Democratic and Popular Republic of Algeria

Ministry of Higher Education and Scientific Research.

University of May8,1945-Guelma-

Faculty of Mathematics, Computer Science, and Material Sciences

Department of Computer Science

Master's Thesis

Field: Computer Science

Option: ICT/CS

Theme:

Smart Monitoring of Forest and Agricultural Health Using AI and IoT

Case of Thaumetopoea pityocampa Nest Detection

Presented by: *Aya AMIRI ,Amani DOUAFER*

Jury members:

- **President**:
- **Supervisor**: Dr. KhaledHALIMI
- **Examiner**:
- **Examiner**:

June 2025

Acknowledgments

*First and foremost, we express our heartfelt gratitude to **Allah**, whose blessings and guidance have enabled us to undertake and complete this work.*

We extend our sincere thanks to our supervisor, Mr. Halimi Khaled, for his invaluable guidance, continuous support, and insightful advice throughout this project. Choosing to work under his supervision once again was a deliberate and rewarding decision.

We also wish to thank Mr. Abdelaziz Benkirat, Director of the Professional Pole, for his encouragement and involvement, as well as all the faculty members of the Computer Science Department, whose dedication and expertise greatly contributed to our academic journey.

Our appreciation goes as well to Mr. Karim Amiri for his essential assistance during the fieldwork phase of this study.

To all those who supported and believed in us throughout this experience, we are sincerely grateful.

Dedications

In the name of Allah, the Most Gracious, the Most Merciful.

✧*All praise is due to Allah, who granted me strength, patience, and guidance throughout this journey.*

✧*To my grandfather, You believed in me long before I believed in myself.*
You're not here to see it, but I carried your love and pride with me every step of the way.
Your favorite grandchild — Youyou — did it.

✧*To my parents, Hacene and Karina, Your sacrifices and prayers shaped every step of this path. If I shine today, it's because of you.*

✧*To my sister Ines, Thank you for being my constant joy and support.*

✧*To my dear grandmother, Your prayers have always been my shield. Your love, my quiet strength.*

✧*To my uncle, You've been like a second father to me. Your support has meant the world.*

✧*To Imen, My childhood friend and forever bestie, thank you for growing with me and always believing in me.*

✧*To Amani, My partner through every challenge, we started together, and we finish together. I couldn't imagine doing this without you.*

Aya Amiri

Dedications

❀*To acknowledge the invisible is the first step toward true gratitude*❀

First and foremost, all praise is due to Allah, who granted me the strength, patience, and success to reach this stage of my academic path.

❀I dedicate this thesis to everyone who values knowledge. I am truly grateful to be surrounded by outstanding individuals in their fields who constantly encourage learning.

❀To my mother, my first teacher, the reason I fell in love with mathematics, and my inspiration.

❀To my father, a true lover of knowledge, who still studies to this day. Your dedication and excellence are unmatched.

❀To my sister, the distinguished doctor, always the first, always the example.

❀To Aya, my right hand, whose presence, support, and collaboration were essential to the completion of this work.

Amani Douafer

Abstract

With the increasing spread of harmful species such as the pine processionary caterpillar (Thaumetopoea pityocampa), which poses a threat to biodiversity and ecological balance, the monitoring of forest health has become a critical concern. Traditional detection methods are often expensive, time-consuming, and unsuitable for large-scale or real-time surveillance.

To overcome these limitations, an AI-based system is introduced that combines computer vision techniques with geolocation. The system employs a YOLOv11 object detection model enhanced by the Multi-Scale Patch Analysis (MSPA) method, which improves the detection of small or partially visible nests by generating image patches at multiple scales.

In addition to accurate nest detection, the system extracts GPS metadata from the captured images when available to enable spatial localization. This allows for the visualization of detected nests on an interactive map, supporting environmental monitoring and spatial analysis.

Experimental results demonstrate high detection performance, achieving a mAP@0.5 of 98.4%. This integrated approach represents a promising solution for automated forest surveillance and may be extended to applications in agriculture, ecology, and environmental management.

