IECAG 2021

MDPI

Proceedings

Weed Detection using Computer Vision and Artificial Intelligence in the Raspberry Pi Platform as an Edge Device

Luiz Carlos Marques Junior[1] and José Alfredo Covolan Ulson [2]

[1] Department of Electrical Engineering, School of Engineering, São Paulo State University (UNESP), Bauru 17033-360, Brazil; luiz.c.marques-junior@unesp.br
[2] Department of Electrical Engineering, School of Engineering, São Paulo State University (UNESP), Bauru 17033-360, Brazil; alfredo.ulson@unesp.br
* Correspondence: luiz.c.marques-junior@unesp.br;
† Presented at the 1st International Electronic Conference on Agronomy, 3–17 May 2021; Available online: https://sciforum.net/conference/IECAG2021

Abstract: Retaining the balance between high productivity and quality while efficiently managing available resources is one of the biggest challenges facing the agricultural sector. As a possible solution to overcome these challenges and obstacles, precision agriculture has emerged in recent years. Among the promising solutions that precision agriculture offers, is the use of edge computing devices for monitoring and acquiring data in the rural environment, processing information locally and in real time. Computer Vision and Artificial Intelligence, more specifically referred to as Deep Learning, have also being applied recently in agriculture for different tasks such as image classification, object detection and semantic segmentation. However, there is a challenge and limitation in transferring this technology to more affordable platforms to process large quantities of data. Therefore, in this work, we explored the use of Computer Vision and Deep Learning applied to the object detection task in edge devices, specifically the Raspberry PI 4 platform, without hardware acceleration. We decided to apply this methodology for weed detection, since weeds are currently one of the pests that result in greatest loss of productivity in agriculture, and also have rapidly developed resistance to commercial herbicides. Also, in order to evaluate the performance gain for real-time weed detection on the Raspberry Pi platform, quantizaion of the deep neural network architectures using TensorFlow Lite was tested. The experimental results suggests that the proposed methodology is functional, and it was achieve real time weed detection on Raspberry Pi 4 platform, making it possible to reproduce the experimental results on similar edge devices.

Keywords: Weed Detection; Precision Agriculture; Artificial Intelligence; Deep Learning; Raspberry Pi; Edge Computing

Citation: Lastname, F.; Lastname, F.; Lastname, F. Title. *Proceedings* **2021**, *68*, x. https://doi.org/10.3390/xxxxx

Published: date

1. Introduction

World population is expected to achieve 10 billion by 2050, according with FAO (Food and Agriculture Organization of the United Nations). Population increase will boost agriculture demand as well as add pressure on natural resources [1]. In this scenario agroindustry is a sector that represents key importance for every country, thus, it is essential to maintain a high crop yield and yet manage with efficiency and sustainability the resources used for crop production.

A factor that directly affects crop yield is the infestation of weeds and pests, since many weed species have become resistant for commercial herbicides, especially glyphosate [2], causing crops losses and depreciation. Data from EMBRAPA (Brazilian Agriculture Research Corporation), indicated that weed plants appear as one of the factors that most affect agricultural productivity. Damaging the planted crop either by competing for resources such as sun, water and nutrients, or by allelopathy, which is the ability of plants

to produce substances that are harmful to other plants and consequently causing loss of production yield. In cases where no weed control is done, losses can reach more than 90% of production, with control the average is 13 to 15% loss in production [3].

Therefore, there is an urge to apply new technologies to enable weed spot elimination, whether it is mechanical, electrical or chemical in a controlled amount of herbicide. As a possible solution for the challenge of identifying weeds for local elimination deep learning associated with computer can be applied. Deep Learning belongs to the Artificial Intelligence and Machine Learning fields of research, and achieved remarkable results recently on renowned computer vision datasets for tasks such image classification [4],[5],[6],[7] and object detection [8], [9], [10].

However, two major issues appear when deploying deep learning architectures, first the large amount of data necessary for training the model, second the necessity of a powerful hardware enhanced with GPU (Graphical Processing Unit) to allow real time weed detection. To address these issues, the tested architectures were trained on an image dataset which consists of 5 weed species resistant of the major herbicides used in Brazil. The image dataset was composed of 2000 images, where 1500 images were used for training and 500 images were used for validation which is a reduced dataset size. Second it was explored a hybrid approach, where the deep learning architecture was trained using the cloud resources on Google Collab and after training, the weights were quantized using post training quantization on the Tensorflow Lite Converter.

Results indicate that it is possible to perform real time weed detection (FPS =>30) on the Raspberry Pi 4 platform. The SSD MobileNet V2 FPNLite 320x320 model, quantized to int8 achieved the best detection results, with an FPS varying of 65 to 70. To the best of our knowledge our approach is the first to fully deploy deep learning models on the Raspberry Pi 4 platform and achieve real time weed detection without hardware acceleration.

