Article

# Real-time Image Detection for Edge Device: A Peach Fruit Detection Application

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

[1] University of Beira Interior, Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã, Portugal; eduardo.assuncao@ubi.pt (E.A.); k.alibabaei@ubi.pt (K.A.); hugomcp@di.ubi.pt
[2] C-MAST Center for Mechanical and Aerospace Science and Technologies, University of Beira Interior, 6201-001 Covilhã, Portugal
[3] Instituto de Telecomunicações, Rua Marquês d'Ávila e Bolama, 6201-001, Covilhã, Portugal (V. N. G. J. S.): vasco.g.soares@ipcb.pt
[4] School of Agriculture, Polytechnic Institute of Castelo Branco, 6000-084 Castelo Branco, Portugal; mpaulasimoes@ipcb.pt
[5] Polytechnic Institute of Castelo Branco, Av. Pedro Álvares Cabral nº 12, 6000-084, Castelo Branco, Portugal; jcaldeira@ipcb.pt
* 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 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 inference at high speed. In this area, most modern applications use an integrated graphics processing unit (GPU). In this work, we propose to use a Tensor Processing Unit (TPU) accelerator with the Raspberry Pi target device and the state-of-the-art, lightweight, and hardware-aware MobileDet detector model. Our contribution is to extend the possibilities of using accelerators (TPU) for edge devices in precision agriculture. The proposed method was evaluated in 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 frame 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 Device: A Peach Fruit Detection Application . *Journal Not Specified* **2022**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:



## 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 world population and consequent higher food demand, associated with food habits change for healthier foods such as fruits and vegetables increasing the specific demand of this type of product, the impact of climate change in agricultural activities, and the human exodus to urban areas reducing the workforce available in rural areas enhance the improvement of the efficiency and efficacy 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 the management of these agricultural practices [3].

The detection of fruits through automatic systems, and particularly of peaches, can contribute to improving the efficiency of agricultural cultivation processes, whether through the adequate and sufficient supply of water [4–6], fertilizers supply, evaluation of the vigor and health state [7], ripening state, and diseases [2], and even improve weed control [8].

The management of agricultural practices supported by artificial intelligence decision-making systems helps yield estimation, resources management, and circular economy [9,10]. These approaches can contribute to increasing production and rentability, through improved supply contracts and fixed costs reduction. Additionally, these results are part of an improvement in the crop's environmental sustainability by the reduction of fertilizers and contributing to the food loss reduction.

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 inference 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 rates 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 inference time of 0.304 s. It is important to emphasize that the approach used in this study is 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, 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 uses 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 inference time of 54 ms in a PC target device. Because the target device is 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 is 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 speed of the Jetson Xavier reaches 46.9 ms, 40.3 ms, and 45.0 ms (orange, tomato, and apple, respectively) for image resolutions of different sizes. This approach falls 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 608 × 608 resolution images, they achieved a detection accuracy of 93.32% AP and an edge-computing processing speed of 180 ms. Based on the model used and 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 evaluate the proposed adapted model in tomato fruit detection and classification for three maturity stages of each tomato fruit. In the image resolution of 515× 512 using a PC with a standard GPU (not portable) the model perform inference speed of 200 FPS. In addition, the model was not optimised in terms of edge divice 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 was developed for general image detection tasks [30]. In this work, we propose to use these two technology with the 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 embedded in a Raspberry Pi target device along with a Coral edge TPU accelerato.
- We present a novel dataset of the three peach cultivars with annotations and make it available for further study (to our knowledge, the first work of its kind).

The remainder of the 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 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 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.

The images were taken with a Sony DSC-RX100M2 red, green, blue (RGB) camera. The images were then resized to $640 \times 480$ pixels resolution. Subsequently, the images were manually labelled 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 at *Data Availability* section at the end of this article.

### *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- An accelerator Coral TPU [29]; 3- A Raspberry Pi Camera Module 2 [34]; 4- A DC to DC Converter [35]; and 5- Three Li-Ion Battery [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 has a quad-core Cortex A72 processor, 8 GB RAM, and Linux operating system with a Python interpreter and TensorFlow Lite library. The coral TPU accelerator, which connects to Raspberry Pi via a USB, is an integrated edge TPU coprocessor designed to perform machine learning operations in an optimized manner (e.g., four Tera operations per second).

### *2.3. SSD: Single Shot Detector*

A single-shot detector (SSD) [37] is an state-of-the-art object detection model that outperforms its competitors You only look once (YOLO) [38] [31] 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. Similar to any model for computer vision tasks based on deep learning, the first step of SSD is the 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].

