, an inference time of 0.04 s was achieved resulting in 25 FPS. The results presented in this paper provide a deeper insight into how the performance of the semantic segmentation model of crops and weeds degrades when optimization is applied to adapt the model to run on edge devices. Lastly, an application is described for the semantic segmentation of weeds embedded in the edge device (Jetson Nano) and integrated with the robotic orchard. The results show good spraying accuracy and feasibility of the method.

Keywords: semantic segmentation; crop and weed; edge device; precision agriculture

Citation: Assunção, E.; Gaspar, P.D.; Mesquita, R.; Simões, M.P.; Alibabaei, K.; Veiros, A.; Proença, H. Real-Time Weed Control Application Using a Jetson Nano Edge Device and a Spray Mechanism. *Remote Sens.* **2022**, *14*, 4217. https://doi.org/10.3390/rs14174217

Academic Editor: Mario Cunha

Received: 30 July 2022
Accepted: 18 August 2022
Published: 26 August 2022

1. Introduction

Weed control is a practice aimed at reducing competition among plants for water and nutrients. Cultivation techniques to control the development of weeds are commonly referred to as soil management. The application of herbicides at the beginning of the vegetative cycle is especially important in weed control because it is one of the critical factors for success [1]. Automatic weed detection is one of the viable solutions for the efficient reduction in or exclusion of chemical products in the field. Studies have focused

on modern approaches that automatically analyze and evaluate weeds in images (e.g., crop and weed segmentation).

Edge computing approaches have been proposed where the computation of data is performed locally, as in robots for agriculture. Examples of technical and technological solutions that can benefit from this approach can be found in [2–12]. Furthermore, it is desired to balance accuracy, throughput, and power management, which is essential in various fields such as the Internet of Things (IoT), robotics, autonomous driving, and drone-based surveillance [13]. Most modern computer vision systems for weed control are based on desktop computers (i.e., the computer is heavy, not portable, and must be connected to the power grid) [14–18]. This means that these systems are heavy, expensive, and require much electrical power. These problems render this approach (i.e., desktop-based) impractical for use in small and low-cost orchard robots.

Milioto et al. [14] proposed an encoder–decoder model based on convolutional neural networks (CNNs) for crop and weed segmentation. They used RGB images and produced the excess green, excess red, color index of vegetation extraction, and normalized difference vegetation indices as inputs to the model. The best performance was 59% mean intersection over union (mIOU) on the desktop device, and an inference time of 190 ms on the Jetson TX2. This model is relatively heavy compared to others, being able to perform only 5.2 frames per second (FPS) inferences at an image resolution of 512 × 384 on the Jetson TX2 device.

McCool et al. [15] implemented a lightweight segmentation model based on deep convolutional neural networks (DCNNs) that could be used in robotic platforms to segment crops and weeds in images. They used a desktop computer with a graphics processing unit (GPU) card for training and inference. The model was trained on the Crop Weed Field Image Dataset (CWFID) with an image resolution of 1296 × 966, and achieved an accuracy of 93.9% with an inference time of 8.33s using an Inception-v3 backbone.

Khan et al. [16] proposed the CED-NET encoder–decoder semantic segmentation method for crop and weed segmentation. Their model had four networks, where two networks are trained independently for segmenting crops, and the other two for weeds. They claimed that this approach to extract features at different scales provides coarse-to-fine predictions and reduces the number of network parameters. Using a desktop with a GPU, they achieved a performance of 77% mIOU in the CWFID dataset.

Wang et al. [17] implemented an encoder–decoder deep-learning network for pixel-wise semantic segmentation of crops and weeds. In addition, three images pre-processing were proposed to improve the model performance. They investigated the model by training and testing in two different datasets: sugar beet and oilseed. The best segmentation performance was 89% mIOU. Using a desktop with a GPU, the inference time for an image with a dimension of 1296 × 966 was 100 ms.

Fawakherji et al. [18] proposed a deep-learning-based method for crop and weed classification using two different convolutional neural networks (CNNs). The first network performs pixelwise segmentation between vegetation and soil. After segmentation, each plant is classified into crops or weeds using the second network. For the semantic segmentation network, they used the UNet structure with the VGG16 backbone. For the classification network, they used a fine-tuned model of VGG16 that leveraged deep CNN's object classification capabilities. For training and testing, they used the NVIDIA GTX 1070 GPU and achieved 87% correctly detected crops and 77% correctly detected weeds.

Olsen [19] proposed a real-time weed control system for a mobile robot. The author used a set of deep neural networks to classify nine types of weed images. In the Jetson Nano device, the MobileNetV2 model achieved an inference time of 29.0 ms; the ResNet-50 architecture achieved an inference time of 59.8 ms; the Inception model ran at 91.9 ms; and the VGG16 model achieved an inference speed of 166.3 ms. This work addresses the problem of image classification. That is, the model performs inference on whether weeds are present and which type they are. However, there is no spatial information (location) about the detected weeds in the image, unlike the semantic segmentation model.

Partel et al. [20] developed an innovative sprayer that uses object detector Tiny Yolov3 to distinguish target portulaca plants, sedge weeds, from nontarget pepper plants and precisely spray to the desired location. Using an NVIDIA Jetson TX2 device and an image resolution of 1024 × 256, it achieved overall precision and recall of 59% and 44%, respectively, and the framework could handle 22 FPS. The experiment was carried out in a simulated field of crops and weeds that did not correspond to the natural environment, which is normally a dense vegetable field.

Using the SegNet encoder–decoder network for semantic segmentation, Abdalla et al. [21] conducted a study on weed segmentation using fine-tuning and data-augmentation techniques. They used SegNet with a VGG16 backbone at an image resolution of 3888 × 5184, and the model achieved an inference time of 230 ms (i.e., 4.3 FPS) and mIOU of 87% in a desktop computer equipped with a GPU card.

Asad and Bais [22] used the SegNet and UNET semantic segmentation models with VGG16 and ResNet-50, classifying crop and background pixels as one class, and all other vegetation as the second class. At an image resolution of 1440 × 960, the SegNet model based on ResNet-50 showed the best results with an mIOU of 82%. The experiments were performed on a desktop computer equipped with a GPU card, and the authors did not report the inference time.

Ma et al. [23] used the SegNet semantic segmentation model based on a fully convolutional network with the AlexNet backbone to segment rice seedlings, weeds, and the background. At an image resolution of 912 × 1024, the model achieved an mIOU of 62% and an inference time of 0.6 s (it could process 1.6 FPS). The experiments were performed on a desktop computer equipped with a GPU card.

Lameski et al. [24] presented a carrot–weed dataset with RGB images of young carrot seedlings. The dataset contains 39 images with the same size of 3264 × 2448. The dataset consists of 311,620,608 pixels, comprising 26,616,081 pixels of carrot plants, 18,503,308 pixels of weed plants, and 266,501,219 pixels of soil. The authors conducted the initial experiments on the dataset using the SegNet semantic segmentation model and achieved the best accuracy of 0.641 for the segmentation of weeds, plants, and soil. In this work, there is no information about the inference time and the device for training and inference.

Naushad et al. [25] used VGG16 and wide residual networks (WRNs) to classify land use and land cover in RGB band images (at 64 × 64 resolution) from the EuroSAT dataset. A transfer-learning technique was used to improve the accuracy. The best accuracy was achieved by the WRN model with 99.17%.

Nanni et al. [26] proposed an ensemble method for semantic segmentation combining DeepLabV3, HarDNet-MSEG, and Pyramid Vision Transformers. The model was trained and evaluated on different datasets covering five scenarios: polyp detection, skin detection, leukocyte recognition, environmental microorganism detection, and butterfly recognition. According to the authors, the model provides state-of-the-art results.

Autonomous mobile robots use remote-sensing techniques and technologies for several agricultural tasks such as monitoring, fertilizing, herbicide spraying, or harvesting.

Computer vision applications can run on mobile devices by using deep-learning frameworks (e.g., TensorRT or Tensorflow Lite) and accelerators. For this, some optimization is needed in the original or standard computer-vision models to minimize memory and computational footprints [27]. Although preliminary studies investigated the inference performance for the semantic segmentation of crops and weeds in mobile edge devices [14,19,20], to the best of our knowledge, no study has been conducted to show the performance degradation (mIOU) when applying the required optimization to the model to operate in such edge devices. The purpose of this study is to:

1. Perceive the relationship between the model tuning hyperparameters of the DeeplabV3 semantic segmentation model with MobileNet backbone to improve inference time and its impact on segmentation performance (using the mIOU metric).
2. Evaluate the suitability of the Cityscapes, PASCAL, and ADE20K datasets for a transfer-learning strategy in crop and weed segmentation.

3. Propose and evaluate a real-time weed control application using a Jetson Nano Edge device and a spray mechanism.

The remainder of this paper is organized as follows. The hardware used to train and infer the model, the dataset used, an explanation of the semantic segmentation model, including the used backbone, the model optimization, and the practical application, are presented in Section 2. The results of the proposed model when applied to the weed datasets and the practical application are presented along with the related analysis and discussion in Section 3. The conclusion of the paper is in Section 4.

2. Materials and Methods

2.1. Resources for Network Training and Deployment

A desktop computer with an Intel Core i7-4790 CPU (3.60 GHz), GeForce RTX 2080 8 GB GPU, 16 GiB RAM, and Ubuntu 18.04 was used for training and obtaining the base model.

A Jetson Nano edge device was used for the deployment. NVIDIA's Jetson family features a heterogeneous CPU–GPU architecture, small form factor, light weight, and low power consumption, and is one of the most widely used edge device families with accelerators for machine-learning inference. It can run neural networks in parallel for applications such as image classification, object detection, and segmentation (Mittal, 2019). It works with an inserted micro-SD card with the system image. Table 1 shows the technical specifications of the Jetson Nano Developer Kit.

Table 1. Technical specifications of Jetson Nano.

Item	Spec
AI performance	472 GFLOPs
GPU	128-core NVIDIA
CPU	Quad-Core Arm
Memory	4 GB 64-bit
Power	5 W | 10 W

Figure 1 shows the general outline of this work, starting with the acquisition and preparation of the dataset (Figure 1A). The model was trained to obtain the base models (Figure 1B). Then, the model was optimized (post-training) (Figure 1C). Lastly, the model was deployed on the Jetson Nano device (Figure 1D). The next subsections provide more details.

2.2. Dataset

Two datasets were used in this work: CWFID (Section 2.2.1) and the Weeds dataset (Section 2.2.2). The CWFID was used to investigate how a model degrades when optimization is applied to adapt the model to run in edge devices, and the Weeds dataset was used for the real-time application proposal.

2.2.1. CWFID

The benchmark Crop Weed Field Image Dataset (CWFID) [28] consists of crops (carrots) and weeds. It was used to train and evaluate the models. For each image, a manual annotation (ground truth) per pixel (e.g., green mask for crops, and red mask for weeds) is available for crops and weeds. Tables 2 and 3 summarize the details of the dataset.

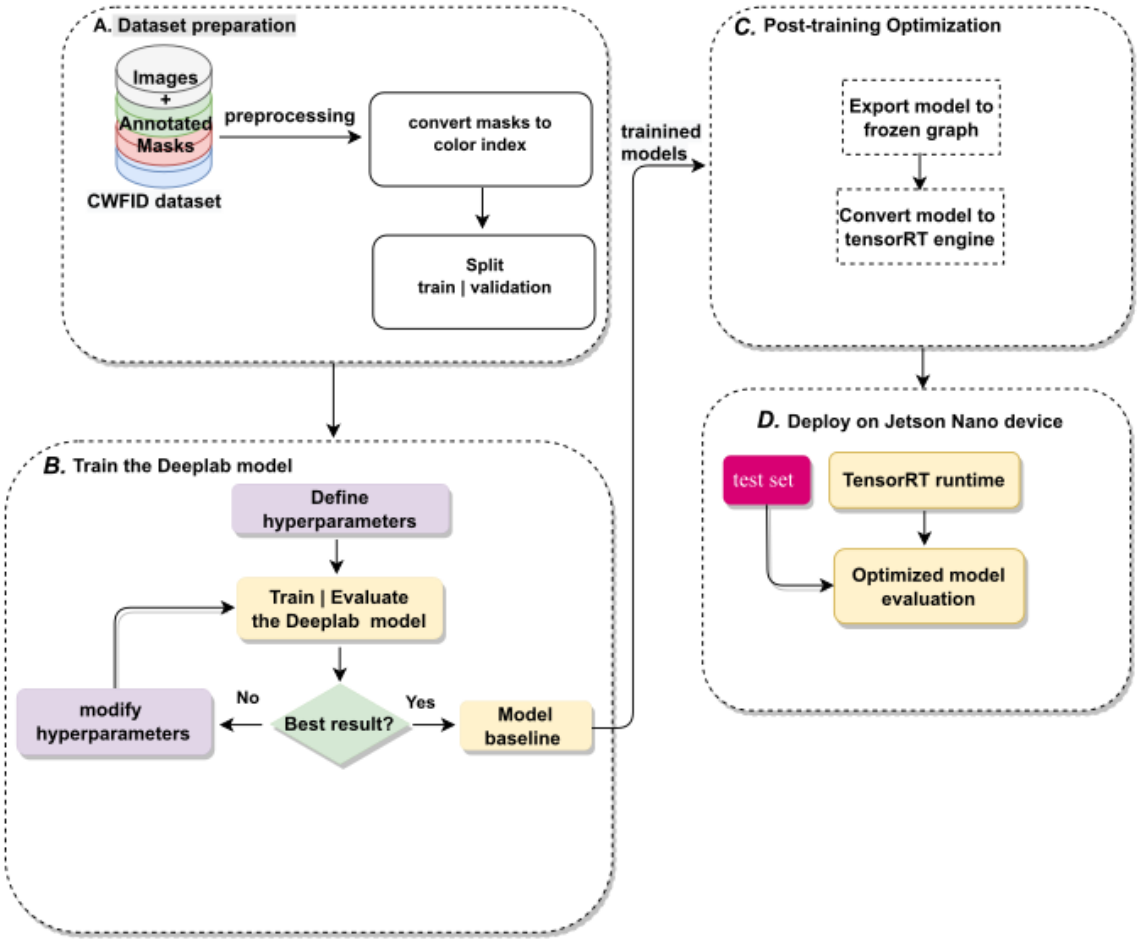


Figure 1. General overview of the model training and inference process carried out in this work. (**A**) Illustrate the collection and preparation of the dataset. (**B**) Show the training process to obtain the base models. (**C**) Show the model optimization. (**D**) Show the deployment on the Jetson Nano device.

Table 2. Scope of the CWFID dataset and split.

Parameter	Value
Number of images	60
Number of labeled plants	494
Number of labeled crop	162
Number of labeled weed	332
Dataset split	
Number of training images	42
Number of test images	18

Table 3. CWFID dataset pixels information.

Item	Back Ground	Weed Class	Crop Class
Total of pixels	69,580,840	4,322,897	1,212,423
Training pixels	49,922,402	3,093,709	817,137
Test pixels	19,658,438	1,229,188	395,286

2.2.2. Weed Dataset for Real-Time Application

A weed dataset was assembled to train the model for the real-time application experiment. The images were captured using a Sony DSC-RX100M2 camera. A total of 12 RGB images were selected and resized into a resolution of 513 × 513 (the input size of the

pretrained model used for fine tuning). Subsequently, the images were manually labeled. Figure 2 shows two examples of images in the weed dataset for the real-time application experiment. The images are shown at the top, and the respective image, labeled weed (i.e., ground-truth mask) , is shown at the bottom. Tables 4 and 5 summarize the weed dataset for the real-time application experiment.