2. Materials and Methods

The materials and methods are organized as follows: The data preparation and augmentation are described in Section 2.1. In Section 2.2, the selection of the deep neural networks models for object detection and training are described. Finally, in Section 2.3 the quantization and deployment of quantized models on the raspberry pi are described.

2.1. Data preparation and Augmentation

In order to train and evaluate the performance of the deep learning models an image dataset was created. The dataset consists of 5 weed species resistant to the major herbicides used commercially in Brazil, especially glyphosate. The species are *Capim-Azevém* (*Lolium Multiflorum*), *Buva* (*Conyza Bonariensis*), *Capim-Amargoso* (*Digitaria Insularis*), *Capim-Pé-de-Galinha* (*Eleusine Indica*) and *Caruru Palmeri* (*Amaranthus Palmeri*). Pictures of each weed are shown on figure 1.

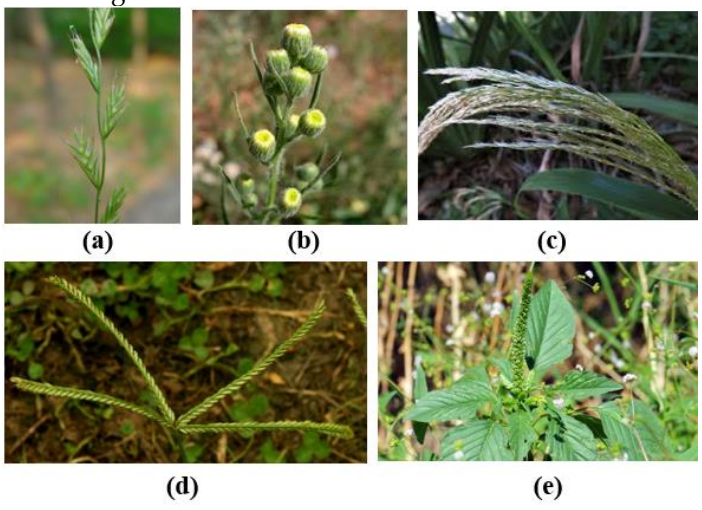

Figure 1. Sample images of weed species resistant of herbicides. (a) Capim-Azevém, (b) Buva, (c) Capim-Amargoso, (d) Capim pé de Galinha, (e) Caruru. (Source: Designed by the authors).

Data augmentation was used to create the dataset. This is an important step, because it increases the number of images for training, as well as provides difficulty scenarios, which are important to validate the model robustness for classification and object detection. The transformations applied for augmentation were rotation, horizontal and vertical flip, the addition of image blurring and noise. By the end of data augmentation, the image dataset was composed of 2000 images, where 500 images were used for validation, being 100 images for each weed class, and 1500 images used for training, being 300 images for each class. Posteriorly, all images were cropped using the labelImg tool [11], used to select and save the weed species of interest inside every image.

After image labeling and cropping a CSV (Comma-separated values) format file was created based on the XML files from training and testing dataset. The CSV files contain information regarding image height and width, weed class, number and coordinates of the weeds inside the image. The CSV were used to generate test and train files in TFRecord file format. The TFRecord is Tensorflow own binary storage format and it is used as basis input data for training and validation of the deep learning models. Once all data preparation was finished the next step was to select the deep learning models.

2.2. *Selection of the Deep Neural Networks for Object Detection and Models Training*

Object detection is one of the lines of research and development of computer vision, in which video and image processing techniques are used, along with algorithms to perform the extraction of characteristics and, from these characteristics, detect an object of interest. In this work the deep learning architectures were selected from Tensorflow Hub [12], which is a Google library containing a series of applied neural network architectures and Machine Learning algorithms in various tasks involving, image classification and detection, speech recognition, semantic segmentation, etc.

The object detection models available in Tensorflow Hub, were originally trained and tested in the Microsoft COCO (Common Objects in Context) image dataset, [13]. Originally the COCO dataset contains 90 classes of common objects and has been trained in approximately 2.5 million labeled images, also it is widely recognized and used by the scientific community in the field of computer vision and artificial intelligence.

Speed was the main architectures selection factor, followed by mAP (Mean Average Precision). The mAP is the major accuracy metric used on COCO dataset. Based on these parameters the selected models were SSD MobileNet V2 FPNLite 320x320[14], SSD MobileNet V2 FPNLite 640x640[14] and SSD ResNet50 V1 640x640[15]. In table 1 it is shown information regarding the selected models, such as the speed for processing the algorithm in milliseconds (ms), the average precision obtained by these algorithms in the COCO dataset (COCO mAP) and the type of output generated by the model, in this case all architectures create a box (boxes) around the detected object.

