FEDERAL UNIVERSITY OF RIO GRANDE DO SUL
ENGINEERING SCHOOL
CONTROL AND AUTOMATION ENGINEERING

DAVID RUTHERFORD ARMSTRONG - 00275628

MACHINE LEARNING FOR IDENTIFICATION AND CLASSIFICATION OF CROPS AND WEEDS

Porto Alegre
2021

DAVID RUTHERFORD ARMSTRONG - 00275628

MACHINE LEARNING FOR IDENTIFICATION AND CLASSIFICATION OF CROPS AND WEEDS

Graduation Project presented to COMGRAD-CCA of Federal University of Rio Grande do Sul in partial fulfillment of the requirements for the degree of *Control and Automation Engineering*.

ADVISOR:
Prof. Dr. Marcelo Götz

COADVISOR:
Dr. Vitor Camargo Nardelli

Porto Alegre
2021

DAVID RUTHERFORD ARMSTRONG - 00275628

MACHINE LEARNING FOR IDENTIFICATION AND CLASSIFICATION OF CROPS AND WEEDS

This Project was considered adequate for obtaining the credits of the course TCC (Diplom Project) of *Control and Automation Engineering* and approved in its final form by the Advisor and the Examination Committee.

Advisor: _____
Prof. Dr. Marcelo Götz, UFRGS
PhD, Universität Paderborn – Paderborn, Germany

Examination Committee:

Prof. Dr. Marcelo Götz, UFRGS
PhD, Universität Paderborn – Paderborn, Germany

Prof. Dr. Heraldo José Amorim, UFRGS
PhD, Federal University of Rio Grande do Sul – Porto Alegre, Brazil

Prof. Dr. Renato Ventura Bayan Henriques, UFRGS
PhD, Federal University of Minas Gerais – Belo Horizonte, Brazil

_____
Marcelo Götz
Course Coordinator
Control and Automation Engineering

Porto Alegre, maio 2021.

ABSTRACT

Machine learning for computer vision tasks is an area which has grown significantly in recent years. At the same time, there has been a marked increase in demand for better, more environmentally friendly farming methods. The current study uses machine learning applied to computer vision to develop a system that is able to identify and classify plants as crops and weeds with a view to enabling robotic appliances to perform plant cultivation and monitoring. This is performed using a state-of-the-art object detection architecture known as Faster R-CNN, trained and tested on a publicly available crop/weed dataset. Various configurations of this network were compared and tested on a Raspberry Pi 4 in order to gauge the usefulness of this detection system in field conditions, in terms of both speed and accuracy. The results show that relatively high speed predictions are possible while using low resolution images, but that significantly more accurate results can be obtained by using larger images, albeit with a significant speed penalty. In the near future, advances in computing capacity of small computers such as the Raspberry Pi, or the use of graphics processing units, will enable faster processing and a more useful system. Depending on the application, however, the tested hardware may already be adequate.

Keywords: Neural networks, Advanced agriculture, Computer vision, Embedded systems, Object detection.

RESUMO

Aprendizado de máquina para tarefas de visão computacional é uma área de muito crescimento nos últimos anos. Paralelamente, a demanda por técnicas de agricultura melhores e menos destrutivos tem aumentado significativamente. Este projeto utiliza técnicas de aprendizado de máquina aplicadas a visão computacional para desenvolver um sistema capaz de identificar e classificar plantas como úteis ou ervas daninhas, a fim de possibilitar a utilização de veículos robóticos para efetuar cultivação e monitoramento de plantios. Isso é realizado utilizando um sistema de detecção de objetos denominado Faster R-CNN, treinado e testado com uma base de dados de acesso livre. Várias configurações desta rede foram comparadas e testadas no microcomputador Raspberry Pi 4 a fim de avaliar seu desempenho em condições de campo, em termos de velocidade e exatidão. Resultados mostram que, utilizando imagens de baixa resolução, é possível atingir velocidades relativamente altas, mas resultados muito mais exatos podem ser obtidos aumentando a resolução das imagens, embora isso resulte numa queda de velocidade. Entretando, avanços tecnológicos de microcomputadores, num futuro próximo, ou a utilização de uma placa gráfica para realizar a predição, resultarão num sistema mais rápido e útil. Não obstante, dependendo da aplicação, as ferramentas testadas já podem ser adequadas.

Palavras-chave: Redes neurais, Agricultura avançada, Visão computacional, Sistemas embarcados, Detecção de objetos.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Growing awareness of the environmental impact of large scale agriculture has led to increasing interest in more sustainable farming methods (LOCKERETZ et al., 2007). In parallel with this there has been growing public concern about the indiscriminate use of pesticides in agriculture. This causes pollution beyond the area originally sprayed and frequently infects nearby water sources. This can cause widespread contamination, affecting non-target organisms, such as livestock and other plantations, ultimately resulting in negative effects on human health (AKTAR; SENGUPTA; CHOWDHURY, 2009).

To reconcile these concerns with the ever-growing demand for higher productivity, producers have increasingly turned to technology. One technique that has recently made significant advances in the management of these problems is the use of small robotic vehicles to perform crop monitoring and cultivation (BASSO, 2018). These devices have become more readily available and at a lower cost, and also now have built-in cameras capable of capturing high quality data for disease monitoring. This data can then be analysed by a human operator or an automated system, depending on availability. However, because of processing constraints, this frequently has to be run on a remote computer.

While the processing power of these embedded systems is limited, advances in computing capacity and lower costs mean that more powerful computers can be installed directly on the device. As a result, their efficacy is greatly increased, since they are no longer limited by communication or location constraints.

The current project studies the state-of-the-art for object detection and applies it to the problem of crop and weed identification. It then analyses the practicality of performing inference directly on an embedded device by implementing a plant identification algorithm on a Raspberry Pi 4 and analysing its performance.

In terms of structure, Chapter 2 contains an overview of the technology used, including an analysis of computer vision, convolutional neural networks, and embedded devices. The dataset is presented in Chapter 3, along with the process used for training the neural network. Chapter 4 presents various experiments performed to obtain the best configuration for detection, and finally, the fifth chapter provides a final analysis of the project.

2 LITERATURE REVIEW

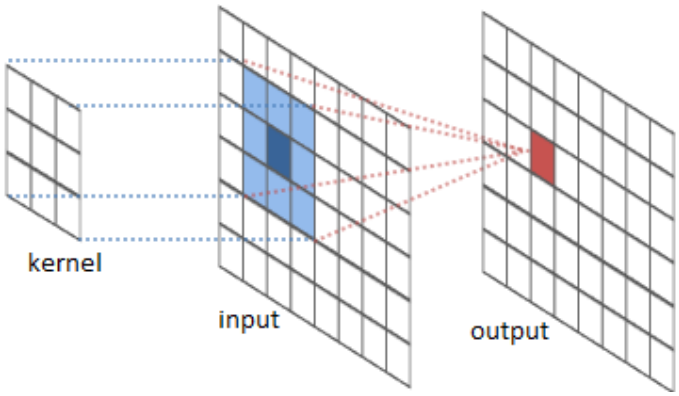
This chapter presents an overview of the current literature on the subject of machine learning and computer vision with their relevant application in embedded systems.

2.1 Computer Vision

Computer vision is the area of computing dedicated to the analysis of images using automated systems. Though fundamentally a problem of signal processing, it is a significantly more complex field, given the vast number of interconnected variables. While there is a significant relationship between the different pixels (thus making the field practical), each pixel is, essentially, an individual sensor, producing its own stream of data and resulting in multiple individual values. The objective of computer vision, and, specifically, object detection algorithms, is to produce a relationship between these individual values in order to derive useful information.

The connection between individual pixels and their meaning is known as a Feature. Traditionally, these were handcrafted shapes and groups of pixels applied to an image in order to identify the location of the greatest correlation between the input image and the feature. This process normally employs convolution, which involves the application of a small multiplication matrix (kernel) to each image pixel (and its neighbours). The sum of the values produces the pixel in the output image (Figure 1). To obtain better results, the input image can be filtered in order to enhance a specific colour or characteristic (such as edges) by performing convolution using specially designed kernels.

