Automatic insect count in trap images using deep learning

(Contagem automática de insetos em imagens de armadilhas utilizando deep learning)

Por

Ana Cláudia Carvalhais Teixeira

Orientador: António Manuel Trigueiros da Silva Cunha

Co-orientador: Raul Manuel Pereira Morais dos Santos

Co-orientador: Alessandro Matese

Dissertação submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
MESTRE
em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no
DR – 2.ª série–Nº214 – 7 de novembro de 2018 e no
Regulamento de Estudos Pós-Graduados da UTAD
DR, 2.ª série – Deliberação n.º 658/2016

Automatic insect count in trap images using deep learning

(Contagem automática de insetos em imagens de armadilhas utilizando deep learning)

Por

Ana Cláudia Carvalhais Teixeira

Orientador: António Manuel Trigueiros da Silva Cunha

Co-orientador: Raul Manuel Pereira Morais dos Santos

Co-orientador: Alessandro Matese

Dissertação submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
MESTRE
em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no
DR – 2.ª série–Nº214 – 7 de novembro de 2018 e no
Regulamento de Estudos Pós-Graduados da UTAD
DR, 2.ª série – Deliberação n.º 658/2016

Orientação Científica :

António Manuel Trigueiros da Silva Cunha

Professor Auxiliar do
Departamento de Engenharias
Escola de Ciências e Tecnologia
da Universidade de Trás-os-Montes e Alto Douro

Raul Manuel Pereira Morais dos Santos

Professor Associado com Agregação do
Departamento de Engenharias
Escola de Ciências e Tecnologia
da Universidade de Trás-os-Montes e Alto Douro

Alessandro Matese

Senior researcher do
National Research Council (CNR-ITALY)
in Florence in the Institute of BioEconomy (IBE)

"In the middle of difficulty lies opportunity" | "No meio da dificuldade encontra-se a oportunidade."

Einstein (1879 – 1955)

-

UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

Mestrado em Engenharia Electrotécnica e de Computadores

Os membros do Júri recomendam à Universidade de Trás-os-Montes e Alto Douro a aceitação da dissertação intitulada "**Automatic insect count in trap images using deep learning (Contagem automática de insetos em imagens de armadilhas utilizando deep learning)** " realizada por **Ana Cláudia Carvalhais Teixeira** para satisfação parcial dos requisitos do grau de **Mestre**.

agosto 2022

Presidente: **Doutor Pedro Alexandre Mogadouro do Couto**,
Professor Auxiliar da Escola de Ciências e Tecnologia da
Universidade de Trás-os-Montes e Alto Douro

Vogais do Júri: **Doutor João Dallyson Sousa de Almeida**,
Professor Adjunto da Escola do Departamento de Informática da
Universidade Federal do Maranhão

António Manuel Trigueiros da Silva Cunha,
Professor Auxiliar do Departamento de Engenharias da Escola de
Ciências e Tecnologia
da Universidade de Trás-os-Montes e Alto Douro

Contagem automática de insetos em imagens de armadilhas utilizando deep learning

Ana Cláudia Carvalhais Teixeira

Submetido na Universidade de Trás-os-Montes e Alto Douro
para o preenchimento dos requisitos parciais para obtenção do grau de
Mestre em Engenharia Electrotécnica e de Computadores

Resumo — As pragas de insetos são a principal causa de perda de produtividade e qualidade nas culturas em todo o mundo. O percevejo e a traça da uva são exemplos de duas das pragas mais significativas que afetam o arroz e as vinhas, respetivamente. As armadilhas para insetos estão entre as soluções mais adequadas para monitoramento e contagem influenciando a seleção e dosagem do pesticida a ser aplicado no controlo de pragas. No entanto, a monitorização e contagem baseiam-se na visita frequente de técnicos ao local e são suportadas por métodos de contagem ineficientes, sendo uma tarefa exigente e demorada. A gestão integrada de pragas foi desenvolvida para melhorar a gestão de pragas de insetos, reduzir o uso excessivo de pesticidas e aumentar a qualidade e o rendimento das culturas. Com o aprimoramento das tecnologias de inteligência artificial, diversas aplicações surgiram no contexto agrícola, incluindo detecção automática, monitorização e contagem de insetos.

Este estudo propõe a contagem automática de insetos em armadilhas usando algoritmos de deep learning. Foram utilizados três bases de dados diferentes, *Pest24*, *Bedbug* e *Grape moth*. *Pest24* é um conjunto de dados público com uma grande diversidade de insetos. Os conjuntos de dados *Bedbug* e *Grape moth* são conjuntos de dados privados fornecidos pelo mySense, uma plataforma de agricultura de precisão desenvolvida e gerida por investigadores da UTAD.

Nossa metodologia foi dividida em duas partes, o uso de detetores de um estágio (YOLOv5) e de dois estágios (Faster R-CNN). Para a YOLOv5, realizamos um total de 8 experimentos, nos quais foram analisados o impacto do transfer learning e o hyperparameter tuning. Para a *Pest24*, obtivemos um desempenho superior ao

relatado no estado da arte (72,1% de mAP), com o método YOLOv5. Em seguida, usando os hiperparâmetros e weights obtidos do conjunto de dados *Pest24*, treinamos os conjuntos de dados *Bedbug* e *Grape moth*. Os melhores resultados para o dataset do *Bedbug* foram obtidos com o YOLOv5 com tranfer learning com um AP de 95,9%. O melhor resultado foi obtido com YOLOv5 usando o ajuste de hiperparâmetros da *Pest24* com AP de 91,5% para o *Grape moth*. Para o detector de dois estágios, realizamos 8 experimentos, nos quais foram combinados o uso de CNN (VGG ou Inception ResNet V2) e a aplicação de otimização de âncora, ajustando a âncora ao tamanho dos insetos. A otimização da âncora mostrou-se eficaz no conjunto de dados *Grape moth*, atingindo um AP de 80%, mas para o conjunto de dados da *Grape moth*, o método que teve o melhor desempenho foi sem a otimização âncora com um AP de 78.3%.

Como resultado, YOLOv5 foi a melhor arquitetura, apresentando apenas dificuldades de detecção em imagens com elevado número de insetos. Além disso, foi possível verificar que a escala relativa dos insetos é o fator que mais afeta a tarefa de detecção e que a otimização da âncora pode melhorar os resultados de detecção.

Palavras Chave: Deep learning, contagem de insetos, deteção de insetos, monitorização inteligente de pragas

Automatic insect count in trap images using deep learning

Ana Cláudia Carvalhais Teixeira

Submitted to the University of Trás-os-Montes and Alto Douro
in partial fulfillment of the requirements for the degree of
Master in Electrical Engineering and Computers

Abstract — Insect pests are the leading cause of yield loss and quality in crops worldwide. The bedbug and the grape moth are examples of two the most significant pests affecting rice and vineyards. Insect traps are among the most appropriate solution for monitoring and counting influencing the selection and dosage of the pesticide to be applied for pest control. However, the counting and monitoring operations are based on the frequent visit of technicians to the site and are supported by inefficient counting methods, which is a challenging and time-consuming task. Integrated pest management was developed to improve insect pest management, reduce the overuse of pesticides, and increase the quality and yield of crops. With the improvements of artificial intelligence technologies, several applications have emerged in the agricultural context, including automatic detection, monitoring and counting of insects.

This study proposes the automatic counting of insects in traps using deep learning algorithms. Three different databases, *Pest24*, *Bedbug* and *Grape moth* were used. *Pest24* is a public dataset with a great diversity of insects. The *Bedbug* and the *Grape moth* datasets are private datasets provided by mySense, a precision agriculture platform developed and managed by researchers from the UTAD.

Our methodology was divided into two parts, the use of one-stage (YOLOv5) and two-stage (Faster R-CNN) detectors. For the one-stage detector, we carried out a total of 8 experiments, in which the impact of transfer learning and hyperparameter tuning were analyzed. First, for *Pest24* dataset we obtained a performance superior to that reported in state of the art, with the YOLOv5 method with standard hyperparameters, with an mAP of 72.1%. Then, using the hyperparameters and weights obtained from the *Pest24* dataset, we trained the *Bedbug* and *Grape moth* datasets. The best results for the *Bedbug* dataset were obtained with the YOLOv5

with transfer learning with an AP of 95.9%. The best result was obtained with YOLOv5 using the hyperparameters tuning of *Pest24* with an AP of 91.5% for the *Grape moth*. For the two-stage detector, we performed 8 experiments, in which the use of CNN (VGG and Inception ResNet V2) and the application of anchor optimization or not (adjusting to the size of the insects) were combined. The anchor optimization proved to be effective in the *Bedbug* dataset, reaching an AP of 80%, but for the *Grape moth* dataset, the method that performed the best was without the anchor optimization with an AP of 78.3%.

As a result, YOLOv5 was the best architecture, presenting only detection difficulties in images with a high number of insects. Furthermore, it was possible to verify that the relative scale of the insects is the factor that most affects the detection task and that the anchor optimization can improve the detection results.

Key Words: deep learning, insect counting, insect detection, smart pest monitoring

Agradecimentos

Agradeço à UTAD, que me acolheu durante estes anos e que foi determinante para a minha formação académica.
À minha família por sempre acreditarem em mim, por toda a sua dedicação, esforço, companhia e amor dado ao longo de toda a vida.
Ao meu namorado pela força transmitida em ultrapassar os momentos de maiores dificuldades, por todo o seu apoio, carinho e compreensão.
Aos meus orientadores, Professor António, Professor Raul e Professor Alessandro, pela confiança, colaboração, apoio ao longo do trabalho e pela disponibilidade de orientação desta dissertação.
Ao Professor Joaquim, o meu orientador da bolsa de investigação.
A todos os que, direta ou indiretamente, contribuíram para este meu percurso.
A todos eles, o meu muito obrigado!

UTAD, Ana Cláudia Carvalhais Teixeira
Vila Real, 17 de junho de 2022

Table of contents

List of figures

Abbreviation	**Expansion**
AI	*Artificial Intelligence*
AP	*Average precision*
CNN	*Convolutional neural network*
DL	*Deep learning*
FPN	*Feature pyramid networks*
GIS	*Geographic information systems*
GPS	*Global positioning systems*
IoT	*Internet of things*
IPM	*Integrated pest management*
mAP	*Mean average precision*
NMS	*non-maximum suppression*
PAN	*Path aggregation network*
R-CNN	*Region-based Convolutional Neural Networks*
RPN	*Region proposal network*
SGD	*Stochastic gradient descent*
SPM	*Smart pest monitoring*
SSD	*Single Shot Multibox Detector*
SVM	*Support vector machine*
YOLO	*You Only Look Once*
ZF	*Zeiler and Fergus*

1 Introduction

Insect pests are one of the biggest problems in agriculture, causing between 20 and 40 percent losses in global agricultural production each year (13). The prevention of pests can avoid the loss of a large part of the harvest and, consequently, avoid high economic losses, with this objective, farmers apply pesticides on crops. However, continued use of this product causes harmful effects on human health, the environment, and natural resources (14). Thus, to avoid the undesirable consumption of pesticides and detect pests early, it is necessary to apply advanced technical solutions.

Precision agriculture consists of a set of advanced tools and technologies that make it possible to evaluate or monitor the conditions of a particular agricultural plantation. The main objectives of precision agriculture are the increase in farmers' income, associated with the reduction of production costs, the increase in productivity, and the reduction of the environmental impact of agricultural activity (15).

Pest detection is a useful precision farming tool (16). Insect monitoring is therefore necessary for the early detection of pests and thus to prevent the excessive use of pesticides. One form of monitoring is counting the insects attracted to traps. Your count is essential to determine the need to apply pesticides or not. Traditionally, insect counting is done visually. However, as each trap can contain dozens of insects

of different species, the counting task becomes very laborious, susceptible to errors, and time-consuming (17).

1.1 Motivation

Since insects are very small and similarities between different insects are reduced, counting becomes a difficult task. With the rapid development of image processing and artificial intelligence (AI) techniques, the possibility of automating the counting of insects in traps emerged. Thus, counting automatically would be a plus, as it would make the counting procedure faster, safer, more effective and is essential for a smaller and more assertive use of phytopharmaceuticals.

In the literature, some works use this approach for the task of identifying and accounting for insects. Nieuwenhuizen et al. (18) presented a methodology to detect and count the whitefly in sticky traps. They used the Faster R-CNN with Inception Resnet v2 and obtained 87.40% mean average precision (mAP). Nieuwenhuizen's method has two limitations, the fact that it only works on images with a yellow background and the images have to be obtained under controlled conditions. Hong et al. (19) proposed several algorithms that detect and count M. thunbergianae from pheromone trap images. The authors trained a Faster R-CNN with Resnet101, EfficientDet D4, Retinanet50, and SSD Mobilenetv2 architectures. The Faster R-CNN achieved the best result, with an AP of 85.63%. Yun et al. (20) applied different methodologies for black pine detection from adhesive trap images. The methodology that had the best performance was YOLOv5, with an AP of 94.7%. Tang et al. (21) proposed a modified YOLOv4 to detect insect pests in a dataset with 28,000 images. The method developed by them obtained an mAP of 71.6%.

Thus, the development of a model for the detection and automatic counting of insects in traps using images acquired in a natural environment will represent an advance in the state of the art, considering the data provided by the mySense platform.

mySense is an innovative platform consisting of a set of devices that collect numerical

data or images to offer a set of solutions for specific problems in precision agriculture. This platform has an image collection system of insect traps in several geographically dispersed observatories and different cultures (22).

1.2 Objectives

This study aims to develop a system capable of performing the automatic counting of insects in trap images using deep learning (DL).

The specific objectives are enumerated below:

1. Research state-of-the-art methods for detection insects in trap images using DL.

2. Identify public database with images of insect traps that allow better testing and evaluation of the methods developed.

3. Organize and annotation a database with the images provided by the mySense platform.

4. Implement state-of-the-art DL methods and developed methods for automatic detection and counting the insects present in them.

1.3 Contributions

This dissertation presents the following contributions:

- A review of the state of the art of small object detection methods and more focused on insect detection;

- Construction of two databases (*Bedbug* and *Grape moth*);

- A comparative approach between the one-stage detector and two-stage detector method for the detection of insects in traps;
- Analysis of transfer learning and hyperparameter tuning application;
- Adaptation of the Faster R-CNN detector for objects of reduced dimensions;

The code is available in a git repository[1].

1.4 Publications and awards

This dissertation involved the following publication during its period:

1. A. C. Teixeira, J. Ribeiro, A. Neto, R. Morais, J. J. Sousa and A. Cunha, "Using deep learning for detection and classification of insects on traps", 2022 IEEE International Geoscience and Remote Sensing Symposium IGARSS, 2022.
2. A. C. Teixeira, R. Morais, J. J. Sousa and A. Cunha, "A deep learning approach for automatic counting of bedbugs and grape moth", International Conference on ENTERprise Information Systems, CENTERIS, 2022.
3. A. C. Teixeira, R. Morais, J. J. Sousa and A. Cunha," Using deep learning for automatic detection of insects in traps", International Conference on ENTERprise Information Systems, CENTERIS, 2022.

During this dissertation we participate and awarded 3rd place in "Concurso de ideias INOVA@UTAD 2022" with "TRAPInspector – Sistema inteligente de deteção e contagem automática de insetos em armadilhas". TRAPInspector is a device based on a single board computer (Raspberry Pi/Orange Pi) connected to an instrumented trap and where an DL algorithm responsible for the detection and counting of insects

[1] $https://github.com/anaclaudia13ct/insect_detection.git$

is executed locally, thus making this a standalone solution that does not depend on any external service. This count is finally sent to the farmer via SMS/email. Authors: Raul Manuel Pereira Morais dos Santos, Jorge Miguel Ferreira da Silva Mendes, Ana Cláudia Carvalhais Teixeira, António Manuel Trigueiros da Silva Cunha, Emanuel Soares Peres Correia. UTAD, May 30, 2022.

1.5 Document Structure

This document is divided into 7 chapters. In each chapter there is a introductory note with the purpose of the chapter, specifying its content. The Chapter 2 contextualizes the theme at the agricultural sector. The detailed study of concepts related to deep learning fundamentals, inclued the neural networks, convolutional neural networks, transfer learning and fine tuning, and lastly, the architecture of object detection methods in the Chapter 3. In the Chapter 4 a literature review about small object detection and insect detection with DL methods is presented. The materials and methods are presented in the Chapter 5, the results and discussion in the Chapter 6. And, the conclusions and future work in the Chapter 7.

2 Contextualisation

To better understand the context of this work, we carried out a preliminary study aimed at the agricultural sector. In section 2.1, we present the evolution of agriculture up to the present day. Section 2.2 describes the various types of traps and their benefits. And section 2.2 covers an overview of precision agriculture and the mySense platform.

2.1 Evolution of agriculture

Agriculture is the oldest economic activity. It is essential for the satisfaction of countless human needs, for example, the production of food, clothes, and energy. For thousands of years, agriculture evolved and developed very slowly. Good production performance depends on natural factors, such as weather conditions, soil quality, relief, and humidity. These factors determined the quality and quantity of production. With the creation and development of new technologies and tools, it was possible to bypass and, in some cases, eliminate natural obstacles and thus achieve the productivity and income desired by farmers (23).

Intensive agriculture is a type of agricultural system that has high productivity.

The main features of this mode of production are crop rotation, use of fertilisers and pesticides, selection and genetic modification of seeds and species, mechanised production, and skilled labour (24).

Agricultural activity is associated with a set of variables in the ecosystem, including pests and diseases and the presence of other plant species that compete for food, water, and light. These agents are responsible for reducing the productive and economic yield of crops. Therefore, it is necessary to protect crops from unwanted agents, so pesticides become the most appropriate solution (25). Unfortunately, this has created the misperception that pesticides and herbicides are safe and have little impact on the environment (14). Thus, agricultural practices have been dependent on these components for several years.

Due to the chemical properties of pesticides and their continued use over decades, there has been an increase in the number of resistant pests, the poisoning of useful living beings, air pollution, water pollution, soil mobility, poisoning, and other health problems (25). In other words, the use of pesticides has harmful consequences for human health, the environment, and natural resources (26).

In this context, insect monitoring becomes necessary for the early detection of pests and thus avoiding the excessive use of pesticides which can lead to savings in the order of several thousand euros (27). Integrated pest management (IPM) systems have begun to be developed in recent years by the research community, monitoring pests and applying specific pesticides when needed (28) (29). These systems can provide farmers with a decision-making tool (30). One form of monitoring approach is the detection and counting of insects that are attracted to traps. Typically this task is made by specialists (31) (27). However, each trap can contain dozens of insects of different species. Therefore, the counting task becomes very laborious, susceptible to errors, time-consuming, subjective, and expensive (32).

Given the rapid advancement of technologies in the field of AI, the internet of things (IoT), smart pest monitoring (SPM) has emerged, allowing automatic data acquisition, remote transmission, data processing, and decision making (28) (33).

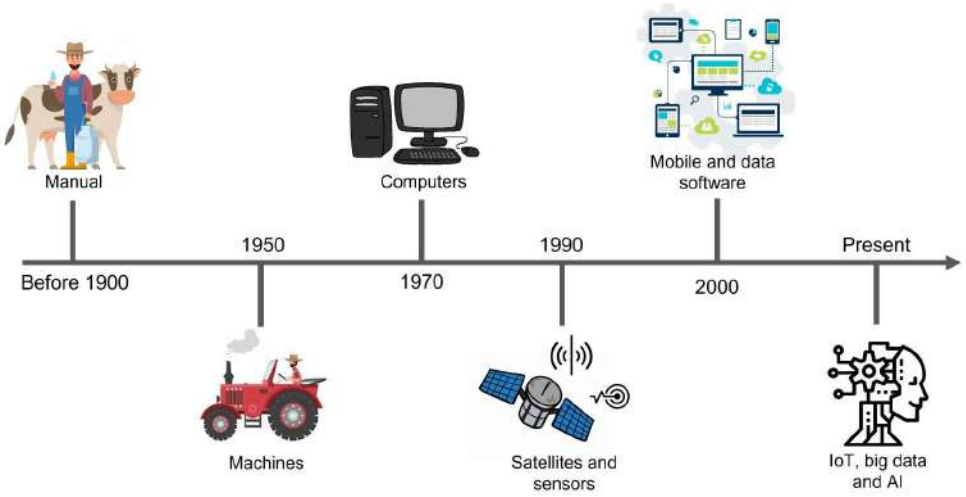

Figure 2.1 – Timeline of the evolution of agricultural activity and technologies. Based on (1).

Figure 2.1 is a chronological representation of the evolution of agricultural activity and technologies.

2.2 Insects traps

Pest control seeks to follow a diversified pest reduction strategy, combined with other forms of control and the use of chemical components. One way to apply this methodology is through physical barriers, such as traps (34). Insect traps are essential elements of IPM. These can be sex pheromone traps, sticky traps and light traps (35) (36) (37). The type of trap is chosen according to the kind of plantation or the pest to be monitored (38). Traps are frequently observed by qualified personnel to determine the number of insects that have been trapped in each trap. There is a need to travel regularly to each location to carry out this task, making this work

expensive (31).