**Table 1.** Examples of peach tree cultivar with their predominant fruit density.

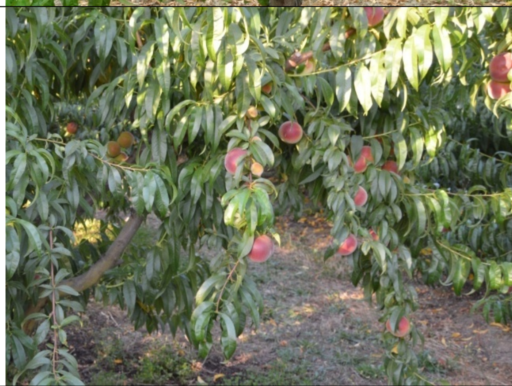
| Cultivar | Sample Image | Fruit Density | Color |
|---|---|---|---|
| Royal Time |  | Low | Red |
| Sweet Dream |  | Medium | Dark Red |
| Catherine |  | High | Yellow |

**Table 2.** Statistics of the dataset.

| Split | Cultivar | Images | Fruits (Labels) |
|---|---|---|---|
| Train | Sweet Dream | 270 | 2,015 |
| | Royal Time | 248 | 1,066 |
| | Catherine | 305 | 4,564 |
| Test | Sweet Dream | 66 | 453 |
| | Royal Time | 63 | 270 |
| | Catherine | 76 | 1,480 |
| **Total of train** | | **823** | **7,645** |
| **Total of test** | | **205** | **2,203** |

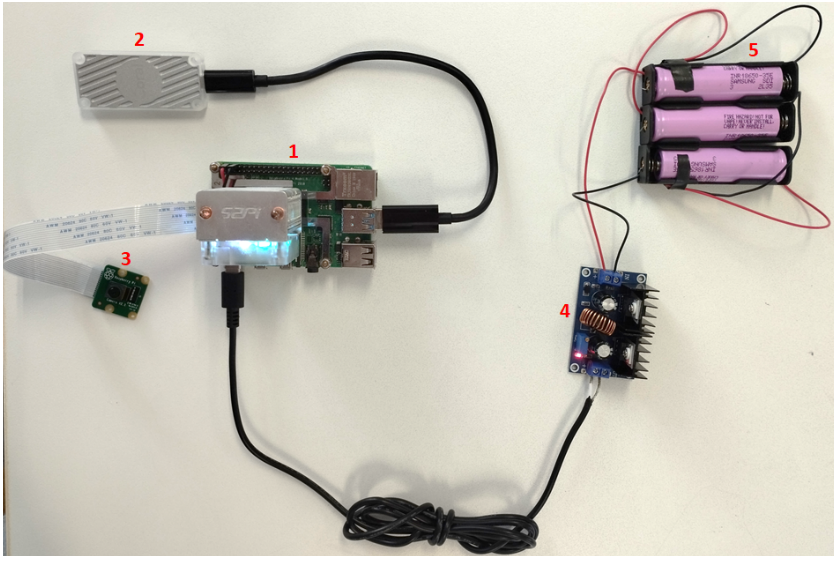



**Figure 1.** Hardware platform (edge device) for performing inference.

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 $1 \times 1$ convolution). MobileNetV1 uses $3 \times 3$ depth-wise separable convolutions, which require eight–nine times less computation than standard convolutions, with only slightly lower accuracy [15].

#### 2.3.2. MobilenetV2

MobiliNetV2 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 depthseparable 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 an opposite manner 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 learn (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

MobiliDet is the latest version of the SSD model based on the MobileNet family. Again, the AutoML approach is 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, native efficient models were created by considering model size and computational power. This is the case with several models such as MobileNet and SqueezeNet. Another approach for increasing the performance of an edge device (faster inference and memory accesses) is to apply 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 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 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 @ 3.60GHz, 16 GB RAM, and an NVIDIA RTX 2080 graphics card with 8 GB of memory. Software tools included Linux OS with Python 3.6 and the TensorFlow Model Garden framework. The fine-tuning strategy was performed using pre-trained models in the COCO dataset. The learning rate was set to 0.02 for the MobileNetV1 and MobileNetV2 models and 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 is used to evaluate 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 the Equation 3.

$$AP = \int_0^1 P_{(R)}\, dR, \tag{3}$$

## 3. Results and Discussions

### *3.1. Ablation Studies*

Figure 2, 3 and 4 show the detection samples for each peach cultivar (from different orchards).