Figure 1: *Diagram of image convolution*



Source: Intel Labs (2016)

2.2 Machine Learning

The field of Artificial Intelligence (AI) has been the subject of much study. While AI is not new, it has recently gained prominence because of the increased availability of greater computing power. This is specifically due to the development of high performance Graphic Processing Units (GPUs, capable of performing large numbers of calculations in parallel (SHALLUE et al., 2018). One sub-area of particular interest is Machine Learning (ML), where computers are used to replicate the learning capacity of the human of the human brain. This specific area of interest relates to the brain's capacity to learn patterns and structures, whether in time series, in sound (such as speech recognition), in speech syntheses or in natural language processing. Also included are visual applications such as face recognition and object detection.

While the process of human learning is not fully understood, the method used for teaching a machine is, in theory, relatively straightforward. The whole process consists of the analysis of known data and the subsequent fitting of a multi-variable equation to this information. The exact manner in which the data is fitted is defined, based on the application and type of the data used.

2.2.1 Neural networks

Although there are numerous types of machine learning and the specifics of their training methods are outside the scope of this paper, the type chosen for this project, known as supervised learning, functions by comparing the output of the fitted equation to prior known data. The difference between the computed (predicted) result and the known data (ground truth) is then calculated and used to update the predictive function. One of the requirements, therefore, for a project involving supervised machine learning is the existence of a collection of data to be used for training. The acquisition of such datasets is frequently difficult and costly, and the output of the machine learning system is directly related to the quality of the dataset used during the training stage.

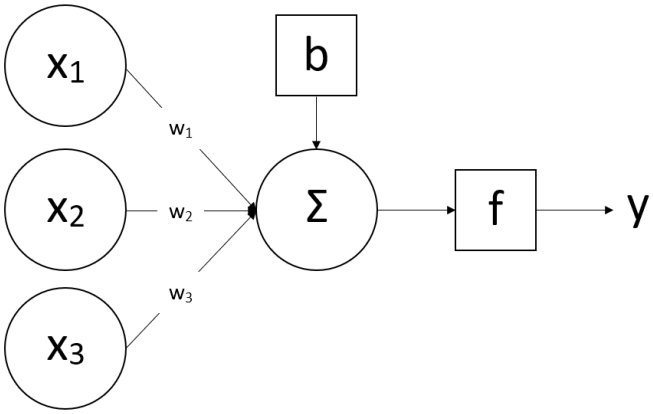
One specific particularly successful area of supervised learning in computer vision applications is the study of Neural Networks. These represent the predictive function as a network of interconnected neurons (known as perceptrons in the context of machine learning), each with its own parameters, which are set during the training process. This is understood to be similar to the functioning of the human brain and often expressed as neurons that fire together, wire together (HEBB, 1949). In short, the training phase calculates which neurons and connections are most important in representing the training data and these are strengthened to better predict new data.

In its simplest form, an individual perceptron in a neural network consists of various trainable parameters, namely, a neuron bias and multiple weights, one for every neuron in the preceding layer. Each of these are tuned throughout the training process. The outputs of the previous layer's neurons are multiplied by the weights and summed with the bias.

$$y = f\Big(\sum(x_i \cdot w_i) + b\Big),$$

where y is the output of the perceptron, x_i are its inputs, with their corresponding weights, w_i, b is the bias, and f is the activation function. An activation function is used to add a non-linear characteristic to the neuron, and is applied to its output. The diagram in Figure 2 shows the layout of a basic perceptron.

Figure 2: *Diagram of a perceptron*



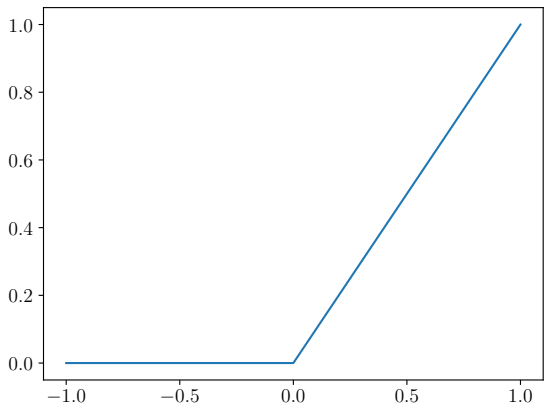
Source: Created by the author

While the exact function applied varies according to the application, the most frequently used in computer vision problems is the Rectified Linear Unit (ReLU), defined by

$$y = max(x, 0).$$

According to Ramachandran, Zoph, and Le (2017), although many other activation functions have been proposed, the effectiveness, simplicity and reliability of ReLU make it the activation function of choice for most applications. This function, displayed in Figure 3, rejects all negative inputs while maintaining positive values. Ramachandran, Zoph, and Le (2017) note that this property has the advantage of letting the gradients flow during the training phase, in contrast to other activation functions such as Tanh and the Sigmoid function.

Figure 3: *ReLU activation function*



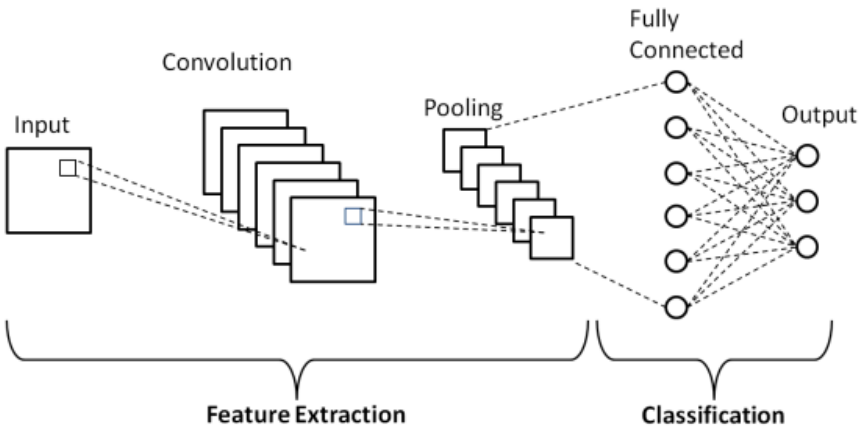
Source: Created by the author

The architecture of the whole neural network can be described as a collection of layers, each layer with connections to others. At its most simple, this can be in the form of a series where each layer is connected to all the neurons in the next, and is known as a fully-connected network. However, more complex configurations are

used, especially for complicated problems such as object detection, where layers are built in parallel to represent object location and classification (GIRSHICK, 2015).

However, in most cases involving image analysis, a fully-connected network would require an unreasonable amount of memory to run and store the model. This is because every pixel of the original image is connected to numerous perceptrons in subsequent layers. Because of this, an alternative type of neural network known as a Convolutional Neural Network (CNN), is used. Instead of learning the weights and biases for the entire size of the image, a smaller kernel is calculated and applied to the input as a convolution (shown schematically in Figure 4). These convolutions are then layered, along with a few fully-connected layers. Such an approach is similar to traditional computer vision techniques, with this difference, that instead of the features being handcrafted, they are learned by the machine learning algorithm.

Figure 4: *Schematic representation of a CNN*



Source: Phung and Rhee (2019)

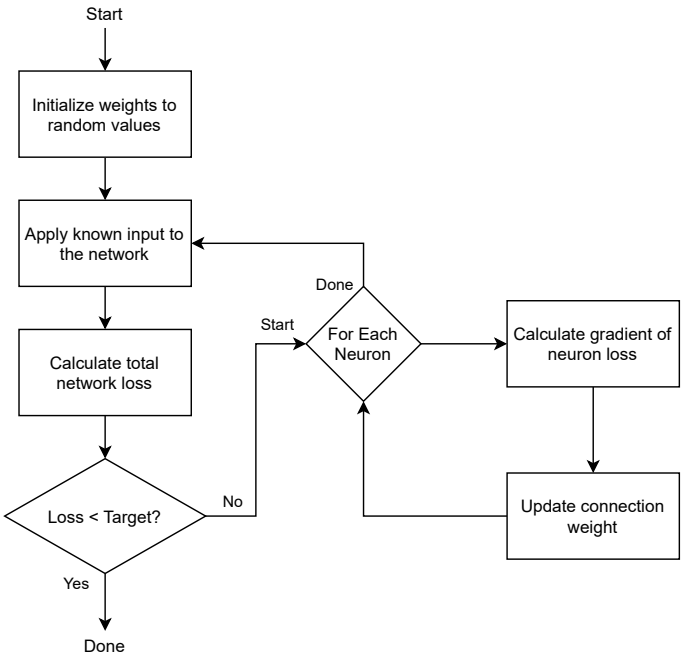
The process of fitting an equation to a specific set of data is well known and relatively simple when applied to less complex equations with few variables. In many cases, such as with Least Mean Squares, the calculation can be performed in a single step. However, for a complex non-linear equation such as that used in a neural network, this becomes impractical. In such cases, an algorithm called backpropagation (BP) is used to iteratively approach the optimal solution. This process is realized mathematically by minimizing the network's loss function.