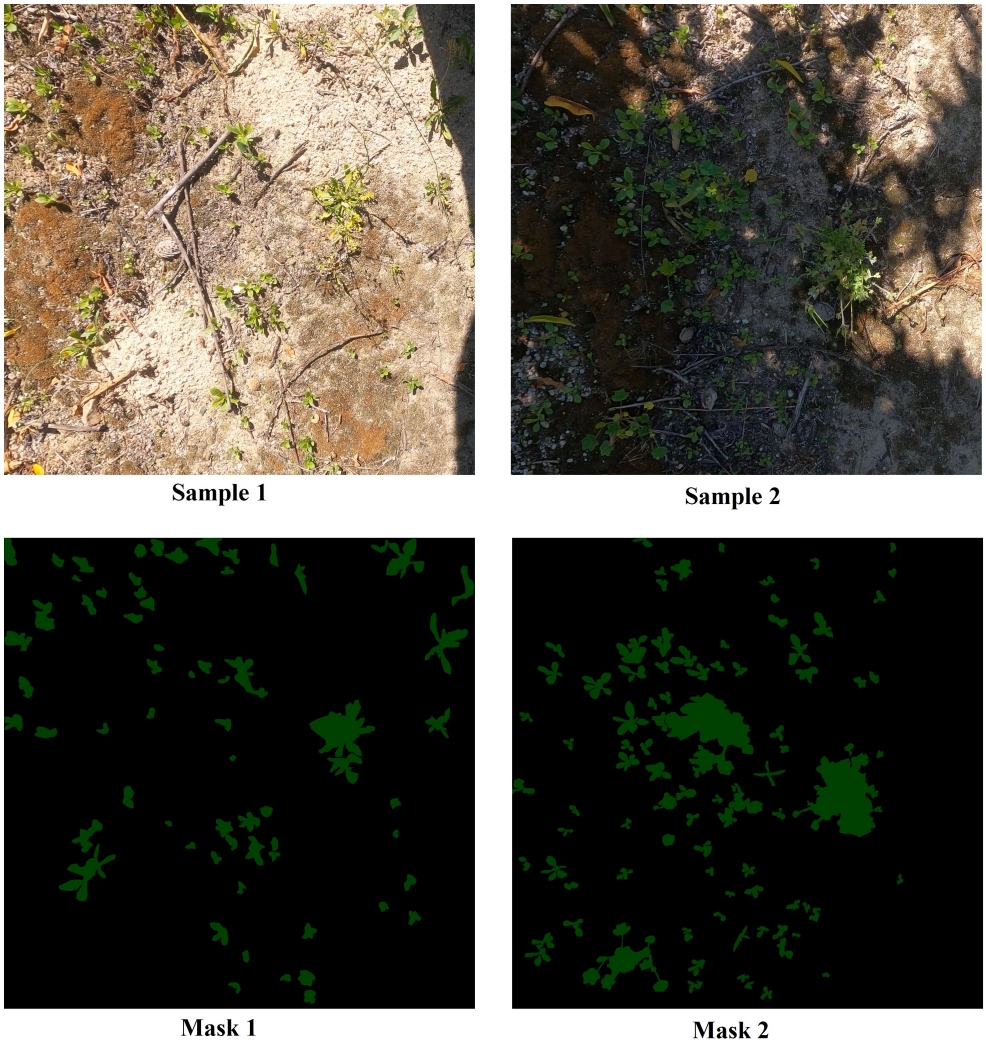


Figure 2. Examples of images in the weed dataset for the real-time application experiment.

Table 4. Scope of the weed dataset for the real-time experiment.

Parameter	Value
Number of images	12
Number of labeled weeds	847
Dataset split	
Number of training images	10
Number of validation images	2

Table 5. Information about the pixels of the weed dataset.

Item	Background	Weed Class
Total of labeled pixels	25,934,802	1,789,998

2.3. DeepLab Model and Training

2.3.1. Semantic Segmentation

Semantic segmentation is a dense prediction task that assigns semantic labels to each pixel in an image [29]. Some state-of-the-art models are based on an encoder–decoder network structure, as shown in Figure 3. Encoder–decoder is a group of models that learn to map data from an input space to an output space over a two-level network [30].

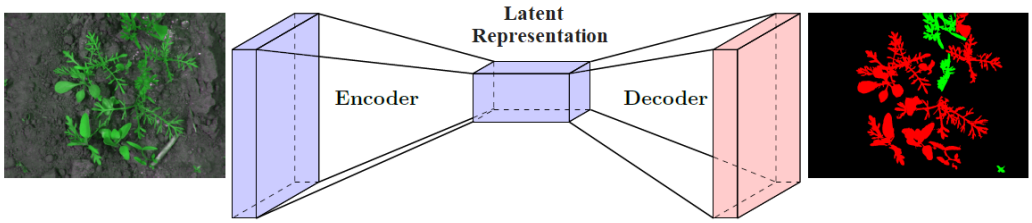


Figure 3. General structure of semantic segmentation based on DCNN.

Long et al. [29] developed a method for semantic segmentation by adapting a deep-learning classification network to perform dense prediction. A fully convolutional network (FCN) generates coarse output maps with reduced dimensions (i.e., downsampling). The feature maps are brought back to the input size by stacking deconvolutional layers (i.e., upsampling).

An encoder–decoder semantic segmentation model for medical images, U-NET, was developed by Ronneberger et al. [31]. Similarly, in [29], an FCN was used in the encoder part to capture contextual information. In this phase, due to the max-polling operation, the feature maps are contracted. In the decoder, successive expansion is performed using the mirrored structure of the encoder, but upsampling operators replace max polling.

Chen et al. [32] proposed a semantic segmentation model that combines the responses at the last CNN layer with a fully connected conditional random field (CRF). This approach adds more context to the model and helps in locating segment boundaries with better accuracy. In [30] contains the main semantic segmentation strategies.

Defining a network backbone while considering the most important aspects for a particular application is a crucial task. In our study, time is an important aspect, since the application needs to perform inferences in real-time. Another aspect is the lightness of the model, since the hardware is a small, low-cost, and low-power portable device with limited processing power (cores) and memory. Considering these aspects, the MobilenetV2 [33] was selected, as it met these specifications.

In this work, state-of-the-art semantic segmentation model DeepLabV3 [34] was employed with the MobilenetV2 backbone by using the Tensorflow Model Garden (TMG) framework [35] to perform the experiments. The DeelLabV3 was chosen because it is a very versatile model where the Mobilenet backbone could be used for the purposes of this work, such as model shrinking with hyperparameters. Figure 4 illustrates the architecture of the DeepLabV3 model, as originally described.

2.3.2. Network Backbone

In an image segmentation model based on deep learning, the first part consists of a DCNN. Its main goal is to extract features from the input image. This feature extractor is usually referred to as the backbone of the model. In this work, the MobilenetV2 DCNN was used as the backbone. The architecture of MobilinetV2 was customized by adjusting the depth multiplier (DM) and output stride (OS) hyperparameters to achieve the desired performance for the application (i.e., light weight and fast inference time). To optimize the model for light weight, the DM hyperparameter for model shrinkage was adjusted. In particular, the DM hyperparameter affects the number of model input channels for each layer. For example, at DM = 1.0, the model had the original number of input channels; at DM = 0.5, the model had half the number of input channels. In this study, the experiments

were conducted with DM = 1.0 and DM = 0.5. In addition, the DeepLabV3 model implements atrous convolution to explicitly control how densely features should be computed rather than using pooling in a fully convolutional network. The OS hyperparameter is the ratio of the size of the input image and the size of the last output feature map in the encoder. Values of 8, 16, and 32 were used for OS to investigate the trade-off between accuracy and inference time, as this hyperparameter affects segmentation accuracy and inference time.

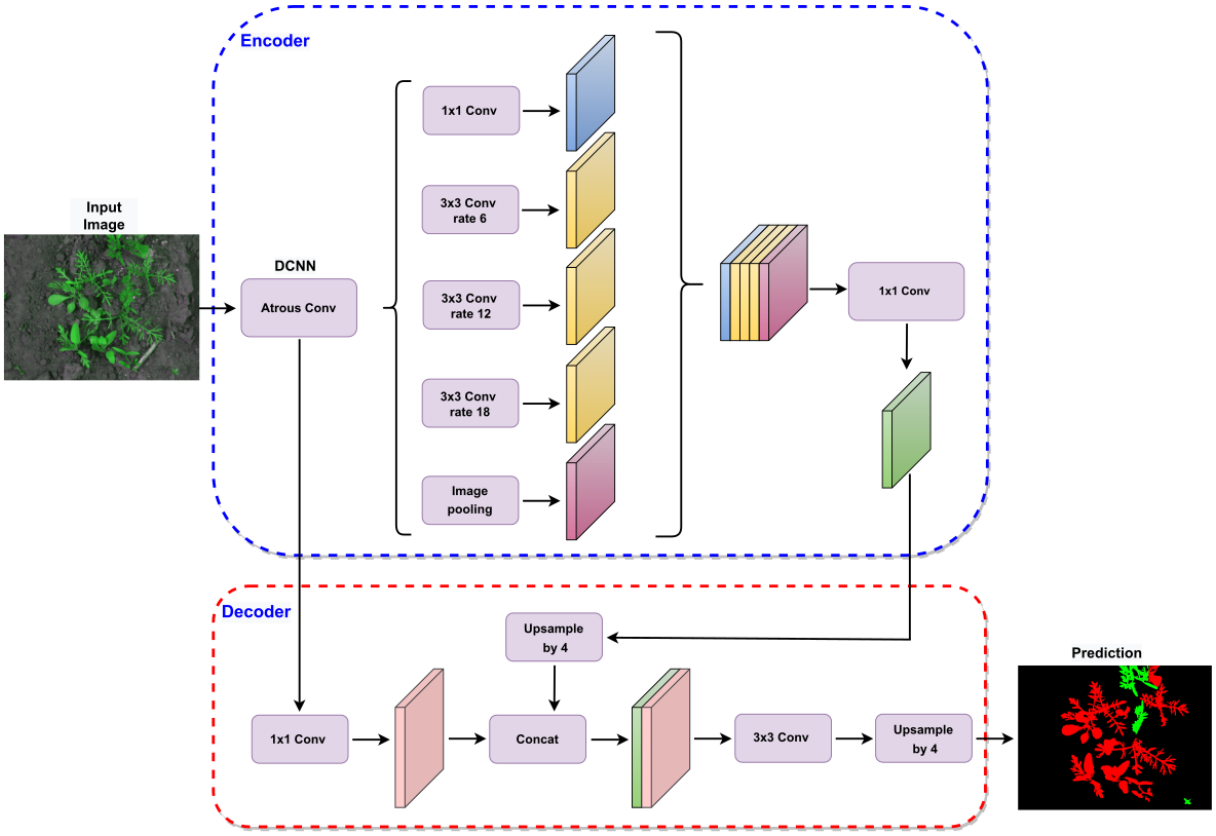


Figure 4. DeepLabV3 model illustration, adapted from [34].

2.3.3. Network Training

Software tools were Python 3.6 with the TensorFlow Deep Learning package and the Tensorflow Model Garden (TMG) framework. Using a fine-tuning strategy is more effective than training deep networks from scratch [36] . In this work, we performed fine tuning as recommended in [36]. TMG provided a fine-tuning strategy to train the last layers of the DeepLabV3 model for a new segmentation dataset. DeepLabV3 was trained on the basis of pretrained models on the Cityscape [37], ADE20k [38], and PASCAL [39] datasets to reduce training time and investigate which of these models provided the best segmentation results. The training configurations were as suggested in [35]. Specifically, ADAM learning rate = 0.001, ADAM momentum = 0.9, ADAM epsilon = 1×10^{-8}, and base learning rate of 0.0001 were used for 30,000 iterations with a batch size of 2.

2.4. Post-Training Optimization

After training, the TMG framework provided a set of *checkpoint* files (i.e., the trained model). Then, the checkpoint files were converted into a frozen graph using a tool (script) available in the TMG framework. Then, the frozen graph was converted (optimized) to run on the Tensorflow Real-Time (TensorRT) *engine* using the TensorRT class converter.

TensorRT is a software development kit (SDK) developed by NVIDIA Corporation to optimize inference time for deep learning on NVIDIA graphics processing units (GPUs) [40]. It works on top of already trained networks by combining layers and optimizing kernel

selection to improve latency, output capacity, energy efficiency, and memory consumption [41]. An example of TensorRT optimization is model quantization from 32-bit floating-point numbers to 16-bit (FP16) or 8-bit integers (INT8). In addition, TensorRT performs on an optimized inference engine based on a trained network.

2.5. Model Assessment

2.5.1. Computation Footprint

The metric of floating-point operations (FLOPs) was used to measure the computational complexity of the models, where addition and multiplication were both considered as one FLOP [42]. The number of FLOPs required by the model to perform an inference is an indicator of model latency, which is reflected in the inference time.

2.5.2. Segmentation Performance

The metric used to evaluate pixelwise segmentation performance is (mIOU), defined by Equation (1):

$$\text{mIOU} = \frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i + FP_i + FN_i}, \quad (1)$$

where TP stands for true positive, FP stands for false positive, FN stands for false negative, and C is the number of classes.

2.6. Real-Time Application

The semantic segmentation model (computer vision) was integrated into the robotic orchard rover developed by Veiros et al. [43] (Figure 5). The goal of this experiment was to investigate the accuracy and feasibility of the computer-vision framework in conjunction with a mechanism for applying a herbicide spray to weeds. The experiments were conducted in the laboratory.

Figure 5. Robotic Rover for agricultural applications: (1) herbicide reservoir; (2) camera; (3) Cartesian robotic manipulator with spray nozzle; (4) control system.

2.6.1. Mechanism

Figure 6 shows the main components of the Cartesian robotic manipulator for spray applications. Three actuators must be controlled by the output pins of the Jetson Nano device: pressure motor: a DC motor that applies pressure to the herbicide container; manipulator motor: a stepper motor that moves the axis of the Cartesian manipulator;

nozzle relay: a relay that opens and closes the spray valve; spray nozzle; Cartesian robot manipulator; herbicide container.