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

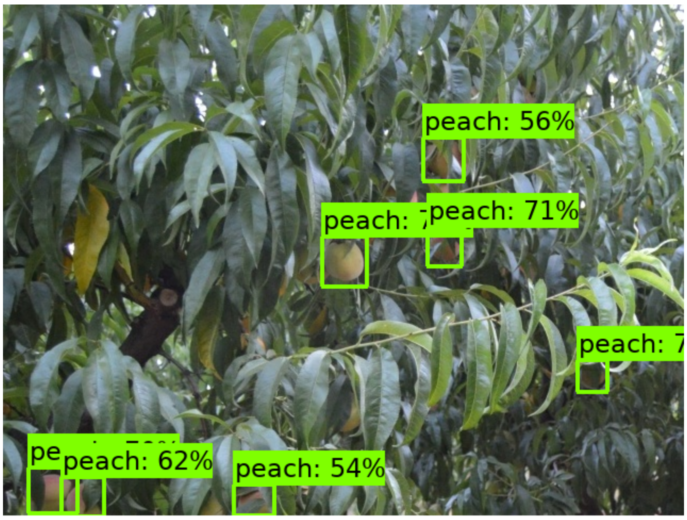



**Figure 3.** Detection sample for Sweet Dream peach cultivar.

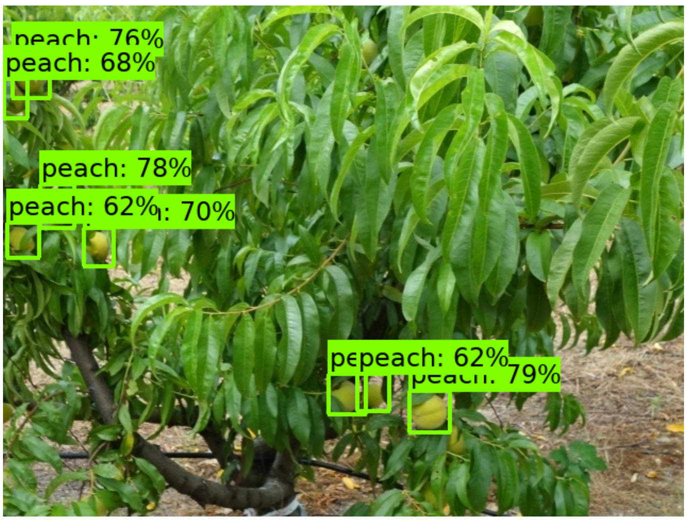



**Figure 4.** Detection sample for Catherine peach cultivar.

Table 3 lists the performance of the models and their degradation when converted to inference models (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 TPU target device. The model with the least degradation (performance drop) was SSD MobilNet EdgeTPU with a drop of 0.5%, and the most affected model was SSD MobileNetV2 with a drop of 1.5%. The results, shown in Table 3, indicate that models designed (native) to run on a TPU device (SSD MobileDet and SSD EdgeTPU) are approximately 4% better than models not designed (native) to run on a TPU, and that converting models to run on a TPU accelerator only slightly affects the model detection accuracy. However, the advantage of conversion in terms of inference time is 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 | EdgeTPU | Baseline to TPU |
| 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 CPU and TPU target 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 is 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 TPU and perform inference 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 | 1,045.9 | 50.4 | 19.84 |
| MobileNet EdgeTPU | 1,232 | 50.5 | 19.80 |
| SSD MobileNetV2 | 773.1 | 48.4 | 20.66 |

### *3.3. Accuracy and Inference Time Trade-off*

In subsections 3.1 and 3.2, the accuracy (AP) and inference time (ms) of the models for the TPU target device are 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 is a trade-off between the accuracy and latency, as shown in Figure 5. Comparing the fastest model (SSD MobileNetV1) with the model that has the best detection accuracy (SSD MobileDet), there is 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.

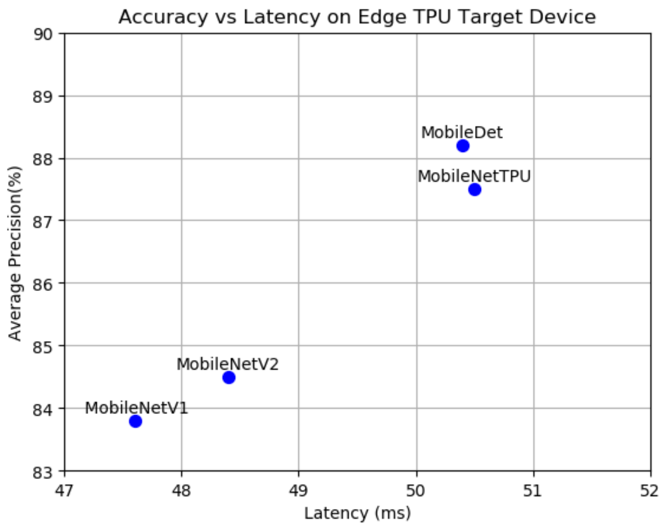



**Figure 5.** Models performance on Edge TPU device.

The performance of the SSD MobileDet model presented in this study is 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 is the most expensive, almost four times cheapest, and the least accurate. Nevertheless, they had the best inference times.