According to (ZHAO et al., 2018), the loss function is the effective driver of network learning. It is a numerical calculation of inaccuracy in the predicted result, and can be specified in such a way as to make the system approach specific objectives, depending on the given problem. This includes, for example, prioritizing identification of specific categories in a classification problem. While there are numerous loss functions for use in different situations, L2 (the difference between the predicted value and the ground truth squared) is the most commonly used for computer vision applications.

Backpropagation can be viewed as a generalization of the Least Mean Squares algorithm (LEUNG; HAYKIN, 1991). The main difference is that backpropagation, because of its iterative nature, can be applied to a multilayer perceptron (such as a neural network). The BP algorithm, shown schematically in Figure 5, typically involves initializing the weights of a neural network to random values and calculating the output of the network on known data. Its loss is then calculated, and the

resultant value propagated backwards through the neural network, while, at the same time, modifying the weights and biases in each layer in order to approach the known ground truth training data. This is done using a technique known as Gradient Descent which moves towards an optimum solution by calculating the gradient of the loss function relative to the parameters of the neuron. These parameters are then adjusted in the opposite direction to the gradient (RUDER, 2017).

Figure 5: *Backpropagation process*



Source: Created by the author

2.3 Embedded Systems

An embedded system is a combination of hardware and software designed to perform a dedicated function (BARR, 2012). Frequently, this forms part of a larger system and is used to control it. An example of this is the cruise control system in cars.

In most situations, the processors used in embedded systems have more limited processing capability than traditional standalone computers. Since the task of image processing is computationally intensive, it has conventionally been performed on a remote computer and transmitted wirelessly to the target embedded device. This process introduces latency, and at times may be impossible due to environmental constraints such as lack of connectivity.

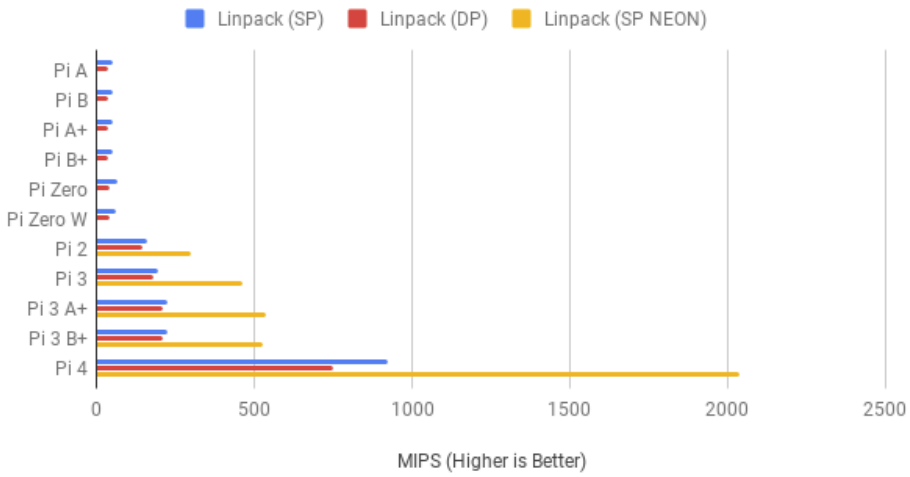
However, according to Udendhran et al. (2020), the recent availability of high performance processors is encouraging the development of vision applications in embedded systems. Examples of this can be seen in devices such as the NVIDIA Jetson series of boards — small, single-board computers with built-in GPUs, specifically developed for advanced computer vision applications. The capacity of cheaper boards, such as the Raspberry Pi, has also greatly improved in recent years.

Since these devices can theoretically be used as general purpose computers, they do not adhere to the formal definition of an embedded system, even though they

still share many of their characteristics, including low processing capacity and small form factor. In addition, these microcomputers are frequently embedded in larger mechanical systems and used exclusively for the purpose of control, thus justifying the use of the nomenclature.

A comparison of the performance of different versions of the Raspberry Pi is shown in Figure 6, the most recent version being over three times as fast as the one released three years before. These values are measured using the Linpack benchmark, a metric showing the speed at which a given system can compute the result of a matrix multiplication on floating point numbers. While these metrics do not always directly translate to better performance on a given algorithm, they are relevant here since this project deals with the same calculation as is performed by a neural network.

Figure 6: *Linpack benchmark of Raspberry Pi versions*



Source: Halfacree (2019)

3 METHODOLOGY

This chapter presents the methodology used for developing and testing the detection system. It includes the process of dataset selection, the choice of neural network architecture and its specific implementation, followed by an overview of the training and testing process.

3.1 Dataset

As noted in Section 2.2.1, one of the most important parts of any supervised machine learning project is a high quality dataset. Since these datasets are typically large and require expert knowledge of the specific application domain, manual acquisition and annotation of a suitable dataset is often complex and costly, and as such, is outside the scope of this paper.

Other methods for obtaining an acceptable dataset include generating synthetic images from varying images of three-dimensional models, as demonstrated by Rodrigues (2020). This has the advantage of performing the data acquisition and image annotation in a single step, which can be repeated and varied as necessary. However, while this method works well for consistent or well-defined objects, such as machine parts, it is difficult to use in the generation of plant imagery.

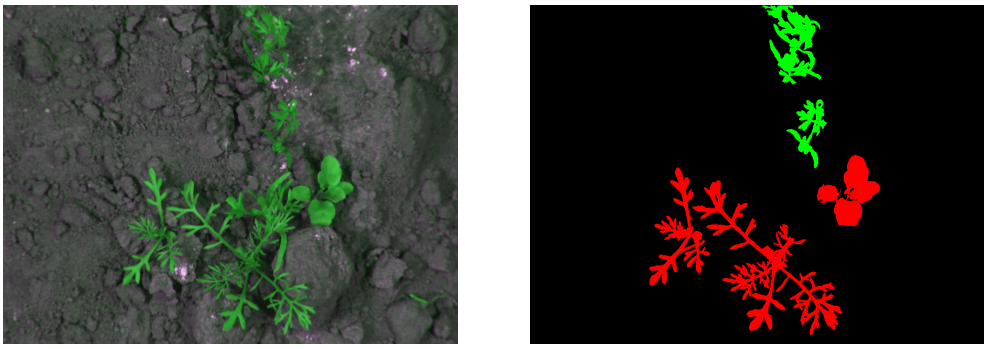
Having thus considered the various options and limiting factors, the method of choice was a publicly available dataset, created with the purpose of facilitating advanced agricultural projects. While there are various such datasets available, some with as many as 8,000 plants (SUDARS et al., 2020), the one chosen for the purpose of this project consisted of 494 high quality annotations of carrot plants (HAUG; OSTERMANN, 2015).

The decision to choose the smaller dataset was driven by the quality of the annotations. Although the quantity of images is important for the accuracy of the trained model, inconsistencies in the training data make accurate prediction impossible. In this case, the smaller dataset was deemed more appropriate for the task in hand. Figure 7 shows an example of an image and its corresponding annotation. These annotations were then converted to bounding boxes for use with object detection networks.

Osako et al. (2020), amongst others, has shown that it is possible to train a high performance deep neural network using a relatively small dataset. However, to help produce better results, a technique known as image augmentation is used to artificially create extra images. In the current project, the input dataset was randomly flipped and cropped by up to 10% in order to expand the original.

Figure 7: *Sample image from crop/weed dataset with annotation*

(a) *Image* **(b)** *Annotation*



Source: Haug and Ostermann (2015)

3.2 Neural Network

The selection of a neural network architecture for performing any AI task is a complex problem, since there are many possible configurations. Some of the parameters that must be selected include the number of neurons per layer and the number of layers. In addition, the interconnection between layers must also be defined, since at times these are not sequential. These parameters are selected based on the complexity of the problem and the computing power available for training and deployment of the model.