Keywords: Automatic detection, Geolocation, Computer vision, YOLOv11, Forests, *Thaumetopoea pityocampa*, Multi-Scale Patch Analysis (MSPA), Artificial Intelligence.

Résumé

Avec la propagation croissante d'espèces nuisibles telles que la chenille processionnaire du pin (Thaumetopoea pityocampa), qui menace la biodiversité et l'équilibre écologique, la surveillance de la santé des forêts devient une priorité. Les méthodes de détection traditionnelles s'avèrent coûteuses, lentes et peu adaptées à une surveillance à grande échelle ou en temps réel.

Pour pallier ces limitations, un système intelligent basé sur la vision par ordinateur et l'intelligence artificielle est proposé. Ce système repose sur le modèle de détection YOLOv11, optimisé par la méthode Multi-Scale Patch Analysis (MSPA), qui améliore la détection des nids de petite taille ou partiellement visibles grâce à la génération d'images en sous-parties à différentes échelles.

En complément de la détection automatique, le système exploite les métadonnées GPS contenues dans les images capturées, lorsque celles-ci sont disponibles. Cela permet d'associer chaque détection à une position géographique réelle et de visualiser les résultats sur une carte interactive, facilitant l'analyse spatiale et la surveillance environnementale.

Les résultats expérimentaux montrent une performance élevée avec un mAP@0.5 de 98,4 %. Cette approche intégrée représente une solution prometteuse pour la surveillance automatisée des forêts, avec des perspectives d'extension vers d'autres domaines tels que l'agriculture et la gestion environnementale.

Mots-clés : Détection automatique, Vision par ordinateur, YOLOv11, Forêts, Thaumetopoea pityocampa, Multi-Scale Patch Analysis (MSPA), Intelligence Artificielle.

الملخص:

مع الانتشار المتزايد للأنواع الضارة مثل يرقة دودة الصنوبر (Thaumetopoea processionnaire pityocampa)، والتي تُشكّل تهديدًا للتنوع البيولوجي والتوازن البيئي، أصبحت مراقبة صحة الغابات أولوية بيئية ملحّة. تُعتبر الطرق التقليدية للكشف عن هذه الآفات مكلفة وبطيئة وغير ملائمة للرصد الواسع النطاق أو في الوقت الحقيقي.

تم اقتراح نظام ذكي يعتمد على تقنيات الرؤية الحاسوبية والذكاء الاصطناعي لرصد أعشاش هذه الآفة بشكل تلقائي. يعتمد النظام على نموذج YOLOv11 المحسّن باستخدام تقنية تحليل البقع متعددة المقاييس(MSPA) ، والتي تسمح بتوليد صور فرعية على مقاييس مختلفة، مما يُحسّن اكتشاف الأعشاش الصغيرة أو الظاهرة جزئيًا.

كما يستفيد هذا النظام من بيانات تحديد الموقع الجغرافي (GPS) المدمجة في الصور الملتقطة، عند توفرها، لربط كل كشف بموقع جغرافي دقيق. يتم بعد ذلك عرض النتائج على خريطة تفاعلية تسهّل التحليل المكاني والمتابعة البيئية .أظهرت التجارب دقة أداء عالية، حيث بلغ متوسط الدقة (mAP@0.5) حوالي 98.4٪. وتُعدّ هذه المقاربة حلاً واعدًا للمراقبة الآلية للغابات، مع إمكانية توسيع استخدامها لتشمل مجالات أخرى مثل الزراعة وإدارة البيئة.

الكلمات المفتاحية:

الكشف التلقائي، تحديد الموقع، الرؤية الحاسوبية، YOLOv11، الغابات، يرقة الصنوبر Thaumetopoea pityocampa، تحليل البقع متعددة المقاييس(MSPA) ، الذكاء الاصطناعي

Contents

List of Figures

List of Tables

General Introduction

Forest ecosystems play a vital role in maintaining environmental balance, preserving biodiversity and mitigating climate change. However, these vital environments are under increasing threat from human activities and invasive species. One of the most destructive insects for coniferous forests, particularly pine trees, is the pine processionary caterpillar (Thaumetopoea pityocampa). Its larvae cause massive defoliation by weakening trees and also pose a danger to human and animal health due to their stinging hairs.