Traps can control large areas and not interfere with crop quality as chemical compounds do. The main advantages of traps are the practical and reliable response for pest monitoring, identification of the right time to intervene with pesticides, they allow the identification and quantification of pests, and the reduction of costs and harmful effects on human beings, the environment, and resources natural (34).

Pheromone traps use a pheromone to attract pests. Pheromones are chemical substances produced by insects and released to the outside. These chemical signals are intended to communicate and trigger behaviours between the same species. The most common pheromone is the sexual attraction pheromone (36). Figure 2.2 shows an example of a pheromone trap. These are characterised by low cost, selective pest capture, detection of the pest species, and identification of the need or not to use pesticides on-site (36).

As shown in Figure 2.2 , light traps are intended to lure and capture insects attracted to light. This acts on insects that present phototropism, that is, insects with nocturnal activity and consequently are attracted to the trap when natural lighting is reduced. This trap aims to capture those insects and will cause their death. The attraction of female insects may represent the elimination of hundreds of insects in the future, as there is an interruption in the life cycle (35).

Sticky traps are the most common, composed of sticky cards with resin or wax, making the insects stick to their surface. Figure 2.2 shows an example of adhesive traps. This trap makes it possible to monitor and identify the periods of greatest infestation and thus eliminate them (37). The use of adhesive traps makes it possible to reduce pests by being trapped on its surface, and their monitoring allows the identification of the use or not of pesticides. The greater the distribution of these traps among the crops, the lesser the need to use pesticides to combat pests; however, they can also trap insects that are important for the cultivation of crops; that is, it is not a selective trap (37).

Figure 2.2 – Example of pheromones traps, light traps and sticky traps. Images provided by mySense (2)

2.3 Precision Agriculture

Precision agriculture is associated with the use of sophisticated technologies to assess and monitor the conditions of the agricultural field and thus apply the best solution for the benefit of the farmer and the environment. Precision agriculture has the spatial and temporal identification of the farm field. Its main objective is to improve and reduce production costs, increase crop productivity and quality, and decrease environmental impact (15).

To practice precision agriculture, several recent technologies are required, such as global positioning systems (GPS), geographic information systems (GIS), or electronic sensors (15). However, precision agriculture still needs personnel, farmers, or technicians, with sufficient knowledge to work and use these technologies.

The application of fertilisers, seeds, pesticides, and irrigation water is the most

common application of precision agriculture, as they represent some economic weight in the production cost. The application of different pesticides, in doses and at a specific time, may represent an economic yield for the farmer and reduce the contamination of the environment and crops. Thus, in the context of pest monitoring, the automatic counting of insects in traps represents economic and sustainable benefits.

An example of a Portuguese platform for precision agriculture is mySense. mySense is an innovative platform for the IoT oriented to agricultural applications developed at the University of Trás-os-Montes and Alto Douro. This platform consists of devices that collect numerical data or images to provide solutions for specific problems in precision agriculture. In addition, these devices perform periodic measurements of various parameters (22).

3 Deep learning fundamentals

DL is a branch of machine learning that provides solutions for computational vision tasks suited for image classification, segmentation, detection, and other tasks related to image recognition (39). DL methods are an extension of neural networks that process data and thus imitate the behaviour of the human brain when processing this data (3).

In this chapter, we made a detailed study of concepts related to the bibliography. In section 3.1, we cover an overview of neural networks. In 3.2, we describe the convolutional neural networks. Section 3.3 approach the transfer learning and fine tuning. And, section 3.4 presents in detail the object detection task and several methodologies such as Region-based Convolutional Neural Networks (R-CNN), Faster R-CNN, You Only Look Once (YOLO), and Single Shot Multibox Detector (SSD).

3.1 Neural networks

Neural networks are made up of dense layers. These layers are the grouping of several perceptrons. The perceptron receives several inputs, processes them, passes

through an activation function, and returns (40). In Figure 3.1, a perceptron is illustrated, where the input signals are represented by the vector x=[x_1, x_2], it can correspond, for example, to the pixels of an image. Input signals are multiplied by the synaptic weights that are the elements of the vector w=[w_1, w_2, w_3, w_4, w_5], generating the z value.

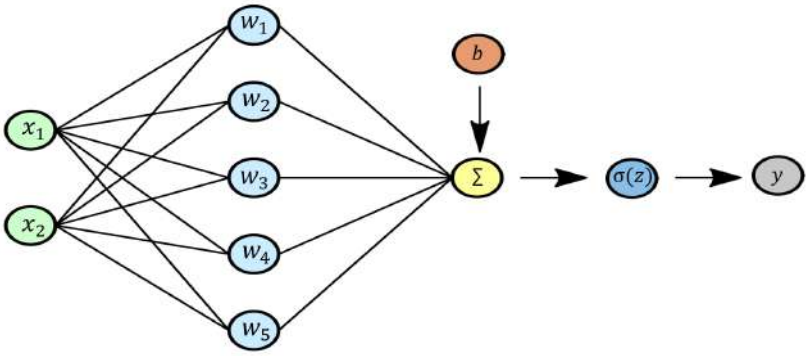

Figure 3.1 – Single layer neural network.

$$z = \sum_{i=1}^{n} x_i w_i + b \tag{3.1}$$

The bias, b, has the effect of increasing or decreasing the activation function net inflow, depending on whether it is positive or negative, respectively. The z value goes through the activation function to limit the amplitude of a neuron's output. The activation function limits the allowable amplitude range of the output signal y to some finite value (3).

Thus, the neural network is an approximation function in which a network learns the parameters (weights) in hidden layers that, when multiplied by the input, provide the predictions, which will be close to the true target. The true targets will be compared to the predictions to determine the loss. An optimiser is applied to the result of the loss score, which will influence the neural network's weights, as seen in Figure 3.2. This figure shows all the elements involved in training a neural network. Initially, the weights of the network are random, so the predictions are not ideas and the loss score is very high. During training, the weights are being adjusted and the loss score is decreasing (3). This process is repeated hundreds or thousands of times,

the objective is to reduce the loss and optimise parameters, making the predictions closer and closer to the true target (40).

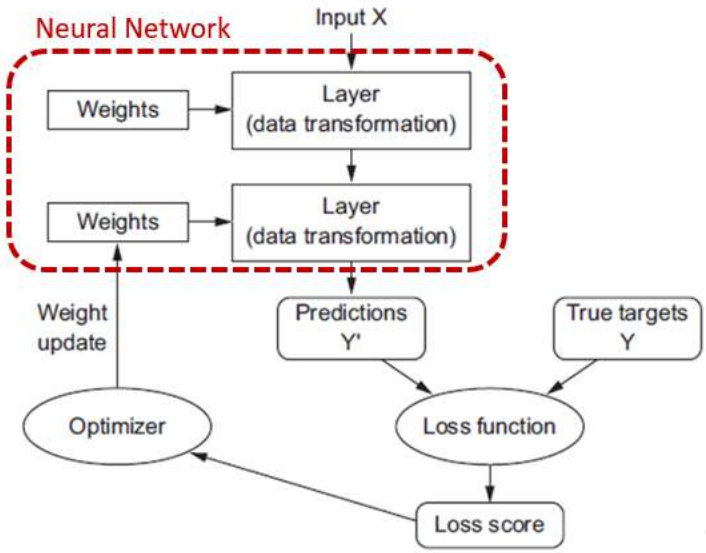

Figure 3.2 – Relationship between the network, layers, loss function, and optimiser. Adapted from (3)

The networks use several parameters and hyperparameters. Parameters are learned or estimated by the model during training, like weights and bias (3). Hyperparameters are variables that are defined in advance before training begins. Some hyperparameters are the learning rate, number of hidden layers, activation function, number of epochs, optimiser, batch size, pooling size and filter size in convolution layers (39). The hyperparameter tuning compares and adjusts current hyperparameters of the model with those obtained previously. Hyperparameter tuning aims to maximise the performance of the model (41).

3.1.1 Activation function

The activation function of a node defines the finite output of that node, making the linear combination of the inputs with the layer weights, limiting the output amplitude range. The most used activation functions are as follows:

Sigmoid function

The sigmoid function is the activation function that produces positive values in the interval $[0, 1]$. Its graph is in the form of *"S"*. The sigmoid function is differentiable; that is, we can find the slope of the sigmoid curve at any two points. The most significant advantage is that the value of the derivative is maximum when x tends to 0, pushing the result to the end of the interval $[0, 1]$, this feature is useful for binary classification problems (42).

$$\sigma(Z) = \frac{1}{(1 + e^{(-Z)})} \tag{3.2}$$

$$\sigma^{'}(Z) = \sigma(Z)(1 - \sigma(Z)) \tag{3.3}$$

Tanh function

The tanh function is a nonlinear function with interval $[-1, 1]$. This function is also sigmoidal in *"S"* shape. This function is centred on zero, so its derivative converges to zero more quickly, being this feature the main advantage. The tanh function is often used to classify two classes (43).

$$\sigma(Z) = \tanh(Z) = \frac{e^{Z} - e^{-Z}}{e^{Z} + e^{-Z}} \tag{3.4}$$

$$\sigma^{'}(Z) = 1 - tanh(Z)^{2} \tag{3.5}$$

ReLu function

The ReLu activation function is used in almost all convolutional neural networks or deep learning. This function produces results in the range $[0,\infty]$. Returns zero for all negative values and the value itself for positive values (42).

$$\sigma(Z) = \begin{cases} z & z > 0 \\ 0 & z <= 0 \end{cases} \tag{3.6}$$

$$\sigma'(Z) = \begin{cases} 1 & z > 0 \\ 0 & z < 0 \end{cases} \tag{3.7}$$

Leaky ReLu function

The Leaky ReLu function modifies the ReLu function; it can be used when the network has difficulty converging. Instead of setting the negative values to zero, this activation function applies a division factor, making them minimal and close to zero (42).

$$\sigma(Z) = \begin{cases} z & z > 0 \\ \alpha z & z <= 0 \end{cases} \tag{3.8}$$

$$\sigma'(Z) = \begin{cases} 1 & z > 0 \\ \alpha & z < 0 \end{cases} \tag{3.9}$$

Softmax function

The softmax function generalises the sigmoid function for non-binary classification cases. This is applied to the output layer, where the values are in the range $[0, 1]$. Their sum is equal to 1, whereas, in a problem with four classes, the softmax function produces four values, and their sum is equal to 1. So, each value represents the probability of each class.

$$\sigma(Z)_i = \frac{e^{Z_i}}{\sum_{j=1}^{k} e^{Z_j}} \tag{3.10}$$

3.1.2 Loss function

The loss function measures the compatibility of predictions and true targets (39). It is critical to use the appropriate loss function as they can interfere with the effectiveness of the neural network (44). Depending on the type of application, the loss functions can be divided in three types: classification loss, regression loss and unsupervised learning (39). Object detection includes two tasks: classification loss and regression loss (45). Regarding the classification, the loss function can be for binary or multi-class classification. For binary classification, the most used loss is the binary cross-entropy, which is defined by the Equation 3.11, where p is the predict probability and y is the label which can be 0 or 1. For multi-class classification

the most used loss is the categorical cross-entropy defined in the Equation 3.12, where t_i and s_i are the true target and the score for each class i, respectively. Regarding the regression, the commum loss used is the mean squared error, defined in Equation 3.13, which determines squared differences between the true and predicted values (3).

Problem	Loss function	Equation
Binary classification	Binary cross-entropy	$CE(p,y) = \begin{cases} -\log(p) & if\ y = 1 \\ -\log(1-p) & otherwise \end{cases}$ (3.11)
Multi-class classification	Categorical cross-entropy	$CE = -\sum_{i}^{C} t_i log(s_i)$ (3.12)
Regression	Mean squared error	$MSE = \frac{1}{C}\sum_{i}^{C}(y - p_i)^2$ (3.13)

Table 3.1 – Loss functions and their equations used depending on the problem

3.1.3 Optimisers

Optimisers emerged to improve the performance of neural networks; for this, the result must be continuously measured, comparing the obtained result with the expected result. Therefore, the proper choice of the optimiser is essential (46). There are several types, below are some of the most used:

Adam

Adam (47) is an optimisation algorithm that calculates the learning rate for each parameter. Adam behaves like a heavy ball that will suffer friction when going

down a hill and prefers flats of lower altitude. This algorithm stores the decreasing exponential mean of the previous square gradients v_t. And keeps the decreasing exponential mean of the previous gradients m_t. These expressions can be calculated as follows:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.14}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{3.15}$$

Where, β_1 and β_2 are the decay rates, these parameters have very small values, close to 1. The update rule for this optimisation algorithm is:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{v_t}} + \epsilon} \widehat{m_t} \tag{3.16}$$

The advantages of this method are fast convergence, easy implementation, and the ability to rectify the learning rate. The downside is the high computational cost (46).

RMSprop

RMSProp is an adaptive learning rate optimiser that handles updating the rate of parameters in a decreasing and automatic way; it combines the use of the gradient signal with the adaptation of the step size t, for each weight (46). The running average $E[g^2]_t$, depends on the previous average and the current gradient:

$$E[g^2]_t = 0.9 E[g^2]_{t-1} + 0.1 g_t^2 \tag{3.17}$$

The update rule can be defined by:

$$\theta_{t+1} = \theta_t - \frac{\nu}{\sqrt{[g^2]_t + \epsilon}} g_t \tag{3.18}$$

Updating the parameters assigns a different learning rate for each parameter, making it reach convergence faster (48).

Stochastic gradient descent

Stochastic gradient descent (SGD) is a variant of the gradient descent algorithm. The gradient descent is an iterative algorithm that starts at a random point and runs along its slope until it finds the lowest point. The SGD runs faster because

it updates the parameters for each training example $x^{(i)}$ and label $y^{(i)}$however, it causes more significant fluctuations in the objective function (49). Therefore, the equation can define the update rule for this algorithm:

$$\theta = \theta - \eta \nabla_\theta J\left(\theta; x^{((i))}; y^{((i))}\right) \tag{3.19}$$

3.2 Convolutional neural networks

Convolutional neural networks (CNNs) are neural networks that follow a feed-forward pattern, where all layers connect, following the path from the input to the output of the network. CNNs are inspired by biological processes, more specifically by the organisation of the animal visual cortex (42). This type of neural network is often applied in image recognition and video processing, thus, becoming the "state of the art" in object classification and detection problems. The disadvantage of CNNs is the need for many labelled data for feature extraction (50). The typical CNN architecture is constituted of 3 layers: (1) convolution layer, (2) pooling layer, and (3) classification layer.

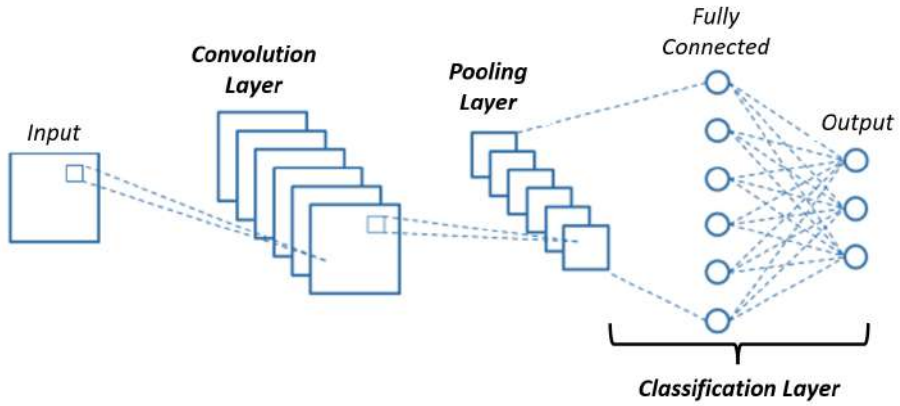

Figure 3.3 – Schematic diagram of a basic convolutional neural network architecture. Adapted from (4)

The convolution layer consists of reduced-size convolutional filters that extract input features like corners and edges. Convolutional filters, also called kernels, run through the input data along the width, height, and dimension, and perform

the mathematical operation of convolution. Convolution operation consist of a linear operation between two arrays, an input layer and a kernel, as shown in Figure 3.4 (50). During the network training, the kernels are adjusted to trigger when the input contains a particular characteristic that is common to the information. An activation function is generally applied to the convolution result, used in CNN to detect and learn nonlinear features (51).

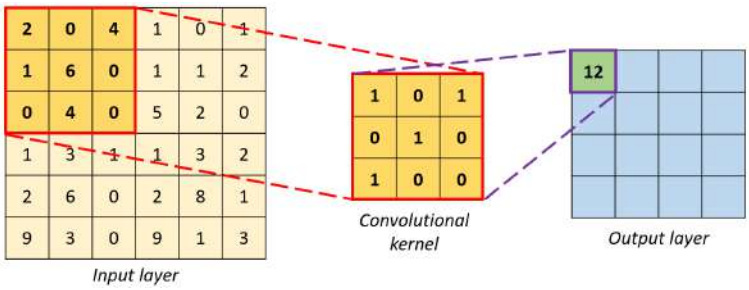

Figure 3.4 – Representation of the operation in the convolutional layer.

The pooling layer reduces the input data size and thus lowers the computational cost. The pooling layer is usually used after the convolutional layer, where the convolution layers will be given another way to represent the data and thus avoid overfitting (39). The most used techniques are max-pooling and average pooling. The max-pooling extracts the maximum value from the sub-region, as shown in Figure 3.5; and the average pooling returns the average of subregion values.

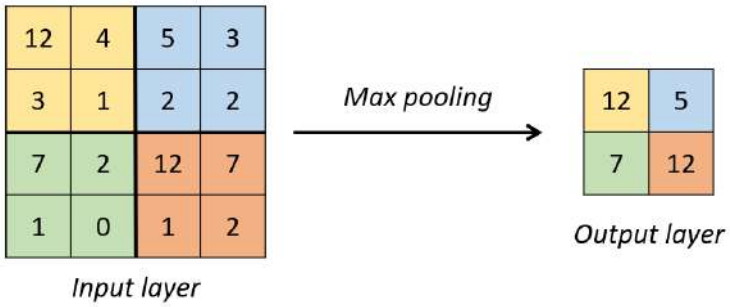

Figure 3.5 – Representation of the max pooling operation.

Classification layers are located at the end of CNNs. The inputs to this layer are the features extracted by the convolution and pooling layers. This layer produces the result of the problem and is constituted by flatten layer and a fully connected layer.

The flatten layer converts the data to a 1-dimensional array to be incremented in the fully connected layer's next layer. The fully connected layer is fully wired, which means that each neuron is wired together in the next layer. The size of the output layer is equal to the number of classes in the problem; that is, a network with two classes has two output neurons. After applying the activation and optimiser functions, each output neuron has the probability associated with it (51).

There are some CNN architectures available that are widely used. They are AlexNet (52), VGG (53), ZF (54), GoogLeNet (55) and ResNet (56).

3.3 Transfer learning and fine tunning

The success of DL depends in part on the amount of data. Sometimes the available data are scarce, and private or the costs associated with the acquisition or annotation are very high. In these situations, it is common to use transfer learning (44). Transfer learning consists in using the knowledge learned for a task in each domain, to improve the learning of another domain in another task (57), i.e., a network is pre-trained on a large dataset, such as ImageNet (58) or MS COCO (59), and then applied to the dataset that we intend to train (44). If the source dataset is large and complete, the learned features can be useful for the problem we want to solve (3). The Figure 3.6 shows a representation of the learning process of traditional DL and the learning process of transfer learning.

There are two ways to use a pre-trained network: (1) fixed feature extraction and (2) fine-tuning (3). The fixed feature extraction consists of removing the fully connected layers, that is the convolutional layers are frozen, of the pre-trained network, and add a new classifier. Considering the extracted resources, the classifier is trained from scratch (5). Fine-tuning consists of replacing and training the classifier that was added to the pre-trained network; and in tuning part of the pre-trained network kernels through backpropagation (57). Normally, the initial layers do not change, as they contain more generic resources, while the later layers become more specific

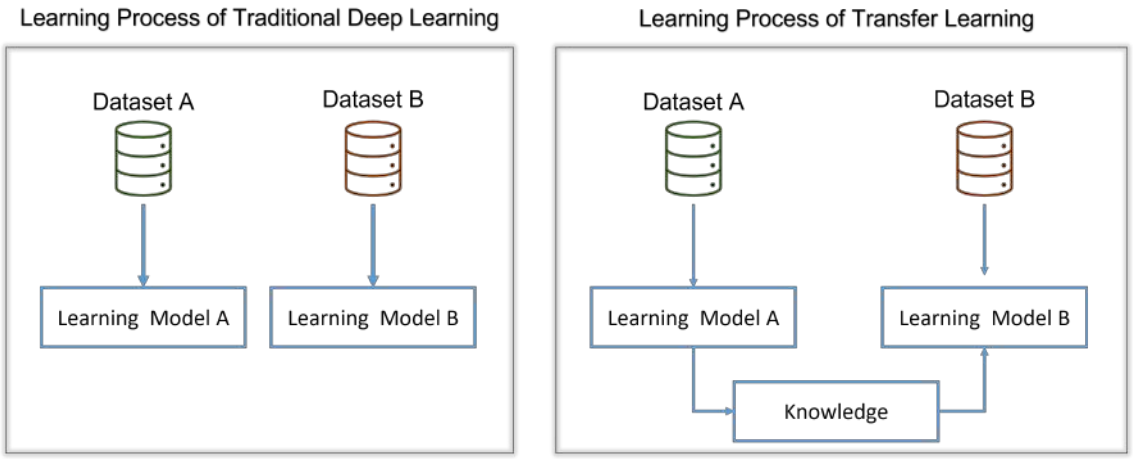

Figure 3.6 – Different learning processes between traditional DL and transfer learning. Adapted from (5)

to our dataset, so they are adjusted by backpropagation (44).