As discussed in Chapter 2, machine learning applied to computer vision is a complex task. While custom-built neural networks perform well for specific applications in other areas of signal processing, image recognition is dominated by a series of continuously improving "families" of standard neural networks, along with their variations.

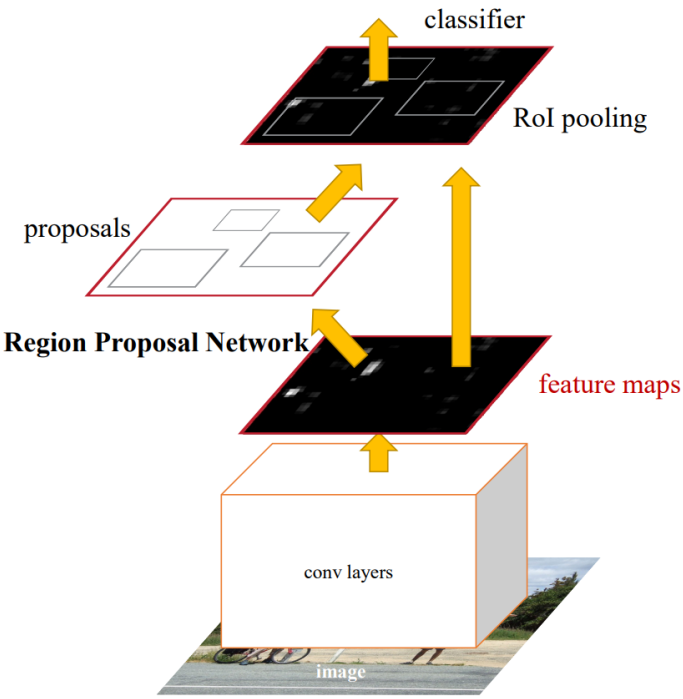
3.3 Faster R-CNN

The current state-of-the-art neural network configuration for object detection – the process of identifying and classifying multiple objects in a larger image – is the Region based Convolution Neural Network (R-CNN) (GIRSHICK et al., 2014). This network can be divided into two stages: the first uses a technique known as selective search to identify possible object locations, while the second attempts to classify the detected objects using a CNN.

In order to address specific shortcomings in the original paper, certain variations have been proposed. Girshick (2015) improved speed by applying the CNN once for the whole image, instead of every possible bounding box. Faster R-CNN (REN et al., 2016) is a further improvement and was the version chosen for this current project.

Faster R-CNN improves on the previous version by substituting the slow selective search step with a CNN based region proposal network (RPN). One of the advantages of this approach is that the two convolutional steps can be combined by performing one convolution step on the original image and connecting its output to a small number of fully connected layers. Three of these were used in the original paper's implementation, for both detection and classification of images and are shown in Figure 8.

Figure 8: *Architecture of the Faster R-CNN network*



Source: Ren et al. (2016)

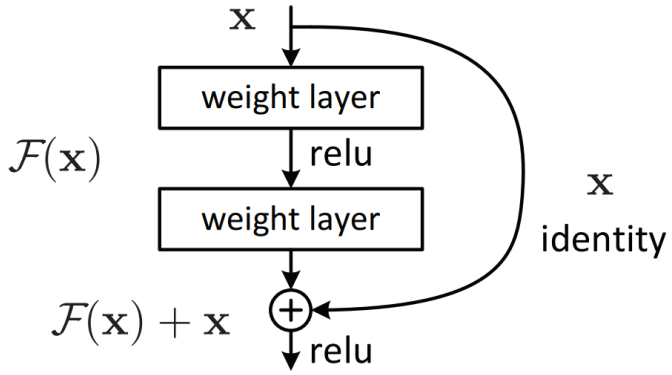
Because of the sharing of the convolutional neural network between the two steps of the architecture, the traditional backpropagation algorithm cannot be used directly. As a result, the training method proposed involves separate convolution steps for the RPN and the classification network. This is followed by retraining only the fully connected layers of the RPN in order to join the two again, allowing it to be initially trained to a near optimum state where the final step is responsible for fine-tuning the fully connected layers.

One of the more useful features of the Faster R-CNN architecture is its modular nature. The convolutional network, known as the backbone, can be replaced by many existing successful image classification networks, such as VGG (SIMONYAN; ZISSERMAN, 2015) or ResNet, short for residual network, (HE et al., 2015), amongst others. This gives significant freedom to the system designer regarding the trade-off between detection accuracy and prediction speed. Sometimes simpler classification CNNs can adequately model the training data while requiring less computational capacity to run.

In this project, variations of the ResNet neural network were tested both for classification accuracy and prediction speed. ResNet is a type of neural network that includes both a primary connection between consecutive layers and a secondary connection that skips some layers and facilitates the training of arbitrarily deep architectures. This concept is presented schematically in Figure 9. The ResNet configuration was chosen because of its learning advantages over preceding network architectures while not significantly compromising speed. Figure 10 presents the schematic representation of a 34 layer version of the network, known as ResNet34.

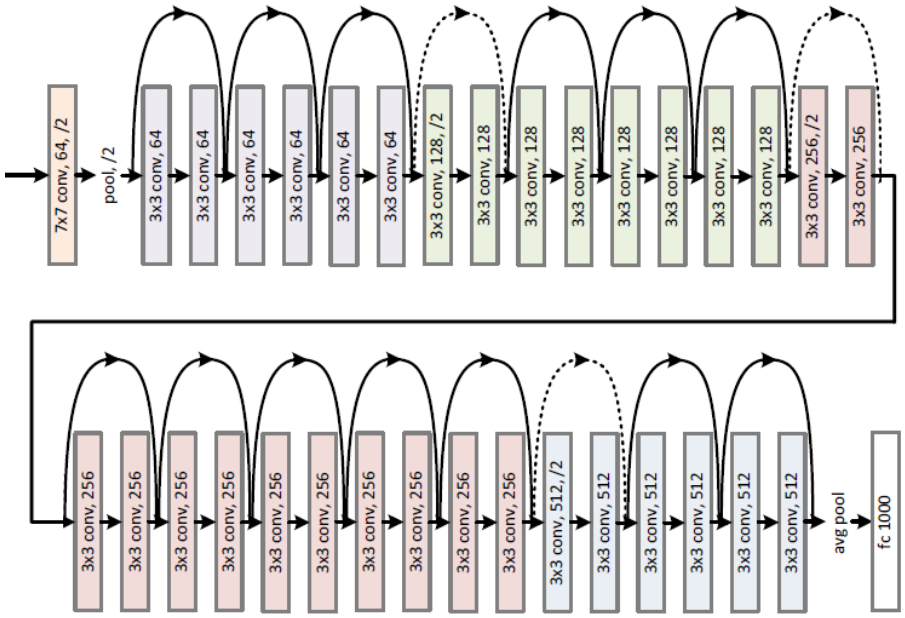
The utilization of existing standard classification networks also facilitates a method called transfer learning to speed up the training process. This technique is based on the assumption that image recognition tasks, even in different subject

Figure 9: *ResNet building block*



Source: He et al. (2015)

Figure 10: *ResNet34 architecture*



Source: He et al. (2015)

domains, rely on similar image features to perform the detection. Therefore, instead of initializing the network weights to random values, as suggested in Chapter 2, the training process can be accelerated by using values from another object classification task. Because of the recent interest in object detection using neural networks, there are many publicly available neural network models trained on large object datasets, such as ImageNet (RUSSAKOVSKY et al., 2015) – an object dataset with over 14 million images – which was used with Faster R-CNN for transfer learning in this project.

3.4 Detectron 2

The implementation of a neural network architecture has recently been facilitated by the advent of standardized easy to use libraries such as TensorFlow (AGARWAL; BARHAM, et al., 2015) and PyTorch (PASZKE et al., 2019). These packages, backed by Google and Facebook Research, respectively, consist of simple interfaces in the Python programming language with most of the network's computations being performed in a low level programming language such as C++. In addition, it is possible to perform parallel computing tasks on the computer's GPU, if available, which decreases the training time by up to a factor of ten (AWAN; SUBRAMONI; PANDA, 2017). While these libraries do not negate the need for an understanding of concepts such as backpropagation, these being necessary for problem-solving in neural networks, for example, they do allow the user to focus on the implementation of the system while guaranteeing good performance.