Traditionally, the detection of processionary nests was carried out manually by forestry experts. While this method can be effective on a small scale, it is very time-consuming and unsuitable for large-scale monitoring. The emergence of artificial intelligence (AI) and computer vision (CV) has opened up the possibility of automated solutions for environmental monitoring. Object detection models such as YOLO (You Only Look Once) enable real-time detection and can effectively locate processionary nests over large forest areas.

This thesis presents a deep learning based approach for detecting Thaumetopoea pityocampa nests, using the YOLOv11 model. In order to overcome the difficulties associated with detecting small or distant nests, a specific pre-processing method called MSPA has been integrated. A customized dataset was created from annotated images from various sources, including photos taken in the field using GPS-equipped devices. This wealth of data, both in terms of quality and geography, enabled a more detailed spatial analysis.

The final system was integrated into an interactive platform accessible to users, whether they are forest rangers, researchers, or ordinary citizens, allowing them to submit images or videos and view the results, including the location of nests on a map. The model's performance was evaluated using several metrics and compared to that of other state-of-the-art approaches.

Our goal is to contribute to forest health monitoring by demonstrating how artificial intelligence can be used for sustainable environmental management, particularly by facilitating early pest detection and promoting rapid intervention.

This thesis is organized as follows:

Chapter 1 : Introduction to Forest Health and AI Methodologies

This chapter discusses the importance of forest ecosystems and the threats they face, with a particular focus on those caused by the pine processionary caterpillar (Thaumetopoea pityocampa). It outlines the limitations of traditional nest detection methods and puts forward artificial intelligence (AI) and computer vision as promising solutions for monitoring forest health.

Chapter 2 : Deep Learning Approaches for PPM Nest Detection

It reviews deep learning models used for object detection, including YOLO, SSD, and Faster R-CNN, with a focus on the YOLOv11 model adopted in this study. It addresses the challenges of detecting small or distant nests and shows how current models can effectively address them.

Chapter 3 : System Design and Model Development

It describes the system architecture and the dataset preparation process. It also presents the multi-scale patch analysis (MSPA) technique, which improves detection accuracy, details the collection, annotation, and preprocessing of data, as well as the customization and training of the YOLOv11 model.

Chapter 4 : Implementation and Results

This chapter describes how the detection system has been implemented within an interactive platform. The platform enables users to upload images or videos and view detections on a map. The chapter also describes the development environment, evaluation metrics, and experimental results. Comparisons are provided to demonstrate the model's effectiveness ,and concludes with a discussion of the system's limitations and potential future improvements, including integration with drones.

Chapter 1

Introduction to Forest Health Monitoring and AI Approaches

1.1 Introduction

1.1.1 Forest health monitoring

Forest health monitoring enables landowners and forestry organizations to gain a deeper understanding of the forest biome's condition and to detect potential threats, such as those that could fuel wildfires. Effective monitoring should be capable of identifying changes that indicate more serious issues, such as droughts or tree diseases, and alerting forest managers. While many government-run programs track trends on a larger scale, local monitoring is often conducted to better address immediate concerns [1].

Forest managers may be particularly concerned with the following key elements of forest health:

- Tree density
- Tree species
- Tree size
- Tree health
- Acres under management
- Forest growth rate
- Ecosystem diversity
- Wildlife habitats
- Air quality

1.1.2 Methods of forest health monitoring

In general, there is no magic bullet when it comes to forest monitoring, forest managers will use a variety of methods to guarantee a healthy forest. When combined, the following techniques ought to produce reliable forest health monitoring.

- **Field inspections:** As a sort of spot check, forest managers will physically spend time in the forest gathering samples and documenting visual data to provide a more comprehensive picture of the health of the forest. Since this approach depends on managers, rangers, or wardens physically conducting inspections, it can be time consuming and challengingto cover large areas, but it is useful for obtaining a detailed look at the forest and identifying specific issues.