3.4 Object detection methods

The object detection task can be associated with two important concepts: (1) object classification; and (2) detection, as seen in Figure 3.7. Classification is the assignment of a class to the principal object in the image. Object detection consists of the object localisation and classification of multiple objects in an image (60). This technique uses rectangular bounding boxes to locate and classify the categories of the objects (61). Object detection is an essential area of computer vision. It plays a crucial role in many applications, such as video, medical image, vehicle, pedestrian, and face detection.

There are two significant groups of detectors: one-stage detectors and two-stage detectors. One-stage detectors solve the detection task by directly predicting object categories and regression object locations (60), such as YOLO (8) and SSD (62). This method does not require the region proposal process, so the detection is faster; however, the precision is generally less than the two-stage object detector

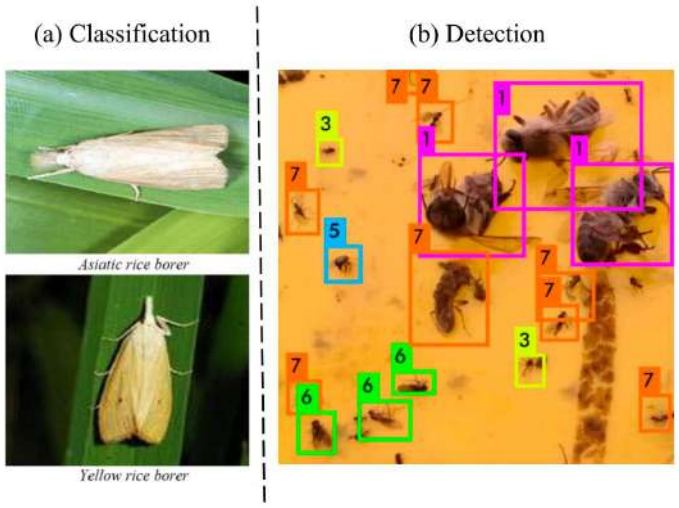

Figure 3.7 – Classification and detection of insects. (1) classification of insects on plants images (6); (2) detection of insects in images in traps (7)

architecture. The two-stage detectors initially extract the regions of interest from the input image and then classify and redefine the location of the object through the first proposed regions; examples are R-CNN (63), Fast R-CNN (64), Faster R-CNN (65), Mask R-CNN (66) and R-FCN (67). The most significant advantage is the high precision, and the disadvantage is the high detection time (61).

3.4.1 R-CNN

Girshick et al. (63) proposed R-CNN. Their model is divided into three steps: (1) region proposal, (2) feature extraction, and (3) classification of regions. The region proposal is to propose a region containing the object or part of it. A selective search is performed to obtain approximately 2000 regions designed for various bounding boxes of different sizes and dimensions.

In step 2, independent of the candidate regions size or aspect ratio, we distort all the pixels in a bounding box to the default size of 227x227 (63). Each proposed region is trained by a CNN network that extracts the characteristics. The architectures used were AlexNet and VGG.

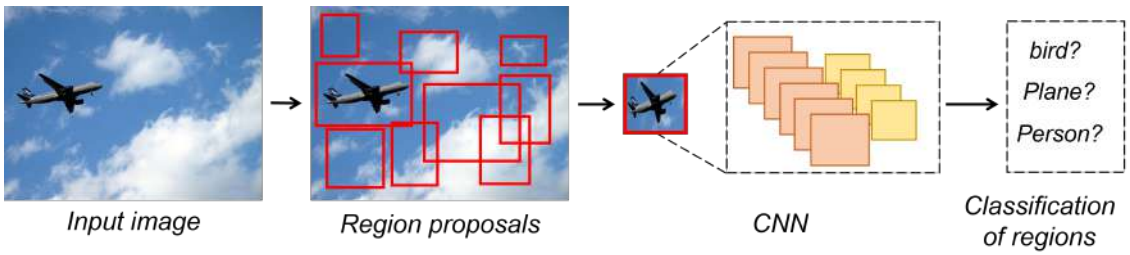

Figure 3.8 – The R-CNN model

In the final step, there is a need to classify the objects within each region, for which the support vector machine (SVM) (68) classifier is applied. SVM is a learning algorithm often used in machine learning capable of doing classification and regression. Simply put, SMV is an algorithm that seeks to find a hyperplane that best divides a dataset into different classes.

After the application of the SVM classifier, the IoU metric is used, which allows calculating the score of the proposed regions (69); the proposed redundant regions, which have an IoU less than a threshold, are discarded by the non-maximal suppression (NMS) algorithm (70). R-CNN effectively detects objects. However, training takes a long time, so it is not implemented in real-time. These disadvantages are due to the need to classify 2,000 proposed regions per image. Another downside is the need for the proposed regions to be distorted to a predefined size.

3.4.2 Faster R-CNN

Ren et al. proposed the Faster R-CNN architecture (65). As already mentioned, the R-CNN generates several region proposals, generating around 2,000 regions, which makes it a slow procedure. To reduce training time without losing accuracy, the Faster R-CNN proposes the region proposal network (RPN) (Ren et al., 2015). Faster R-CNN is composed of two main networks, the RPN that generates the region proposals and a network that uses these regions to detect objects. This model is divided into four steps, CNN, region proposal network, ROI pooling, and

classification.

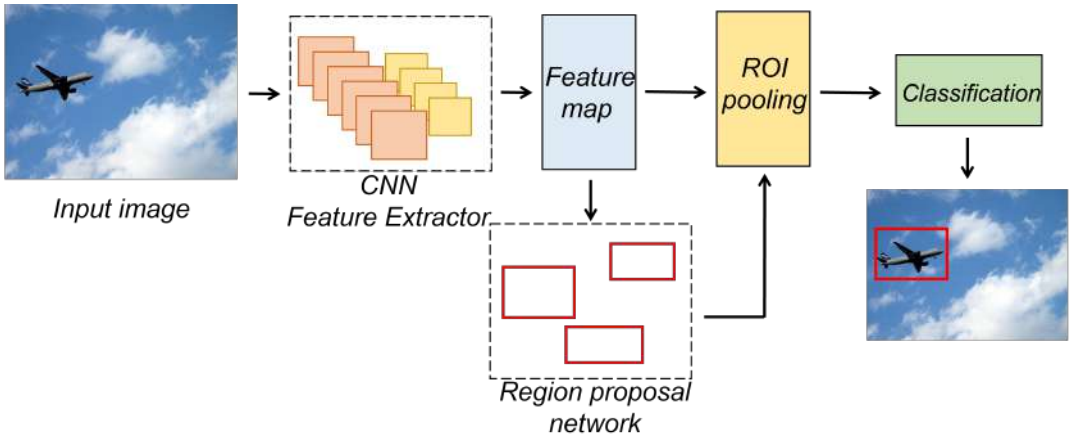

Figure 3.9 – The Faster R-CNN model

CNN

Input images go through a CNN, usually pre-trained for the classification task. The authors of this model used the architectures VGG and the ZF model, pre-trained on ImageNet. In the CNN, only the output of an intermediate layer will be used. The objective is to extract the features of the image and obtain a feature map (65).

RPN

After extracting the features from the image, we slide a small net over the output of the feature map and apply a convolutional layer. Several anchor boxes with different scales and proportions are generated at each sliding window location. Then, two 1x1 convolutional sister layers (two siblings) are applied, which generate the classification box and the regression box, respectively, whose number of channels depends on the number of anchors.

The box generates two predictions: the foreground score, that is, the object, and the non-object score. The regression box generates four predictions related to the location of the anchor. All anchors are placed in two categories for training, foreground and background. Foregrounds are those that have an IoU value greater than 0,7. Backgrounds are those that do not overlap any object or that have an

IoU less than 0,3. Then they are placed in a mini-batch, with samples of 50% each and trained. Anchors that are predicted and contain objects can sometimes overlap, and there is objects duplication. NMS is used to solve this problem, which discards those with an IoU less than a predefined threshold. Thus, the proposed regions containing the objects are obtained (65).

RoI pooling

After the second stage, we have several object proposals, and it is necessary to place the bounding boxes and classify them in the intended categories. In this step, for each proposed region, it selects the feature map section and applies max pooling, dividing the proposed region into sections of equal sizes to find the highest value. This value is copied to the output. Thus, it is possible to extract the resources of the regions of interest in the input image from the regions proposed in the previous step (65).

Classification

After extracting resources from the image, it is necessary to proceed with the classification and adjustment of the bounding boxes. For this task, two fully connected layers are used. Then, for each object, two different fully connected layers are applied. A layer with C+1 units, where C are classes and 1 is the background; this layer makes object classification. And another layer with 4C units, where we have four parameters (x and y from the centre, width, and height of the bounding box) for each class C, does bounding boxes regression (65).

3.4.3 YOLO

The YOLO has continued to evolve since its initial release in 2016, as can be consulted in the Figure 3.10.

In 2016, Redmon et al. (8) proposed YOLO, one of the first detector one-stage objects. This method initially divides the input image into an SxS grid; if a grid

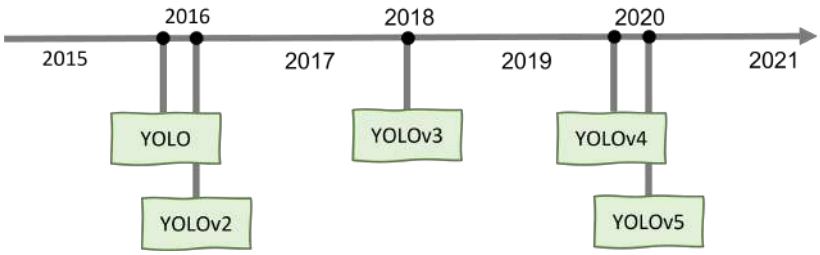

Figure 3.10 – YOLO timeline

cell contains the object's centre, that cell is responsible for detecting that object. Each cell predicts bounding boxes and assigns a confidence score that reflects the probability value that bounding boxes contain the object of interest. Each cell also makes the prediction of the class, in which the probability for each of the possible classes is provided. In the end, the predictive value of the class and the confidence score are combined into a final score, which will determine the probability that the box has the object it is intended to detect (8). This methodology is represented in the Figure 3.11. Some objects can be detected by only one cell, however, when objects are large, they can be found by multiple cells. To correct multiple detections of the same object is used NMS, this method allows selecting only one bounding box, the one with the best score.

The YOLO architecture was inspired by the GoogLeNet model. The first version has 24 convolutional layers followed by two fully connected layers. This version is fast; however, it makes more localisation errors and has limitations for small objects. To solve these issues, in the same year, Redmon proposed the YOLOv2 (71). This version introduces batch normalisation layers after convolution layers and anchor boxes. The anchor boxes are assumed of bounding boxes, they are positioned in the centre of each cell. The network adjusts the width and height of the box to obtain the box with the object to be detected. For YOLOv2, it was used Darknet-19, a classification model that is made of nineteen convolutional layers and five max-pooling layers. This version is still bad for small objects (71).

YOLOv3 was released in 2018 by Redmon et al. (72). This release also uses anchor boxes like YOLOv2, with the ability to predict boxes at 3 different scales to detect

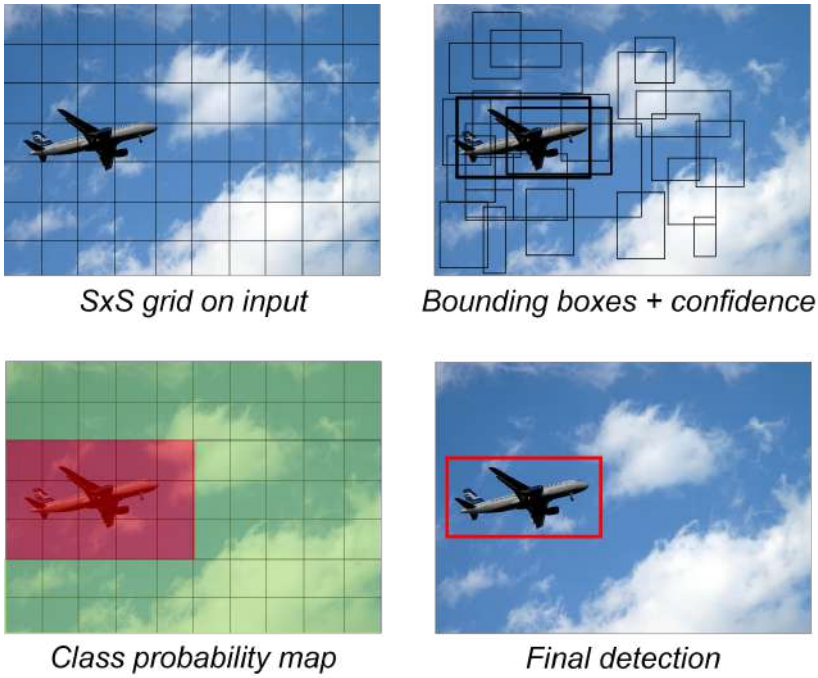

Figure 3.11 – The YOLO model. Adapted from (8)

objects of different sizes, from the smallest to the largest. This feature makes it possible to predict 9 anchor boxes, 3 per scale.

The authors considered that the architecture is divided into the backbone, head, and neck. The backbone is composed of convolutional layers to detect the main features of an image and process them. The backbone is first trained on a classification dataset, such as ImageNet, at a lower resolution than the final detection model. Neck uses the capabilities of convolution layers in the backbone with fully connected layers to make predictions for classes and bounding boxes. The head is placed between the backbone and the neck and is used to collect feature maps at different stages (72). In YOLOv3, Darknet-53 is used as the backbone, this architecture contains 53 convolutional layers. As a neck used, the feature pyramid networks (FPN). FPN is a top-down architecture with lateral connections that allows for high-level semantic feature maps at all scales, favouring small objects' detection. The head is constituted by a YOLO layer, which performs the prediction of the bounding box. As it detects objects at 3 different scales, YOLOv3 minimises the disadvantages

of YOLOv2 and YOLO, making it possible to detect small objects (72).

Bochkovskiy et al. (73) propose the YOLOv4 to prioritise real-time detection. The authors of YOLOv4 uses the CSPDarknet-53 as backbone (74). The CSPDarknet-53 is the Cross Stage Partial Network (CSPNet) application in Darknet-53 architecture. Wang et al. (74) propose the CSPNet that can be used with several architectures to mitigate the required computational cost. This strategy allows the architecture to achieve a richer gradient combination by separating the feature map of the base layers into two parts, one part passes through a dense layer, the result of this layer is concatenated with the other part and transmitted to the next stage, like show Figure 3.12.

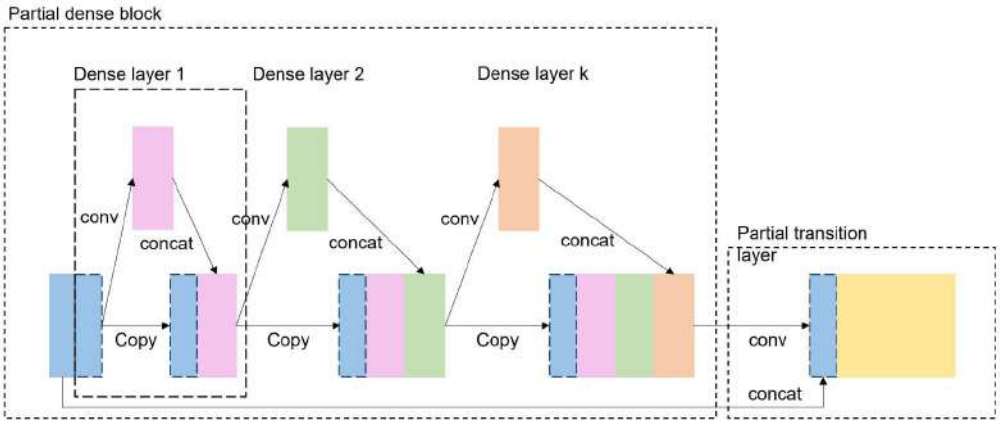

Figure 3.12 – Ilustrations of CSPDenseNet. Based on (8)

In this way, the architecture achieves a richer gradient combination, there is a reduction in computational cost and an increase in inference speed (74). The neck is composed of spacial pyramid pooling (SPP) (75) and path aggregation network (PAN) (76). The SPP was created with the aim of eliminating the need for input images to have a fixed size predefined by the architecture. This strategy generates a fixed-length representation regardless of the size of the input image (75). The PAN is an instance segmentation framework with the aim of increasing the flow of

information in the framework. It is used to predict class labels and can use pixel-by-pixel instance masks for locating objects (76). In addition, the head is composed of the YOLO layer, who is able to for detecting objects at three different scales.

For last, in June 2020, Glenn Jocher[1] developed YOLOv5. This version is the only one that uses PyTorch instead of Darknet. They use CSPDarknet-53 as a backbone, like YOLOv4. The Figure 3.13 shows the YOLOv5 architecture. The focus layer is the junction of the first three layers of YOLOv3 into a single layer, SPP was also added so that there is no restriction on the size of the image. The neck uses PAN with FPN, which will improve the propagation of low-level features in the model, which increases object location accuracy, even for small objects. The head is the same as YOLOv3 and YOLOv4 (77). The Table 3.2 compares the backbone, neck and head between each method of YOLOv3, YOLOv4 and YOLOv5.

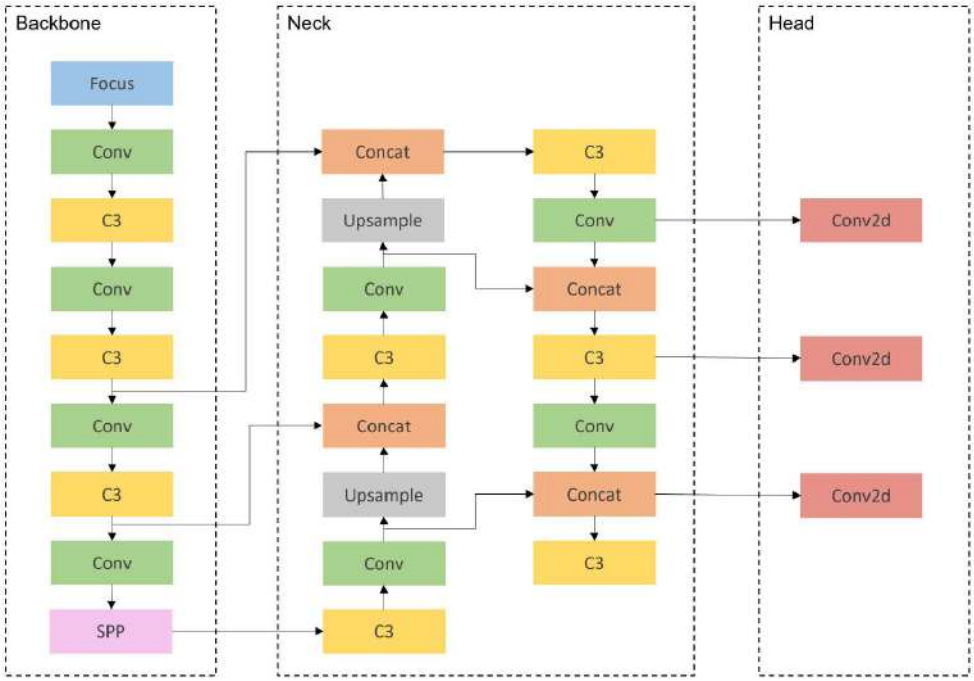

Figure 3.13 – The architecture of the YOLOv5 method. Based on (9)

[1]https://github.com/ultralytics/yolov5

	YOLOv3	YOLOv3	YOLOv5
Backbone	Darknet-53	CSPDarknet-53	CSPDarknet-53
Neck	FNP	SPP and PAN	PAN with FPN
Head	YOLO layer	YOLO layer	YOLO layer

Table 3.2 – Comparison between YOLOv3, YOLOv4 and YOLOv5.

3.4.4 SSD

SSD was designed for real-time object detection, proposed by Liu et al. (62). This methodology does not use a region proposal network, which makes the process faster. SSD has two components, the backbone model and the SSD head. The backbone model is a pre-trained classification network; they used the VGG architecture. The final fully connected layer has been removed. The SSD head is a set of auxiliary convolutional layers that will generate the bounding boxes and classification of objects. These layers progressively decrease in size and allow detection predictions at various scales; that is, it generates multi-scale feature maps, producing the results of the detection task (62).