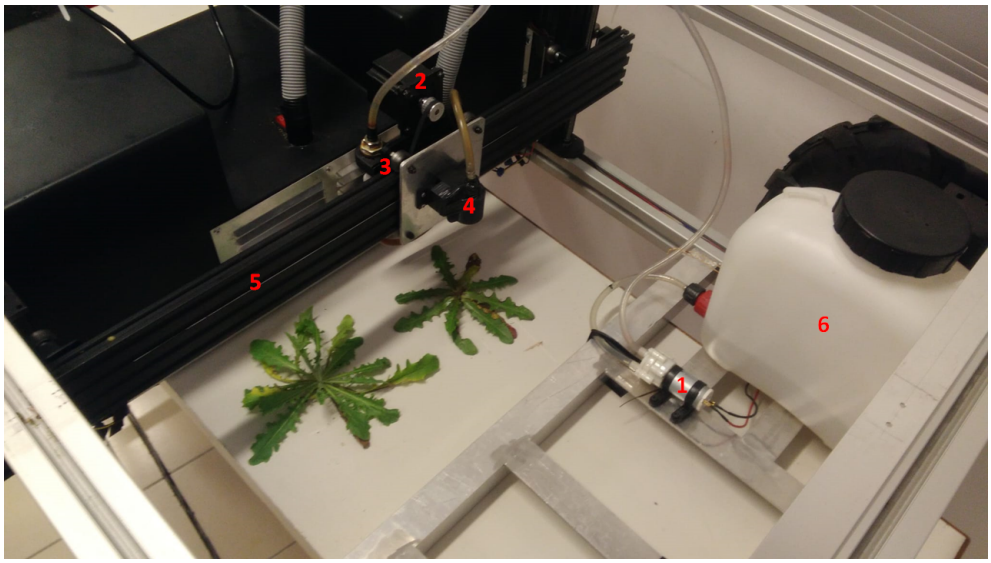


Figure 6. Main components of the robotic rover for weed control. (1) pressure motor: The DC motor that applies pressure to the herbicide container; (2) manipulator motor: The stepper motor that moves the axis of the Cartesian manipulator; (3) nozzle relay: The relay that opens and closes the spray valve; (4) spray nozzle; (5) Cartesian robot manipulator; (6) herbicide container.

2.6.2. Camera

A Raspberry Pi v2 camera module with a Sony IMX219 8-megapixel sensor was used to capture the video images. Figure 7 shows the camera location embedded on the robotic orchard rover.

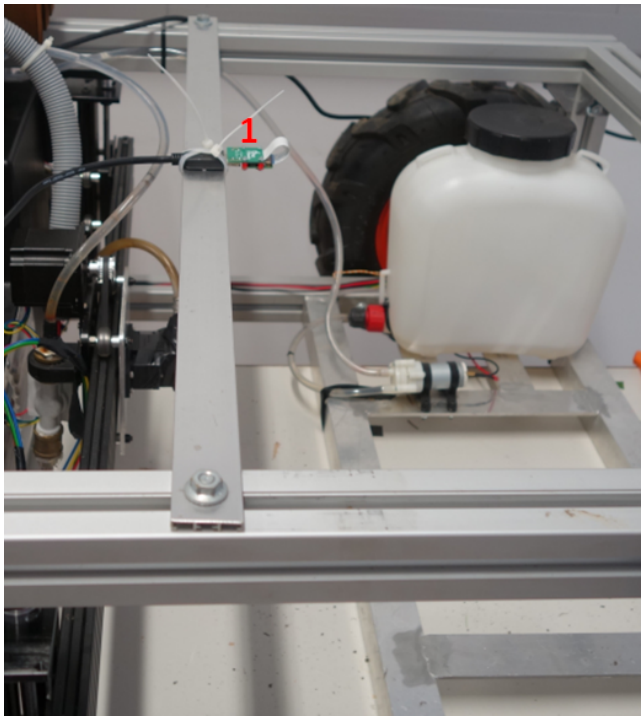


Figure 7. Camera location (1) embedded on the robotic orchard rover.

2.6.3. Computer Vision and Mechanism Integration

The DeepLabV3 model was trained with the dataset described in Section 2.2.2 for practical application. Training was performed as described in Section 2.3.3, fine tuning on the PASCAL dataset, with hyperparameters OS = 16 and DM = 1.0. Although the dataset consisted of only a few images, each image contained many labeled weeds (see Figure 2). Another fact that allows for training deep learning models with few data is the use of a fine-tuning strategy.

Figure 8 provides a general view of the practical experiment involving integration between the computer vision and the mechanism. The output of the computer-vision model was a segmented image with a set of (X, Y) coordinates (centroid) of each weed (green dots). These coordinates were converted into the manipulator's Cartesian manipulator coordinates (X′, Y′). The (X′, Y′) coordinates were then fed into the control algorithm that calculated the sequence of movements to position the spray nozzle on the detected weed and execute the spray.

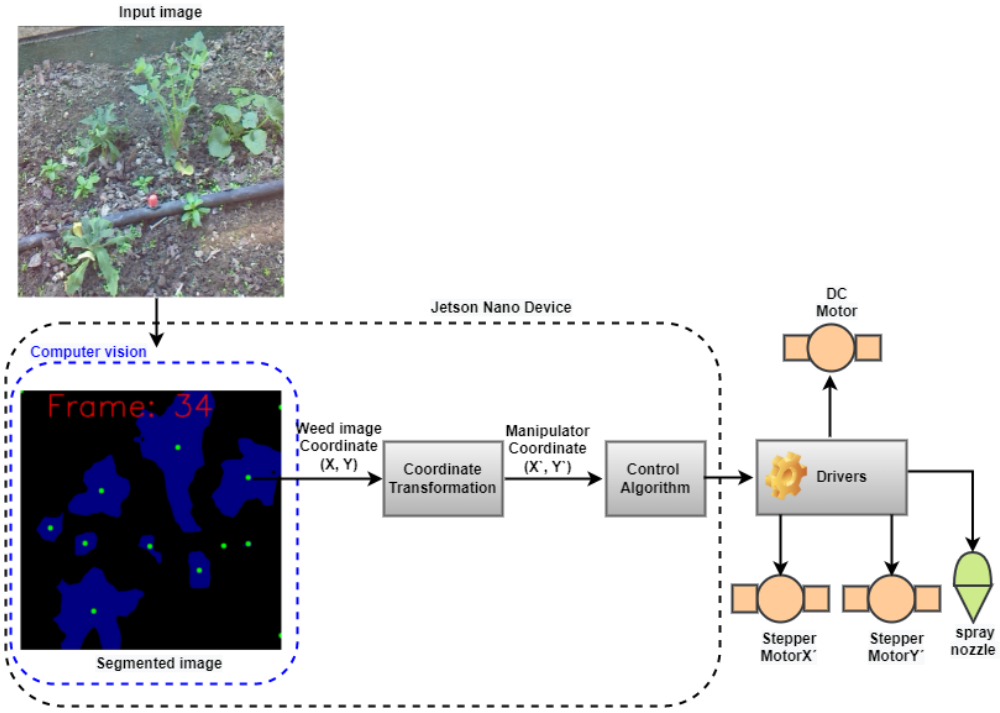


Figure 8. General view of the practical experiment.

2.6.4. Coordinate Transformation

The spray nozzle was attached to the Cartesian axis of the robot manipulator. The axis was driven by the stepper motorX′ (Figure 8), which was configured to perform one revolution when 1000 pulses are applied. The image output from the computer-vision framework had a resolution of 513 × 513.

For the spray nozzle to cover the spatial camera view, the stepper motor X′ should receive 4350 steps. This relationship is shown in Figure 9. Given an (X, Y) weed position in the image, the number of pulses (steps) that must be applied to stepper motor X′ for the nozzle to hit the weed position is calculated by Equation (2).

$$Pulses = \frac{4351}{513} \times X, \tag{2}$$

where X stands for the position of the weed in the image. That means that the coordinate transformation converts the pixel position into the number of pulses (steps).

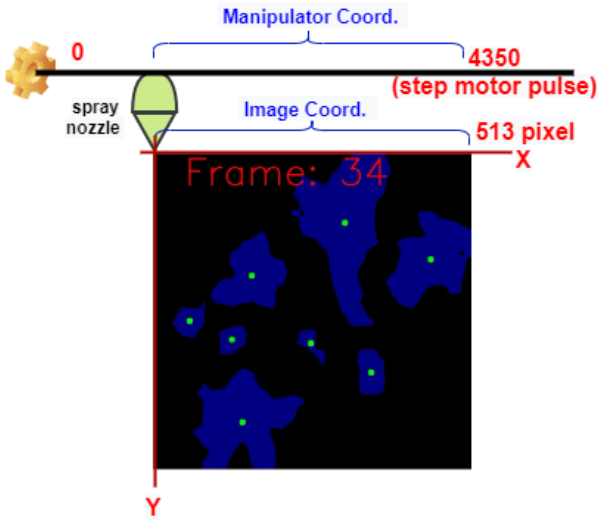


Figure 9. Image coordinate and Cartesian robot coordinate.

3. Results and Discussions

This section presents the results of the DeepLabv3 experiments on the semantic segmentation of the WCFID dataset where hyperparameters were set, an optimization framework was applied, and fine tuning was performed.

3.1. Pretrained Models and Fine-Tuning Performance

To investigate how the pretrained models affected segmentation performance for the use case (i.e., crop and weed segmentation), experiments were conducted with the fine-tuning strategy using the Cityscape, ADE20k, and PASCAL datasets. Table 6 shows the results for this experiment in combination with hyperparameters OS and DM = 1.0.

Table 6. Fine-tuning comparison (mIOU).

Pre-Trained Dataset	OS = 8	OS = 16	OS = 32
PASCAL	75%	72%	66%
ADE20k	76%	74%	68%
Cityscape	78%	74%	68%

As shown in Table 6, when fine tuned using hyperparameter OS = 8, the Cityscape dataset performed the best with 78% mIOU. However, when using OS = 16 and OS = 32, the performance for fine tuning with Cityscape and ADE20k was the same. OS = 8 gave the best result compared to OS = 16 and OS = 32 because it captured more spatial information from the input. The fine-tuning process uses knowledge gained from pretraining in other datasets. For example, the Cityscape dataset likely had more similarities with plants, e.g., the Vegetation class in the Nature group, which is probably the reason for the best result.

3.2. Model Optimization and Performance Degradation

This experiment investigates the relationship between model optimization (i.e., tuning hyperparameters and applying the TensorRT framework to improve inference time) and segmentation degradation. Although fine tuning with the Cityscape dataset gave the best results, as shown in a previous experiment, fine tuning in this experiment was used for the PASCAL dataset because a pretrained model with DM = 0.5 and DM = 1.0 was only available for the PASCAL dataset [44]. Table 7 shows the computational cost of each model measured in giga floating-point operations (GFLOPs). It is clear how much the optimization reduced the computational cost: using hyperparameters DM = 0.5 and OS = 32 in the CNN

backbone, the computational cost of the model was reduced from 81.9 to 4.4 GFLOPs. This performance was due to the reduction in convolutional layers in the MobileNet backbone by hyperparameters DM, which shrank the model by a factor of 0.5, and OS.

Table 7. Temporal computational cost (in GFLOPs) with different parameterisations.

Optimization	OS = 8	OS = 16	OS = 32
DM = 1.0	81.9	25.1	14.7
DM = 0.5	26.1	7.9	4.4

Table 8 shows how segmentation performance is affected by optimization to reduce inference time. The best model segmentation performance was 75% mIOU for the model with OS = 8 and DM = 1.0. The worst performance was 64% mIOU for the model with successive and more stringent optimization to improve the inference time, i.e., depth multiplier of 0.5, output stride of 32, and TensorRT. Reducing the model size to achieve fast inference time by using DM = 0.5 and OS = 32 led to a decrease in segmentation accuracy. The model became shallower and lost information about the features of the input image. An important result is also that using the TensorRT framework for enhancement did not affect the segmentation accuracy. TensorRT is an inference-oriented framework whose main purpose is efficient inference. It also supports dynamic graphs, which enables the efficient reuse of memory and improves inference performance [27].

Table 8. Summary of segmentation performance with respect to different tested parameterisations.

Optimization	OS = 8	OS = 16	OS = 32
DM = 1.0	75%	72%	66%
DM = 1.0 + TensorRT	-	72%	66%
DM = 0.5	73%	68%	64%
DM = 0.5 + TensorRT	-	68%	64%

The corresponding segmentation quality results are shown in Figure 10. Segmentation performance (quality) varied with different hyperparameters DM and OS. Comparing the results with the same OS but different DM (i.e., (c) with (h), (d) with (i), and (e) with (j)), the results with DM = 0.5 had more mis-segmentations between crops and weeds than those with DM = 1.0. This result can be explained by the fact that the DM hyperparameter affected the complexity of the model (size). A model with DM = 1.0 was deeper than a model with DM = 0.5.

If we now compare the results with the same DM but different OS (8, 16 and 32), we can see that the results with O = 8 were finer (fine details) than those with OS = 16 and OS = 32. The results with OS = 16 were finer than those of models with OS = 32. This result can be explained by the fact that the OS hyperparameter influenced the sparsity of the input calculation of the model.

Table 9 shows how optimization affected the time of model inference. The best result was 0.17 s using DM = 0.5, OS = 32 and TensorRT. Table 7 shows that this configuration resulted in the smallest model, which reduced the inference time, as expected .

Table 9. Inference time performance (s).

Optimization	OS = 8	OS = 16	OS = 32
DM = 1.0	2.51	0.85	0.77
DM = 1.0 + TensorRT	-	0.57	0.31
DM = 0.5	1.19	0.43	0.33
DM = 0.5 + TensorRT	-	0.3	0.17

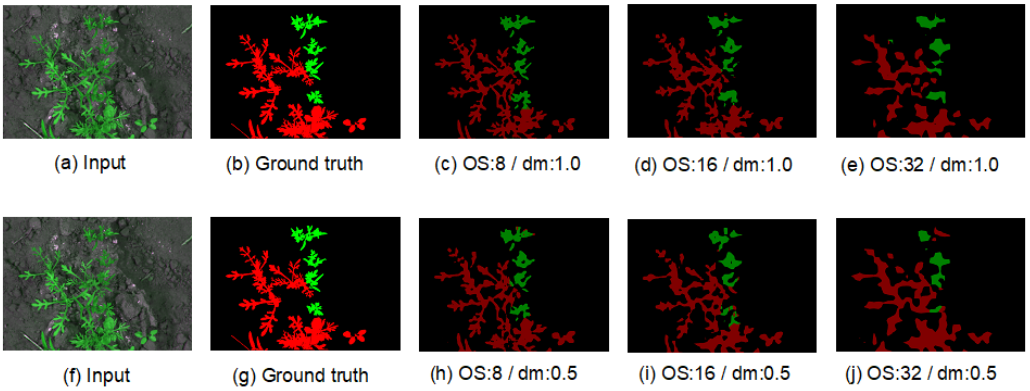


Figure 10. Qualitative segmentation results for the CWFID dataset. The labels of each subimage, except input and ground truth, denote the optimization used in the model to obtain the segmentation results according to Table 8.

Tables 10 and 11 show the relationship (correlation) between model segmentation performance and inference time. The inference time could be accelerated by a factor of 14.8, while the segmentation performance (mIOU) decreased by 14.7%. With these optimizations and an image resolution of 1296 × 966, the model could perform segmentation in 5.9 FPS.

Table 10. Variation in inference time with respect to different parameterisations when compared to the baseline configuration.

Optimization	OS = 8	OS = 16	OS = 32
DM = 1.0	1	3.0	3.3
DM = 1.0 + TensorRT	-	4.4	8.1
DM = 0.5	2.1	5.8	7.6
DM = 0.5 + TensorRT	-	8.4	14.8

Table 11. Variations in model performance (mIOU%) with respect to different parameterisations considering the optimal configuration tested.