Because of the complexity of the training step of Faster R-CNN, as mentioned in Section 3.3, this project uses an implementation recommended by Ren et al. (2016) called Detectron 2 (WU et al., 2019). This framework, which uses the PyTorch library, contains high performance implementations of various recent papers in the field of object recognition. It also gives the user full control over neural network parameters, including facilitating transfer learning on standard datasets and, in the case of Faster R-CNN, replacement of the network backbone.

3.5 Training Process

As mentioned in Chapter 2, training a neural network is a process involving the iterative optimization of the network loss function. As such, every step of the training process consists of thousands of floating point operations to perform gradient descent. This results in a training process that can take many hours to achieve the desired training loss.

In order to obtain useful results in an acceptable timeframe, all the networks were trained using an NVIDIA Quadro M5000 GPU. The number of training iterations was empirically set to 20,000 as an acceptable level for performance comparison. With this configuration, training each variation of the network took approximately 4 hours.

With the purpose of evaluating the network's accuracy, the dataset was divided into a train and a test set, with 80% being used for training and 20% used for testing. This is required to guarantee that the network is not simply memorizing the training data.

3.6 Inference and Analysis

The inference of a neural network should produce essentially the same results regardless of the type of processor on which it is performed. As such, initial accuracy tests can be carried out on the training computer, using the GPU to accelerate the process.

However, to examine the speed performance of the neural network running in near real-world conditions, a Raspberry Pi 4 was used. As noted in Chapter 2, the Raspberry Pi is a series of small, low-cost computers. The Raspberry Pi 4 is the latest in the series and the model used has 4 GB of RAM and a quad-core 64-bit ARM processor.

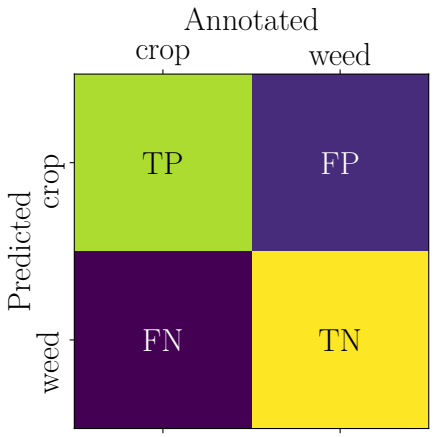
In order to perform inference on the Raspberry Pi, parts of the Detectron 2 framework must be installed on the device, along with their dependencies. However, because of the ARM architecture of the Raspberry Pi's processor, most of these components are not directly installable and must be compiled from source. These include both the Python 3.8 binary and Detectron 2, along with other auxiliary packages and libraries. The main PyTorch and TorchVision packages, however, were installed from pre-compiled experimental binaries.

4 RESULTS

In order to evaluate and compare the performance of the trained models, the method chosen must account for both stages of the object recognition process – detection and classification. While there are many possibilities, the most frequently used for evaluating the simpler classification problem utilizes two metrics, precision and recall, along with a confusion matrix.

A confusion matrix, frequently used in binary classification problems, is a table containing the intersection between annotations and their respective classifications, as shown in Figure 11. This matrix shows four important values: "true positives" (TP), the number of correctly classified positive instances; "true negatives" (TN), which corresponds to the correctly classified negative instances; "false positives" (FP), which defines how many negative instances were incorrectly classified as positive; and "false negatives" (FN), which shows how many positive samples were incorrectly classified as negative.

Figure 11: *Confusion Matrix layout*



Source: Created by the author

The precision metric is defined as the number of true positives out of the total positive predictions. Mathematically, this is defined as

$$P = TP/(TP + FP).$$

Analogously, recall is defined as the ratio of true positives to the total number of positive annotations, or mathematically

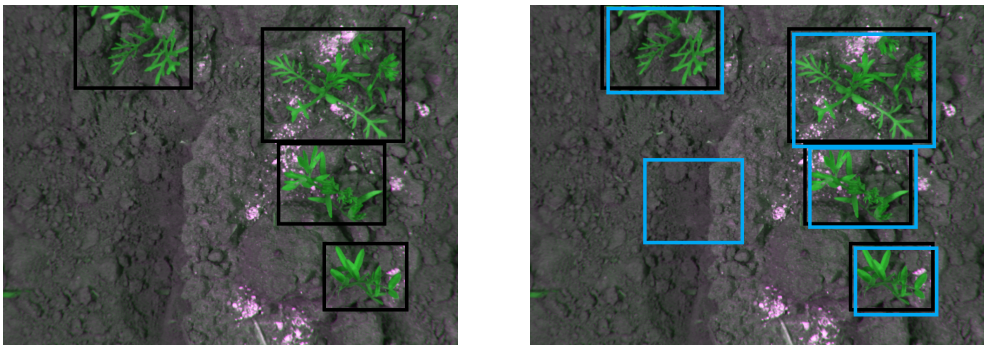
$$R = TP/(TP + FN).$$

These metrics provide a comprehensive understanding of the performance of a classification system: precision indicates how likely it is that a positive prediction is actually a positive sample and recall indicates how likely the model is to predict all positive samples.

With the aim of analysing the identification step performance of the neural network, a slight variation of these metrics was used: true positives were defined as the detected bounding boxes which significantly overlapped with the ground truth, while false positives and false negatives correspond to cases where a bounding box was detected but not annotated, or vice-versa. A visual explanation of precision and recall in these conditions is shown in figures 12 and 13, respectively. With this definition, the concept of true negatives is no longer applicable, as "undetected boxes" cannot be quantitatively defined.

Figure 12: *Example of identification precision of 80%*

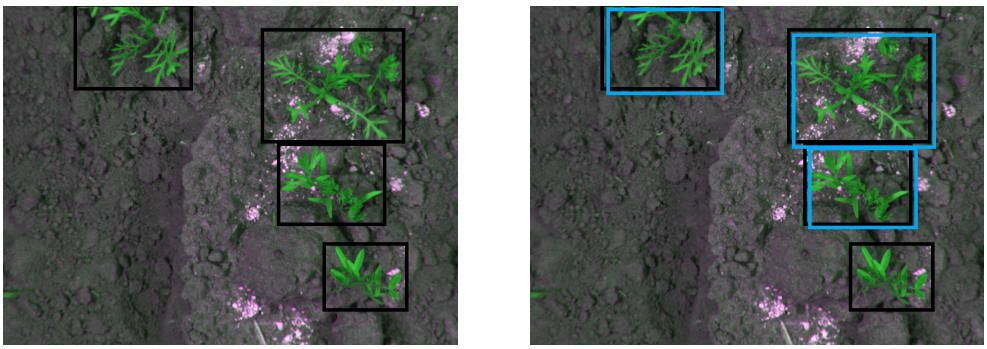
(a) *Ground truth* **(b)** *Prediction*



Source: Created by the author

Figure 13: *Example of identification recall of 75%*

(a) *Ground truth* **(b)** *Prediction*



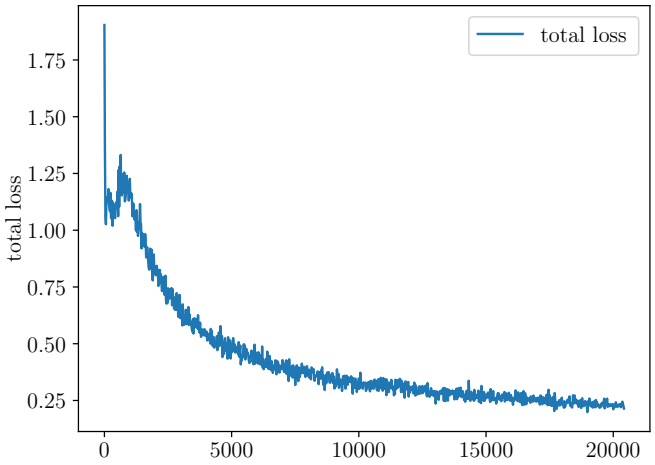
Source: Created by the author

In order to calculate whether two boxes corresponded to the same plant in the original image, a technique was developed involving the calculation of the largest intersection ratio (known as the Intersection Over Union – IOU) between predicted and ground truth boxes, discarding any with an IOU below a certain threshold. In this project, any bounding box pairs with an IOU of less than 0.4 were considered as referring to separate plants, for the purpose of calculating the accuracy metrics.