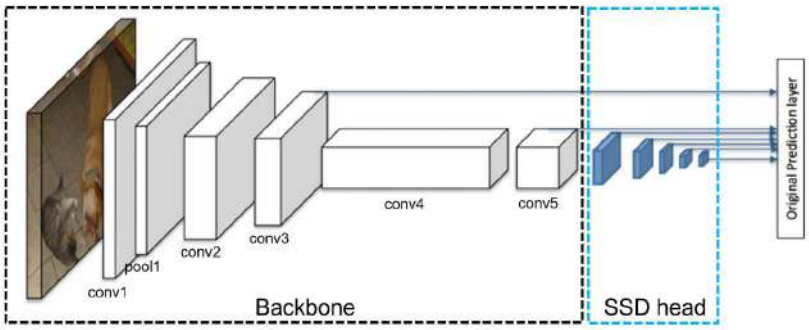

Figure 3.14 – SSD architecture. Adapted from (10)

Like YOLO, SSD splits the image using an SxS grid, where each grid cell is responsible for detecting the object. Each cell can also be assigned several anchor boxes, which

can have different sizes and shapes from the grid cells, depending on the dimensions of the objects to be detected. In this context, the SSD allows you to define a hierarchy of grid cells in different layers, using grids of various dimensions depending on the size of the objects you want to find (62).

The authors of this method showed that feature maps at various scales improve object detection and that the more significant number of bounding boxes results in better performance. Compared to Faster R-CNN, it is three times faster; however, it has worse performance for detecting small objects. Compared to the R-CNN, this model has fewer errors in the location of bounding boxes, but it has more classification errors for similar categories (62).

3.4.5 Summary of detection methods

Aiming to compare all the detection methods, the table 3.3 shows each detection performance. In addition, in the table are presented the image input size, the mAP and the time in frames per second (FPS). The methods were executed in two extensive public datasets, the Pascal VOC 2007+2012 and MS COCO. The mAP was measured at a 50% IoU threshold.

It is possible to verify that the detection performance is higher for one-stage methods. In general, YOLO is the method that have the best mAP and time, and the performance improvement with the evolution of this architecture is remarkable. Thus, the one-stage method with a higher mAP for the MS COCO dataset was YOLOv5. Meanwhile, Faster R-CNN was the two-stage method with the highest mAP in the Pascal VOC dataset.

Methods	Input size	mAP (%)	Time (FPS)
Train on Pascal VOC 2007+2012			
R-CNN	227x227	66.0	0.1
Fast R-CNN	300x400	70	0.5
Faster R-CNN	300x400	73.2	7
YOLO	448x448	63.4	45
YOLOv2	416x416	76.8	67
YOLOv2	480x480	77.8	59
YOLOv2	544x544	78.6	40
Train on MS COCO			
YOLOv3	320x320	51.5	22
YOLOv3	416x416	55.3	29
YOLOv3	608x608	57.9	51
YOLOv4	416x416	62.8	96
YOLOv4	512x512	64.9	83
YOLOv4	608x608	65.7	62
YOLOv5	640x640	66.9	59
SSD-300	300x300	41.2	46
SSD-512	512x512	46.5	19

Table 3.3 – Performance of different detection methods.

4 Literature review

Given the dimensions of insects, in section 4.1, we approach the problem of detecting small objects and present some methods described in the literature to solve this task. Section 4.2 describes some works presented to solve the detection and count of insects.

4.1 Small object detection

The task of detecting small objects is very challenging in computer vision problems. Small objects occupy areas smaller than or equal to 32x 32 pixels. Although many methods used for detection give good results for medium to large objects, they perform poorly when used to detect small objects. Small object detection history is relatively short compared to other computer vision tasks (78).

Most of the algorithms used in the object detection task are based on CNNs. To reduce the sampling of feature maps, pooling layers are applied after the convolutional layers, thus reducing image and feature map dimensions (42). Due to this characteristic of CNN, and as small objects are represented by a few pixels, their characteristics extracted in the initial layers end up being eliminated and do not reach the detection

and classification steps (78).

Tong et al. (78) pointed out five important aspects that can improve the task of detecting small objects, multi-scale feature learning, data augmentation, training strategy, context-based detection, and GAN-based detection. Some authors have already proposed methods that can contribute to the detection of small objects considering these important aspects. Li et al. (79) used FPN on a Faster R-CNN system. Their model got better results. They were able to increase the mAP from 47.3% to 56.9% in object detection datasets.

Some of the methods proposed for this task are based on modified algorithms previously proposed for object detection. Zhang et al. (80) proposed deconv R-CNN, a network with a deconvolution layer after the last convolution layer of the base network. For the task of detecting small objects in remote sensing images, they obtained an mAP of 55.6%, while for the Faster R-CNN model, they obtained an mAP of 42.5%.

Ren et al. (81) developed a methodology for detecting small remote sensing objects. Their proposal consisted of a modification of the Faster R-CNN. Given the small dimensions of the objects, the authors observed that the proposed conventional Faster R-CNN anchors are much larger than most of the objects they intended to detect. Based on this observation, they defined anchors with dimensions adapted to the sizes of the objects to be detected. Inspired by the FPN, they proposed three feature maps concatenated channel-by-channel with the upward path feature maps. Thus, they obtained a single high-level feature map with a final resolution. The authors of this methodology applied data augmentation during training. For this task, they obtained an mAP of 78.9%.

As we have already seen, object detection by two stages (R-CNN or Faster R-CNN) achieved greater accuracy while the one-stage approach (SSD) achieved greater efficiency. To combine the advantages of the two types of detections, Zhang et al. (82) proposed a new approach to one-stage detection, called RefineDet, which achieves better accuracy than two-stage methods and maintains the efficiency of

one-stage methods. This approach is divided into two modules, the refinement of anchors (a) and the object detection module (b). Module a has the objective of filtering the anchors that are not of interest to improve the search for objects. And module b aims to adjust the locations and sizes of the anchors. The refined anchors, obtained initially, are inputs to module b so that the detection accuracy can be further increased. A connection block is also designed to transfer resources in module a to predict locations, sizes, and labels in the object detection module. This approach showed more accurate detection results, especially for detecting small objects (82). The authors obtained 86.8% mAP for object detection in Pascal VOC 2012.

Ahmad et al. (83) proposed a method based on the RetinaNet (11) architecture to optimise anchors; the objective was to reduce unnecessary region proposals. RetinaNet is an architecture that consists of 4 stages:

(a) ResNet – backbone network that determines the resource maps at different scales; this step can still be called upward path.

(b) FPN – the descending path increases the sampling of feature maps, and the lateral connections allow you to generate a pyramid of convolutional features at various scales.

(c) Class subnet – generates the object classification prediction.

(d) Box subnet – does the regression of the anchor boxes.

The experiments of (83) showed that the anchors used by RetinaNet had sizes, aspect ratios, and scales unable to detect objects of reduced size. They used the Crow Search algorithm (84) to research ideal anchor proportions and scales. This algorithm allows iteratively improved solutions created by the candidate's best performance. They defined five iterations, in which if there is no improvement, the algorithm stops looking for anchors and thus uses the most optimised one. After obtaining the anchor sizes, they trained the detector against these results. Although the results were not good, they showed that optimising anchors improve detection.

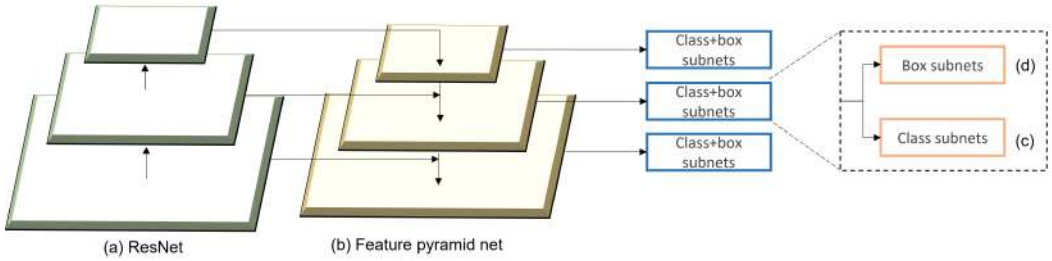

Figure 4.1 – RetinaNet architecture. Source (11)

They obtained for RetinaNet (initial version) 24.9% mAP, and with the optimisation of the anchors, it was 29.5% mAP.

Reference	Dataset	Method	mAP (%)
(79)	COCO	FPN on a Faster R-CNN	56.9%
(80)	Small remote sensing	Deconv R-CNN	55.6%
(81)	Small remote sensing	Modified Faster R-CNN	78.9%
(82)	Pascal VOC	RefineDet	86.8%
(83)	VisDrone	RetinaNet with anchor optimization	29.5%

Table 4.1 – Summary of techniques used for small object detection.

4.2 Insect detection

The task of detecting and counting insects is challenging task due to the size of the insects. Although this task is of great interest to farmers, resolutions have only recently started to emerge, so there is still a lot to be done in this area (6), (85).

Qiao et al. (86) proposed a simple image processing system to estimate the number of whiteflies on sticky traps automatically. Initially eliminated the noise with a low pass filter; then, the images were converted to grayscale and transformed into binary images. The authors used ten different threshold levels to determine the optimal image level. The pixels with a value greater than the defined threshold were white, and the smallest one was black, and thus it was possible to detect the whiteflies.

The method proved to be very effective for adult whiteflies. However, it only worked for whiteflies on sticky traps.

Xia et al. (87) developed an automatic method for whiteflies, aphids, and thrips identification in greenhouses. The method starts using the watershed algorithm to segment insects from the background. With the Mahalanobis distance, the insect's colouring characteristics were extracted to identify the species of different insects. Comparing the proposed identification and the manual identification performed by experts, a correlation of 93.4%, 92.5%, and 94.5% was obtained, respectively, for whiteflies, aphids, and thrips.

Rustia Lin (88) proposed an IoT based remote monitoring system for pests on yellow sticky traps and developed image processing and ML algorithms. The images were divided into four regions and equalised using the histogram based on the brightness adjustment obtained from reference images. A k-means grouping is applied in each image converted into a colour space. The insects and the background are black or white in the obtained image. In the end, the insects can be classified and counted. The method effectively acquired accurate and automatic pest counts getting an average accuracy of 98%. Classification of pests in corn, soybean, wheat and canola is difficult due to the similarity between insect species. Xie et al. (89) proposed an insect recognition system using multiple task sparse representation and multiple-kernel learning techniques. It showed that their method performs well in classifying insect species, outperforming other methods. Ebrahimi et al. (90) and More Nighot (91) implemented an approach based on the support vector machine for classifying and identifying pests. Most of these techniques showed good performance; however, they are only recommended for particular situations and are not adaptable to other scenarios because these techniques can't make intelligent decisions.

With the advancement of science and technology, several DL pest detection methods have been developed. DL is a branch of ML that provides better solutions for computational vision tasks (92). DL can learn and make intelligent decisions using algorithms inspired by the human brain, making it possible to adapt to more complex environments (93), (94). In recent years, several DL applications have emerged to

solve challenges in the agricultural context. Automatic recognition of pest images has become one of the leading research points in DL (12).

Nieuwenhuizen et al. (18) presented a methodology to detect and count the whitefly in yellow traps. The trap images were captured under controlled light conditions. They used Faster R-CNN with inception Resnet v2 (95). For the detection task, they obtained 87,40% mAP. The counting task compares the results obtained with traditionally counting; the correlation was greater than 0,95. However, they state that the quality of the data and annotations present in the images influenced the classification results. Nieuwenhuizen's method has two limitations: it only works on images with a yellow background, and the images must be obtained under controlled conditions.

Li et al. (96) used a CNN from ZF and an RPN with NMS to remove overlapping detections for the location and counting of wheat mites. First, the ZF convolutional layers have no pooling and no fully connected layers. And then optimised several critical parameters like output size, score threshold, and NMS threshold. In the end, they explored various architectures, including AlexNet and ResNet. The architecture that got the best score was ResNet with 88,50% mAP.

Shi et al. (97) proposed an architecture based on the R-FCN method (67) to detect eight species of insects that may be present in stored grains to guarantee the safety of the grains during storage. R-FCN is very similar to Faster R-CNN; only the fully connected layers are replaced after RoI pooling, with a set of position-sensitive score maps to perform average voting. In the method proposed by Shi, it was then based on R-FCN, in which the CNN used was DenseNet (98). To further improve accuracy, they used training techniques on various scales, and in the end, they applied the soft-NMS algorithm (99). The paper's authors applied several models, such as Faster R-CNN and YOLO, to their database to compare results with their proposed method. They showed that the model that obtained the best results was their proposal based on the R-FCN, in which they obtained an mAP of 83,44%.

To develop algorithms that detect and count M. thunbergianae from images of

traps with pheromones, Hong et al. (19) trained various object detection models and evaluated their performance. In addition, the speed of each model was also analysed. For the robustness of the model, they applied transfer learning and data augmentation during training. The model that obtained the best results was Faster R-CNN, with an AP of 85,63%. The model that had a shorter inference time was the SSD; however, the detection results are not as good as the Faster R-CNN. Despite obtaining good results, this work has the limitation that the images are collected in controlled environments. The authors of this paper have concluded that constant monitoring can be achieved with accurate performance for a pest control system using DL techniques.

Q. J. Wang et al. (12) provide a standardised dataset on traps for multiple agricultural pest targets. This database, called *Pest24*, consists of 25,378 high-resolution images with 24 major pest classes specified by the Chinese Ministry of Agriculture. They applied several state-of-the-art object detection methods, Faster R-CNN, SSD, YOLOv3, and Cascade R-CNN. For each technique, they initially used the default settings of their hyperparameters. Then tried different hyperparameter values; for the YOLOv3 method, which showed the best results, they used the k-means clustering algorithm to optimise the parameter's scaling range. The backbone of this method was Darknet-53. YOLOv3 obtained the mAP of 58.79%, proving to be the model that worked best to detect the twenty-four species of insects. Given the size of the dataset and the high number of classes, the authors considered adherence to objects, pest similarity, pest density, relative scale, and colour discrepancy as essential factors in the detection task. The relative scale is the factor that exerts the most significant influence on the AP of detection, and the colour discrepancy has the least significant impact.

Li et al. (100) developed a method based on Faster R-CNN, called 'TPest-RCNN', to automatically detect whitefly and thrips on the sticky trap in greenhouse conditions. the dataset contained 1,400 images. The algorithm proposed has two significant differences from the Faster R-CNN, improved the anchor size, and the RoIPooling design was adjusted to focus on small objects and thus be able to obtain exact

locations. The backbone network used is VGG16. The anchor size present by Faster R-CNN is larger than the insect dimensions, so it adapted the anchor dimensions to the insect dimensions to solve this problem. RoIPooling has been replaced by a method the authors call RoIAlign, inspired by the Mask R-CNN architecture. RoIPooling can produce a deviation between the final and initial position of the bounding box, which may represent the wrong detection. To solve this, RoIAlign divides the proposed region into 4x4 pool sections. Four sampling areas are defined for each section, the centre point of each sampling area representing the sampling location. The pixel values of these points were calculated using the bilinear interpolation method. And finally, max pooling is applied for each compartment. The methodology applied by the authors obtained an mAP of 95.2%. The proposed model surpassed the Faster R-CNN architecture.

Reference	Dataset	Method	Backbone	mAP (%)
(18)	Whitefly	Faster R-CNN	Inception ResNet V2	87.4
(96)	Wheat mite	CNN and RPN	ZF	88.5
(97)	Eight insects in stored grains	R-FCN	DenseNet	83.44
(19)	M. thunbergianae	Faster R-CNN	VGG	85.63
(12)	Pest24	YOLOv3	CSPDarknet-53	58.79
(100)	Whitefly and thrips	Modified Faster R-CNN	VGG	95.2

Table 4.2 – Summary of deep learning methods used for insect detection.

5 Material and methods

In this chapter the materials and methods used to construct the work will be described. The used methodology is organised into three steps, represented in Figure 5.1. First, we describe and prepare the datasets to apply the methodologies. Then, we trained the three datasets separately with YOLOv5: first training the *Pest24*; and then the mySense datasets, *Bedbug*, and *Grape moth*. Finally, the model's performances were evaluated.

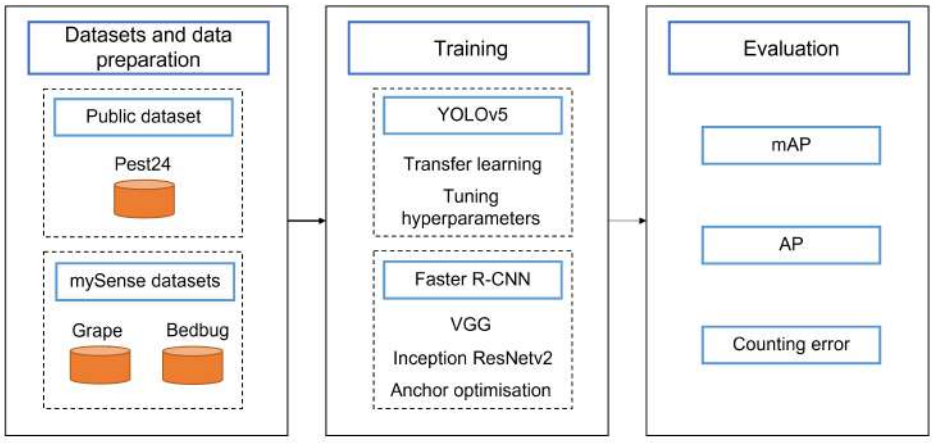

Figure 5.1 – The pipeline of this work.

In the section 5.1 there will be a brief description and characterisation of the

properties of the public dataset and the datasets available by mySense. And, the pre-processing is detailed. The methodology used for detection and counting is presented in the section 5.2, including models used, the training process and the metrics used to evaluate them.

5.1 Datasets and data preparation

5.1.1 Pest24

The *Pest24* dataset contains 25,378 annotated images with 24 major pests specified by the Chinese Ministry of Agriculture. All images were taken by an automatic pest image acquisition device, like show in the Figure 5.2. This device can be set up in the field and automatically trap and take photos of pests (12). The *Pest24* is very distinct from conventional object detection datasets and thus poses new challenges for object detection methods. This dataset is characterised by the large scale of data, tiny relative scales of objects, the high object similarity, and the dense distribution of objects. The Table 5.1 shows the description of the 24 pests, with the index associated with each pest, the portrait, the number of images with each pest, the number of instances, the relative scale and the colour discrepancy.

Figure 5.2 – Devices installed in the cultures to collect traps images and the respective image collected. Source (12).

Index	Specie	Image	Instances	Relative scale (%)
1	Rice planthopper	316	1,511	0.034
2	Rice Leaf Roller	944	1,240	0.123
3	Chilo suppressalis	454	1,285	0.186
5	Armyworm	3,828	8,880	0.394
6	Bollworm	9,049	28,014	0.281
7	Meadow borer	5,526	16,516	0.226
8	Athetis lepigone	7,520	30,339	0.13
10	Spodoptera litura	1,588	1,951	0.458
11	Spodoptera exigua	3,614	7,263	0.138
12	Stem borer	1,357	1,804	0.277
13	Little Gecko	2,503	4,279	0.57
14	Plutella xylostella	531	953	0.043
15	Spodoptera cabbage	1,707	2,302	0.42
16	Scotogramma trifolii Rottemberg	3,223	4,679	0.28
24	Yellow tiger	1,388	1,686	0.398
25	Land tiger	369	475	0.639
28	Eight-character tiger	154	168	0.441
29	Holotrichia oblita	90	108	0.334
31	Holotrichia parallela	3,111	11,675	0.255
32	Anomala corpulenta	5,228	53,347	0.249
34	Gryllotalpa orientalis	3,629	6,528	0.95
35	Nematode trench	118	167	0.32
36	Agriotes fuscicollis Miwa	1,814	6,484	0.114
37	Melahotus	239	768	0.158

Table 5.1 – Description of the 24 classes of pests in *Pest24* (12).

All images are available in *jpg* format with a resolution of 800x600 pixels. The annotation files are in Pascal VOC format. The Pascal VOC is an XML file consisting of:

- Folder: a folder that contains the image;
- Filename: image name;
- Size: image size, with height, width and depth;

- Object: contains the object's class name, the position if known, truncated (if the object is not fully visible, it has a value of 1, else it has a value of 0), the difficulty of detecting the object and the coordinates of the bounding box. The bounding box coordinates are the value of xmin, ymin, xmax and ymax.

These characteristics can be observed in the image and its annotation (Figure 5.3).

Figure 5.3 – Example of image and respective annotation in Pascal VOC format. Source (12).

All images are available in *.jpg* format with a resolution of 800 x 600 pixels, and the images had the date and time of image acquisition in the upper left corner and the acquisition device identification in the lower-left corner. To avoid any interference in learning stage, all images were pre-processed, where a black rectangle was placed over the two watermarks. All datasets have been resized to 640 x 640 pixels because of the resolution indicated for the detection methods applied. Figure 5.4 shows the before (a) and after (b) data preparation. The dataset was divided into training, validation, and testing, with 18,105, 4,857 and 2,417 images, respectively.