Optimization	OS = 8	OS = 16	OS = 32
DM = 1.0	0	−4.0	−12.0
DM = 1.0 + TensorRT	-	−4.0	−12.0
DM = 0.5	−3.0	−9.3	−14.7
DM = 0.5 + TensorRT	-	−9.3	−14.7

In addition, the results were compared with previously published segmentation work in the CWFID dataset. In Table 12, Maxwell is a very small 128-core GPU integrated into the Jetson Nano developer kit, while Titan X is a standard 3584-cores GPU and Titan XP a 3840-cores GPU board.

For work comparison, Table 12 shows the input size of the image, the GPU capabilities, the segmentation performance of the model in terms of accuracy or mIOU, and the inference performance FPS. Regarding GPU characteristics, our approach (DeepLabV3) uses the Maxwell embedded in the Jetson Nano, which is a portable device. In contrast, the other approaches use GPU cards that function in conjunction with a desktop.

Regarding the segmentation performance of the models, CED-Net and our model could be compared because the value of the results is presented in the same metric (mIOU). CED-Net outperformed our approach (DeepLabV3*) by only 2%.

Inference time performance could be compared between our approach and Adapted-IV3. Our approach (DeepLabV3**) outperformed Adapted-IV3 by about 50 times. The primary key to this performance is the use of the MobileNet backbone in the encoder part of the DeepLabV3 model, since the MobileNet DCNN provides light weight by performing depthwise separable convolutions. Another contribution to this performance was the setting of hyperparameters DM = 0.5 and OS = 32, and the use of the TensorRT framework.

Table 12. Comparison of the segmentation performance of other works on the CWFID test dataset. DeepLabV3* (OS = 8 and DM = 1.0) and DeepLabV3** (OS = 32 and DM = 0.5).

Model	Input Size	GPU	Accuracy	mIOU	FPS
Adapted-IV3 [15]	1296 × 966	Titan X	93.9%	-	0.12
CED-Net [16]	1296 × 966	Titan XP	-	77%	-
DeepLabV3*	1296 × 966	Maxwell	-	75%	0.4
DeepLabV3**	1296 × 966	Maxwell	-	64%	5.9

3.3. Image Scale Analysis

In addition, smaller input images were considered to evaluate the inference time under this condition. The experiment was performed by cropping the WCFDI images in their center to a resolution of 513 × 513. Table 13 shows these results. For the quality segmentation shown in Figure 11, an inference time of 0.04 s was achieved, corresponding to processing of 25 FPS.

Figure 11. Qualitative segmentation result for input image resolution of 513 × 513, hyperparameter OS = 32.

Table 13. Analysis of inference time with respect to the size of the input (s).

Optimization	OS = 8	OS = 16	OS = 32
DM = 1.0	0.54	0.18	0.12
DM = 1.0 + TensorRT	0.27	0.09	0.07
DM = 0.5	0.54	0.12	0.06
DM = 0.5 + TensorRT	0.27	0.07	0.04

3.4. Real-Time Application

The goal of this experiment was to demonstrate a practical application of the deep-learning-based computer-vision model, its reliability in real-time, and its integration with mechanical control.

After weeds had been detected by the computer-vision model, a series of actuators were controlled in sequence (e.g., the manipulator motor and the spray solenoid valve). The response time of each actuator was considered by setting the required delay time to finish the current step and start the next step.

We simulated a field soil with weeds by manually positioning them in the camera's field of view. Figure 12 shows the general view of the experiment. The window in which the input image coming from the camera is displayed. The window in which the segmented image is displayed. The area where the weed was placed.

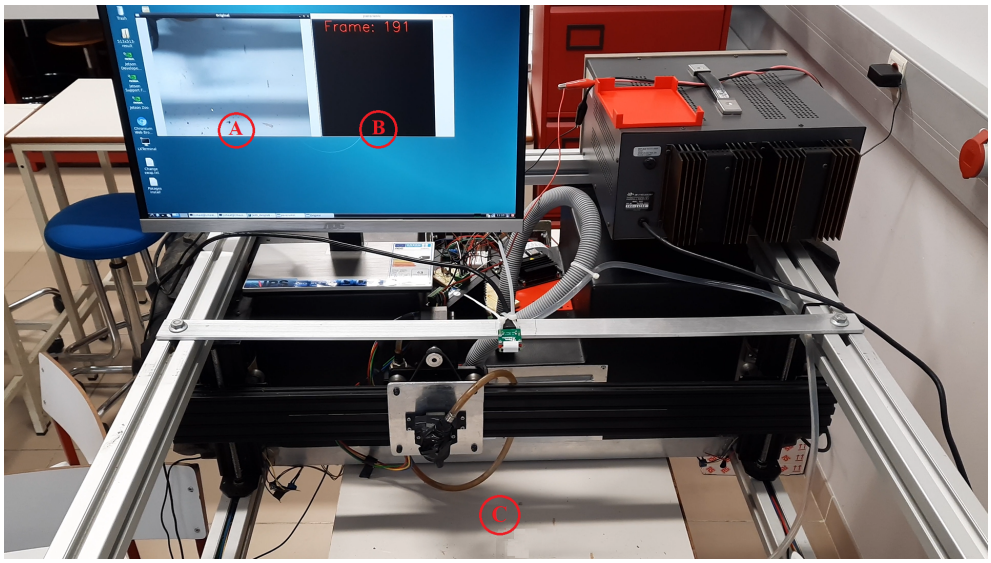


Figure 12. General view of the experimental results. (**A**) is the window in which the input image coming from the camera is displayed. (**B**) is the window in which the segmented image is displayed. (**C**) is the area where the weed was placed.

Figure 13 illustrates the test result for a single image of the image acquisition. In (A), an image with two weeds is shown. In (B), two weeds were segmented with a high-quality border corresponding to the image in (A). The green dots in the center of the blue segmented regions are the references (location coordinates) for the Cartesian manipulator, as described in Section 2.6.4. (C) shows the weeds in the field of view of the camera and the spray nozzle.

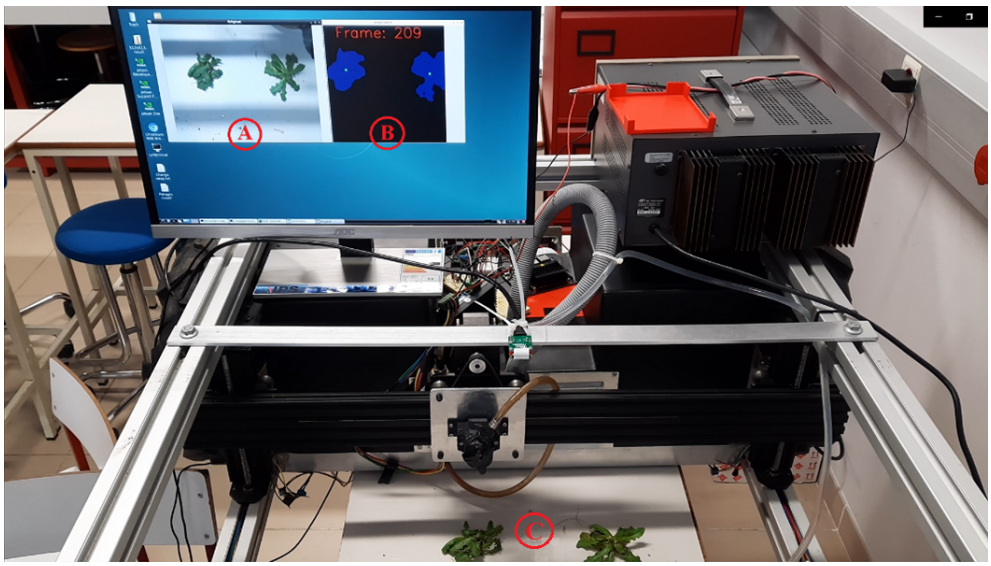


Figure 13. Real-time application result: segmentation of weeds (general view), (**A**) An image with two weeds is shown; (**B**), two weeds were segmented with a high-quality border corresponding to the image in (**A**); (**C**) shows the weeds in the field of view of the camera and the spray nozzle.

Figure 14 shows a sequence of four images. The upper-left partial image (Frame: 189) is an image without weeds. The remaining images (frame: 224, frame: 226, and frame: 241)

show the image with the weeds (input) and the corresponding segmentation results (output). The blue segmented areas had high shape similarity with the corresponding weed image. The green dots indicating the center of gravity of the weeds are clearly visible in the center of each segmented area. Thus, these results show the good accuracy of the computer-vision model.

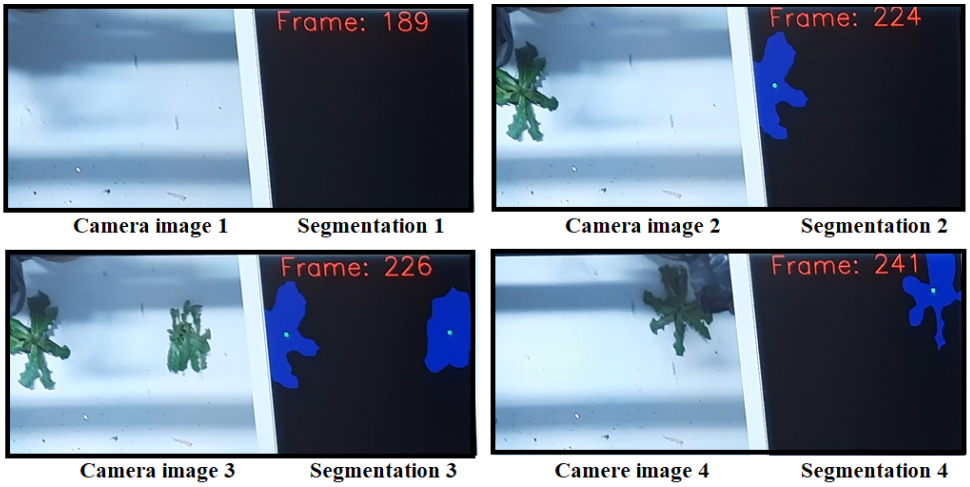


Figure 14. Real-time application result: segmentation of weeds. Details of an image sequence.

The computer-vision model was used in the Jetson Nano device to determine the reference position of the weeds for mechanism control (see Sections 2.6.3 and 2.6.4). Given a set of weed references, the Cartesian manipulator positioned the nozzle on the particular weed and performed the spraying. Figure 15 shows the exact positioning of the nozzle and the spray.

Figure 15. Result of real-time application: positioning of nozzle and spray atomization.

A video demonstration of the experiment result is available in the data-availability statement at the end of the manuscript.

3.5. Final Discussions

Taken together, these results provide important insights into the semantic segmentation accuracy and inference time performance of crop and weed detection when the necessary optimizations are conducted to the model to run it on edge devices. At the cost of worse segmentation performance, inference time can be dramatically increased, enabling real-time inference. The approach proposed in our study can render the inference time faster than that of the compared models even though it is a portable device, while the others

are desktop-based. Compared to an approach with similar hardware and input size [14], our proposed approach performed inference about 5 times faster. DeepLabV3 proved to be a very versatile model for segmentation tasks, since the trade-off between segmentation accuracy and inference time can be controlled by hyperparameters OS and DM.

In addition, fine tuning the model in the Cityscapes, PASCAL, and ADE20K datasets yielded good results. Cityscapes provided the marginally best performance compared to the PASCAL and ADE20K datasets. The results show that, when accuracy requirements are high, fine tuning with the pretrained Cityscape dataset is the best solution. If time inference is a requirement, then the best approach is to fine-tune with the pretrained Cityscape or ADE20k dataset, since both achieved the same accuracy performance. In the latter case, it is recommended to fine-tune with the PASCAL dataset if Cityscape and ADE20k are not available. Real-time application for weed spraying was also accurate and feasible. The segmentation model embedded in the edge device provided all the information (position references) needed to control the mechanism. According to the video demonstration, the mechanism precisely positioned the nozzle at all target weeds and applied the spray.

4. Conclusions

Edge devices play an important role when mobility is required. Computer-vision models running on such devices must be optimized due to the device's limitations in terms of memory and computation. There are many studies and proposals for the optimization of deep learning models, but there is still a lack of research on how the performance of the models is affected by optimization. Moreover, there are not yet many real-world use cases of computer vision and its interaction with mechanisms in the domain of agriculture.

In this work, the optimization of the semantic segmentation model for weed was investigated, and its effects on inference time and segmentation performance were studied. The procedure can be described by using crop and weed images for training and validating the model; the procedure was applied for training the DeeplabV3 semantic segmentation model with the fine-tuning approach. Model optimization was performed before and after training by selecting different model hyperparameters and applying model quantization. Lastly, the performance of the model was evaluated.

Experimental results show that, by using hyperparameters DM = 0.5 and OS = 32 in the CNN backbone, the computational cost of the model was reduced from 81.9 to 4.4 GFLOPs. On the Jetson Nano device, the best inference time of the model was 0.17 s. Compared to the baseline model (i.e., the model without optimization), the inference time was accelerated by a factor of 14.8, while the segmentation performance (mIOU) was decreased by 14.7%. This result illustrates the potential of the optimizations for building lightweight models that still have good predictive accuracy.

Since the technique of fine tuning is important when only a small amount of training data is available, the effects of fine tuning on pretrained models were investigated. With respect to the PASCAL and ADE20K datasets, Cityscapes achieved the marginally best performance.

An extension of this work is to propose a practical application using the weed segmentation model presented in this study, an edge device, and a spraying mechanism. The results show that the proposed method is feasible, and has good accuracy and potential for weed control.

The limitation of this study was that the Cartesian robot manipulator operates in only one axis, while in practice, the other axis motion is the translational motion performed by the robot wheels. Since the practical application was carried out in a simulated environment (laboratory), the main objective of future work is to develop a complete system to study the performance of weed control in a real agricultural environment.

The results presented in this study demonstrate the potential of using lightweight models and portable edge devices for the real-time semantic segmentation of weeds.

Author Contributions: Conceptualization: P.D.G.; Data curation: E.A., A.V. and R.M.; Formal analysis: E.A., P.D.G., M.P.S. and H.P.; Funding acquisition: P.D.G.; Investigation: E.A., A.V. and R.M.; Methodology: E.A. and P.D.G.; Project administration: P.D.G.; Resources: R.M., M.P.S. and K.A.; Software: E.A.; Supervision: P.D.G.; Validation: E.A. and K.A.; Visualization: E.A.; Writing—original draft: E.A.; Writing—review and editing: P.D.G. and H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research work is funded by the PrunusBot project—Autonomous controlled spraying aerial robotic system and fruit production forecast, Operation no. PDR2020-101-031358 (leader), Consortium no. 340, Initiative no. 140, promoted by PDR2020, and cofinanced by the EAFRD and the European Union under the Portugal 2020 program.

Informed Consent Statement: Not applicable.

Data Availability Statement: The video demonstration is available at https://eduardo-assuncao-hub.github.io/Weed_detection/ (accessed on 30 July 2022).

Acknowledgments: The authors are thankful for the opportunity and financial support to carry on this project to Fundação para a Ciência e Tecnologia (FCT) and R&D Unit "Center for Mechanical and Aerospace Science and Technologies" (C-MAST), under project UIDB/00151/2020. Hugo Proença's work was funded by Fundação para a Ciência e Tecnologia in the scope of the UIDB/00151/2020 research project.