Table 1. Deep Learning Object Detection Architectures Selected.

Model Name	Speed (ms)	COCO mAP	Outputs
SSD MobileNet V2 FPNLite 320x320	22	22.2	Boxes
SSD MobileNet V2 FPNLite 640x640	39	28.2	Boxes
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	46	34.3	Boxes

Two important aspects to be emphasized is that although the inference speed (classification or object detection) of these networks is small, in the range of milliseconds, the

networks were tested on devices with higher processing and graphical processing units compared to the Raspberry Pi 4. Therefore, many of the architectures available on the Tensorflow Hub are unfeasible to apply on Raspberry. Second not all models available on the Tensorflow Hub support quantization, therefore this information was also taken into consideration to select the deep learning models.

The training of the architectures was performed on Google Collab, which is a free online platform that enables the training of artificial intelligence algorithms using Google's cloud processing. All models were trained for 20000 iterations, and used Tensorflow, which is an open-source platform for machine learning. [16].

2.4. *Models Quantization and Deployment on the Raspberry Pi 4*

After training, post-training quantization was used. Post-training quantization is as a conversion technique to reduce model size while also improving CPU and hardware accelerator latency, with little degradation in model accuracy [17]. The process of training and quantizing a model to implement it into a hardware less powerful can be considered a hybrid technique, beneficial to edge computing devices. In figure 2 it is shown a representation of the process described, where the steps of training and conversion are done in the server a more powerful computing device, and the inference is done on the client, that can be a smartphone and in our case is the Raspberry Pi 4 platform.

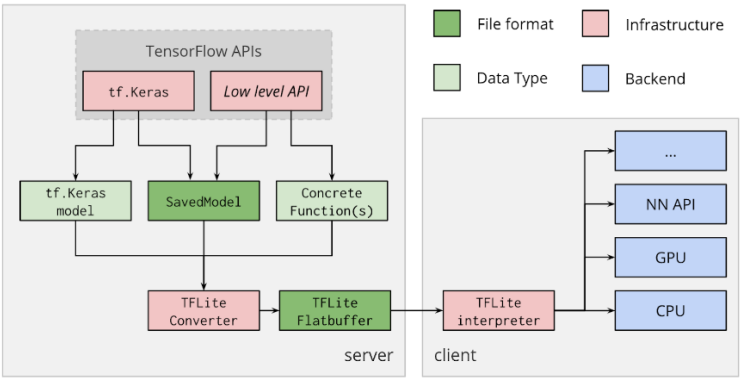

Figure 2. Model training, quantization and deployment pipeline (Source: Tensorflow Lite).

The device used in the experiments is the Raspberry Pi 4, integrated with a Raspberry Pi camera board v.1. The board has a Quad core Cortex-A72 (ARM v8) 64-bit 1.5GHz SoC processor, 4 GB of DDR4 RAM, 2 coolers were installed along with thermal sinks to prevent overheating in the processor. Figure 3 shows the Raspberry 4 device and components used in this study.

Figure 3. Raspberry Pi 4 platform used in experiments (Source: Designed by the authors).

3. Results and Discussion

In this section, the results of quantization and deployment of the deep learning models on the Raspberry Pi are provided. Section 3.1 discusses the effects of quantization on the tested architectures. In Section 3.2 the experimental results of SSD MobileNet 320x320 on the Raspberry are provided.

3.1. Effects of Quantization of Neural Networks for Object Detection

For the experiment, the object detection models were quantized into 3 formats: float32, which is the standard size generated for a tflite file, float16 and int8. In table 2 it is shown the dimension as well as the difference of quantized tflite files.

Table 2. Quantization results and types applied on the Models.

Model Name	Float32	Float16	Int8
SSD MobileNet V2 FPNLite 320x320	11.238 KB	5.702 KB	3.675 KB
SSD MobileNet V2 FPNLite 640x640	11.837 KB	6.001 KB	4.274 KB
SSD ResNet50 V1 FPN 640x640 (RetinaNet50)	198.152 KB	99.153 KB	51.234

The Int8 type was the lightest quantization for all trained architectures, where SSD MobileNet V2 FPNLite 320x320 has the smaller value, almost a third of the size of the float32 model. SSD ResNet50 V1 640x640 also decreased significantly from approximately 198 MB (Megabytes) to 51 MB.

Although all models can be deployed on the Raspberry Pi, only the SSD MobileNet V2 FPNLite 320x320 achieved real time object detection results. Despite the weight size of SSD MobileNet V2 FPNLite 640x640 is slightly bigger than MobileNet V2 320x320, the model was not capable to surpass the 30 FPS mark. First because it demands a higher camera resolution, the minimum resolution that the Raspberry Pi camera can work is 640x480 pixels, which worked for the MobileNet V2 320x320, however, for the 640x640 model it does not, second the image processing task is higher for the 640x640 pixels models, which decreased the performance significantly. On the next section results and performance of real time weed detection using the SSD MobileNet V2 FPNLite 320x320 are sohwn.