4.1 Baseline Configuration

With the objective of establishing a baseline of both prediction time and accuracy, an initial configuration was defined using ResNet50 (50 layer ResNet) as the Faster R-CNN backbone. As mentioned in Section 3.5, 20,000 iterations were used for training. Figure 14 presents the total training loss over time.

Figure 14: *Total training loss*

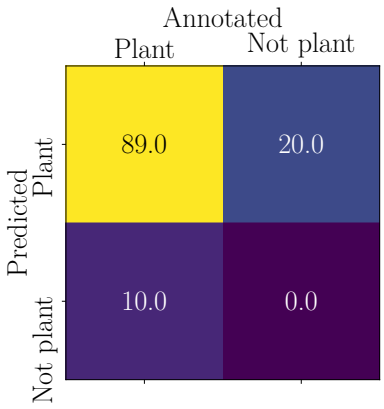


Source: Created by the author

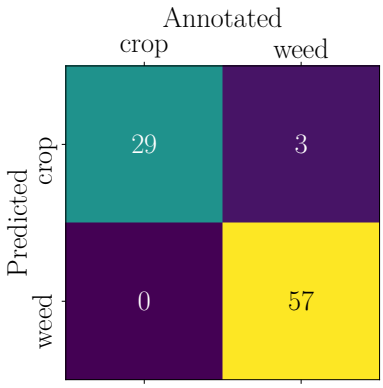
As noted in Section 3.5, the network must be tested on new data in order to meaningfully evaluate its performance. As such, Figure 15 shows the confusion matrices for both the detection and classification phases on the test dataset.

Figure 15: *Confusion matrix for detection and classification*

(a) *Detection*



(b) *Classification*



Source: Created by the author

This results in precision and recall for the identification phase of 0.82 and 0.90, respectively. The classification precision and recall on the identified plants were 0.91 and 1.00, respectively.

Although the accuracy of both identification and classification was acceptable, the prediction time of 42.7 seconds, when performed on the Raspberry Pi, was too high for this application. In order to improve this time, PyTorch was recompiled

with multithreading and Just In Time (JIT) compiler support. This second feature compiles Python code at runtime, significantly improving performance. Table 1 shows the average prediction time on three different processors using these features.

Table 1: *Prediction time on different processors*

Processor	Prediction time (s)
NVIDIA Quadro M5000 (GPU)	0.184
Intel Xeon E5-1650 (CPU)	2.989
Raspberry Pi 4	23.303

Source: Created by the author

The speed on both the GPU and the high-end CPU was acceptable. However, the Raspberry Pi's performance, although better than initial readings, still requires further refinement.

4.2 Network Depth

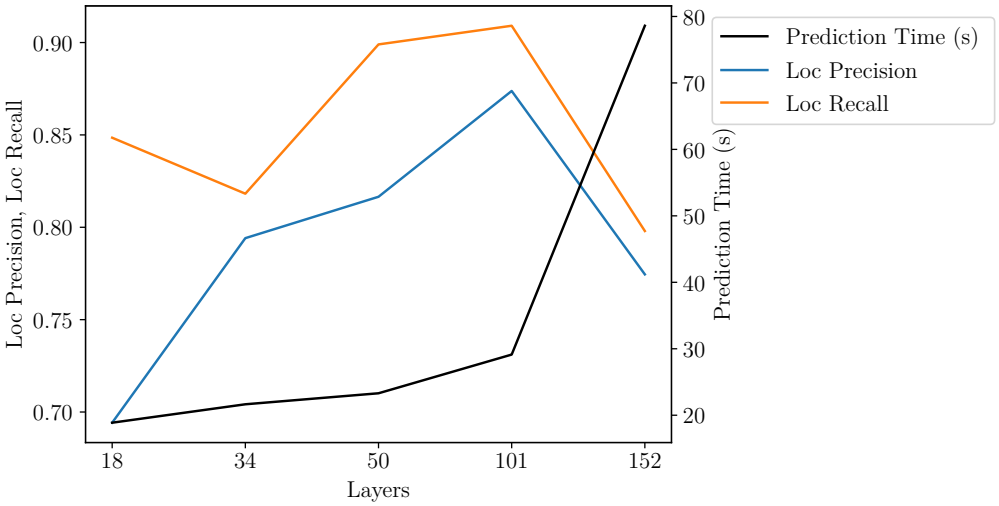
As noted in Section 3.3, one of the advantages of the Faster R-CNN architecture is the ability to replace the backbone network. This can be done in order to simplify the output model, thus diminishing its size and increasing prediction speed. This is especially useful considering the application on an embedded device, where computational power is limited.

Alternatively, more complex backbones (for example, with more layers) can increase the complexity of the data learned by the model, enabling it to correctly identify a larger variety of objects. However, in simpler problems, the added network complexity can cause an issue known as overfitting, in which the network essentially memorizes the training data, leading to low accuracy when analysing real world data.

In order to study the effect of simplifying the network, figures 16 and 17 show the prediction time along with the precision and recall for the RPN phase and the classification, respectively. In addition, the network was also tested with the much larger 101 and 152 layer variants of the ResNet architecture. These, however, would not be acceptable for running on an embedded device.

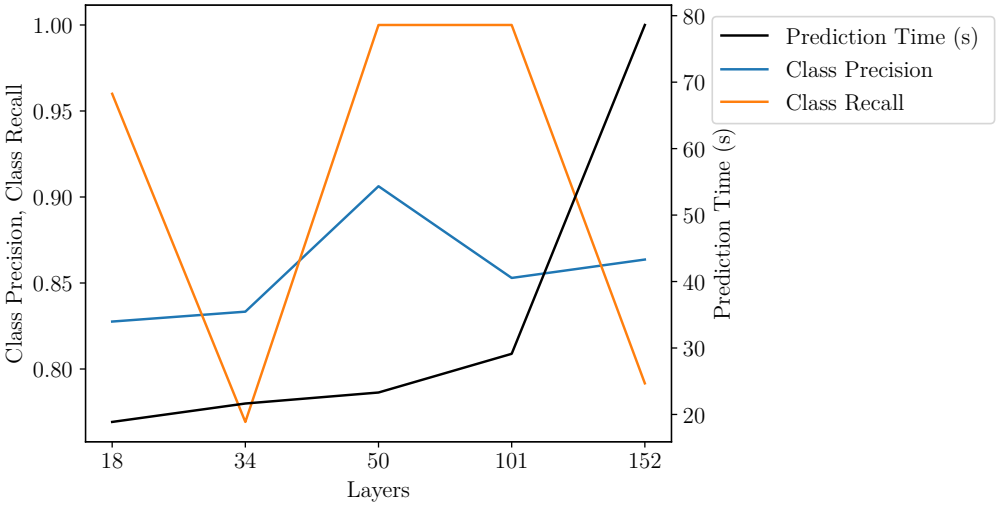
These graphs show that there is some improvement in the network speed, with prediction time falling from 23.30 s to 18.88 s. The impact on other performance metrics, especially classification precision, is more significance, with a decrease from 0.91 to 0.83. However, the effect of reducing the network layers is extremely unpredictable and as such, other approaches should be analysed before considering this possibility.

Figure 16: *Identification precision, recall and time for different network configurations*



Source: Created by the author

Figure 17: *Classification precision, recall and time for different network configurations*



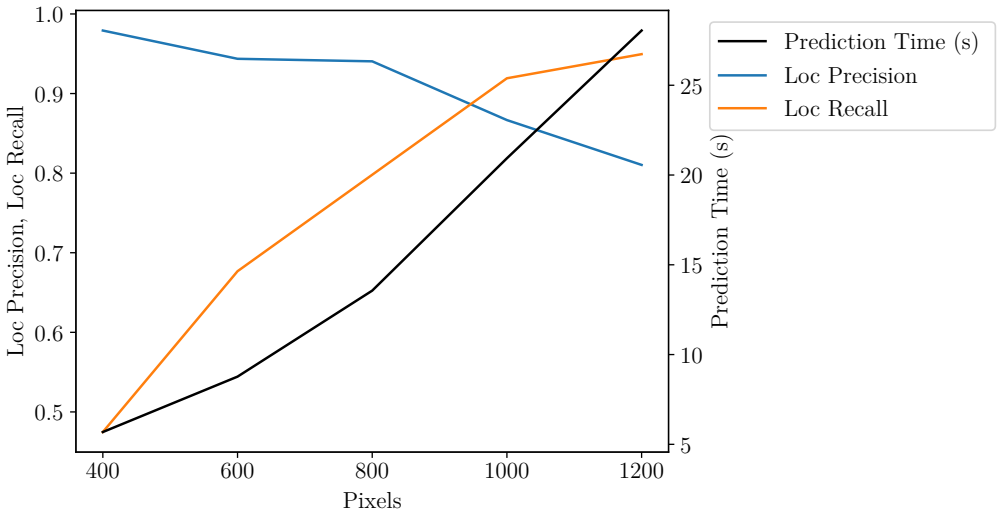
Source: Created by the author

4.3 Image Resolution

Another approach for speeding up object recognition tasks is the lowering of input image resolution. Because the convolutional step works by iterating over all pixels in the image, the relationship between the size of the image side and the prediction speed should be approximately quadratic.