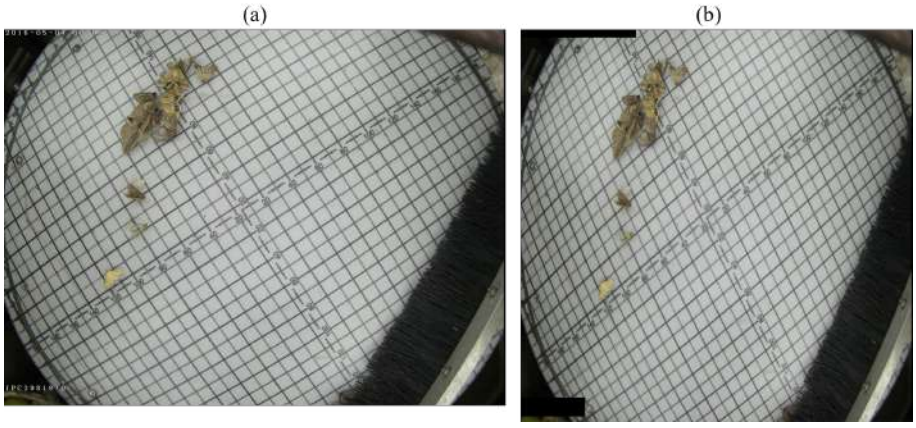

Figure 5.4 – Image example before and after the preparation date. (a) is an image with 800x600 resolution and has the image acquisition data in the upper and lower left corners. And (b), the image has a resolution of 640x640 and a black box cover.

5.1.2 mySense

mySense[1] has several devices installed, as shown in the Figure 5.5, in several geographically dispersed observatories in different cultures and installed at strategic points on the agricultural plot to collect images. For our case study, these devices collect real-time images of insect traps, enabling images in traps in a natural environment. Images were collected from two different observatories, giving rise to two datasets, the *Bedbug*, and the *Grape moth*, as shown in the figure.

Bedbug

Bedbugs are 5mm long insects and brown in colour. These feed on the sap of oat, wheat, corn, cotton, soybeans and rice plants. Bedbug pests are responsible for causing high losses, as they cause damage to the quality and productivity of plantations (101). The installation of light traps for the capture and automatic counting allows for the management of warnings and monitoring of these insects.

A dataset was built with images collected by mySense in light traps with bedbugs, called "*Bedbug*". The *Bedbug* dataset contained only 42 images in *jpg* format with a

[1]https://mysenseapi.utad.pt/

Figure 5.5 – Devices installed in two cultures (rise and vineyard) to collect traps images and the respective image collected.

resolution of 920x840 pixels. An expert in the images identified the bed bugs. And using LabelImg[2], we made the annotation in Pascal VOC format of all images.

Therefore, all images have been resized to 640x640 pixels, as this is the resolution indicated for the detection methods applied. The bedbug has very small dimensions, having a relative scale of 0.019. Since there was a lack of *Bedbug* data, we resorted to a dataset management tool, Roboflow [3], to do data augmentation. The data augmentation techniques applied just in images train, with horizontal and vertical flip and rotation between -14 and +14, as shown in the Figure 5.6. After data augmentation, the dataset contains 102 images with 770 instances. We divided the dataset into training, validation and testing, with 90, 7 and 5 images, respectively.

Grape moth

The grape moth is the main pest of the vine, representing large-scale losses. This insect has several stages of its morphology, egg, caterpillar, pupa and adult. The caterpillar pierces and feeds on the wolves, causing a decrease in production. The protection strategy is pest monitoring, using pheromone traps to attract adult

[2]https://github.com/tzutalin/labelImg
[3]https://roboflow.com/

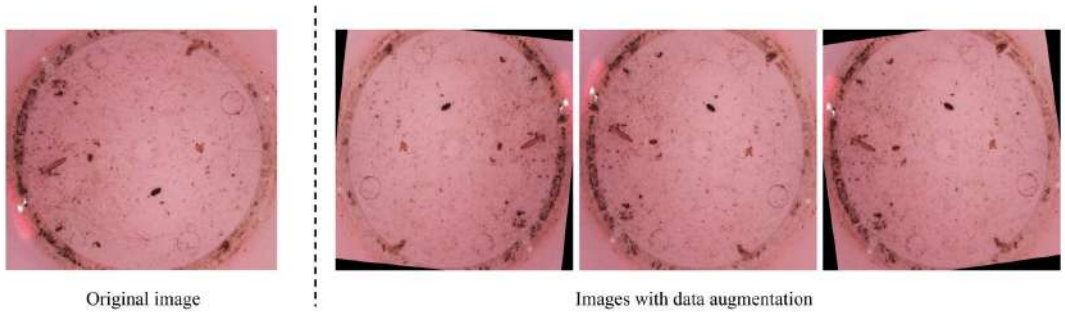

Figure 5.6 – Original images (on the left) and images with data augmentation (on the right) of *Bedbug*.

moths, and thus it is possible to determine the beginning of the morphological cycle of the moth and subsequent application of pesticides. As an adult, the moth is a brownish butterfly about 12 mm long (102).

With the automatic collection of images in pheromone traps of the grape moth, it was possible to build a dataset called "*Grape moth*". The *Grape moth* dataset contained 226 images in jpg format with a resolution of 1600x1200 pixels. This dataset contained 146 annotated images (D1) and 82 unannotated images (D2). All images were resized to 640x640 pixels, with the grape moth having a relative scale of 0.031. And the D2 images were cropped at the bottom of the image to remove a footer, which contained the image's date, time, and location.

The D2 images were collected in an instrumented trap over several months, containing images with sun, different luminosity, shadow and with moths in various positions (which causes confusion to the model). The fact that the images were collected in the same trap without it being replaced, part of the images contains the same insects, so we apply an algorithm to calculate the similarity between images to exclude the most similar ones.

Experts annotated the D1. To annotate the D2, we learned to identify the grape moth adult; however, the annotation task is very time expensive, so we applied a YOLOv5 in D1 and used the knowledge learned during training to generate

annotations of D2. In the end, all the generated annotations were corrected, and we confirmed the veracity of each one. Then, we applied the data augmentation techniques in D2, like the 90º rotate upside and down, and vertical flip and rotation between -14 and +14, resulting in 212 images. After data augmentation, we added D2 to D1, getting 348 images with 12438 instances, and we divided the dataset images into 318 for training, 19 for validation and 11 for testing.

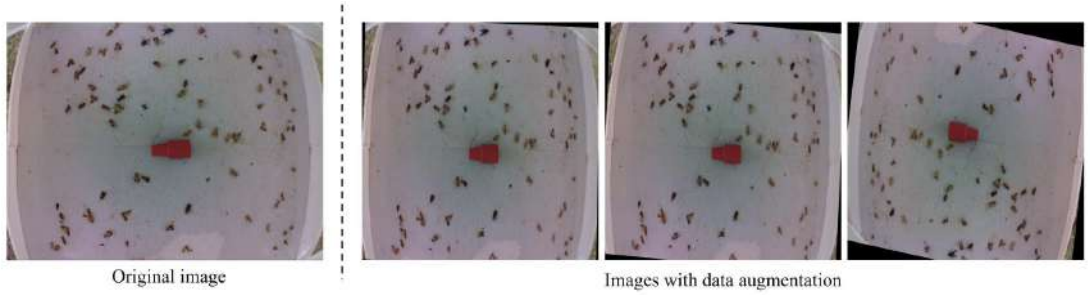

Figure 5.7 – Original images (on the left) and images with data augmentation (on the right) of *Grape moth*.

Dataset	Classes	Train	Validation	Test	Data augmentation
Pest24	24	18,105	4,857	2,417	No
Bedbug	1	90	7	5	Yes
Grape moth (D1+D2)	1	318	19	11	yes

Table 5.2 – Summary of datasets used.

5.2 Insect detection and counting

Our methodology was divided into two parts, detection and counting. First, we use object detection methods, one-stage and two-stage detection methods. Then, considering the result of the detection methods, the number of insects generated by the model was counted. Considering state of the art, the one-stage detection method was YOLOv5, and the two-stage detection method was Faster R-CNN. All experiments were executed on a computer with only one GPU NVIDIA GeForce GTX 1080 TI.

5.2.1 YOLOv5

There are several YOLOv5 models. For our methodology, we chose YOLOv5l, as it has a lower computational cost than other YOLOv5 models and has a considered performance. The YOLOv5 is constituted by CSPDarknet-53 as the backbone, PAN with FPN as the neck, and the YOLO layer as the head, as explained in Chapter 3. The CSPDarknet-53 has multiple convolutional layers and pooling, which extract feature maps. The feature maps have different sizes merged at the neck, reducing the loss of information extracted from the input images. The output network performs object detection and classification (9). The YOLOv5l contain 468 layers with 46,210,094 parameters. This model is trained on MS COCO with an image size of 640x640 pixels and obtains a mAP of 67.3%; the GPU speed is 10.1ms. This model used the SGD optimiser and the loss binary cross-entropy function.

Since DL models are sensitive to hyperparameter selection, better initial estimates will yield better results. However, finding the ideal values can be a challenge. YOLOv5 has 28 hyperparameters that are defined before the start of training and are called default values, using the evolution algorithm, it is possible to train a few epochs with several generations in which the hyperparameters will be adapted to the scenario in question. This evolution algorithm was developed for YOLOv5, and have time-consuming and high computational time. As *Pest24* is characterised by a large scale of data, a small scale of objects, high object similarity, and dense distribution, we applied the evolution algorithm to tune the hyperparameters to the scenario.

Annotations of all images have been converted to YOLOv5 format. Each image has a *txt* file with a line for each bounding box in this format. The format contains class id, centre x, centre y, width and height. A space separates all parameters, and the coordinates are normalised.

5.2.2 Faster R-CNN

The detection with Faster R-CNN was used in four experiments. The Faster R-CNN architecture uses nine anchors in the RPN. Anchors have three scales, with box areas of 128, 256, and 512, and three proportions, 1:1, 1:2, and 2:1. For the detection of small objects, the size of the anchor has dimensions too large compared to the size of the insects. Therefore, we scaled the anchor to 32, 64, and 128. In the Figure 5.8, the left has the anchor with box areas of 128, 256 and 512, and the right has the anchor optimisation with box areas of 32, 64, and 128. In purple are the positive anchors, and the ground-truth bounding box is green.

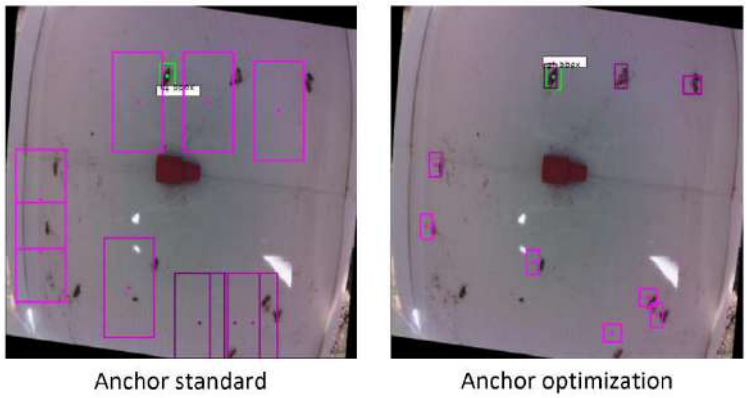

Figure 5.8 – Anchor standard and anchor optimization.

The Faster R-CNN architecture uses the VGG network for feature map extraction. So, we tested the Faster R-CNN architecture with the VGG network with and without anchor optimisation. As some state of the art authors has had some success using the Inception ResNet V2 network, we also tested the architecture with this network with and without anchor optimisation. Weights were used from VGG16 and Inception Resnet v2 were pre-trained in the ImageNet database. The Faster RCNN network was applied in TensorFlow. The learning rate started at $10^{-}5$ with Adam optimiser, and the loss mean squared error function.

The Faster R-CNN use a different annotation format from YOLOv5. So, we need

to convert all annotations. The annotations of all images are in just one *txt* file. Each line of that file contains the path where the image is stored (including the image name) and the coordinates of the bounding box, in order, left, top, right and bottom. A comma separates the coordinates of the bounding box.

5.2.3 Methods

The three datasets were trained separately, first the *Pest24* and then the mySense datasets, *Bedbug* and *Grape moth*.

Pest24

We started by training with YOLOv5 using the standard hyperparameters (PY1), and we trained again with hyperparameters obtained by the evolution algorithm (PY2). This procedure was trained for 400 epochs with a batch size of 16 for training and validation.

Method	Model	Description
PY1	YOLOv5	Detection of 24 insects species using standard hyperparameters
PY2	YOLOv5	Detection of 24 insects species using evolution algorithm for tuning hyperparameters

Table 5.3 – Summary of methodologies used for *Pest24*.

Bedbug

We trained with YOLOv5 using the standard hyperparameters (BY1), with hyperparameters tuning of *Pest24* (BY2), and were trained with transfer learning (BY3). For transfer learning, we use the PY1 weights and train the head of the model, freezing the 12 layers of backbone. For each experiment was trained for 150 epochs with a batch size of 16 for training and validation.

The detection with Faster R-CNN was used in four experiments: the Faster R-CNN using VGG with standard anchor scale (BF1) and with anchor optimization (BF2);

and using Inception ResNet v2 with standard anchor scale (BF3) and with anchor optimization (BF4). Each method were trained for 35 epochs with 200 iterations each. All methods are summarised in table 5.4.

Method	Model	Description
BY1	YOLOv5	Detection of bedbug using standard hyperparameters
BY2	YOLOv5	Detection of bedbug using tuning hyperparameters for Pest24
BY3	YOLOv5	Detection of a bedbug with transfer learning using the PY1 weights
BF1	Faster R-CNN	Detection of bedbug using the VGG
BF2	Faster R-CNN	Detection of bedbug using the VGG and the anchor optimisation
BF3	Faster R-CNN	Detection of bedbug using the Inception ResNet V2
BF4	Faster R-CNN	Detection of bedbug using the Inception ResNet V2 and the anchor optimisation

Table 5.4 – Summary of methodologies used for *bedbug*.

Grape moth

As we had part of the grape moth dataset unannotated, we first trained the annotated dataset (D1) with the YOLOv5 model with standard hyperparameters for 150 epochs with a batch size of 16 for training and validation (GY0). Then, we use the knowledge to predict the annotations of the unannotated images (D2) and correct the generated annotation of each image. Thus, we obtained the complete grape moth dataset (D1+D2), and we repeated the same methodology applied to the *Bedbug*, using the standard hyperparameters (GY1), with hyperparameters tuning of *Pest24* (GY2), and were trained with transfer learning (GY3). All experiments were trained for 250 epochs with a batch size of 16 for training and validation.

For detection with Faster R-CNN, we apply the same methodology as *Bedbug*. The Faster R-CNN using VGG with standard anchor scale (GF1) and with anchor optimization (GF2); and using Inception ResNet v2 with standard anchor scale

(GF3) and with anchor optimization (GF4). Each method were trained for 200 epochs with 200 iterations each. All methods are summarised in table 5.5.

Method	Model	Description
GY1	YOLOv5	Detection of grape moth using standard hyperparameters
GY2	YOLOv5	Detection of grape moth using tuning hyperparameters for *Pest24*
GY3	YOLOv5	Detection of a grape moth with transfer learning using the PY1 weights
GF1	Faster R-CNN	Detection of grape moth using the VGG
GF2	Faster R-CNN	Detection of grape moth using the VGG and the anchor optimisation
GF3	Faster R-CNN	Detection of grape moth using the Inception ResNet V2
GF4	Faster R-CNN	Detection of grape moth using the Inception ResNet V2 and the anchor optimisation

Table 5.5 – Summary of methodologies used for *Grape moth*.

5.3 Models evaluation

The metrics for evaluating the detection models were the AP and mAP. And to evaluate the counting, use the counting error. All metrics were summarised in the Table 5.6 that shows the mathematical equations used to calculate these metrics.

The intersection over union (IoU) is an evaluation metric that measures the common area between the ground truth (G) and prediction (P), divided by the total area of the two regions. The closer to 1 the prediction result is, the better the detection, so if the prediction is 0, the object we want to detect isn't in the bounding box.

The true positives (TP) are the correct number of detections. Detections are correct if the IoU value exceeds the preset threshold value. Usually, by default, the threshold value is 0.5. The false positives (FP) are the wrong number of detections with the IoU less than the threshold. And the false negatives (FN) are the number of objects

that are not detected.

So, we can now define precision and recall. Precision measures how precise your predictions are. That is the percentage of your predictions that are correct. Recall measures how good the true positives are; In other words, it is the fraction of true positives within all positives.

Evaluation metric	Equation
IoU	$\frac{Area(G \cap P)}{Area(G \cup P)}$
Precision	$\frac{TP}{TP+FP}$
Recall	$\frac{TP}{TP+FN}$
AP	$\int_0^1 p\left(r\right) dr$
mAP	$\frac{\sum_c AP_C}{C}$
Counting error	$\left\lvert \frac{C-\hat{C}}{C} \right\rvert$

Table 5.6 – Evaluation metrics for the detection and the counting.

The general definition for average precision (AP) is to find the area under the precision-recall curve. The average precision curve is plotted on the x-axis we recall, and on the y-axis, we have the precision. To do this, predictions and recall are calculated for various threshold values and plotted on a graph. Mean average precision (mAP) is the mean of the AP, that is, the AP for each class is calculated, and, in the end, it is averaged.

Counting error was the metric used to assess insect counts. The measure is the division between the difference between the number of correct detections (C) and the number of predicted detections ($\hat{C}$), by the number of correct detections. The optimal result is closest to 0.

6 Results and discussion

The present chapter will describe the results of the experiments that were conducted to detect and counting of insects. The results are organized by the datasets. In the section 6.1 are presented the results of experiments of *Pest24* dataset. The results of detection and counting of bedbugs are presented in section 6.2. In the section 6.3 are presented the results of grape moth detecting and counting. In the end, in section 6.4 the main found challenges and gaps are provided and discussed.

6.1 Pest24 detection

The evolution algorithm was applied to tunning hyperparameters. Since 300 generations were trained, each taking 45 minutes, it took approximately 10 days to obtain the hyperparameters adjusted to our scenario. The computational cost can be reduced using multiple GPUs or cloud software equipment. The results of 15 hyperparameters are displayed in Table 6.1 where the YOLO standard hyperparameters and the tuning hyperparameters are presented.

The results of the *Pest24* are represented in the Table 6.2. The best results for each method are in bold. The methodology that has the best result is the YOLOv5

Hyperparameter	Standard	Tuning
Initial learning rate (lr0)	0.01	0.0102
Final learning rate (lrf)	0.01	0.0115
Momentum	0.937	0.916
Optimizer weight decay	0.0005	0.00039
Warmup epochs	3.0	2.17
Warmup initial momentum	0.8	0.78
Warmup initial bias lr	0.1	0.125
Box loss gain	0.05	0.0362
Classification loss gain	0.5	0.607
Classification BCE loss positive weight	1.0	1.52
Object loss gain	1.0	0.697
Object BCE loss positive weight	1.0	1.17
IoU training threshold	0.2	0.2
Anchors multiple threshold	4.0	2.29
Anchors per output grid (0 to ignore)	0.0	2.0
Focal loss gamma	0.0	0.0

Table 6.1 – YOLOv5 standard and tuning hyperparameters

model with standard hyperparameters (PY1), obtaining 72.13, 70.7 and 69.4 of mAP, recall and precision, respectively. However, the differences between the models are not relevant. So, optimised hyperparameters did not show improvements in this model performance, which can be explained by the fact that this dataset has high complexity and possibly some hyperparameters did not obtain the optimal value in the evolution algorithm. That is, it would be necessary to train more generations so that all hyperparameters have the optimal value.

Compared to state of the art, our methodology performed better. We use the same dataset and division in training, validation and testing as Q. J. Wang et al. (12). They used YOLOv3 and got an mAP of 58.79%. This difference can be explained by using YOLOv5, a more recent architecture with better performance detecting small objects. Data pre-processing may also have influenced the results.

The Table 6.2 of shows the AP of each class by four experiences developed. This table shows that Gryllotalpa Orientalis (idx 34) has the highest AP of 98.40%,

while the Rice planthopper (idx 1) has the minor AP of 9.87%, both by PY1. This is related to the relative scale and number of instances of each insect. Gryllotalpa Orientalis (idx 34) is the insect that has a larger scale (0.95%) while 1 is one of those with a smaller scale (0.034%) and has an insignificant representation in the dataset, being represented in 316 images and in 1,511 instances.