Our approach is inexpensive and has 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 is not possible because different datasets and image sizes are used. Considering the price, AP, and latency, our approach of using a TPU accelerator is a good alternative for practical application.

**Table 5.** Models comparison. Approach_1: Modified YOLOv5 [23], Approach_2: Modified YOLOv3 [22], Our: SSD MobileDet.

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

## 4. Conclusions

In this study, we propose 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 the 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 not designed to run on a TPU (native). An important result is that the inference speed is on average 20 times faster when the model runs on a TPU device than on a CPU. The MobileNetV1 model running on a TPU device performs 21.01 FPS and the MobileDet model performs 19.84 FPS. At a loss of 1.17, the FPS did not significantly affect the performance of the practical computer vision applications. Therefore, it is reasonable to use SSD MobileDet to improve the detection accuracy. A comparison was made with the other approaches. However, a direct comparison between the approaches is 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 application. Further research could also be conducted to explore a fruit yield estimate 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.

**Institutional Review Board Statement:** Not applicable.

**Data Availability Statement:** The dataset is available at https://github.com/PeachDataset/Dataset.

**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. V.N.G.J.S. and J.M.L.P.C. acknowledge that this work is funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the project UIDB/EEA/50008/2020.

**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.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| AP | Average precision |
| FPS | Frame 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 learn |
| 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. *Computers and Electronics in Agriculture* **2019**, *164*, 104897.
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). IEEE, 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 Sensing* **2022**, *14*, 638.

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. *Agricultural Water Management* **2022**, *263*, 107480.
5. Alibabaei, K.; Gaspar, P.D.; Lima, T.M. Modeling soil water content and reference evapotranspiration from climate data using deep learning method. *Applied Sciences* **2021**, *11*, 5029.
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.
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. Springer, 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 Sensing* **2022**, *14*, 4217.
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.
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.
11. FARM_VISION. PRECISION MAPPING FOR FRUIT PRODUCTION. https://farm-vision.com/#news, 2021. Accessed: 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). IEEE, 2016, pp. 1–6.
13. Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342* **2018**.
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 Proceedings of the IEEE conference on computer vision and pattern recognition, 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 preprint arXiv:1704.04861* **2017**.
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, 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 Proceedings of the IEEE/CVF International Conference on Computer Vision, 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 Proceedings of the IEEE conference on computer vision and pattern recognition, 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 preprint arXiv:1602.07360* **2016**.
20. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.
21. NVIDIA. NVIDIA TensorRT. https://developer.nvidia.com/tensorrt, 2021. Accessed: 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. *Frontiers in Plant Science* **2021**, *12*. https://doi.org/10.3389/fpls.2021.740936.
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.
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. *Computers and Electronics in Agriculture* **2019**, *157*, 417–426. https://doi.org/https://doi.org/10.1016/j.compag.2019.01.012.
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. *Biosystems Engineering* **2020**, *197*, 245–256.
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*. https://doi.org/10.3390/s20072145.
27. Tsironis, V.; Stentoumis, C.; Lekkas, N.; Nikopoulos, A. Scale-Awareness for More Accurate Object Detection Using Modified Single Shot Detectors. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* **2021**, *43*, 801–808.
28. Tsironis, V.; Bourou, S.; Stentoumis, C. TOMATOD: EVALUATION OF OBJECT DETECTION ALGORITHMS ON A NEW REAL-WORLD TOMATO DATASET. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences* **2020**, *43*.
29. Coral. USB Accelerator. https://coral.ai/products/accelerator, 2021. Accessed: 05-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 Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 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**.
32. Tzutalin. LabelImg. https://github.com/tzutalin/labelImg, 2015. Accessed: 03-May-2021.
33. Raspberry-Pi, F. Raspberry Pi 4. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/, 2021. Accessed: 05-May-2021.
34. Raspberry. Raspberry Pi Camera Module 2. https://www.raspberrypi.com/products/camera-module-v2/, 2016. Accessed: 18-September-2022.
35. XLSEMI. 8A 180KHz 40V Buck DC to DC Converter. https://www.alldatasheet.com/datasheet-pdf/pdf/1134369/XLSEMI/XL4016.html, 2021. Accessed: 18-September-2022.
36. Mouser. Li-Ion Battery. https://mauser.pt/catalog/product_info.php?products_id=120-0445, 2022. Accessed: 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. Springer, 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 Proceedings of the IEEE conference on computer vision and pattern recognition, 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 transactions on pattern analysis and machine intelligence* **2016**, *39*, 1137–1149.
40. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition, 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 preprint arXiv:2102.10423* **2021**.
42. Howard, A.; Gupta, S. Introducing the Next Generation of On-Device Vision Models: MobileNetV3 and MobileNetEdgeTPU, 2020.
43. Menghani, G. Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better. *arXiv preprint arXiv:2106.08962* **2021**.