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

Abbreviations

The following abbreviations are used in this manuscript:

DNNs	Deep neural networks
DM	Depth multiplier
mIOU	Mean intersection over union
FPS	Frames per second
CNNs	Convolutional neural networks
DCNNs	Deep convolutional neural networks
GPU	Graphics processing unit
CWFID	Crop Weed Field Image Dataset
WRNs	Wide residual networks
FCN	Fully convolutional network
CRF	Conditional random field
TMG	Tensorflow model garden
FLOPs	Floating-point operations

References

1. Simões, M. *+Pêssego – Resultados de Apoio à Gestão*; Technical Report; Centro Operativo e Tecnológico Hortofrutícola Nacional: Alcobaça, Portugal, 2017.
2. Alibabaei, K.; Gaspar, P.D.; Lima, T.M. Modeling evapotranspiration using Encoder-Decoder Model. In Proceedings of the 2020 International Conference on Decision Aid Sciences and Application (DASA), Sakheer, Bahrain, 8–9 November 2020; pp. 132–136.
3. Assunção, E.; Diniz, C.; Gaspar, P.D.; Proença, H. Decision-making support system for fruit diseases classification using Deep Learning. In Proceedings of the 2020 International Conference on Decision Aid Sciences and Application (DASA), Sakheer, Bahrain, 8–9 November 2020; pp. 652–656.
4. Shanmugam, S.; Assunção, E.; Mesquita, R.; Veiros, A.; Gaspar, P.D. Automated weed detection systems: A review. *KnE Eng.* **2020**, 271–284. Available online: http://3.65.204.3/index.php/KnE-Engineering/article/view/7046 (accessed on 30 July 2022).
5. Cunha, J.; Gaspar, P.D.; Assunção, E.; Mesquita, R. Prediction of the Vigor and Health of Peach Tree Orchard. In Proceedings of the International Conference on Computational Science and Its Applications, Cagliari, Italy, 13–16 September 2021; Springer: Cham, Switzerland, 2021; pp. 541–551.
6. Mesquita, R.; Gaspar, P.D. A Novel Path Planning Optimization Algorithm Based on Particle Swarm Optimization for UAVs for Bird Monitoring and Repelling. *Processes* **2021**, *10*, 62. [CrossRef]
7. Alibabaei, K.; Gaspar, P.D.; Lima, T.M. Modeling soil water content and reference evapotranspiration from climate data using deep learning method. *Appl. Sci.* **2021**, *11*, 5029. [CrossRef]

8. Alibabaei, K.; Gaspar, P.D.; Lima, T.M. Crop yield estimation using deep learning based on climate big data and irrigation scheduling. *Energies* **2021**, *14*, 3004. [CrossRef]
9. Alibabaei, K.; Gaspar, P.D.; Lima, T.M.; Campos, R.M.; Girão, I.; Monteiro, J.; Lopes, C.M. A Review of the Challenges of Using Deep Learning Algorithms to Support Decision-Making in Agricultural Activities. *Remote Sens.* **2022**, *14*, 638. [CrossRef]
10. Alibabaei, K.; Gaspar, P.D.; Assunção, E.; Alirezazadeh, S.; Lima, T.M. Irrigation optimization with a deep reinforcement learning model: Case study on a site in Portugal. *Agric. Water Manag.* **2022**, *263*, 107480. [CrossRef]
11. Alibabaei, K.; Gaspar, P.D.; Assunção, E.; Alirezazadeh, S.; Lima, T.M.; Soares, V.N.; Caldeira, J.M. Comparison of On-Policy Deep Reinforcement Learning A2C with Off-Policy DQN in Irrigation Optimization: A Case Study at a Site in Portugal. *Computers* **2022**, *11*, 104. [CrossRef]
12. Alibabaei, K.; Assunção, E.; Gaspar, P.D.; Soares, V.N.; Caldeira, J.M. Real-Time Detection of Vine Trunk for Robot Localization Using Deep Learning Models Developed for Edge TPU Devices. *Future Internet* **2022**, *14*, 199. [CrossRef]
13. Mittal, S. A Survey on optimized implementation of deep learning models on the NVIDIA Jetson platform. *J. Syst. Archit.* **2019**, *97*, 428–442. [CrossRef]
14. Milioto, A.; Lottes, P.; Stachniss, C. Real-Time Semantic Segmentation of Crop and Weed for Precision Agriculture Robots Leveraging Background Knowledge in CNNs. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 2229–2235. [CrossRef]
15. McCool, C.; Perez, T.; Upcroft, B. Mixtures of Lightweight Deep Convolutional Neural Networks: Applied to Agricultural Robotics. *IEEE Robot. Autom. Lett.* **2017**, *2*, 1344–1351. [CrossRef]
16. Khan, A.; Ilyas, T.; Umraiz, M.; Mannan, Z.I.; Kim, H. CED-Net: Crops and Weeds Segmentation for Smart Farming Using a Small Cascaded Encoder-Decoder Architecture. *Electronics* **2020**, *9*, 1602. [CrossRef]
17. Wang, A.; Xu, Y.; Wei, X.; Cui, B. Semantic Segmentation of Crop and Weed using an Encoder-Decoder Network and Image Enhancement Method under Uncontrolled Outdoor Illumination. *IEEE Access* **2020**, *8*, 81724–81734. [CrossRef]
18. Fawakherji, M.; Youssef, A.; Bloisi, D.; Pretto, A.; Nardi, D. Crop and Weeds Classification for Precision Agriculture Using Context-Independent Pixel-Wise Segmentation. In Proceedings of the 2019 Third IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 25–27 February 2019; pp. 146–152. [CrossRef]
19. Olsen, A. Improving the Accuracy of Weed Species Detection for Robotic Weed Control in Complex Real-Time Environments. Ph.D. Thesis, James Cook University, Douglas, Australia, 2020.
20. Partel, V.; Charan Kakarla, S.; Ampatzidis, Y. Development and evaluation of a low-cost and smart technology for precision weed management utilizing artificial intelligence. *Comput. Electron. Agric.* **2019**, *157*, 339–350. [CrossRef]
21. Abdalla, A.; Cen, H.; Wan, L.; Rashid, R.; Weng, H.; Zhou, W.; He, Y. Fine-tuning convolutional neural network with transfer learning for semantic segmentation of ground-level oilseed rape images in a field with high weed pressure. *Comput. Electron. Agric.* **2019**, *167*, 105091. [CrossRef]
22. Asad, M.H.; Bais, A. Weed detection in canola fields using maximum likelihood classification and deep convolutional neural network. *Inf. Process. Agric.* **2020**, *7*, 535–545. [CrossRef]
23. Ma, X.; Deng, X.; Qi, L.; Jiang, Y.; Li, H.; Wang, Y.; Xing, X. Fully convolutional network for rice seedling and weed image segmentation at the seedling stage in paddy fields. *PLoS ONE* **2019**, *14*, e0215676. [CrossRef].
24. Lameski, P.; Zdravevski, E.; Trajkovik, V.; Kulakov, A. Weed detection dataset with RGB images taken under variable light conditions. In Proceedings of the International Conference on ICT Innovations, Skopje, Macedonia, 18–23 September 2017; Springer: Cham, Switzerland, 2017; pp. 112–119.
25. Naushad, R.; Kaur, T.; Ghaderpour, E. Deep Transfer Learning for Land Use and Land Cover Classification: A Comparative Study. *Sensors* **2021**, *21*, 8083. [CrossRef] [PubMed]
26. Nanni, L.; Lumini, A.; Loreggia, A.; Formaggio, A.; Cuza, D. An Empirical Study on Ensemble of Segmentation Approaches. *Signals* **2022**, *3*, 341–358. [CrossRef]
27. Hadidi, R.; Cao, J.; Xie, Y.; Asgari, B.; Krishna, T.; Kim, H. Characterizing the Deployment of Deep Neural Networks on Commercial Edge Devices. In Proceedings of the 2019 IEEE International Symposium on Workload Characterization (IISWC), Orlando, FL, USA, 3–5 November 2019; pp. 35–48. [CrossRef]
28. Haug, S.; Ostermann, J. A crop/weed field image dataset for the evaluation of computer vision based precision agriculture tasks. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; Springer: Cham, Switzerland, 2014; pp. 105–116.
29. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3431–3440. [CrossRef]
30. Minaee, S.; Boykov, Y.Y.; Porikli, F.; Plaza, A.J.; Kehtarnavaz, N.; Terzopoulos, D. Image Segmentation Using Deep Learning: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2021**, *44*, 3523–3542. [CrossRef]
31. Ronneberger, O.; P.Fischer.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*; Springer: Cham, Switzerland, 2015; Volume 9351, pp. 234–241.
32. Chen, L.C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; Yuille, A.L. SSemantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs. *arXiv* **2016**. [CrossRef]

33. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
34. Chen, L.C.; Zhu, Y.; Papandreou, G.; Schroff, F.; Adam, H. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In Proceedings of the ECCV 2018, Munich, Germany, 8–14 September 2018.
35. Chen, C.; Du, X.; Hou, L.; Kim, J.; Li, J.; Li, Y.; Rashwan, A.; Yang, F.; Yu, H. TensorFlow Official Model Garden. 2020. Available online: https://github.com/tensorflow/models/tree/master/official (accessed on 30 July 2022).
36. Chu, B.; Madhavan, V.; Beijbom, O.; Hoffman, J.; Darrell, T. Best Practices for Fine-Tuning Visual Classifiers to New Domains. In Proceedings of the ECCV Workshops 2016, Amsterdam, The Netherlands, 8–10 and 15–16 October 2016.
37. Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; Schiele, B. The Cityscapes Dataset for Semantic Urban Scene Understanding. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016.
38. Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A.; Torralba, A. Scene Parsing through ADE20K Dataset. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017.
39. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
40. NVIDIA. TensorRT Release Notes. 2021. Available online: https://docs.nvidia.com/deeplearning/tensorrt/release-notes/ (accessed on 30 July 2022).
41. NVIDIA. NVIDIA TensorRT. 2021. Available online: https://developer.nvidia.com/tensorrt (accessed on 30 July 2022).
42. Tang, R.; Adhikari, A.; Lin, J. FLOPs as a Direct Optimization Objective for Learning Sparse Neural Networks. *arXiv* **2018**, arXiv:1811.03060.
43. Veiros, A.; Mesquita, R.; Gaspar, P.D.; Simões, M.P. Multitask Robotic rover for agricultural activities (R2A2): A robotic platform for peach culture. In Proceedings of the X International Peach Symposium, Naoussa, Greece, 30 May–3 June 2022.
44. Yu, H.; Chen, C.; Du, X.; Li, Y.; Rashwan, A.; Hou, L.; Jin, P.; Yang, F.; Liu, F.; Kim, J.; et al. TensorFlow DeepLab Model Zoo. 2020. Available online: https://github.com/tensorflow/models/blob/master/research/deeplab/g3doc/model_zoo.md/ (accessed on 30 July 2022).

Chapter 6

Real-Time Image Detection for Edge Devices: A Peach Fruit Detection Application

The following work was developed by myself, P. Gaspar, K. Alibabaei, M. Simões, H. Proença, V. Soares and João Caldeira between January and September 2022. The aim was to develop an algorithm for fruit detection targeting an edge device for real-time application, generating and using a new dataset.

1

[1]This work was published in [**?**], E. Assunção, P. D. Gaspar, K. Alibabaei, M. P. Simões, H. Proença, V. N. G. J. Soares, J. M. L. P. Caldeira, "Real-Time Image Detection for Edge Devices: A Peach Fruit Detection Application," Future Internet, vol. 14, no. 3, 2022. [Online]. Available:`https://www.mdpi.com/1999-5903/14/11/323`

future internet

MDPI

Article

Real-Time Image Detection for Edge Devices: A Peach Fruit Detection Application

Eduardo Assunção [1,2], Pedro D. Gaspar [1,3,*], Khadijeh Alibabaei [1,3], Maria P. Simões [4,5], Hugo Proença [2], Vasco N. G. J. Soares [2,4] and João M. L. P. Caldeira [2,4]

1 C-MAST Center for Mechanical and Aerospace Science and Technologies, University of Beira Interior, 6201-001 Covilhã, Portugal
2 Instituto de Telecomunicações, University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
3 Department of Electromechanical Engineering, University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal
4 Polytechnic Institute of Castelo Branco, Av. Pedro Álvares Cabral nº 12, 6000-084 Castelo Branco, Portugal
5 CERNAS, Research Center for Natural Resources, Environment and Society, Escola Superiora Agrária de Coimbra Bencanta, 3045-601 Coimbra, Portugal
* Correspondence: dinis@ubi.pt

Abstract: Within the scope of precision agriculture, many applications have been developed to support decision making and yield enhancement. Fruit detection has attracted considerable attention from researchers, and it can be used offline. In contrast, some applications, such as robot vision in orchards, require computer vision models to run on edge devices while performing inferences at high speed. In this area, most modern applications use an integrated graphics processing unit (GPU). In this work, we propose the use of a tensor processing unit (TPU) accelerator with a Raspberry Pi target device and the state-of-the-art, lightweight, and hardware-aware MobileDet detector model. Our contribution is the extension of the possibilities of using accelerators (the TPU) for edge devices in precision agriculture. The proposed method was evaluated using a novel dataset of peaches with three cultivars, which will be made available for further studies. The model achieved an average precision (AP) of 88.2% and a performance of 19.84 frames per second (FPS) at an image size of 640 × 480. The results obtained show that the TPU accelerator can be an excellent alternative for processing on the edge in precision agriculture.

Keywords: deep learning; edge device; object detection; precision agriculture; TPU accelerator

Citation: Assunção, E.; Gaspar, P.D.; Alibabaei, K.; Simões, M.P.; Proença, H.; Soares, V.N.G.J.; Caldeira, J.M.L.P. Real-Time Image Detection for Edge Devices: A Peach Fruit Detection Application. *Future Internet* **2022**, *14*, 323. https://doi.org/10.3390/fi14110323

Academic Editor: Paolo Bellavista

Received: 25 September 2022
Accepted: 2 November 2022
Published: 8 November 2022

1. Introduction

Precision agriculture can be used to increase yields and provide information for decision making. The application of precision agriculture in fruit detection has attracted considerable attention from researchers. Examples of benefits of fruit detection include yield estimation and mapping [1] and disease control [2]. The increase in the world's population and the resulting higher demand for food are accompanied by a change in dietary habits toward healthier foods such as fruits and vegetables, increasing the specific demand for this type of produce and the impact of climate change on agricultural activities; the migration of people to cities leads to a reduction in the labor force available in rural areas, which requires an improvement in the efficiency and effectiveness of agricultural practices. Technological evolution allows the automation and robotization of some of these practices, as well as the development of decision support systems that help in the management of these agricultural practices [3].

The detection of fruits—and, particularly, peaches—through automatic systems can contribute to the improvement of the efficiency of agricultural cultivation processes, whether that be through the adequate and sufficient supply of water [4–6], the fertilizer supply, evaluation of the vigor and state of health [7], ripening state, and diseases [2],

or even the improvement of weed control [8]. The management of agricultural practices supported by artificial intelligence systems for decision making helps in yield estimation, resource management, and circular economy [9,10]. These approaches can contribute to an increase in production and rentability through improved supply contracts and the reduction of fixed costs. Additionally, these results are part of an improvement in a crop's environmental sustainability through the reduction of fertilizers and are a contribution to the reduction of food loss.