Classe	PY1	PY2	Q. J. Wang et al. (12)
1	**9.87**	8.18	0.60
2	62.30	**64.60**	51.70
3	**79.30**	75.40	72.10
5	**86.30**	84.60	82.90
6	**94.40**	93.90	91.70
7	**86.70**	85.20	80.70
8	**78.60**	76.70	68.90
10	**81.70**	79.20	76.80
11	**62.20**	59.90	52.50
12	**78.90**	75.40	75.90
13	**91.00**	88.90	88.70
14	**17.70**	15.90	1.60
15	**68.00**	66.90	60.40
16	**69.90**	66.90	51.50
24	**66.30**	65.00	50.20
25	**81.50**	80.80	74.20
28	29.30	**30.20**	1.50
29	43.60	37.80	**61.40**
31	**93.40**	92.60	93.30
32	**97.40**	97.30	97.30
34	**98.40**	98.39	98.60
35	**77.70**	67.00	40.40
36	**87.20**	86.70	79.70
37	**89.60**	88.20	73.60
mAP	**72.13**	70.20	58.79

Table 6.2 – APs of 24 classes of pests by two methods (PY1 and PY2), and with the method obtained by the author of this dataset (12)

The graphs in the Figures 6.1 and 6.2 show a bar graph of the relationship between the AP, obtained in the method with the best performance (PY1), with the number of instances and the relative scale of each class. It is possible to verify an evident relationship between the number of instances and the relative scale of the insect in the AP. The greater the relative scale and the number of insect instances, the greater the AP value. Insects with higher relative scales are represented by a more significant number of pixels, which allows the extraction of more characteristics of the same by the model.

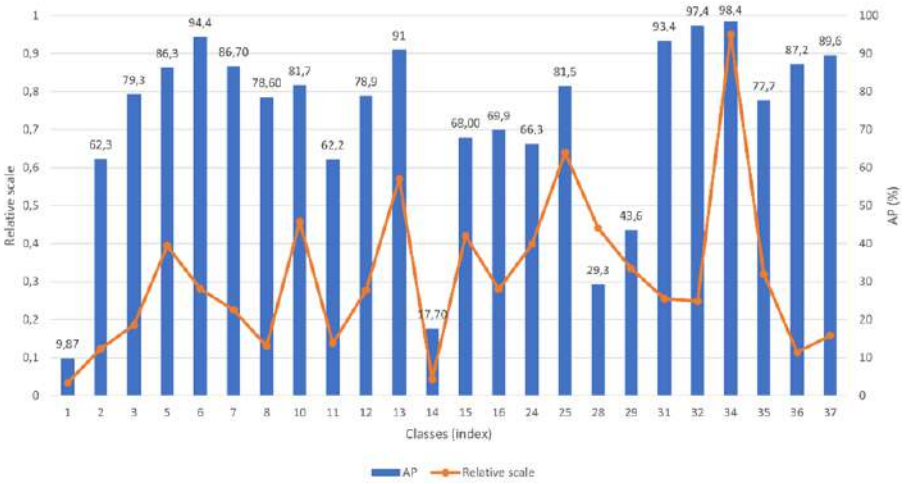

Figure 6.1 – Graph of the relationship of the AP obtained in each class with its relative scale.

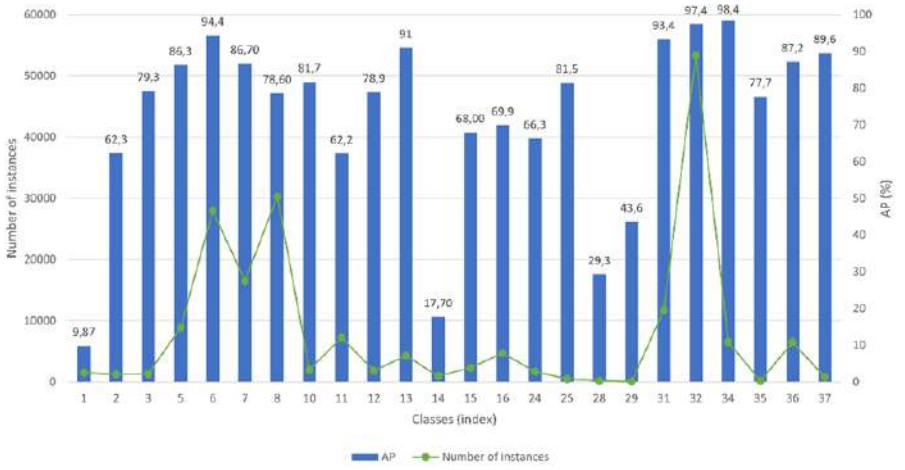

Figure 6.2 – Graph of the relationship of the AP obtained in each class with its number of instances.

To verify which of the parameters has the most influence on the value of AP, we plotted the scatter plots (Figure 6.3) as a function of the AP obtained with the relative scale (a) and with the number of instances (b). The graphs show that the relative scale has the highest correlation (0.308), it is the factor that has the most significant influence on the AP.

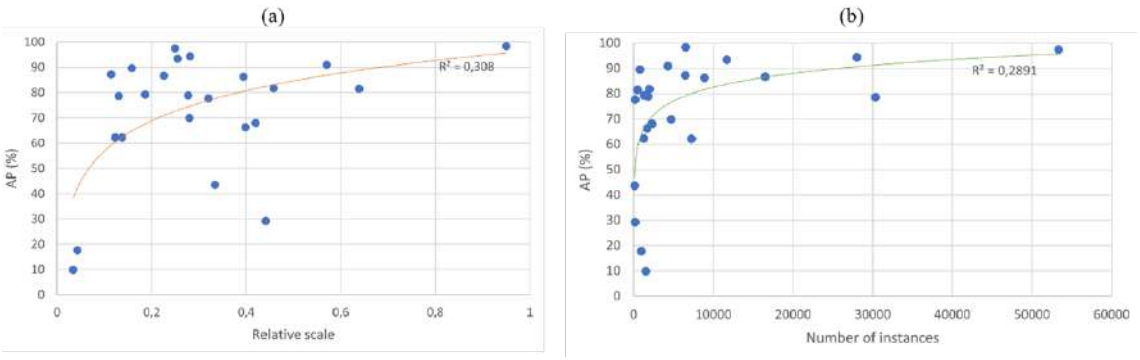

Figure 6.3 – Scatter plots with the correlation value of the trend curve (R), for the relative scale (a) and for the number of instances (b).

The figure 6.4 illustrates the predictions for three different images. The first row of images is the ground truth. The PY1 and PY2 models correctly predicted images (a), correcting the location and classification of all insects. Images (b) and (c) contain some wrong detections, but not very relevant. Comparing the images generated between the two models, there are no major discrepancies.

6.2 Bedbug detection and counting

The *Bedbug* results are summarized in the Table 6.3 and graph in the Figure 6.9. The best results are in bold. The methodology that obtained a superior AP was BY3, the detection of a *Bedbug* with YOLOv5 model with transfer learning using the PY1 weights. Analyzing the results of the one stage detector, we can verify that the BY1 method is the one with lower results than the others, as this method directly applied the YOLOv5 model with standard hyperparameters. While the rest

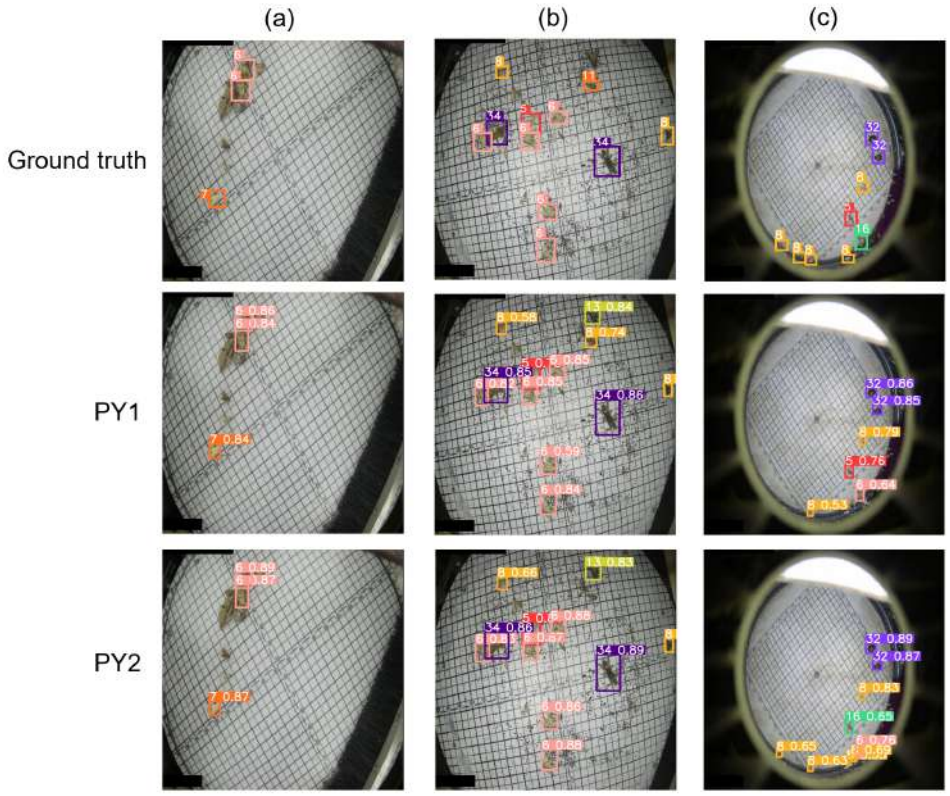

Figure 6.4 – Example of predictions from three *Pest24* images, (a), (b) and (c). The first line is the ground truth, the second is the prediction of the PY1 model, and the last is the prediction of the PY2 model.

applied methodologies with improved hyperparameters and transfer learning from *Pest24*.

Metric	BY1	BY2	BY3	BF1	BF2	BF3	BF4
AP	78.4	87.7	**95.9**	73.5	80	64.2	64.5
Counting error	70.2	92.2	**64.2**	67.4	77.1	95.4	94.5

Table 6.3 – Results obtained by each method for the *Bedbug* dataset.

Analyzing the two-stage method, the model that obtained the highest AP was BF2, detection using the VGG and the anchor optimization. We can verify that the methods that used the VGG architecture were the ones that presented the highest results (BF1 and BF2). The methods that improved their performance with anchor optimization, obtaining mAP superior to those that used the pre-defined anchor

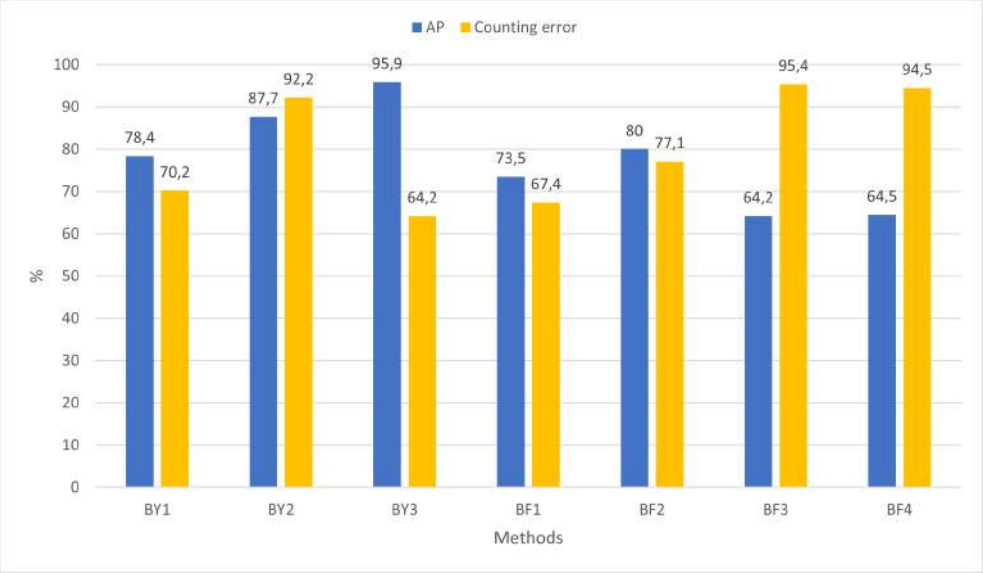

Figure 6.5 – Bar graph with AP and counting error obtained by each method for the *Bedbug* dataset.

(BF2 mAP is superior to BF1 mAP; and BF4 mAP is superior to BF3 mAP). Thus, we can conclude that in this case the optimization of the anchor showed to improve the performance of the methodology and that the best architecture was the VGG. To reduce the sampling of feature maps, after the convolutional layers, pooling layers are applied, thus allowing to reduce the image and feature map dimensions. The Inception Resnet v2 architecture is about 164 layers deep while the VGG16 architecture is 16 layers deep. The small objects are represented by a few pixels, and having the bedbug relative scale of 0.019, their characteristics extracted in the initial layers end up being eliminated and do not reach the detection and steps.

Regarding the counting error, we can verify that it is greater than 60% in all the methods, which reveals that is, in this dataset, the methods were not effective in performing the counting. BY3 was also the method that had the lowest counting error (64.2%) despite being a very high value.

The Figure 6.6 illustrates the predictions of three methods by YOLO for four images, (a), (b), (c) and (d). The first row of images are the ground truth. The predicted bedbug count for each image is in the upper left corner of each image. In green is

represented the ground truth bounding box and in red is the prediction. The BY1 model correctly predicted images (b) and (c), correcting the detection and counting of all bedbugs. Images (a) and (d) contain 52 TP, 2 FP and 154 FN. The BY2 model just correctly predicted image (b). Images (a), (c) and (d) have 13 TP, 0 FP and 200 FN. The BY3 model correctly predicted images (b) and (c). Images (a) and (d) have 63 TP, 5 FP and 201 FN. Analyzing the predicted images, we can conclude that the methodologies are more effective for images with fewer bedbugs.

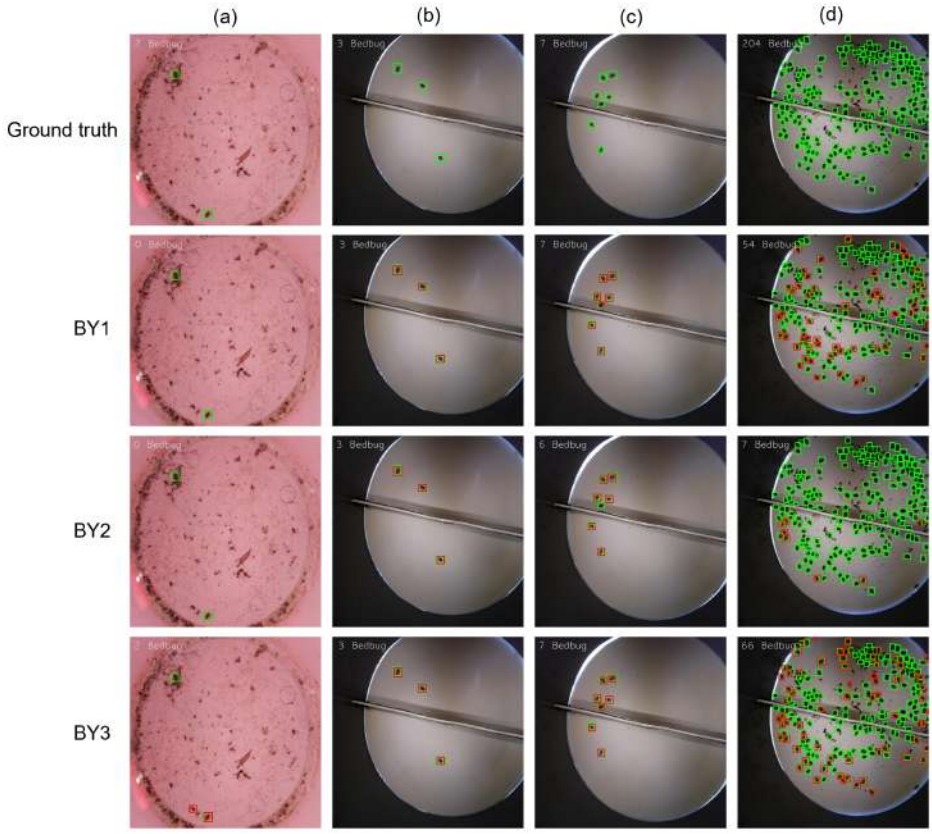

Figure 6.6 – Example of predictions from four *Bedbug* images. The first row is the ground truth, and each row are the prediction of BY1, BY2 and BY3. In the predicted images, green is represented the ground truth bounding box, and red is the prediction.

The Figure 6.7 illustrates the predictions of four methods by faster R-CNN for the same images analyzed above, (a), (b), (c) and (d). We can conclude that methodologies with the Faster R-CNN generally performed poorly. The models that used the VGG (BF1 and BF2) had more TP, while the models that used the

CNN Inception ResNet V2 (BF3 and BF4) had a greater number of FP and FN.

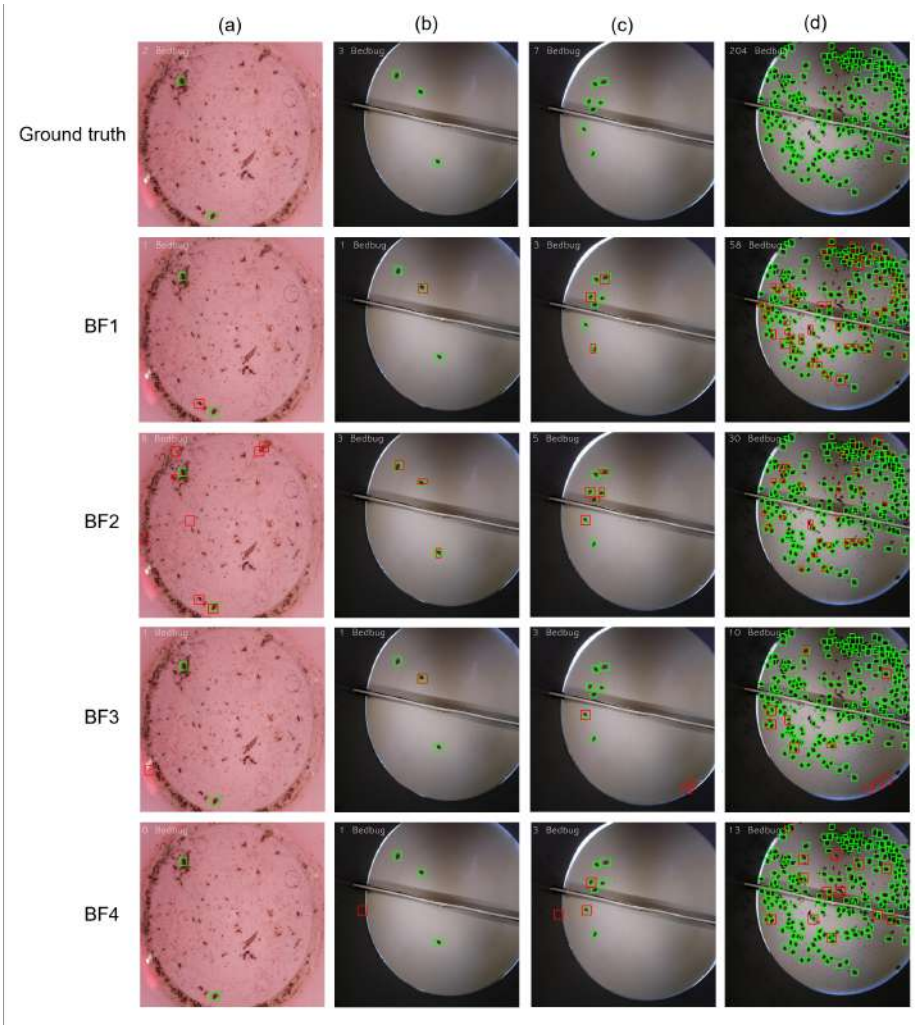

Figure 6.7 – Example of predictions by Faster R-CNN methods from four *Bedbug* images. The first row is the ground truth, and each row are the prediction of BF1, BF2, BF3 and BF4. In the predicted images, green is represented the ground truth bounding box, and red is the prediction.

Model BF1 did not correctly predict detection and count on any image. Images (a), (b), (c) and (d) contain 30 TP, 33 FP and 186 FN. The BY2 model just correctly predicted image (b). Images (a), (c) and (d) have 31 TP, 25 FP and 195 FN. The BF3 model did not correctly predict detection and count on any image. Images (a), (b), (c) and (d) have 4 TP, 11 FP and 212 FN. And, the BF4 model also did not correctly predict detection and count on any image. Images (a), (b), (c) and (d)

have 4 TP, 13 FP and 212 FN.

6.3 Grape moth detection and counting

First, we apply YOLOv5 (GY0) to dataset D1. Despite this dataset being of smaller dimensions, the model performed well, obtaining 95.2%, 91.3% and 89.2% of mAP, recall and precision, respectively. Next, we use the knowledge learned by the network to predict the D2 images. In the figure, the first column corresponds to D2 images predicted by the model, and the second column corresponds to the image with the respective annotation after correcting and verifying its veracity. As seen in the Figure 6.8, we had to correct or add a few annotations, and the model could identify most of the grape moths. This type of approach made the annotation process much faster.

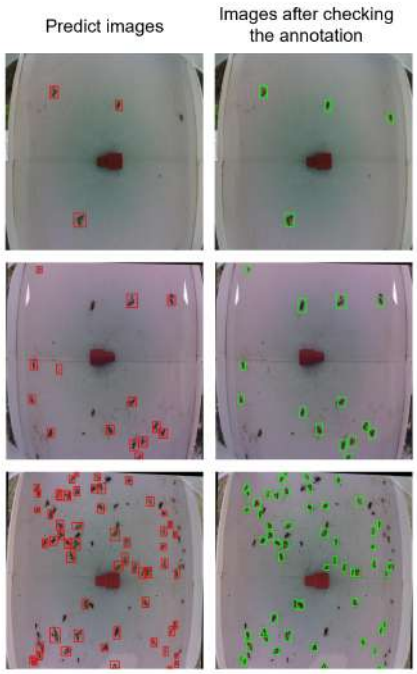

Figure 6.8 – Example of three images predicted by the GY0 method (first column) and the same image after checking the annotation.