- **Aerial surveys:** A forest manager can gain a good view over a large area by using drones to examine the forest canopy ,they can gather data and are quite good at mapping and surveying terrain when combined with infrared technology. Even though aerial surveys are always improving, they are not as precise as being able to observe something on foot.

- **Remote sensing technique:** Although they are still in their infancy, remote technologies like satellites for collectingforest data are advancing quickly. Although remote sensing can swiftly cover wide regions and yield detailed information,it might be costly and might not be able to identify some problems. Wireless sensor networks, or IoT sensors, are another remote method for monitoring forest health, but they are frequently more useful. Using a long-range radio network,a network of sensors gathers data on temperature, humidity, and natural gasses and transmits i nformation and alerts inreal time. Because sensors requirerelatively little maintenance over a period of 10 to 15 years, forest health monitoring is now quick and easy [2].

1.2 Impact of Thaumetopoea pityocampa

1.2.1 Pine Processionary Moth Life Cycle

Thaumetopoea pityocampa, the pine processionary moth, is a pest that feeds on the needles of pine trees, causing heavy defoliation. It is endemic to the Mediterranean and southern Europe but has spread to other parts of the world. The larvae are well known for their "processions" when they move in lines to find suitable places for pupation.

Figure 1. 1:Thaumetopoea pityocampa [3]

The moth poses both ecological and economic threats by weakening trees and exposing them to other pests and diseases. Adult moths have a lifespan of just one day during summer, during which they mate and deposit eggs on pine trees. The caterpillars, hatched from these eggs, begin consuming the needles of the trees during autumn.

In mid-January, they build impressive white silken nests, about the size of a football, in the pine tree foliage and branches. There may be many nests in a single tree. The remainder of the winter is spent in these nests high in the trees; they occupy the nests during the day and venture out at night to feed on the needles [4].

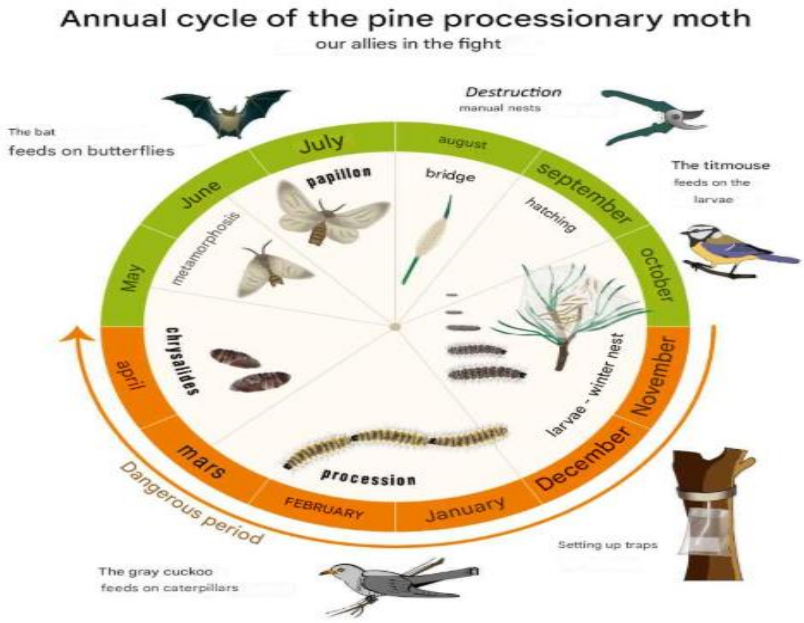


Figure 1. 2:Life cycle of PPM [5]

1.2.2 PPM Damage Symptoms

Pine processionary moth caterpillars feed on pine needles and some other species of conifer trees, and in severe infestations cause severe defoliation of trees. This can stress the trees to be more vulnerable to attack from other insects or disease, and to environmental stresses such as flood or drought. PPM caterpillars have thousands of tiny hairs that carry an urticating, or irritating, protein called thaumetopoein, which accounts for its scientific name. Upon contact with humans and animals, these hairs can cause painful eye, skin and throat irritations and rashes and, in some exceptional cases, allergic reactions.