3.2. Weed Detection Results Using Quantized MobileNet 320x320

Originally the tflite file has a format float32, during the tests with the float32 file it was obtained from 32 to 35 frames per second. Next, a float16 quantized file was tested, with the file in float16, 35 to 38 frames per second were obtained. Finally, a quantized tflite file of int8 was tested. The int8 file obtained 66 to 70 frames per second, the best results. Experimental results are shown in figure 4.

Figure 4. Real Time Weed Detection Results: (a) Capim pé de Galinha weed detection with float32 quantization, (b) Capim pé de Galinha weed detection with float16 quantization (c) Capim pé de Galinha weed detection with int8 quantization. (Source: Designed by the authors).

5. Conclusions

In this work, the use of the Raspberry Pi platform as an edge device for computer vision applications involving the weed detection was explored. The experiment proved to be functional, which is an important advance, since deep neural network architectures are generally implemented in devices with higher processing capacity than Raspberry and at higher cost. The possibility of using this device as a low-cost, real-time weed detection system allows even small farmers to have access to a technology that uses computer vision and artificial intelligence.

The SSD MobileNet V2 FPNLite 320x320 implemented in Raspberry, had low accuracy in detection, where the network repeatedly confused the classes of weeds. Therefore, for future work, it is intended to explore more complex networks with greater average precision and to verify whether it is possible to implement them on the Raspberry Pi platform.

The methodology used proved to be functional to detect weeds in real time, therefore it is possible to extend the applications of this study to other edge devices with hardware characteristics similar to Raspberry Pi, such as smartphones for real time weed and diseases detection.

Author Contributions: Conceptualization, Methodology, Software, Writing—original draft L.c. m.J., Writing—review and editing, J.a.c.U. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by CAPES Foundation, grant number #88887.489872/2020-00.

Acknowledgments: The authors of this work would like to gratefully CAPES (Higher Education Personnel Improvements Coordination). We also are grateful to the anonymous reviews for improving this article.

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

References

1. FAO. 2017. ***The future of food and agriculture – Trends and challenges***. Rome.
2. Aegro, *"9 fatos primordiais para o manejo de ervas daninhas resistentes ao glifosato"*. Avaliable online: https://blog.aegro.com.br/ervas-daninhas-resistentes-a-glifosato/ accessed on 22/11/2020.
3. Brazilian Agriculture Research Corporation (EMBRAPA), *"Weeds."* Avaliable online: https://www.embrapa.br/en/tema-plantas-daninhas/sobre-o-tema accessed on 22/11/2020.
4. Krizhevsky, A; Sutskever, I; Hinton, G. E. ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*, San Diego, **2012**, *v. 25*, p. 1097-1105.
5. Simonyan, K; Zisserman, A. Very deep convolutional networks for Large-Scale image Recognition. *ICLR*, San Diego, **2015**.
6. Szegedy, C. et al. Going Deeper with Convolutions. *CVPR*, Boston, **2015**, v. 1, p. 1 - 9.
7. He, K. et al. Deep Residual Learning for Image Recognition. *CVPR***,** Las Vegas, **2016**, v. 1, p. 770-778.
8. Sermanet, P et al. Overfeat: Integrated recognition, localization and detection using convolutional networks. *ICLR*, Banff, **2014**.
9. Ren, S et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *TPAMI*, **2016**, v.39, p. 1137 - 1149.
10. Redmon, J et al. You Only Look Once: Unified, Real-Time Object Detection. *CVPR***,** Las Vegas, **2016**, v. 1, p. 779 - 788.
11. Tzutalin, D. *Labellmg*. Avaliable online: https://github.com/tzutalin/labelImg accessed on 15/12/2020.
12. Tensorflow Hub. *Object_Detection***.** Avaliable online: https://tfhub.dev/tensorflow/collections/object_detection/1 accessed on 10/01/2021.
13. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. *ECCV*, Zurich, Switzerland, **2014**, p. 740–755.
14. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A. & Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. *CVPR*, Salt Lake City, USA, **2018,** p. 4510-4520.
15. He, K.; Zhang, X.; Ren, S. & Sun, J. Deep Residual Learning for Image Recognition. *CVPR*, Las Vegas, USA, **2018,** p. 770-778.
16. Bisong E. (2019) Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-4470-8_7
17. Tensorflow Lite. *Post-training quantization***.** Avaliable online: https://www.tensorflow.org/lite/performance/post_training_quantization accessed on 05/02/2021.