After all the D2 annotated and verified, we were ready to train the dataset with all the experiments explained above. The *Grape moth* results are summarised in the Table 6.4 and graph in the Figure 6.9. The best results are in bold. The methodology that obtained the best performance was the GY2, the detection of the *Grape moth* with YOLOv5 model using tuning hyperparameters for *Pest24*. Therefore, the differences between the models trained with YOLO are not relevant. However, compared to methods that used Faster R-CNN, there are many discrepancies. We can check this in the count error metric: the methods with YOLO have values less than 12%, and Faster R-CNN has values between 62% and 79%. Therefore, there is a huge difference in the performance of these two architectures.

Metric	GY1	GY2	GY3	GF1	GF2	GF3	GF4
AP	90.3	**91.5**	86.6	78.3	75	78	54.5
Counting error	10.8	**5.6**	11.6	72.4	62.9	78.7	62.1

Table 6.4 – Results obtained by each method for the *Grape moth* dataset.

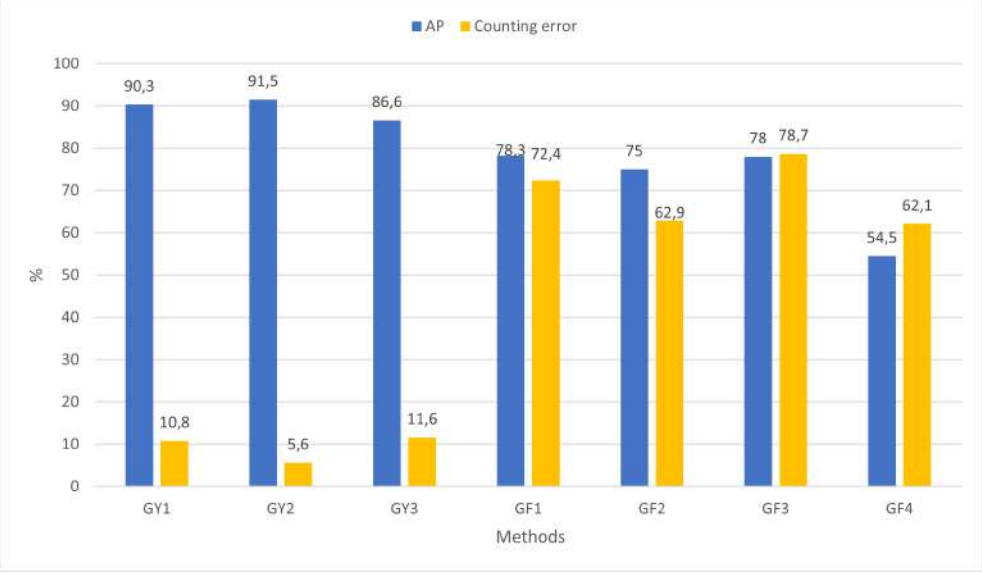

Figure 6.9 – Bar graph with AP and counting error obtained by each method for the *Grape moth* dataset.

Analyzing the results of the YOLO, we can verify that the GY3 is the one with lower results than the other, this model was the one that used transfer learning of *Pest24*. We can thus conclude that for this dataset, the use of *Pest24* tuning

hyperparameters was better than transfer learning. We think this is due to the fact that the *Pest24* dataset has several insects with similar characteristics to the grape moth.

Comparing the GY0 model with GY1, which used the same methodology in the D1 and D2 datasets, we can see that the detection results were better for GY0 than for GY1. The D2 images were collected in a trap over several weeks without being replaced, containing images with the sun, different luminosity, shadow, and moths in various positions (since their death is not immediate) and with different characteristics as they deteriorate over time. These characteristics of the dataset may have generated confusion in learning the models.

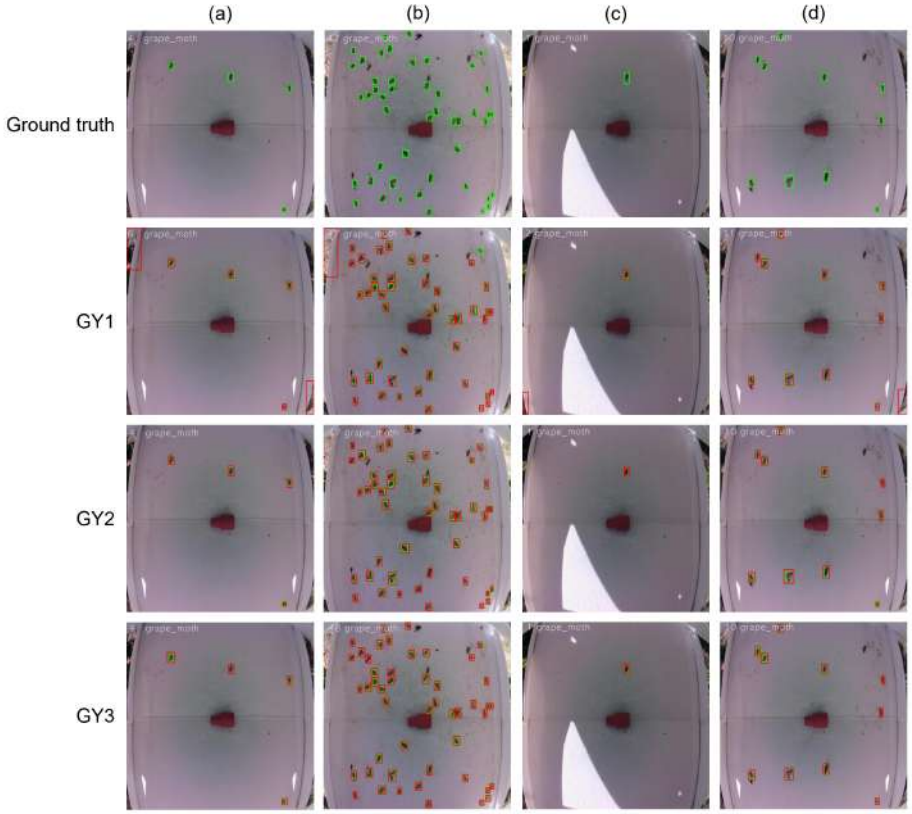

Figure 6.10 – Example of predictions from four *Grape moth* images. The first row is the ground truth, and each row are the prediction of GY1, GY2 and GY3. In the predicted images, green is represented the ground truth bounding box, and red is the prediction.

Analyzing the results of Faster R-CNN, the model that obtained the highest AP

was GF1, detection using the VGG. There is the same trend as in the *Bedbug* dataset, as the methods that used the VGG architecture were the ones that presented the highest results (GF1 and GF2). Although the AP is smaller with the anchor optimization and the counting error are better, i.e. the counting error is smaller.

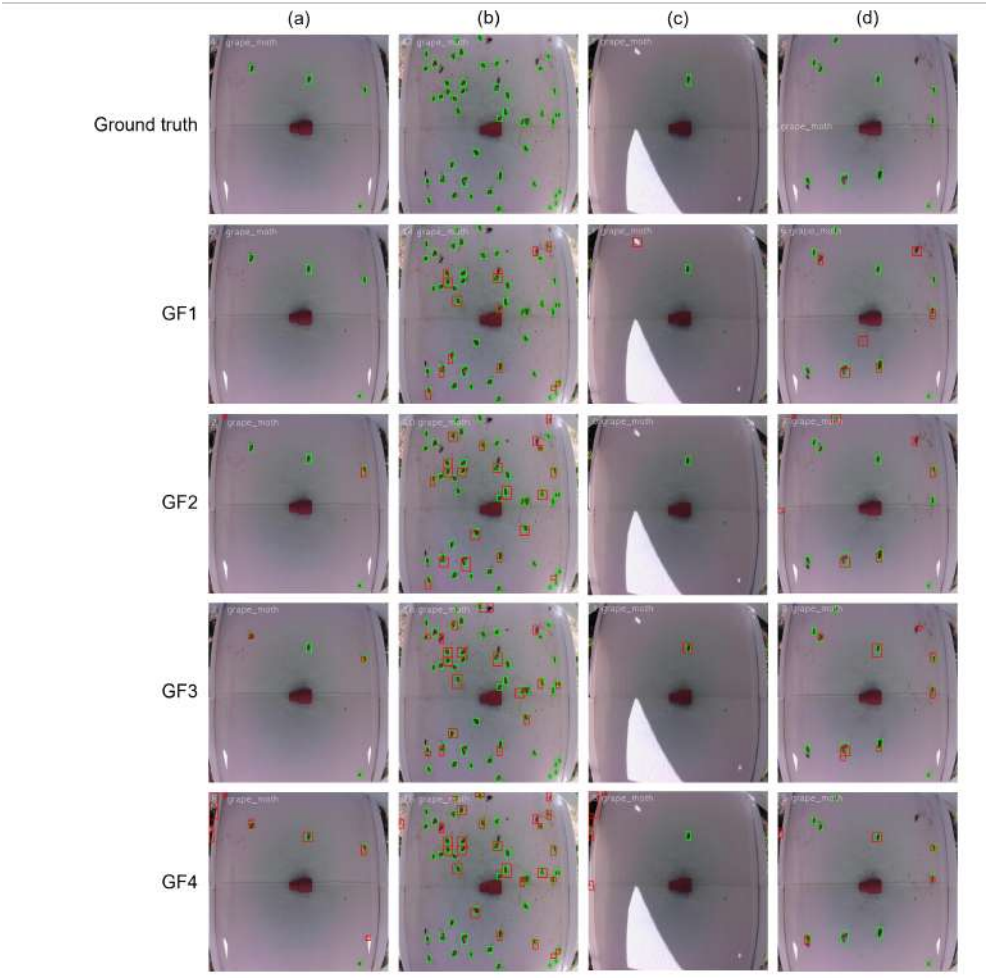

Figure 6.11 – Example of predictions by Faster R-CNN methods from four *Grape moth* images. The first row is the ground truth, and each row are the prediction of GF1, GF2, GF3 and GF4. In the predicted images, green is represented the ground truth bounding box, and red is the prediction.

The Figure 6.10 illustrates the predictions of six methods by YOLO for four images, (a), (b), (c) and (d). The first row of images are the ground truth. The predicted bedbug count for each image is in the upper left corner of each image. In green is represented the ground truth bounding box and in red is the prediction. In general,

YOLO hit all the insects but made some wrong predictions (FP), showing confusion outside the trap. Model GY1 did not correctly predict detection and count on any image. Images (a), (b), (c) and (d) contain 61 TP, 5 FP and 1 FN. The GY2 model correctly predicted all images, correcting the detection and counting of all grape moths. The GY3 model correctly predicted images (a), (c) and (d). Images (b) have 47 TP and 1 FP.

The Figure 6.11 illustrates the predictions of four methods by Faster R-CNN for four images, (a), (b), (c) and (d). Analyzing the predictions by the Faster R-CNN, we found that the methods work is unsatisfactory, and there were few correct detections. No model got all the images right. In model GF1, the images (a), (b), (c) and (d) contain 9 TP, 12 FP and 53 FN. In model GF2, the images (a), (b), (c) and (d) have 12 TP, 17 FP and 50 FN. In model GF2, the images have 10 TP, 22 FP and 52 FN. And, in model GF4, the images contains 25 TP, 19 FP and 37 FN.

6.4 Discussion

Insects are the most biodiverse group of animals. They can present some challenges related to your physical characteristics, such as its size, the similarity between species, the different positions that can have in images and the different morphological characteristics of the same insect. As we know, insects are living beings of reduced dimensions. An image can have a high resolution, being represented by a large set of pixels, or it can be represented by a set of smaller pixels, having a lower resolution. Analyzing the results obtained, the relative scale, that is, the size of the insects in proportion to the image is the factor that exerts the most significant influence on the detection task. As shown in Figure 6.12, the trap includes dozens of insects that are represented with low resolution. We verified that the models are not as effective in images with a high number of insects. So, regular replacement of traps and increase of resolution of the image is encouraged.

Given the incredible biodiversity of insects, there are very similar species; at the

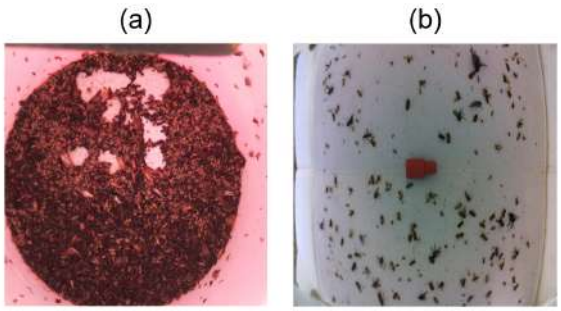

Figure 6.12 – Examples of the high amount of insects in traps and the size of the insects. Image (a) has 111 bedbugs and image (b) has 66 grape moths.

time of image capture, insects of the same species may be in different positions and with different brightness. These characteristics can generate significant challenges in the task of insect detection. For example, in Figure 6.13, the similarity between the three species, Armyworm (idx 5), Bollworm (idx 6) and Yellow tiger (idx 24), three different species with identical morphological characteristics. Also, two examples of different positions of the same insect in the same image can be observed. And, examples of the same grape moth in different lighting conditions could be confused with another insect.

Some images collected in the field associated with SPM systems may present some challenges. Once images are collected in a trap for several weeks without being replaced, containing images with the sun, different luminosity, shadow, and insects in various positions and with different characteristics as they deteriorate over time. This challenge can be solved by carefully choosing the hour when an image is captured, choosing strategic points for the placement of SPM devices, and avoiding areas with trees and shadows. Another characteristic of these systems is the acquisition of very similar images since the collection of the image is continuous, therefore, an algorithm of comparison and exclusion of very similar images was applied. The impact of these challenges was observed in the grape moth dataset since the D2 images were collected in an instrumental trap.

The application of methodologies with DL requires large data sets for model training and thus can achieve human-level results. However, there is a lack of labeled data,

sufficient sizes to datasets, and insects class unbalanced. We used the GY1 model learning to generate the annotations of the remaining *Grape moth* dataset because the remaining data were not annotated. However, there was a need to confirm and validate the generated annotations. Therefore, we believe that semi-supervised is an ideal application for future work. Furthermore, an effective semi-supervised model can perform better than a supervised model. Semi-supervised is an ideal application for when it is not possible to have all the data labelled. Another alternative involves focal loss. Focal loss is beneficial for unbalanced training datasets, being a simple and very effective approach to solving this challenge.

Insect detection is a challenging task in computer vision and raises many challenges. While many methods used for detection give good results for medium and large objects, the performance is not so good when used to detect small objects. Faster R-CNN is a great example of this. This architecture performs well on medium and large objects but poorly on small ones. We found the same, and even with the adaptation of the anchor to the size of the insect, the result was undesired. YOLOv5 performed better as it is a newer architecture, and in the neck uses PAN with FPN, which will improve the propagation of low-level features in the model and increases object location accuracy, even for small objects. So, the use of architectures with multi-scale resource learning improves detection performance. As pointed out by Tong et al. (78), the object context can play an important role in the performance of methodologies. The context in which the object is detected can help improve object detection performance, especially for detecting small objects. Observing the obtained results, we verified that some methods generated predictions in the background of the outside of the trap, so, for future work, we propose the segmentation of traps before detection. Observing the obtained results, we verified that some methods generated predictions in the background of the outside of the trap, so, for future work, we propose the segmentation of traps before detection.

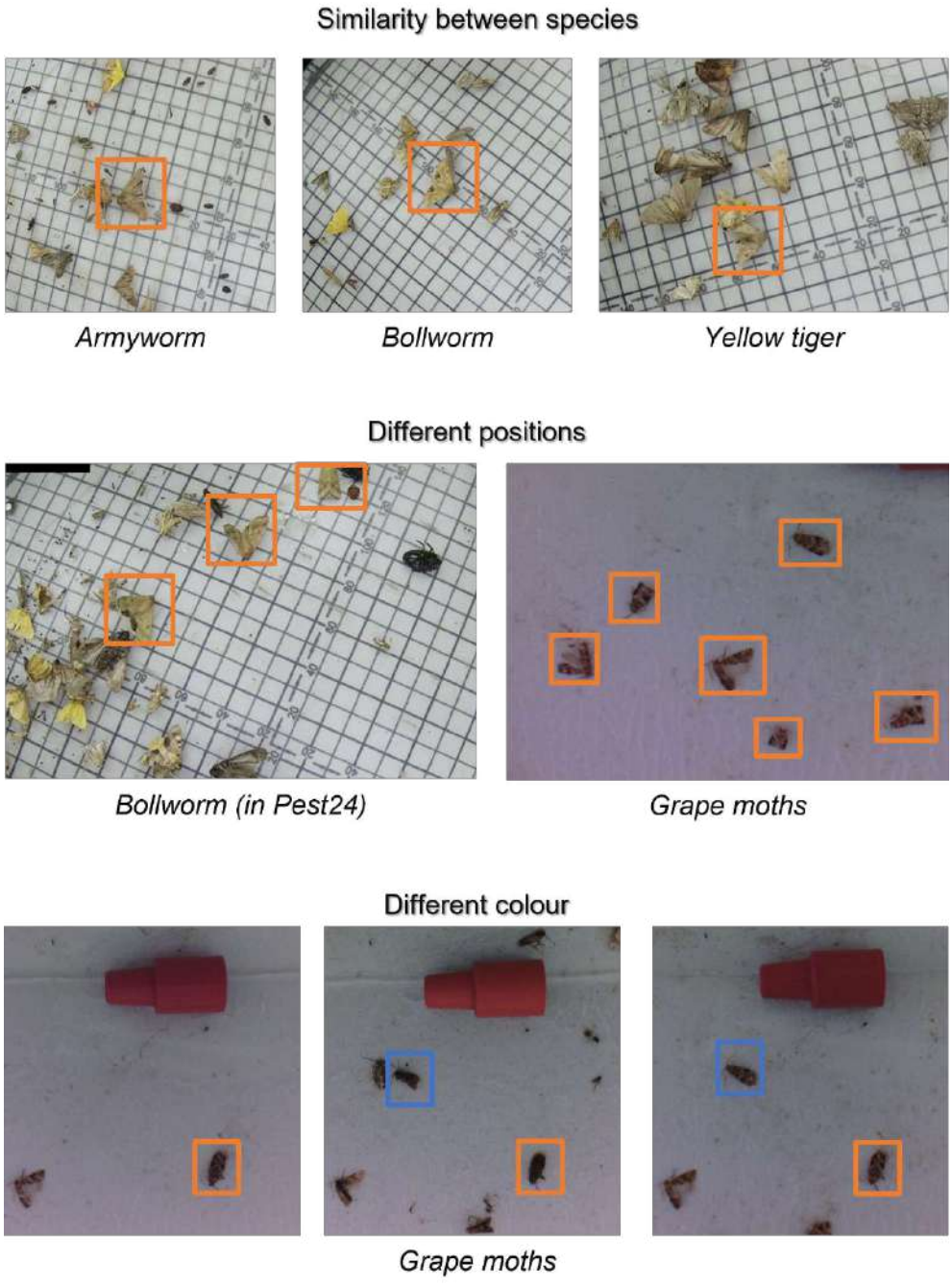

Figure 6.13 – Examples of similarity between species, different positions of the same insect and the different colour of the same insect.

7 Conclusions and Future work

Insect traps are among the most appropriate solution for monitoring and counting, influencing the selection and dosage of the pesticide to be applied for pest control. However, the counting and monitoring operations are based on the frequent visit of technicians to the site and are supported by inefficient counting methods, which is a challenging and time-consuming task. With the improvements of DL technologies, several applications have emerged in the agricultural context, including automatic detection and counting of insects.

7.1 Conclusions

This work presented an exploration in DL models, aimining to count insects in trap images using DL. Our approach was based on two different detection methods, a one-stage detector (YOLOv5) and a two-stage detector (Faster R-CNN). Two datasets were generated from the images provided by the mySense platform. As images are still being collected, these datasets are still small. So, we include in our study the public dataset *Pest24*.

For YOLO, we started by training the *Pest24* dataset with different configurations

to optimise the detection results of *Pest24*. We proposed detection with YOLOv5 with standard hyperparameters and with hyperparameter tuning. The one with the best performance was YOLOv5 with the standard hyperparameters, with an mAP of 72.1%. This result is much higher than state-of-the-art (58.79%). We concluded that hyperparameter tuning was ineffective, which can be justified by this dataset's high variability and complexity. For the *Bedbug* dataset, the best method was YOLOv5 with transfer learning from *Pest24*, reaching an AP of 95.9% and a counting error of 64.2%. For the *Grape moth* dataset, the best method was the YOLOv5 using the hyperparameters tuning of *Pest24* with an AP of 91.5% and a counting error of 5.6%.