Figure 1. 3:Human Reactions to Thaumetopoea pityocampa Hairs [6]

1.2.3 PPM Nest Characteristics

The most noticeable indication of the presence of pine processionary moth is the large, white, silken nests that the Caterpillars spin high in the trees during January. They construct these nests amongst the pine leaves and can grow as large as a football. These are the winter refuges of the caterpillars, where they huddle together to keep warm and safe while still eating and developing. The nests are generally placed at tip ends of branches, especially towards the top canopy of the tree, and therefore seen from a distance but less easy to study up close [7].

Figure 1. 4:Pine Processionary Moth Nest [8]

1.2.4 Manual Detection Methods for PPM Nests

Traditional PPM infestation detection relies to a great extent on ground-level visual surveys. Scanners walk along wooded areas, checking individual trees for signs of PPM activity, such as the presence of silk nests constructed by wintering larvae. Such nests are often found on the periphery parts of host plants, often at the top of branches in the higher crown. While this method allows for direct observation, it is laborious and may not effectively cover broad or heavily wooded areas [9].

1.2.5 Limitations and Challenges

Manual techniques of PPM infestations detection are confronted with several limitations. They are extremely time-consuming and labor-intensive as people need to walk through the forests to inspect every tree individually, especially in dense or large forests where the nests are hard to find. The nests are habitually hidden deep in the trees or among the branches, hence hard to find. Weather is also a problem rain, snow, or fog can make visibility and inspection impossible. There is also the risk of human error, where the inspector might miss a nest or get it wrong. Because the process is slow, infestations may spread significantly before they are even noticed.

To overcome the many limitations of manual detection methods, researchers and practitioners have increasingly turned to automated solutions powered by computer vision and deep learning. These advanced technologies offer a scalable, accurate, and time-efficient alternative to traditional inspection. By analyzing images captured from drones or ground-based cameras, computer vision systems can rapidly detect pine processionary moth nests with high precision, even in challenging environments or at great heights. This shift from manual to automated methods represents a significant advancement in forest health monitoring, enabling early detection, faster response, and more efficient management of PPM infestations.

1.3 Computer Vision Introduction

1.3.1 Computer Vision Overview

Computer Vision is a field of Deep Learning and Artificial Intelligence where human beings train computers to see and interpret the world they live in. While humans and animals automatically solve vision as a problem even at a very young age, helping machines see and interpret their surroundings through vision is a large unsolved problem. Limited view of the human eye and the infinitely changing landscape of our dynamic world is what renders Machine Vision challenging at its very core.

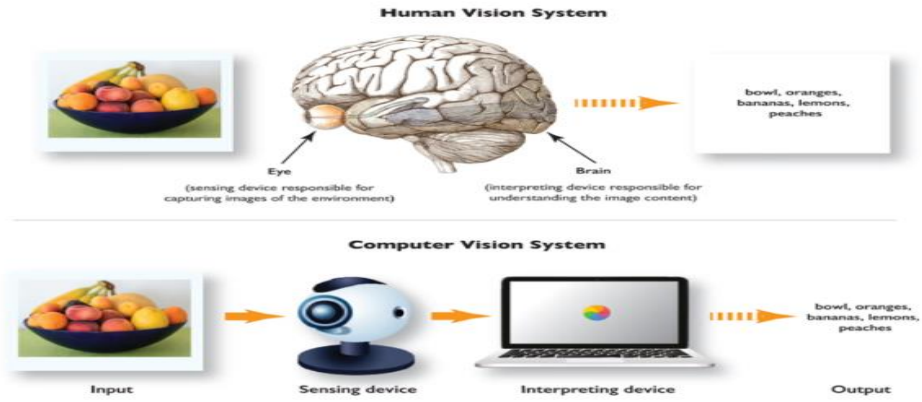


Figure 1. 5:Human vision system VS cv system [10]

1.3.2 Common Computer Vision Tasks

Computer vision assignments are basically making computers understand digital images and also visual data from the actual world. This could involve extracting, processing, and analyzing data from such types of inputs for decision-making. The past of machine vision consisted of formalizing tough problems on a grand scale into well-liked solvable problem statements. Splitting subjects into well-organized groups with nice naming conventions helped researchers around the globe to identify problems and resolve them efficiently. Some of the most common computer vision tasks in AI today include image classification, object detection, and image segmentation, among others.