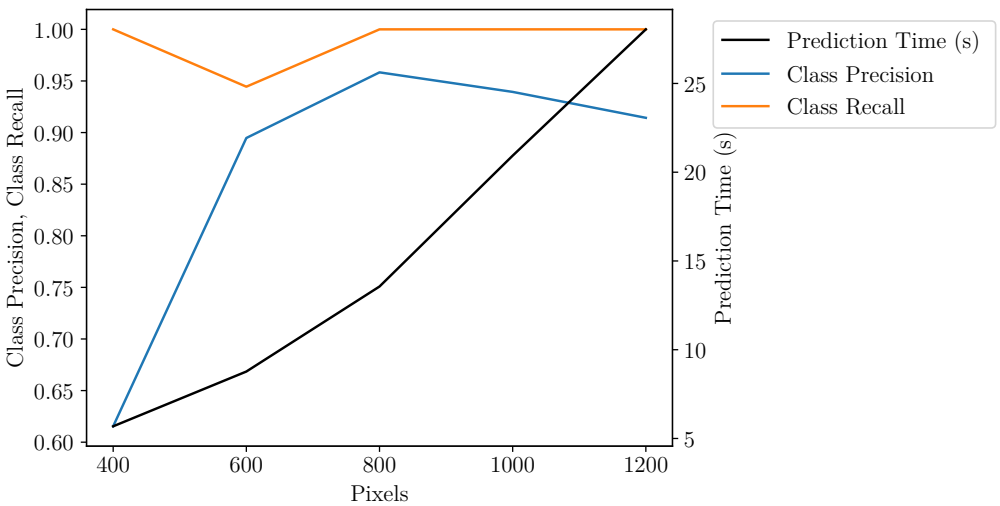
In order to test the effect of the image resolution on precision and recall, the network inference was run on image resolutions where the largest dimension varied between 400 and 1,200 pixels, using the ResNet50 architecture as the backbone. The results are displayed in figures 18 and 19.

Figure 18: *Identification precision, recall and time for different image resolutions*



Source: Created by the author

Figure 19: *Classification precision, recall and time for different image resolutions*

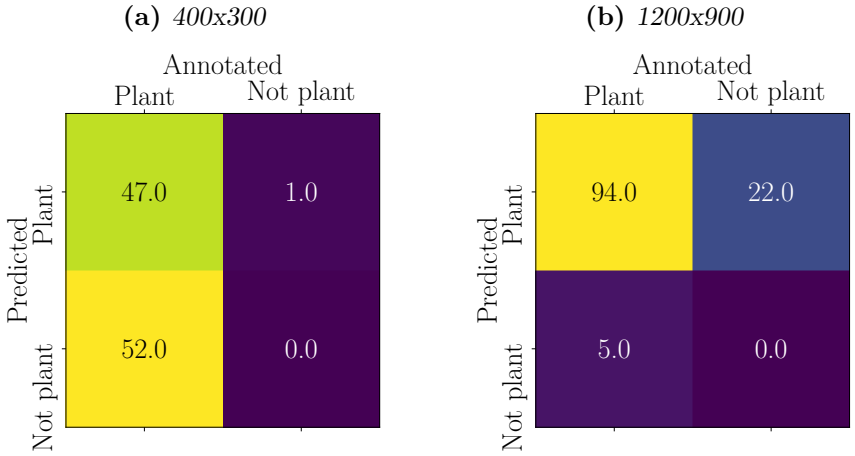


Source: Created by the author

In the identification phase, an inverse relationship is observed between the

network's precision and its recall. This characteristic is frequently repeated when adjusting object detection parameters and is explained by the confusion matrices in Figure 20.

Figure 20: *Confusion matrix for detection at 400x300 and 1200x900 pixels*



Source: Created by the author

The parameter change results in more predictions overall, some of which are correct, thereby increasing recall, while others are incorrect and result in lowered precision. Therefore, a compromise must be made when changing any parameter. In this case, since prediction time is important, a resolution of 800x600 pixels is acceptable for this application.

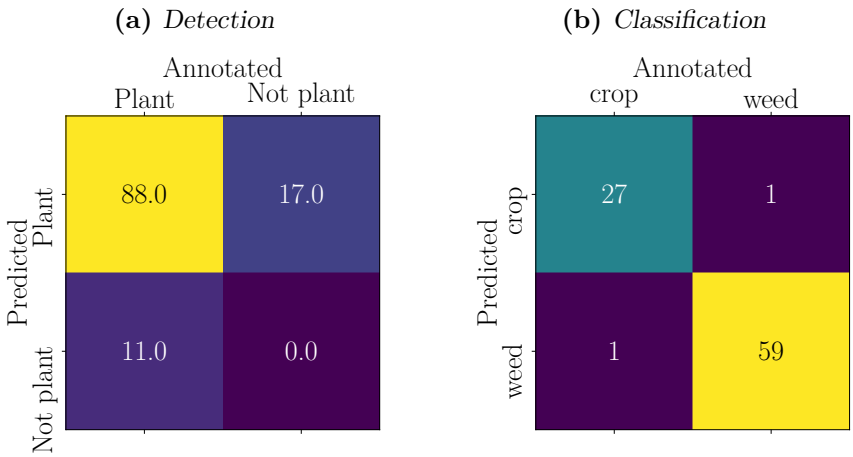
4.4 Fine Tuning

Based on the results presented in sections 4.2 and 4.3, the decision was taken to use ResNet50 as the backbone with 800x600 image resolution and adjust other parameters in order to attain better results. To this end, a system was implemented to plot precision and recall metrics while sweeping various parameters of interest. As a result, the following changes were made to the network configuration:

- The classification threshold below which region proposals are discarded was lowered from 0.8 to 0.45.
- The acceptable overlap (IOU) between bounding boxes of the same class was increased from 0.5 to 0.6.
- The permissible overlap (IOU) between bounding boxes of different classes was increased from 0.5 to 0.7.
- The model was trained for a further 10,000 iteration, to a total of 30,000.

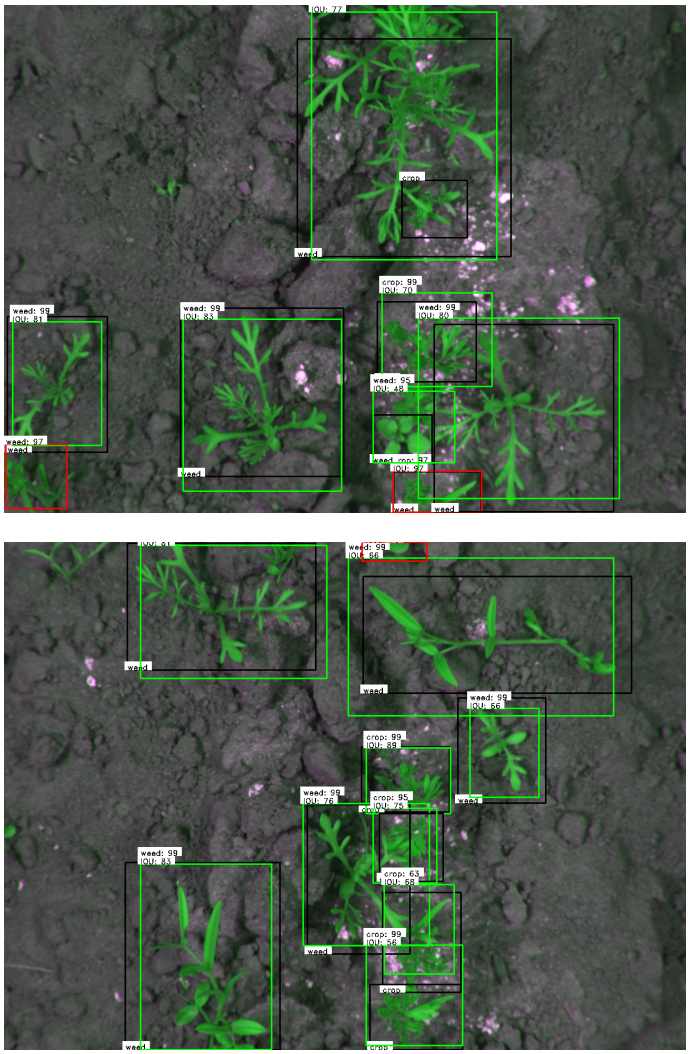
The confusion matrices resulting from changing these parameters are presented in Figure 21. The precision and recall values for the identification stage were 0.84 and 0.89, and for the classification stage were 0.96 and 0.96, respectively. The inference took an average of 12.80 seconds per image. Figure 22 presents two example outputs from the algorithm.