Computer vision for fruit detection can be developed such that it cannot be used in real time. That is, images or videos are first captured and stored for later use (processing) [11]. This type of computer vision model was developed to run on a cloud or desktop computer, which typically requires large amounts of computing resources and memory. However, in certain applications, computer vision models must run on an edge device while performing inferences at high speed. This is the case with robot vision applications [12]. In general, edge devices are limited in terms of processing, memory, and power consumption [13,14]. To adapt an image processing application to these constraints, models such as MobileNets [15–17], ShuffleNet [18], Squeezenet [19], and DenseNet [20] have been developed. Because these models are optimized to run on a CPU, they are only suitable for "light applications" (e.g., processing only approximately one frame per second (FPS)). This is because these models have a high latency. However, after training, these models can be optimized to run on a graphics processing unit (GPU) with much better inference time performance [21–23].

Tian et al. [24] proposed a modified version of the YOLOv3 detector model to detect apples at different growth rate stages in orchards. The authors used an NVIDIA Tesla V100 server GPU for the training and testing. Using 3000 × 3000 resolution images, they achieved an F1 score of 0.817 and an inference time of 0.304 s. It is important to emphasize that the approach used in this study was not portable. Fu et al. [25] developed a vision system based on RGB and Kinect sensors for detecting apples in outdoor orchards. They used the faster R-CNN model and a desktop PC equipped with a GPU NVIDIA TITAN XP card. For original RGB images at a resolution of 1920 × 1080, they reported a detection performance of 0.79 AP and an inference time of 0.125 s. The approach used in this study was not portable. Liu et al. [26] proposed a modified version of YOLOv3 for detecting tomatoes. The detection used circles instead of boxes to locate the tomatoes. The model received 416 × 416 pixel images as inputs and achieved a detection accuracy of 96.4% AP and an inference time of 54 ms in a PC target device. Because the target device was a PC, this approach does not fall into the portable category.

Zhang et al. [22] proposed a lightweight fruit detection algorithm designed specifically for edge devices. The algorithm was based on a light-CSPNet network and YOLOv3. The model was deployed in the NVIDIA Jetson family (Jetson Xavier, Jetson TX2, and Jetson NANO). The detection accuracies for the orange, tomato, and apple datasets were 93, 88, and 85% AP, respectively. The detection speeds of the Jetson Xavier reached 46.9, 40.3, and 45.0 ms (orange, tomato, and apple, respectively) for image resolutions of different sizes. This approach fell into the portable category. Huang et al. [23] proposed a modified version of the YOLOv5 detector by adding an attention mechanism and an adaptive fusion method to the citrus detection model. The target device was an NVIDIA Jetson Nano integrated graphics processor. Using images with a resolution of 608 × 608, they achieved a detection accuracy of 93.32% AP and an edge-computing processing speed of 180 ms. Based on the model used and the target device, this approach falls into the portable category. Tsironis et al. [27] adapted the single-shot object detector (SSD) to the underlying object size distribution of the target detection area. They evaluated their proposed adapted model in tomato fruit detection and classification for three maturity stages of each tomato fruit. With the image resolution of 515× 512, by using a PC with a standard GPU (not portable), the model performed inferences at a speed of 200 FPS. In addition, the model was not optimized in terms of the edge device approach. In another work, Tsironis et al. [28] created a specialized tomato dataset with more than 250 images and a total of 2400 annotations. In this work, the dataset was evaluated for six object detection models.

Recently, a state-of-the-art TPU accelerator [29] and the MobileDet detector were developed for general image detection tasks [30]. In this work, we proposed the use of these two technologies with a Raspberry Pi target device for a real-time peach fruit detection application. The main contributions of this paper include the following:

- We propose the use of a lightweight and hardware-aware MobileDet detector model for a real-time peach fruit detection application while embedded in a Raspberry Pi target device along with a Coral edge TPU accelerator.
- We present a novel dataset of three peach cultivars with annotations and have made it available for further study (to our knowledge, this is the first work of its kind).

The remainder of this paper is organized as follows. Section 2 presents the equipment used for inference, the image dataset, the object detection model, and the mathematical formulation for the model evaluation. To confirm the performance of the proposed method, the results and discussions are presented in Section 3. Finally, Section 4 concludes the paper and provides guidelines for future work.

2. Materials and Methods

2.1. Dataset Description

An image dataset of the following three fruit peach cultivars was created: Sweet Dream, Royal Time, and Catherine. The images were taken in peach orchards in the Beira Interior region, the main peach-growing area in Portugal [31]. Table 1 shows the characteristics of each peach cultivar and describes the predominant fruit density for each cultivar.

Table 1. Examples of peach tree cultivars with their predominant fruit densities.

Cultivar	Sample Image	Fruit Density	Color
Royal Time		Low	Red
Sweet Dream		Medium	Dark Red
Catherine		High	Yellow

The images were taken with a Sony DSC-RX100M2 red–green–blue (RGB) camera. The images were then resized to a resolution of 640 × 480 pixels. Subsequently, the images

were manually labeled using the LabelImg annotation tool [32], which generated an XML file for each image. The information for the dataset is presented in Table 2. The dataset can be downloaded from the link provided in the *Data Availability* section at the end of this article.

Table 2. Statistics of the dataset.

Split	Cultivar	Images	Fruits (Labels)
Train	Sweet Dream	270	2015
	Royal Time	248	1066
	Catherine	305	4564
Test	Sweet Dream	66	453
	Royal Time	63	270
	Catherine	76	1480
Total of training		823	7645
Total of testing		205	2203

2.2. Hardware for Inference

The hardware platform (edge device) used to perform inferences consists of the following parts, as shown in Figure 1: (1) A microcontroller development kit—Raspberry Pi 4 [33]; (2) a Coral TPU accelerator [29]; (3) a Raspberry Pi Camera Module 2 [34]; (4) a DC-to-DC converter [35]; (5) three Li-ion batteries [36]. Note: The battery in Figure 1 is only an illustration for the application, as the capacity of the battery used depends on the application. The Raspberry had a quad-core Cortex A72 processor and 8 GB of RAM, and it used the Linux operating system with a Python interpreter and the TensorFlow Lite library. The Coral TPU accelerator, which was connected to the Raspberry Pi via USB, was an integrated edge TPU coprocessor designed to perform machine learning operations in an optimized manner (e.g., four Tera operations per second).

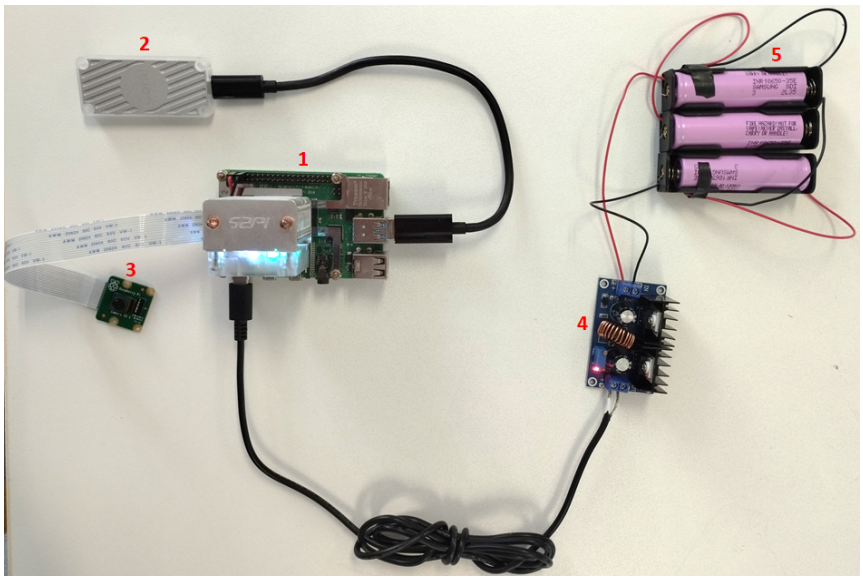


Figure 1. Hardware platform (edge device) for performing inferences.

2.3. SSD: Single Shot Detector

A single-shot detector (SSD) [37] is n state-of-the-art object detection model that outperforms its competitors, "you only look once" (YOLO) [38] and faster R-CNN [39], in terms of accuracy and inference time [37]. Therefore, the SSD model was used as a

detector in this study. Similarly to any model for computer vision tasks that is based on deep learning, the first step of SSD is feature extraction. This block is a convolutional neural network (CNN) and is usually referred to as the backbone of the model. The output of the backbone is a feature map containing the relevant information required to solve computer vision tasks. It is important to emphasize that the variations in the SSD model are on the backbone when selecting the CNN and performing optimizations (as described in Section 2.4). The remainder of the SSD model is constructed by adding additional layers of functionality at the end of the backbone. The SSD model partitions specific feature maps into standard boxes and generates scores for the presence of objects in each box. Additional technical details of the SSD model can be found in [37].

As mentioned previously, SSD variations were performed on the backbones. In this study, experiments were conducted using a MobileNet CNN as the backbone for the SSD model to investigate the trade-off between the detection accuracy and inference time. The backbones used were MobileNetV1, MobileNetV2, MobileNet EdgeTPU, and MobileDet.

2.3.1. MobileNetV1

MobileNetV1 is a lightweight model designed for use in mobile devices that typically has limited computing resources and memory. The main idea for achieving this goal is the implementation of a depthwise separable convolution. Depthwise separable convolution factorizes a conventional convolution into depthwise and pointwise convolutions (i.e., a 1×1 convolution). MobileNetV1 uses 3×3 depthwise separable convolutions, which require eight to nine times less computation than standard convolutions, with only slightly lower accuracy [15].

2.3.2. MobileNetV2

MobileNetV2 is a second-generation MobileNet. It was developed based on MobileNetV1. In MobileNetV2, linear bottlenecks between layers and connections between bottlenecks (residual connections) were included in the convolutional structure. MobileNetV2 also uses depthwise separable convolution, but adds the concepts of inverted residuals and linear bottlenecks to the building block. The concept of an inverted residual comes from an earlier idea of creating a connection (shortcut) between the layers. However, in MobilenetV2, this process is performed in a manner opposite to the original concept [40], allowing for faster training and better accuracy. In summary, linear bottlenecks are related to the last activation function of the block, which is replaced by a nonlinear function with a linear function. This approach avoids information degradation [16].

2.3.3. MobileNet Edge TPU

MobileNetV1 and MobileNetV2 were designed manually, entirely by hand. In contrast, the MobileNet edge TPU was developed using the accelerator-aware auto-machine learning (AutoML) [41] approach, which significantly reduces the manual process of designing and optimizing neural networks for hardware accelerators [42]. MobileNet Edge TPU is a version of MobileNet that has been adapted to run optimally on edge TPU devices (and take advantage of their features). In this study, this model was expected to perform significantly better in terms of accuracy and latency than MobileNetV1 and MobileNetV2 when running on a TPU device.

2.3.4. MobileDet

MobileDet is the latest version of the SSD model based on the MobileNet family. Again, the AutoML approach was used to create the model. The backbone has a hybrid convolution that includes depthwise and conventional convolution [30].

2.4. Model Optimizations

As mentioned in the introduction, edge devices have limited resources for computation and memory. To address this problem, efficient native models were created by

considering the model size and computational power. This is the case with several models, such as MobileNet and SqueezeNet. Another approach to increasing the performance of an edge device (faster inference and memory access) is the application of quantization techniques, where the model becomes simpler by reducing the precision of the weights and activation functions of the model (e.g., from 32-bit floating-point representations to 8-bit representations) [13]. Quantization approaches can broadly be divided into two categories. The first category is post-training quantization (PTQ), which quantizes the floating-point models. This technique reduces the size of the models by a factor of four and reduces inference time [13]. However, PTQ leads to degradation in model performance during inference. One reason for this is the smaller number of bits allocated [43].

The second category, quantization-aware training (QAT), attempts to mitigate the error caused by quantization by simulating the effects of quantization on weights and activation functions during the training process. This means that the model compensates for the loss due to the application of quantization. For this reason, QAT provides higher accuracy than PTQ [13]. We used QAT in all of the implementations of the detection models used in the experiments.

2.5. Network Training

Training was performed on a desktop PC with an Intel(R) Core(TM) i7-4790 CPU at 3.60 GHz, 16 GB of RAM, and an NVIDIA RTX 2080 graphics card with 8 GB of memory. The software tools included Linux OS with Python 3.6 and the TensorFlow Model Garden framework. The fine-tuning strategy was performed using models that were pre-trained on the COCO dataset. The learning rate was set to 0.02 for the MobileNetV1 and MobileNetV2 models and to 0.0455 for the MobileNet Edge TPU and MobileDet models. The number of training steps was 30,000 for the MobileNetV1 and MobileNetV2 models and 35,000 for the MobileNet Edge TPU and MobileDet models.

2.6. Model Assessment

The average precision (AP) metric was used to evaluate the model performance. The AP is defined as the area over the curve of precision (P) and recall (R). P was calculated using Equation (1), and R was calculated using Equation (2).

$$P = \frac{TP}{TP + FP}, \tag{1}$$

$$R = \frac{TP}{TP + FN}, \tag{2}$$

where TP, FP, and FN represent true-positive, false-negative, and false-positive results, respectively. The AP is calculated using Equation (3).

$$AP = \sum_n (R_n - R_{n-1}) \times P_n, \tag{3}$$

where P_n, R_n is the respective precision and recall at threshold index n.

3. Results and Discussions

3.1. Ablation Studies

Figures 2- 4 show the detection samples for each peach cultivar (from different orchards).

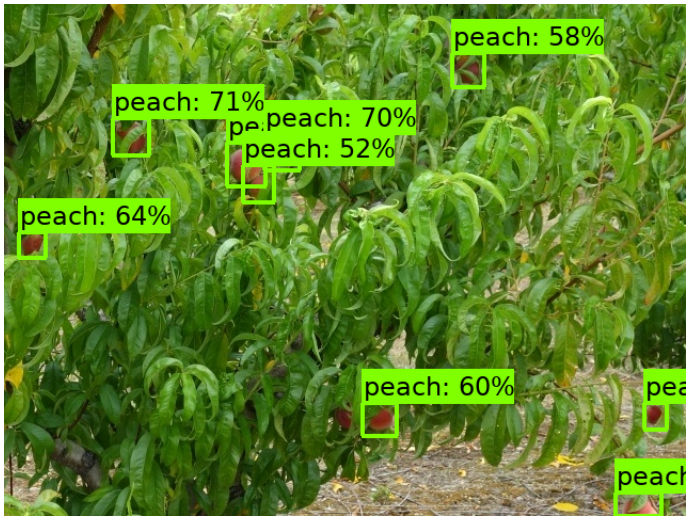


Figure 2. Detection sample for the Royal Time peach cultivar.

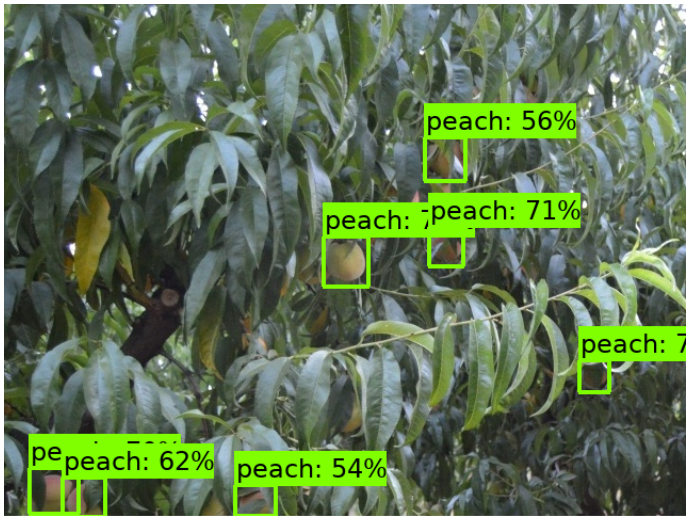


Figure 3. Detection sample for the Sweet Dream peach cultivar.