Figure 1. 6:The most common cv tasks [11]

Computer vision tasks	Definition	Common Models
Image Classification	Image classification tasks involve CV models classifying images into user-defined classes for various applications.	BLIP, ResNet, VGGNet
Object Detection and Localization	While image classification categorizes an entire image, object detection and localization identify specific objects in an image.	Faster R-CNN, YOLO, SSD
Semantic Segmentation	Semantic segmentation tries to label each pixel in an image for a finer classification. The approach gains more classification accuracy by labeling the individual pixels of an object.	FastFCN, DeepLab, U-Net

Table 1. 1:Common Computer vision tasks

1.4 Related Works

Ref	Year	Title	Approach	Data Type	Dataset	Accuracy/FPS	Results
[12]	2023	Testing Early Detection of Pine Processionary Moth (Thaumetopoea pityocampa) Nests Using UAV-Based Methods	Object detection using YOLOv5 and Faster R-CNN,	RGB UAV imagery	Custom dataset (forests in Catalonia)	YOLOv5 0.826 (presence/absence) 0.696 (per nest) Faster R-CNN: slightly lower	Best performance with high-altitude RGB images
[13]	2019	Detection and Mapping of Pine Processionary Moth Nests in UAV Imagery Using Semantic Segmentation	Semantic segmentation using U-Net CNN	RGB UAV imagery	Custom dataset		Accurate mapping Of nests
[14]	2022	Eco-Friendly Fight Against Thaumetopoea pityocampa Infestations in Pine Forests Using Deep Learning on UAV Imagery	Object detection using YOLOv5	RGB UAV imagery	Custom dataset	/	Detection was robust and reliable Suitable for real-time field deployment
[15]	2023	Deep-Pest-Detector:	CNN with	RGB +	Multimodal	97%	Real-time

		Automated Detection and Localization of Processionary Moth Nests via Aerial Drones and DNN	RGB + Thermal image fusion	Thermal UAV imagery	dataset collected during field surveys		detection enabled by onboard drone processing
[16]	2023	Palm Tree Disease Detection Using Residual Networks	ResNet and transfer learning with Inception ResNet	RGB images of palm leaves	Custom-collected dataset	high classification accuracy	Effectively identified palm leaf diseases using deep learning
[17]	2020	VddNet: Vine Disease Detection Network Based on Multispectral Images and Depth Map	Custom CNN architecture (VddNet)	Multispectral and depth images	UAV-collected imagery	/	High precision in detecting and localizing vine diseases
[18]	2023	Deep Learning-Based Trees Disease Recognition and Classification	Deep CNN model for tree leaf disease classification	High-resolution RGB images of tree leaves	Custom dataset	high accuracy	Accurate and efficient classification of leaf diseases
[19]	2021	Automated Detection of Olive Tree Diseases Learning and Drone Imagery	YOLOv4 for object detection and classification	RGB UAV imagery	Custom dataset	90%	Enabled early detection of olive diseases
[20]	2020	Aerial Spectral Imaging	CNN	Hyperspectral UAV	Field data from	85%	Detected early signs

and CNNs for Early Detection of Ash Dieback in Forests	imagery	European ash forests	of ash dieback -Combined spectral bands improvedclassification accuracy compared to RGB-only models

Table 1. 2:Overview of UAV-Based Deep Learning Methods for Tree Pest and Disease Detection

The review of related works highlights the growing shift toward automated approaches powered by deep learning models integrated into UAV-based systems. Whether through object detection, semantic segmentation, or multimodal image fusion, these methods are all grounded in the principles of artificial intelligence. It is the advancements in AIparticularly in machine learning (ML) and deep learning (DL)that have made forest monitoring and pest detection faster, more accurate, and more adaptable to complex conditions.