Figure 21: *Confusion matrix for detection and classification*



Source: Created by the author

Figure 22: *Example images of identification and classification*



Source: Created by the author

5 CONCLUSION

This project set out to create a system capable of detecting and classifying crops and weeds using AI. The state-of-the-art object detection systems were analysed and adapted to suit the needs of the project, tuning the parameters to achieve the best possible results in terms of speed and accuracy.

In addition, the inference system was implemented on a small microcomputer, in order to test its performance in near field conditions. The trained models were analysed relative to the trade-off between speed and accuracy, and adapted to improve their performance on devices with limited computing power.

The experiments performed on the Raspberry Pi 4 demonstrate that, depending on the specific application and accuracy required, various network parameters can be modified to speed up the inference, achieving a minimum of 5.69 s, in the tested configurations. They also show the effect that these changes have on detection and classification accuracy, demonstrating that significantly better results can be obtained with higher resolution images. When tested at a 800x600 pixel resolution, the system achieved 96% precision and 96% recall for classification, while still taking only 12.80 s to perform the inference.

The performance of the Faster R-CNN network, with the ResNet50 backbone, demonstrated that for certain applications, such as ground-based autonomous robots for use in agriculture, the inference speed and accuracy on the Raspberry Pi could be useful. However, for applications such as unmanned aerial vehicles, where movement speed is typically much higher and battery autonomy lower, a GPU based solution would be preferable for online identification.

Another possible avenue for further research would be the use of alternative network architectures. These include the family known as You Only Look Once (YOLO) which focus on achieving higher speed, while sacrificing inference accuracy. Such a solution may be preferable for monitoring applications involving tighter time constraints.

REFERENCES

AGARWAL, A.; BARHAM, P., et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. [S.l.: s.n.], 2015. Software available from tensorflow.org. Available from: <http://tensorflow.org>.

AKTAR, M. W.; SENGUPTA, D.; CHOWDHURY, A. Impact of pesticides use in agriculture: their benefits and hazards. *Interdisciplinary toxicology*, Slovak Toxicology Society, v. 2, n. 1, p. 1, 2009.

AWAN, A. A.; SUBRAMONI, H.; PANDA, D. K. An In-Depth Performance Characterization of CPU- and GPU-Based DNN Training on Modern Architectures. In.

BARR, M. *Embedded Systems Glossary*. [S.l.: s.n.], 2012. Available from: <https://barrgroup.com/embedded-systems/glossary>.

BASSO, M. *A framework for autonomous mission and guidance control of unmanned aerial vehicles based on computer vision techniques*. [S.l.: s.n.], 2018.

GIRSHICK, R. *Fast R-CNN*. [S.l.: s.n.], 2015. arXiv: 1504.08083 [cs.CV].

GIRSHICK, R. et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. [S.l.: s.n.], 2014. arXiv: 1311.2524 [cs.CV].

HALFACREE, G. *Benchmarking the Raspberry Pi 4*. [S.l.: s.n.], 2019. Available from: <https://medium.com/@ghalfacree/benchmarking-the-raspberry-pi-4-73e5afbcd54b>.

HAUG, S.; OSTERMANN, J. A Crop/Weed Field Image Dataset for the Evaluation of Computer Vision Based Precision Agriculture Tasks. In: COMPUTER Vision - ECCV 2014 Workshops. [S.l.: s.n.], 2015. p. 105–116. DOI: 10.1007/978-3-319-16220-1_8. Available from: <http://dx.doi.org/10.1007/978-3-319-16220-1_8>.

HE, K. et al. *Deep Residual Learning for Image Recognition*. [S.l.: s.n.], 2015. arXiv: 1512.03385 [cs.CV].

HEBB, D. O. *The organization of behavior: A neuropsychological theory*. [S.l.]: Psychology Press, 1949.

INTEL LABS. *Bringing Parallelism to the Web with River Trail*. [S.l.: s.n.], 2016. Available from: <http://intellabs.github.io/RiverTrail/tutorial/>.

LEUNG, H.; HAYKIN, S. The complex backpropagation algorithm. *IEEE Transactions on Signal Processing*, v. 39, n. 9, p. 2101–2104, 1991. DOI: 10.1109/78.134446.

LOCKERETZ, W. et al. What explains the rise of organic farming. *Organic farming: An international history*, CABI International: Oxford, UK, 2007.

OSAKO, Y. et al. Cultivar discrimination of litchi fruit images using deep learning. *Scientia Horticulturae*, v. 269, p. 109360, 2020. ISSN 0304-4238. DOI: https://doi.org/10.1016/j.scienta.2020.109360. Available from: <https://www.sciencedirect.com/science/article/pii/S0304423820301886>.

PASZKE, A. et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: WALLACH, H. et al. (Eds.). *Advances in Neural Information Processing Systems 32*. [S.l.]: Curran Associates, Inc., 2019. p. 8024–8035. Available from: <http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf>.

PHUNG; RHEE. A High-Accuracy Model Average Ensemble of Convolutional Neural Networks for Classification of Cloud Image Patches on Small Datasets. *Applied Sciences*, v. 9, p. 4500, Oct. 2019. DOI: 10.3390/app9214500.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. *Searching for Activation Functions*. [S.l.: s.n.], 2017. arXiv: 1710.05941 [cs.NE].

REN, S. et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. [S.l.: s.n.], 2016. arXiv: 1506.01497 [cs.CV].

RODRIGUES, J. V. *Classificação de peças mecânicas a partir de visão computacional e aprendizado de máquina utilizando imagens sintéticas*. [S.l.: s.n.], 2020.

RUDER, S. *An overview of gradient descent optimization algorithms*. [S.l.: s.n.], 2017. arXiv: 1609.04747 [cs.LG].

RUSSAKOVSKY, O. et al. *ImageNet Large Scale Visual Recognition Challenge*. [S.l.: s.n.], 2015. arXiv: 1409.0575 [cs.CV].

SHALLUE, C. J. et al. Measuring the effects of data parallelism on neural network training. *arXiv preprint arXiv:1811.03600*, 2018.

SIMONYAN, K.; ZISSERMAN, A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. [S.l.: s.n.], 2015. arXiv: 1409.1556 [cs.CV].

SUDARS, K. et al. Dataset of annotated food crops and weed images for robotic computer vision control. *Data in Brief*, v. 31, p. 105833, 2020. ISSN 2352-3409. DOI: https://doi.org/10.1016/j.dib.2020.105833. Available from: <https://www.sciencedirect.com/science/article/pii/S2352340920307277>.

UDENDHRAN, R. et al. Enhancing image processing architecture using deep learning for embedded vision systems. *Microprocessors and Microsystems*, v. 76, p. 103094, 2020. ISSN 0141-9331. DOI: https://doi.org/10.1016/j.micpro.2020.103094. Available from: <https://www.sciencedirect.com/science/article/pii/S0141933120301642>.

WU, Y. et al. *Detectron2*. [S.l.: s.n.], 2019. https://github.com/facebookresearch/detectron2.

ZHAO, H. et al. *Loss Functions for Neural Networks for Image Processing*. [S.l.: s.n.], 2018. arXiv: 1511.08861 [cs.CV].