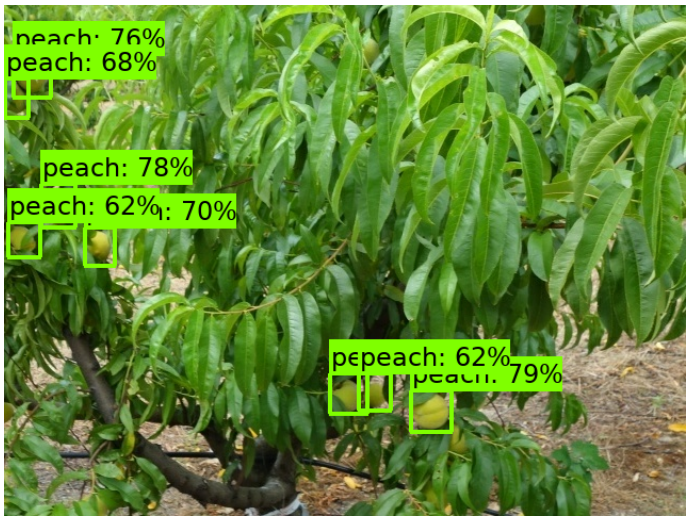


Figure 4. Detection sample for the Catherine peach cultivar.

Table 3 lists the performances of the models and their degradation when converted into inference models (they were optimized to run on the target TPU device). The results showed that SSD MobileDet outperformed the other models and achieved an AP of 88.2%

on the target TPU device. The model with the least degradation (performance drop) was SSD MobileNet Edge TPU with a drop of 0.5%, and the most affected model was SSD MobileNetV2 with a drop of 1.5%. The results, which are shown in Table 3, indicate thatthe models designed (native) to run on a TPU device (SSD MobileDet and SSD EdgeTPU) were approximately 4% better than the models that were not designed (native) to run on a TPU, and that the converting models to run on a TPU accelerator only slightly affected the model detection accuracy. However, the advantage of conversion in terms of inference time was enormous, as described in Section 3.2.

See the *Sample Availability* section at the end of this article for a video demonstration of the detection.

Table 3. Target hardware comparison.

Model	AP (%)		Drop from Baseline to TPU
	Baseline	EdgeTPU	
SSDLite MobileDet	89	88.2	0.8
MobileNet EdgeTPU	88	87.5	0.5
SSD MobileNetV2	86	84.5	1.5
SSD MobileNetV1	85	83.8	1.2

3.2. Inference Time

Table 4 lists the inference times of the models for the target CPU and TPU devices. The model with the lowest latency was SSD MobileNetV1 at 47.6 ms (average). The SSD MobileNet EdgeTPU model exhibited the highest latency (50.5 ms). The maximum difference between the models was 2.9 ms. An important finding was that the inference speed was 20 times faster on average when the model was running on the TPU device and the models designed (native) to run on the CPU (MobileNetV1 and MobileNetV2); however, it was optimized to run on the TPU, and it performed inferences slightly faster than the models designed to run on TPU devices.

Table 4. Inference time comparison.

Model	Latency		
	CPU (ms)	EdgeTPU (ms)	FPS
SSD MobileNetV1	847.9	47.6	21.01
SSDLite MobileDet	1045.9	50.4	19.84
MobileNet EdgeTPU	1232	50.5	19.80
SSD MobileNetV2	773.1	48.4	20.66

3.3. Accuracy and Inference Time Trade-Off

In Sections 3.1 and 3.2, the accuracy (AP) and inference time (ms) of the models for the target TPU device were presented. The models designed specifically for TPU devices had a better detection accuracy, and those designed specifically for CPU (but optimized for TPU) had a better inference time. Thus, there was a trade-off between the accuracy and latency, as shown in Figure 5. Comparing the fastest model (SSD MobileNetV1) with the model that had the best detection accuracy (SSD MobileDet), there was a gain in detection accuracy of 4.4% at the expense of a loss in inference time of 2.8 ms (equivalent to a loss of 1.17 FPS). At a loss of 1.17, the FPS did not significantly affect the performance in the practical applications of computer vision. Therefore, it is justifiable to use SSD MobileDet to improve the recognition accuracy.

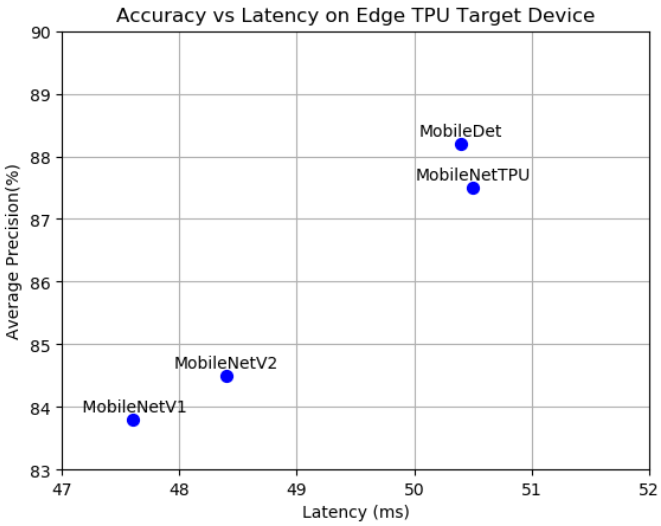


Figure 5. Models' performance on the edge TPU device.

The performance of the SSD MobileDet model presented in this study was compared with the results of other studies. The results are shown in Table 5. Given the lack of practical applications in horticulture for the fruit detection task [22], this comparison provides insight into model performance, edge devices, and price (cost).

Approach 1 was the cheapest and most accurate; however, the combination of the model and device led to a very high inference time. Approach 2 was the most expensive—almost four times the cost of the cheapest—and the least accurate. Nevertheless, they had the best inference times.

Our approach was inexpensive and had a cost similar to that of Approach 1. The accuracy was better than that of Approach 2, but worse than that of Approach 1. The inference time was slightly lower than that of Approach 2, but much better than that of Approach 1. A direct comparison between the approaches in Table 5 was not possible because different datasets and image sizes were used. Considering the price, AP, and latency, our approach of using a TPU accelerator was a good alternative for practical applications.

Table 5. Comparison of the models. Approach_1: Modified YOLOv5 [23], Approach_2: Modified YOLOv3 [22], Ours: SSD MobileDet.

Model	Device | Accel.	Price (€)	Input Size	Fruit	AP (%)	Latency
Approach_1	Jetson Nano | GPU	108	608 × 608	Citrus	93.32	180 (ms)
Approach_2	Jetson Xavier | GPU	429	-	Apple	85	45 (ms)
Our	Raspberry | TPU	141	640 × 480	Peach	88.2	50.4 (ms)

4. Conclusions

In this study, we proposed the use of a lightweight and hardware-aware MobileDet detector model for real-time peach fruit detection applications in conjunction with an edge device and TPU accelerator. A novel annotated dataset of three peach cultivars was created and made available for further studies.

Models designed to run on a TPU device (e.g., SSD MobileDet and SSD EdgeTPU) (hardware-aware) performed approximately 4% (AP) better than models that were not designed to run on a TPU (native). An important result is that the inference speed was, on average, 20 times faster when the model ran on a TPU device than on a CPU. The MobileNetV1 model running on a TPU device performed at 21.01 FPS, and the MobileDet

model performed at 19.84 FPS. At a loss of 1.17, the FPS did not significantly affect the performance for practical computer vision applications. Therefore, it is reasonable to use SSD MobileDet to improve the detection accuracy. A comparison was made with other approaches. However, a direct comparison between the approaches was not possible because different datasets and image sizes were used. Considering the price, AP, and latency, our approach of using a TPU accelerator is a good alternative for practical applications. Further research could also be conducted to explore fruit yield estimates based on the approach presented in this paper.

Author Contributions: Conceptualization: P.D.G. and E.A.; Data curation: E.A.; Formal analysis: E.A., P.D.G., M.P.S. and H.P.; Funding acquisition: P.D.G.; Investigation: E.A. and K.A.; Methodology: E.A. and P.D.G.; Project administration: P.D.G.; Resources: M.P.S., V.N.G.J.S., J.M.L.P.C. and K.A.; Software: E.A.; Supervision: P.D.G.; Validation: E.A. and K.A.; Visualization: E.A.; Writing—original draft: E.A.; Writing—review and editing: P.D.G. and H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded in part by the PrunusBot project—Autonomous controlled spraying aerial robotic system and fruit production forecast, Operation No. PDR2020-101-031358 (leader), Consortium No. 340, Initiative No. 140, promoted by PDR2020 and co-financed by the EAFRD and the European Union under the Portugal 2020 program.

Data Availability Statement: The dataset is available at https://github.com/PeachDataset/Dataset, accessed on 25 September 2022.

Acknowledgments: P.D.G., E.A., and K.A. acknowledge that this work was also supported by the Fundação para a Ciência e Tecnologia (FCT) and C-MAST (Centre for Mechanical and Aerospace Science and Technologies) under project UIDB/00151/2020. E.A, H.P, V.N.G.J.S. and J.M.L.P.C. acknowledge that this work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/50008/2020.

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

Sample Availability: A video demonstration is available at https://user-images.githubusercontent.com/99321854/153617667-830ce756-5e6b-4d08-9011-b68acd001352.mp4, accessed on 25 September 2022.

Abbreviations

The following abbreviations are used in this manuscript:

AP	Average precision
FPS	Frames per second
GPU	Graphics processing unit
TPU	Tensor processing unit
DSP	Digital signal processor
SSD	Single-shot detector
YOLO	You only look once
CNN	Convolutional neural network
AutoML	Auto-machine learning
PTQ	Post-training quantization
QAT	Quantization-aware training

References

1. Roy, P.; Kislay, A.; Plonski, P.A.; Luby, J.; Isler, V. Vision-based preharvest yield mapping for apple orchards. *Comput. Electron. Agric.* **2019**, *164*, 104897. [CrossRef]
2. Assunção, E.; Diniz, C.; Gaspar, P.D.; Proença, H. Decision-making support system for fruit diseases classification using Deep Learning. In Proceedings of the 2020 International Conference on Decision Aid Sciences and Application (DASA), Sakheer, Bahrain, 8–9 November 2020; pp. 652–656.
3. Alibabaei, K.; Gaspar, P.D.; Lima, T.M.; Campos, R.M.; Girão, I.; Monteiro, J.; Lopes, C.M. A Review of the Challenges of Using Deep Learning Algorithms to Support Decision-Making in Agricultural Activities. *Remote Sens.* **2022**, *14*, 638. [CrossRef]

4. Alibabaei, K.; Gaspar, P.D.; Assunção, E.; Alirezazadeh, S.; Lima, T.M. Irrigation optimization with a deep reinforcement learning model: Case study on a site in Portugal. *Agric. Water Manag.* **2022**, *263*, 107480. [CrossRef]
5. Alibabaei, K.; Gaspar, P.D.; Lima, T.M. Modeling soil water content and reference evapotranspiration from climate data using deep learning method. *Appl. Sci.* **2021**, *11*, 5029. [CrossRef]
6. Alibabaei, K.; Gaspar, P.D.; Assunção, E.; Alirezazadeh, S.; Lima, T.M.; Soares, V.N.; Caldeira, J.M. Comparison of on-policy deep reinforcement learning A2C with off-policy DQN in irrigation optimization: A case study at a site in Portugal. *Computers* **2022**, *11*, 104. [CrossRef]
7. Cunha, J.; Gaspar, P.D.; Assunção, E.; Mesquita, R. Prediction of the Vigor and Health of Peach Tree Orchard. In Proceedings of the International Conference on Computational Science and Its Applications, Cagliari, Italy, 13–16 September 2021; pp. 541–551.
8. Assunção, E.; Gaspar, P.D.; Mesquita, R.; Simões, M.P.; Alibabaei, K.; Veiros, A.; Proença, H. Real-Time Weed Control Application Using a Jetson Nano Edge Device and a Spray Mechanism. *Remote Sens.* **2022**, *14*, 4217. [CrossRef]
9. Assunção, E.T.; Gaspar, P.D.; Mesquita, R.J.; Simões, M.P.; Ramos, A.; Proença, H.; Inacio, P.R. Peaches Detection Using a Deep Learning Technique—A Contribution to Yield Estimation, Resources Management, and Circular Economy. *Climate* **2022**, *10*, 11. [CrossRef]
10. Alibabaei, K.; Gaspar, P.D.; Lima, T.M. Crop yield estimation using deep learning based on climate big data and irrigation scheduling. *Energies* **2021**, *14*, 3004. [CrossRef]
11. FARM_VISION. Precision Mapping for Fruit Production. 2021. Available online: https://farm-vision.com/#news (accessed on 11 November 2021).
12. Puttemans, S.; Vanbrabant, Y.; Tits, L.; Goedemé, T. Automated visual fruit detection for harvest estimation and robotic harvesting. In Proceedings of the 2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA), Oulu, Finland, 12–15 December 2016; pp. 1–6.
13. Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv* **2018**, arXiv:1806.08342.
14. Jacob, B.; Kligys, S.; Chen, B.; Zhu, M.; Tang, M.; Howard, A.; Adam, H.; Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 2704–2713.
15. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
16. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
17. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 1314–1324.
18. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6848–6856.
19. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv* **2016**, arXiv:1602.07360.
20. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
21. NVIDIA. NVIDIA TensorRT. 2021. Available online: https://developer.nvidia.com/tensorrt (accessed on 10 December 2021).
22. Zhang, W.; Liu, Y.; Chen, K.; Li, H.; Duan, Y.; Wu, W.; Shi, Y.; Guo, W. Lightweight Fruit-Detection Algorithm for Edge Computing Applications. *Front. Plant Sci.* **2021**, *12*, 2158. [PubMed]
23. Huang, H.; Huang, T.; Li, Z.; Lyu, S.; Hong, T. Design of Citrus Fruit Detection System Based on Mobile Platform and Edge Computer Device. *Sensors* **2022**, *22*, 59. [CrossRef] [PubMed]
24. Tian, Y.; Yang, G.; Wang, Z.; Wang, H.; Li, E.; Liang, Z. Apple detection during different growth stages in orchards using the improved YOLO-V3 model. *Comput. Electron. Agric.* **2019**, *157*, 417–426. [CrossRef]
25. Fu, L.; Majeed, Y.; Zhang, X.; Karkee, M.; Zhang, Q. Faster R–CNN–based apple detection in dense-foliage fruiting-wall trees using RGB and depth features for robotic harvesting. *Biosyst. Eng.* **2020**, *197*, 245–256. [CrossRef]
26. Liu, G.; Nouaze, J.C.; Touko Mbouembe, P.L.; Kim, J.H. YOLO-Tomato: A Robust Algorithm for Tomato Detection Based on YOLOv3. *Sensors* **2020**, *20*, 2145. [CrossRef]
27. Tsironis, V.; Stentoumis, C.; Lekkas, N.; Nikopoulos, A. Scale-Awareness for More Accurate Object Detection Using Modified Single Shot Detectors. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2021**, *43*, 801–808. [CrossRef]
28. Tsironis, V.; Bourou, S.; Stentoumis, C. Tomatod: Evaluation of object detection algorithms on a new real-world tomato dataset. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2020**, *43*, 1077–1084. [CrossRef]
29. Coral. USB Accelerator. 2021. Available online: https://coral.ai/products/accelerator (accessed on 5 October 2021).
30. Xiong, Y.; Liu, H.; Gupta, S.; Akin, B.; Bender, G.; Wang, Y.; Kindermans, P.J.; Tan, M.; Singh, V.; Chen, B. Mobiledets: Searching for object detection architectures for mobile accelerators. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Virtual, 19–25 June 2021; pp. 3825–3834.