For Faster R-CNN, we propose several Faster R-CNN architectures with different configurations to optimize the results obtained. For the *Bedbug* dataset, the best method was the Faster R-CNN with VGG and anchor optimization reaching an AP of 80% and a counting error of 77.1%. For the *Grape moth* dataset, the best method was the Faster R-CNN using VGG with standard anchor, with an AP of 78.3% and a counting error of 72.4%. There is the same trend as in the *Bedbug* dataset, as the methods that used the VGG architecture were the ones that presented the highest results.

From this work, we can conclude that to solve this task, and the Faster R-CNN has a bad performance. On the other hand, the results obtained with YOLO are promising, and it was possible to detect pests with high AP. However, we found that YOLOv5 is not as efficient for images with a high number of insects which had an impact on the counting error, so we encourage frequent replacement of traps to avoid the high number and overlapping insects.

The impact of the number of instances and the relative scale of insects in the AP was also analysed. It shows that the real scale is the factor that has the most impact on the model's performance. This conclusion shows that the task of insect detection is still an open problem since it is a property associated with the morphology of these living beings.

7.2 Future work

For future work, we pretend to train more generations and use multiple GPUs or cloud software for training with the evolution algorithm; we also recommend other data preparation techniques, such as normalisation and increase in image resolution, allowing for greater extraction of insect characteristics with a smaller relative scale.

We identified two key open challenges related to automatic insect detection using DL. The datasets images and the methodologies. For challenges of datasets images, we recommend improving data acquisition, data augmentation, focal loss, and semi-supervised. While for the methodologies, we recommend the multi-scale resource learning and context-based detection. We proposed the segmentation of the traps before the detection task to verify some confusion in the background of the outside of the trap.

As future work, we intend to add the images that have been continuously collected by mySense to our datasets; apply new detection and counting methodologies with DL, and implement the methods developed in the field.

References

[1] R. Kamath, M. Balachandra, and S. Prabhu, "Raspberry pi as visual sensor nodes in precision agriculture: A study," *IEEE Access*, vol. 7, pp. 45 110–45 122, 01 2019. xix, 9

[2] R. Morais. mysense. [Online]. Available: https://mysenseapi.utad.pt/ xix, 11

[3] F. Chollet, *Deep Learning with Python*. Manning Publications Co, 2018. xix, 13, 14, 15, 18, 22

[4] Phung and Rhee, "A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets," *Applied Sciences*, vol. 9, p. 4500, 10 2019. xix, 20

[5] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. xix, 22, 23

[6] T. Kasinathan, D. Singaraju, and S. R. Uyyala, "Insect classification and detection in field crops using modern machine learning techniques," *Information Processing in Agriculture*, vol. 8, no. 3, pp. 446–457,

2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214317320302067 xix, 24, 38

[7] Y. Zhong, J. Gao, Q. Lei, and Y. Zhou, "A vision-based counting and recognition system for flying insects in intelligent agriculture," *Sensors*, vol. 18, p. 1489, 05 2018. xix, 24

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 2015. [Online]. Available: https://arxiv.org/abs/1506.02640 xx, 23, 27, 28, 29, 30

[9] L. Zhu, X. Geng, Z. Li, and C. Liu, "Improving yolov5 with attention mechanism for detecting boulders from planetary images," *Remote Sensing*, vol. 13, no. 18, 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/18/3776 xx, 31, 51

[10] How single-shot detector (ssd) works? [Online]. Available: https://developers.arcgis.com/python/guide/how-ssd-works/ xx, 32

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017. [Online]. Available: https://arxiv.org/abs/1708.02002 xx, 37, 38

[12] Q.-J. Wang, S.-Y. Zhang, S.-F. Dong, G.-C. Zhang, J. Yang, R. Li, and H.-Q. Wang, "Pest24: A large-scale very small object data set of agricultural pests for multi-target detection," *Computers and Electronics in Agriculture*, vol. 175, p. 105585, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169919324123 xx, 40, 41, 42, 44, 45, 46, 58, 59

[13] Food and A. O. of the United Nations, *The State of Food and Agriculture - Innovation in family farming*. Agriculture and Economic Development Analysis Division, 2014. 1

[14] F. Teixeira, *Utilização de Pesticidas Agrícolas*. ACT - Autoridade para as condições do trabalho, 2014. 1, 8

[15] J. S. José Coelho, *Agricultura de precisão*. Produção apoiada pelo Programa AGRO - Medida 7 - Formação Profissional, 2009. 1, 11

[16] M. E. Karar, F. Alsunaydi, S. Albusaymi, and S. Alotaibi, "A new mobile application of agricultural pests recognition using deep learning in cloud computing system," *Alexandria Engineering Journal*, vol. 60, no. 5, pp. 4423–4432, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016821001642 1

[17] E. A. Lins, "Uma metodologia de contagem e classificação de afídeos utilizando visão computacional," Master's thesis, Universidade de Passo Fundo, Passo Fundo, 2018. 2

[18] A. Nieuwenhuizen, J. Hemming, and H. Suh, "Detection and classification of insects on stick-traps in a tomato crop using faster r-cnn," 01 2018. 2, 40, 42

[19] S.-J. Hong, I. Nam, S.-Y. Kim, E. Kim, C. Lee, S. Ahn, I.-K. Park, and G. Kim, "Automatic pest counting from pheromone trap images using deep learning object detectors for matsucoccus thunbergianae monitoring," *Insects*, vol. 12, p. 342, 04 2021. 2, 41, 42

[20] W. Yun, J. P. Kumar, S. Lee, D.-S. Kim, and B.-K. Cho, "Deep learning-based system development for black pine bast scale detection," *Scientific Reports*, vol. 12, 01 2022. 2

[21] Z. Tang, Z. Chen, F. Qi, L. Zhang, and S. Chen, "Pest-yolo: Deep image mining and multi-feature fusion for real-time agriculture pest detection," in *2021 IEEE International Conference on Data Mining (ICDM)*, 2021, pp. 1348–1353. 2

[22] R. Morais, T. Adão, A. Ferreira, J. Mendes, L. Pádua, J. Hruška, J. Sousa, J. Sousa, and E. Peres, "mysense: uma plataforma de iot integradora de dados de múltiplas fontes para aplicações agroflorestais com cariz de polivalência," 10 2018. 3, 12

[23] D. Harris and D. Fuller, "Agriculture: Definition and overview," pp. 104–113, 01 2014. 7

[24] J. Gaffney, J. Bing, P. Byrne, K. Cassman, I. Ciampitti, D. Delmer, J. Habben, H. Lafitte, U. Lidstrom, D. Porter, J. Sawyer, J. Schussler, T. Setter, R. Sharp, T. Vyn, and D. Warner, "Science-based intensive agriculture: Sustainability, food security, and the role of technology," *Global Food Security*, vol. 23, pp. 236–244, 08 2019. 8

[25] C. C. Maria Godinho, "Uso dos pesticidas em portugal. que caminhos?" Quercus Ambiente, Tech. Rep., 2016. 8

[26] D. Oliveira, "Os agrotóxicos e seus efeitos no meio ambiente," Faculdade de Ciências da Saúde, Tech. Rep., 2001. 8

[27] W. Li, T. Zheng, Z. Yang, M. Li, C. Sun, and X. Yang, "Classification and detection of insects from field images using deep learning for smart pest management: A systematic review," *Ecological Informatics*, vol. 66, p. 101460, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S157495412100251X 8

[28] M. Cardim Ferreira Lima, M. E. Damascena de Almeida Leandro, C. Valero, L. C. Pereira Coronel, and C. O. Gonçalves Bazzo, "Automatic detection and monitoring of insect pests—a review," *Agriculture*, vol. 10, no. 5, 2020. [Online]. Available: https://www.mdpi.com/2077-0472/10/5/161 8

[29] D. J. A. Rustia, J.-J. Chao, L.-Y. Chiu, Y.-F. Wu, J.-Y. Chung, J.-C. Hsu, and T.-T. Lin, "Automatic greenhouse insect pest detection and recognition based on a cascaded deep learning classification method," *Journal of Applied Entomology*, vol. 145, no. 3, pp. 206–222, 2021. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/jen.12834 8

[30] M. Preti, F. Verheggen, and S. Angeli, "Insect pest monitoring with camera-equipped traps: strengths and limitations," *Journal of Pest Science*, vol. 94, pp. 1–15, 03 2021. 8

[31] T. Southwood and P. Henderson, *Ecological Methods 3rd edition*, 05 2000. 8, 10

[32] L. Nanni, A. Manfè, G. Maguolo, A. Lumini, and S. Brahnam, "High performing ensemble of convolutional neural networks for insect pest image detection," *Ecological Informatics*, vol. 67, p. 101515, 12 2021. 8

[33] B. Ramalingam, R. E. Mohan, S. Pookkuttath, B. Gómez, C. Borusu, T. Teng, and Y. Tamilselvam, "Remote insects trap monitoring system using deep learning framework and iot," *Sensors (Basel, Switzerland)*, vol. 20, 09 2020. 8

[34] B. C. Gilbert, Nicolas, *Insect trapping guide*, 07 2013. 9, 10

[35] M. Abbas, M. Ramzan, N. Hussain, A. Ghaffar, K. Hussain, S. Abbas, and A. Raza, "Role of light traps in attracting, killing and biodiversity studies of insect pests in thal," *Pakistan Journal of Agricultural Research*, vol. 32, 01 2019. 9, 10

[36] S. Clark, "Pheromone traps for insect pest management," Cornell University, Tech. Rep., 2013. 9, 10

[37] K. R. Steve Dreistadt, Julie Newman, "Sticky trap monitoring of insect pests," University of California Cooperative Extension Farm Advisor, Tech. Rep., 1998. 9, 10

[38] P. Trematerra and M. Colacci, "Recent advances in management by pheromones of thaumetopoea moths in urban parks and woodland recreational areas," *Insects*, vol. 10, no. 11, 2019. [Online]. Available: https://www.mdpi.com/2075-4450/10/11/395 9

[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org. 13, 15, 17, 21

[40] S. Haykin, *Neural networks and learning machines*. Pearson Education, 2009. 14, 15

[41] E. Elgeldawi, A. Sayed, A. Galal, and A. Zaki, "Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis," *Informatics*, vol. 8, 11 2021. 15

[42] F. Beltrão, "Aplicação de redes neurais artificais profundas na deteção de placas pare," Master's thesis, Universidade tecnológica federal do Paraná, 2019. 16, 17, 20, 35

[43] J. Requeijo, *Redes Neurais Artificiais*, 2016. 16

[44] R. Yamashita, M. Nishio, R. Do, and K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, 06 2018. 17, 22, 23

[45] S. Jiang, H. Qin, B. Zhang, and J. Zheng, "Optimized loss functions for object detection and application on nighttime vehicle detection," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 236, no. 7, pp. 1568–1578, jul 2021. 17

[46] F. Gomes, F. Pereira, F. Marins, and M. Silva, "Estudo comparativo entre métodos de otimização de problemas com múltiplas respostas," *Exacta*, vol. 16, 09 2018. 18, 19

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980 18

[48] K. S. Geoffrey Hinton, Nitich Srivastava, "Neural networks for machine learning lecture," Tech. Rep., 2018. 19

[49] S. Ruder, "An overview of gradient descent optimization algorithms," 2016. [Online]. Available: https://arxiv.org/abs/1609.04747 20

[50] J. Pessoa, "Deep learning e redes neurais convolucionais: reconhecimento automático de caracteres em placas de licenciamento automotivo," Master's thesis, Universidade Federal da Paraíba, 2018. 20, 21

[51] R. M. Y. Victor, "Detecção de objetos: estudo e aplicação da arquitetura r-cnn," Master's thesis, Universidade Federal da Paraíba, 2018. 21, 22

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf 22

[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014. [Online]. Available: https://arxiv.org/abs/1409.1556 22

[54] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision, ECCV 2014 - 13th European Conference, Proceedings*, ser. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), no. PART 1. Springer Verlag, 2014, pp. 818–833, 13th European Conference on Computer Vision, ECCV 2014 ; Conference date: 06-09-2014 Through 12-09-2014. 22

[55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014. [Online]. Available: https://arxiv.org/abs/1409.4842 22

[56] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385 22

[57] E. Tetila, B. Brandoli, G. Astolfi, N. A. Belete, W. Amorim, A. Roel, and H. Pistori, "Detection and classification of soybean pests using deep learning with uav images," *Computers and Electronics in Agriculture*, vol. 179, p. 105836, 12 2020. 22

[58] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. 22

[59] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár, "Microsoft coco: Common objects in context," 2014. [Online]. Available: https://arxiv.org/abs/1405.0312 22

[60] Q. Chen, P. Wang, A. Cheng, W. Wang, Y. Zhang, and J. Cheng, "Robust one-stage object detection with location-aware classifiers," *Pattern Recognition*, vol. 105, p. 107334, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320320301370 23

[61] X. Youzi, Z. Tian, J. Yu, Y. Zhang, S. Liu, S. Du, and X. Lan, "A review of object detection based on deep learning," *Multimedia Tools and Applications*, vol. 79, 09 2020. 23, 24

[62] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot MultiBox detector." Springer International Publishing, 2016, pp. 21–37. [Online]. Available: https://doi.org/10.1007%2F978-3-319-46448-0_2 23, 32, 33

[63] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2013. [Online]. Available: https://arxiv.org/abs/1311.2524 24

[64] R. Girshick, "Fast r-cnn," 2015. [Online]. Available: https://arxiv.org/abs/1504.08083 24

[65] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 2015. [Online]. Available: https://arxiv.org/abs/1506.01497 24, 25, 26, 27

[66] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," 2017. [Online]. Available: https://arxiv.org/abs/1703.06870 24

[67] J. Dai, Y. Li, K. He, and J. Sun, "R-fcn: Object detection via region-based fully convolutional networks," 2016. [Online]. Available: https://arxiv.org/abs/1605.06409 24, 40

[68] A. C. Ana Lorena, "Uma introdução às support vector machines," Universidade Federal do ABC, Tech. Rep., 2007. 25

[69] R. B S, "Object detection using region based convolutional neural network: A survey," *International Journal for Research in Applied Science and Engineering Technology*, vol. 8, pp. 1927–1932, 07 2020. 25

[70] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," 2017. [Online]. Available: https://arxiv.org/abs/1705.02950 25

[71] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 2016. [Online]. Available: https://arxiv.org/abs/1612.08242 28

[72] ——, "Yolov3: An incremental improvement," 2018. [Online]. Available: https://arxiv.org/abs/1804.02767 28, 29, 30

[73] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020. [Online]. Available: https://arxiv.org/abs/2004.10934 30

[74] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen, and J.-W. Hsieh, "Cspnet: A new backbone that can enhance learning capability of cnn," 2019. [Online]. Available: https://arxiv.org/abs/1911.11929 30

[75] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 346–361. [Online]. Available: https://doi.org/10.1007%2F978-3-319-10578-9_23 30

[76] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," 2018. [Online]. Available: https://arxiv.org/abs/1803.01534 30, 31

[77] U. Nepal and H. Eslamiat, "Comparing yolov3, yolov4 and yolov5 for autonomous landing spot detection in faulty uavs," *Sensors*, vol. 22, 01 2022. 31

[78] K. Tong, Y. Wu, and F. Zhou, "Recent advances in small object detection based on deep learning: A review," *Image and Vision Computing*, vol. 97, p. 103910, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885620300421 35, 36, 72

[79] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2016. [Online]. Available: https://arxiv.org/abs/1612.03144 36, 38

[80] W. Zhang, S. Wang, S. Thachan, J. Chen, and Y. Qian, "Deconv r-cnn for small object detection on remote sensing images," in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2018, pp. 2483–2486. 36, 38

[81] Y. Ren, C. Zhu, and S. Xiao, "Small object detection in optical remote sensing images via modified faster r-cnn," *Applied Sciences*, vol. 8, p. 813, 05 2018. 36, 38

[82] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," 2017. [Online]. Available: https://arxiv.org/abs/1711.06897 36, 37, 38

[83] M. Ahmad, M. Abdullah, and D. Han, "Small object detection in aerial imagery using retinanet with anchor optimization," in *2020 International Conference on Electronics, Information, and Communication (ICEIC)*, 2020, pp. 1–3. 37, 38

[84] A. Askarzadeh, "A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm," *Computers Structures*, vol. 169, pp. 1–12, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045794916300475 37

[85] Y. He, Z. Zhiyan, L. Tian, Y. Liu, and X. Luo, "Brown rice planthopper (nilaparvata lugens stal) detection based on deep learning," *Precision Agriculture*, vol. 21, 12 2020. 38

[86] M. Qiao, J. Lim, C. W. Ji, B.-K. Chung, H.-Y. Kim, K.-B. Uhm, C. S. Myung, J. Cho, and T.-S. Chon, "Density estimation of bemisia tabaci (hemiptera: Aleyrodidae) in a greenhouse using sticky traps in conjunction with an image processing system," *Journal of Asia-Pacific Entomology*, vol. 11, no. 1, pp. 25–29, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1226861508000071 38

[87] C. Xia, T.-S. Chon, Z. Ren, and J.-M. Lee, "Automatic identification and counting of small size pests in greenhouse conditions with low computational cost," *Ecological Informatics*, vol. 29, pp. 139–146, 2015, special Issue on "Ecosystem Assessment and Management: Selected papers presented at the International Conference on Ecological Informatics 2013, 27-28 June 2013, Seoul, Korea". [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954114001228 39

[88] D. J. Rustia and T.-T. Lin, "An iot-based wireless imaging and sensor node system for remote greenhouse pest monitoring," *Chemical Engineering Transactions*, vol. 58, 01 2017. 39

[89] C. Xie, J. Zhang, R. Li, J. Li, P. Hong, J. Xia, and P. Chen, "Automatic classification for field crop insects via multiple-task sparse representation and multiple-kernel learning," *Computers and Electronics in Agriculture*, vol. 119, pp. 123–132, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169915003282 39

[90] M. Ebrahimi, M. Khoshtaghaza, S. Minaei, and B. Jamshidi, "Vision-based pest detection based on svm classification method," *Computers and Electronics in Agriculture*, vol. 137, pp. 52–58, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016816991631136X 39

[91] S. More and M. Nighot, "Agrosearch: A web based search tool for pomegranate diseases and pests detection using image processing," 03 2016, pp. 1–6. 39

[92] I. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, vol. 2, 11 2021. 39

[93] A. Gutierrez, A. Ansuategi, L. Susperregi, C. Tubío, I. Rankić, and L. Lenža, "A benchmarking of learning strategies for pest detection and identification on tomato plants for autonomous scouting robots using internal databases," *Journal of Sensors*, vol. 2019, pp. 1–15, 05 2019. 39

[94] Y. He, H. Zeng, Y. Fan, S. Ji, and J. Wu, "Application of deep learning in integrated pest management: A real-time system for detection and diagnosis of oilseed rape pests," *Mobile Information Systems*, vol. 2019, pp. 1–14, 07 2019. 39

[95] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," 2016. [Online]. Available: https://arxiv.org/abs/1602.07261 40

[96] W. Li, P. Chen, B. Wang, and C. Xie, "Automatic localization and count of agricultural crop pests based on an improved deep learning pipeline," *Scientific Reports*, vol. 9, 05 2019. 40, 42

[97] Z. Shi, H. Dang, Z. Liu, and X. Zhou, "Detection and identification of stored-grain insects using deep learning: A more effective neural network," *IEEE Access*, vol. 8, pp. 163 703–163 714, 2020. 40, 42

[98] F. Iandola, M. Moskewicz, S. Karayev, R. Girshick, T. Darrell, and K. Keutzer, "Densenet: Implementing efficient convnet descriptor pyramids," 2014. [Online]. Available: https://arxiv.org/abs/1404.1869 40

[99] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-nms – improving object detection with one line of code," 2017. [Online]. Available: https://arxiv.org/abs/1704.04503 40

[100] W. Li, D. Wang, M. Li, Y. Gao, J. Wu, and X. Yang, "Field detection of tiny pests from sticky trap images using deep learning in agricultural greenhouse," *Computers and Electronics in Agriculture*, vol. 183, p. 106048, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169921000661 41, 42

[101] A. R. J. C. C. G. A. L. A. Arias, J. Jiménez, "La chinche del arroz, eysarcoris ventralis (west.), sin. e. inconspicuus (h. sch.), en extremadura: colonización del arroz y estrategias de protección," Tech. Rep., 1998. 47

[102] C. Carlos, *A Traça da Uva*. ADVID - Associação para o Desenvolvimento da Viticultura Duriense, 2007. 49

[103] I. Branco, "Alternative methods in weed managment to glyphosate and other herbicides," Tech. Rep., 2017.

[104] W. Li, P. Chen (), B. Wang, and C. Xie, "Automatic localization and count of agricultural crop pests based on an improved deep learning pipeline," *Scientific Reports*, vol. 9, 05 2019.

[105] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 1, pp. 142–158, 2016.

[106] Bed bugs - niph. [Online]. Available: https://www.fhi.no/en/op/skadedyrveilederen/bedbugs/bed-bugs-/