31. Dias, C.; Alberto, D.; Simões, M. Produção de pêssego e Nectarina na Beira Interior. *pêssego–Guia prático da Produção. Centro Operativo e Tecnológico Hortofrutícola Nacional*. 2016. Available online: http://hdl.handle.net/10400.11/7076 (accessed on 24 September 2022).
32. Tzutalin. LabelImg. 2015. Available online: https://github.com/tzutalin/labelImg (accessed on 3 May 2021).
33. Raspberry-Pi, F. Raspberry Pi 4. 2021. Available online: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/ (accessed on 5 May 2021).
34. Raspberry. Raspberry Pi Camera Module 2. 2016. Available online: https://www.raspberrypi.com/products/camera-module-v2/ (accessed on 18 September 2022).
35. XLSEMI. 8A 180KHz 40V Buck DC to DC Converter. 2021. Available online: https://www.alldatasheet.com/datasheet-pdf/pdf/1134369/XLSEMI/XL4016.html (accessed on 18 September 2022).
36. Mouser. Li-Ion Battery. 2022. Available online: https://mauser.pt/catalog/product_info.php?products_id=120-0445 (accessed on 18 September 2022).
37. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. Ssd: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
38. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
39. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *39*, 1137–1149. [CrossRef]
40. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
41. Yazdanbakhsh, A.; Seshadri, K.; Akin, B.; Laudon, J.; Narayanaswami, R. An evaluation of edge tpu accelerators for convolutional neural networks. *arXiv* **2021**, arXiv:2102.10423.
42. Howard, A.; Gupta, S. Introducing the Next Generation of On-Device Vision Models: MobileNetV3 and MobileNetEdgeTPU. 2020. Available online: https://ai.googleblog.com/2019/11/introducing-next-generation-on-device.html (accessed on 24 September 2022).
43. Menghani, G. Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. *arXiv* **2021**, arXiv:2106.08962.

Chapter 7

Discussion of Results

This thesis, composed of four projects dealing with the application of computer vision to edge devices in precision agriculture, aims to answer five research questions. In this section, the research questions are answered with the results of the project.

7.1 Discussion of project 1

Research Question 1 (**Q1**) is answered with the result of Project 1.

Q1: How does the deep learning based object detection model handle the detection of peach fruits with different colors, overlaps and occlusions?

Project 1 refers to Chapter 3 of this thesis: **Peaches Detection Using a Deep Learning Technique—a Contribution to Yield Estimation, Resources Management, and Circular Economy.**

In Chapter 3, we applied the faster R-CNN object detection model to evaluate the detection of peaches in images of two orchards. Although peaches have a specific color, size, shape, clustering of fruits and distribution in a tree, the results show that the model handles all these specificities well and achieves an AP of 0.90 for the split test images belonging to the same orchard of the training and an AP of 0.77 for the test images belonging to a different orchard.

Q1 answer: Deep learning-based object detection for peach fruit detection in an uncontrolled environment is a very powerful and robust tool capable of handle the specific characteristics of peach fruit, such as color, size, shape, fruit cluster, and distribution within the tree.

7.2 Discussion of project 2

Research Question 2 (**Q2**) is answered with the result of Project 2.

Q2: Is a deep learning model suitable for fruit disease classification when the amount of data to be trained is small and unbalanced?

Project 2 refers to Chapter 4 of this thesis: **Decision-making support system for fruit diseases classification using Deep Learning.**

In Chapter 4, we used the MobileNetV2 classifier to classify diseases of rot, powdery mildew, and scab in peach fruit. The available training data are small and unbalanced. To overcome this drawback, the techniques of data augmentation and transfer learning were used. The performance of the model reached a macro-average F1 score of 0.96.

Q2 answer: By applying data augmentation and transfer learning techniques, a deep learning-based computer vision model trained with a small amount of unbalanced data is suitable for fruit disease classification.

7.3 Discussion of project 3

Research Questions 3.1 and 3.2 (**Q3.1, Q3.2**) are answered with the result of Project 3.
Q3.1: How does optimization of the model affect the inference time and accuracy?
Q3.2: Is it possible to use the optimized model in a robotic application for precision agriculture?
Project 3 refers to Chapter 5 of this thesis: **Real-Time Weed Control Application Using a Jetson Nano Edge Device and a Spray Mechanism.**
In Chapter 5, we implement a semantic segmentation model to segment weeds. The model was optimized and embedded in an edge device with a GPU accelerator. This computer vision system was integrated into an orchard rover with a spraying mechanism to target weeds with herbicides.
Q3.1 answer: By adjusting the appropriate optimization parameters, a base model can be accelerated by a factor of 14.8.
Q3.2 answer: Yes. This study presents a practical application in which a weed segmentation model is embedded in an edge device and integrated into a spraying mechanism.

7.4 Discussion of project 4

Research Questions 4 (**Q4**) is answered with the result of Project 4.
Q4: Is there an advantage to using TPU accelerators instead of GPUs to perform inference in precision agriculture?
Project 4 refers to Chapter 6 of this thesis: **Real-Time Image Detection for Edge Devices: A Peach Fruit Detection Application.**
In Chapter6, a detector model was implemented, optimized and embedded in an edge device with a TPU accelerator. In this study, the SSD detector with MobileDet backbone achieved a detection accuracy AP of 0.88. And the SSD detector with the MobileNetV1 backbone achieved an inference time of 47.ms.
Q4 answer: With approximate performance values for accuracy and inference time, the TPU accelerator has a better cost-benefit ratio than the GPU. The reason is that the cost of the TPU platform is about 3 times lower than the GPU accelerator platform.

Chapter 8

Conclusions

8.1 General Conclusions

On resource-constrained platforms, such as edge devices, the optimization is mandatory due to limited computational resources. The aim of the present thesis was to investigate the consolidated computer vision models based on deep learning and evaluate the impact of applying the necessary optimizations in the models to run them on edge devices for precision agriculture tasks. In this study, lightweight deep learning models were implemented for peach fruit detection in images, peach fruit disease classification and weed segmentation. The benchmark performance of the lightweight MobileNetV2 model was presented for the (small set of images) peach fruit disease dataset and achieved an F1 score of 0.96. By using MobileNetV2 as the backbone for the semantic segmentation model of DeepLabV3 and applying optimizations, the computational cost of the model was reduced from 81.9 to 4.4 GFLOPs. In addition, the models of the MobileNet variant were used as the backbone for the implementation of the object detector SSD for the detection of peach fruit. The two models, semantic segmentation and object detection, were optimized and used on edge devices (Jetson Nano and Raspberry Pi respectively). On the Jetson Nano, the inference time was accelerated by a factor of 14.8 (compared to the model without optimization). On the Raspberry Pi, the inference time was on average 20 times faster when the optimized model was run on the TPU than on the CPU. One of the key findings from this study is that optimization significantly reduces the inference time without compromising the accuracy of the model. Another important finding is that although the performance of deep learning models is highly dependent on a large dataset for training and the image dataset for precision agriculture is usually small (e.g. peach fruit diseases), it is possible to achieve good accuracy performance by applying appropriate techniques such as transfer learning and fine-tuning.

The second objective of this study was the practical application of precision agriculture with optimized models deployed on edge devices and the creation of an image dataset for precision agriculture. In this research, a semantic segmentation model was used in the Jetson Nano device, which serves as a perception system for an orchard robot in weed control. In addition, a detector in a Raspberry Pi was used to detect peach fruits. The relevance of optimizing models that are not usually designed for edge vices is made clear by the results. The weed control system was able to work in real time to target weeds and the fruit detector performed its inference in 21 FPS. In addition, an annotated dataset of three peach varieties was created and made available online.

Prior to this study, it was difficult to make predictions about the trade-off between accuracy and inference time due to deep learning model optimizations such as shrinkage and quantization. This study provides an important insight into this issue in the field of computer

vision applications for precision agriculture that require real-time responsiveness, such as orchard robots. This is because high inference speed is crucial for fast interaction with the environment. The contribution of this study is also to confirm that deep learning models can utilize the pre-trained models to overcome the lack of large datasets in the field of precision agriculture. In addition, the study contributes to the scientific community by creating an annotated image dataset that can be used for future research.

As far as the research methods are concerned, a limitation must be acknowledged. The programming language used in this study was Python. This language offers a large library geared towards machine learning development and allows for rapid prototyping. However, as the focus is on real-time applications, a faster programming language can also be used. This is the case with C++. This programming language has a high performance. However, too much work has to be done in the development phase. Despite its limitations, the study certainly contributes to our understanding of the optimization of the model and practical applications. In this study, an application for fruit detection was presented. This would be a fruitful area for further work. A tracking algorithm can be used to develop a method for estimating fruit yield. Further research opportunities should also arise from the discussion of limitations (e.g. implementing the algorithm in the C++ programming language).

Bibliography

[1] Ontario_Ministry_of_Agriculture, "Introduction to sustainable agriculture," http://www.omafra.gov.on.ca/english/busdev/facts/15-023.htm, 2016. 1

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, 2001, pp. I–I. 1

[3] N. Kanopoulos, N. Vasanthavada, and R. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 358–367, 1988. 1

[4] 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, p. 5029, 2021. 1

[5] T. van Klompenburg, A. Kassahun, and C. Catal, "Crop yield prediction using machine learning: A systematic literature review," *Computers and Electronics in Agriculture*, vol. 177, p. 105709, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920302301 1

[6] J. Chen, J. Chen, D. Zhang, Y. Sun, and Y. Nanehkaran, "Using deep transfer learning for image-based plant disease identification," *Computers and Electronics in Agriculture*, vol. 173, p. 105393, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919322422 1

[7] B. Caulfield, "What's the difference between a cpu and a gpu?" https://blogs.nvidia.com/blog/whats-the-difference-between-a-cpu-and-a-gpu/, 2009. 1

[8] D. Wu, S. Lv, M. Jiang, and H. Song, "Using channel pruning-based yolo v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments," *Computers and Electronics in Agriculture*, vol. 178, p. 105742, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920318986 5, 8

[9] W. Jia, Y. Tian, R. Luo, Z. Zhang, J. Lian, and Y. Zheng, "Detection and segmentation of overlapped fruits based on optimized mask r-cnn application in apple harvesting robot," *Computers and Electronics in Agriculture*, vol. 172, p. 105380, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919326274 5, 8

[10] 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 5, 8

[11] F. Gao, L. Fu, X. Zhang, Y. Majeed, R. Li, M. Karkee, and Q. Zhang, "Multi-class fruit-on-plant detection for apple in snap system using faster r-cnn," *Computers and Electronics in Agriculture*, vol. 176, p. 105634, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169920314009 5, 8

[12] 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. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919314395 5, 8

[13] B. Espejo-Garcia, N. Mylonas, L. Athanasakos, S. Fountas, and I. Vasilakoglou, "Towards weeds identification assistance through transfer learning," *Computers and Electronics in Agriculture*, vol. 171, p. 105306, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919319854 5, 8

[14] 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. 5, 8

[15] V. N. T. Le, S. Ahderom, and K. Alameh, "Performances of the lbp based algorithm over cnn models for detecting crops and weeds with similar morphologies," *Sensors*, vol. 20, no. 8, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/8/2193 5, 8

[16] V. Czymmek, L. O. Harders, F. J. Knoll, and S. Hussmann, "Vision-based deep learning approach for real-time detection of weeds in organic farming," in *2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, 2019, pp. 1–5. 6, 8

[17] V. Partel, S. Charan Kakarla, and Y. Ampatzidis, "Development and evaluation of a low-cost and smart technology for precision weed management utilizing artificial intelligence," *Computers and Electronics in Agriculture*, vol. 157, pp. 339–350, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169918316612 6, 8

[18] T. Kounalakis, G. A. Triantafyllidis, and L. Nalpantidis, "Deep learning-based visual recognition of rumex for robotic precision farming," *Computers and Electronics in Agriculture*, vol. 165, p. 104973, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016816991930331X 6, 8

[19] A. Rahman, Y. Lu, and H. Wang, "Performance evaluation of deep learning object detectors for weed detection for cotton," *Smart Agricultural Technology*, vol. 3, p. 100126, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772375522000910 6, 8

[20] S. I. Moazzam, U. S. Khan, W. S. Qureshi, T. Nawaz, and F. Kunwar, "Towards automated weed detection through two-stage semantic segmentation of tobacco and weed pixels in aerial imagery," *Smart Agricultural Technology*, vol. 4, p. 100142,

2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S27723
7552200106X 6, 8

[21] Glen Darling. Feedforward neural networks. IBM. Accessed June, 2023. [Online]. Available: https://developer.ibm.com/articles/iot-vs-edge-computing/ 6

[22] NVIDIA. Jetson modules. NVIDIA. Accessed June, 2023. [Online]. Available: https://developer.nvidia.com/embedded/jetson-modules#jetson_orin_nx 6, 7

[23] Raspberry.org. Raspberry cameras. Raspberry Pi. Accessed June, 2023. [Online]. Available: https://www.raspberrypi.com/products/ 7, 8

[24] A. Olsen, "Improving the accuracy of weed species detection for robotic weed control in complex real-time environments," Ph.D. dissertation, James Cook University, Australia, December 2020. [Online]. Available: https://researchonline.jcu.edu.au/65379/ 8

[25] I. Georgievski and M. Aiello, "The potentials of ai planning on the edge," in *2023 IEEE International Conference on Edge Computing and Communications (EDGE)*, 2023, pp. 330–336. 9

[26] 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 techniquemdash;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 10

[27] E. Assunção, C. Diniz, P. D. Gaspar, and H. Proença, "Decision-making support system for fruit diseases classification using deep learning," in *2020 International Conference on Decision Aid Sciences and Application (DASA)*. IEEE, 2020, pp. 652–656. 24

[28] E. Assunção, P. D. Gaspar, R. Mesquita, M. P. Simões, K. Alibabaei, A. Veiros, and H. Proença, "Real-time weed control application using a jetson nano edge device and a spray mechanism," *Remote Sensing*, vol. 14, no. 17, 2022. [Online]. Available: https://www.mdpi.com/2072-4292/14/17/4217 30

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)	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	The sub-field of Machine learning. In deep learning, computational models with numerous processing layers can learn representations of data at different levels of abstraction.
Dropout	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	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)	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)	The average of the squared differences between prediction and actual observation.
Pooling Layer	A form of non-linear down-sampling layer in CNN moldel.
Recurent Neural Network (RNN)	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	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)	A deep learning model designed for real-time object detection.