1.5 Artificial Intelligence Overview

In today's world of rapidly developing technology, one must keep up with advancements in artificial intelligence (AI), machine learning (ML), and deep learning (DL). As per Mark Cuban, a famous American businessman and television personality: "*Artificial Intelligence, deep learning, machine learning whatever you're doing if you don't understand it learn it. Because otherwise, you're going to be a dinosaur within 3 years*." This quote underscores the importance of continual learning in these cutting-edge fields [21].

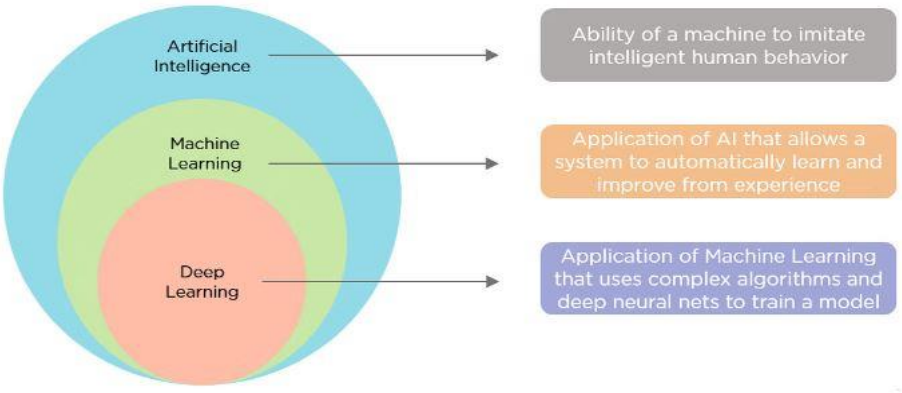


Figure 1. 7:AI and Its Subdomains [21]

As illustrated in Figure 1.8, the relationship between Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) is hierarchical. AI encompasses all techniques that enable machines to mimic human intelligence. ML is a subset of AI focused on learning from data, while DL is a further specialization of ML that leverages deep neural networks to solve complex tasks.

1.5.1 Artificial Intelligence (AI)

Artificial intelligence, simply referred to as AI, is the process of providing data, information, and human intelligence to machines. Artificial Intelligence's main goal is to develop independent machines that can think and act like human beings. The machines can replicate human behavior and carry out tasks through learning and problem solving. The majority of the AI systems replicate natural intelligence to perform complex problems.

"*AI doesn't have to be evil to destroy humanity – if AI has a goal and humanity just happens in the way, it will destroy humanity as a matter of course without even thinking about it, no hard feelings*". Elon Musk, Technology Entrepreneur, and Investor[21].

- **Types of Artificial Intelligence**

  - **Reactive Machines**: Such machines only react. These machines don't create memories, and they don't utilize any experience from the past to make new decisions.

  - **Limited Memory**: The past serves as a guide for these machines, and there is some information accumulated over time. The information utilized is short-term.

  - **Theory of Mind:** These are systems that comprehend human emotions and their influence on decision making. They are trained to adapt their behavior accordingly.

  - **Self-awareness:** These systems are programmed and designed to be self-aware. They have knowledge of their own internal states, forecast other people's emotions, and respond accordingly.

1.5.2 Machine Learning (ML)

Is a field of Artificial Intelligence (AI) that addresses the improvement of AI systems' accuracy using large amounts of data. The data can be in the form of images, messages, documents, or even patterns of human behaviors. The ML algorithms take the data in an attempt to predict or make decisions about future events. The broad types of machine learning are supervised learning, unsupervised learning, and reinforcement learning.

- **Types of Machine Learning**

  - **Supervised learning**: is the process where models are trained from labeled data and inputs are associated with known outcomes. Supervised learning is often used for classification and prediction activities, such as image recognition or spam detection.

  - **Unsupervised learning:** is using unlabeled data, and the model determines patterns or groupings by itself. It's commonly used in clustering or dimension reduction, for instance, customer segmentation.

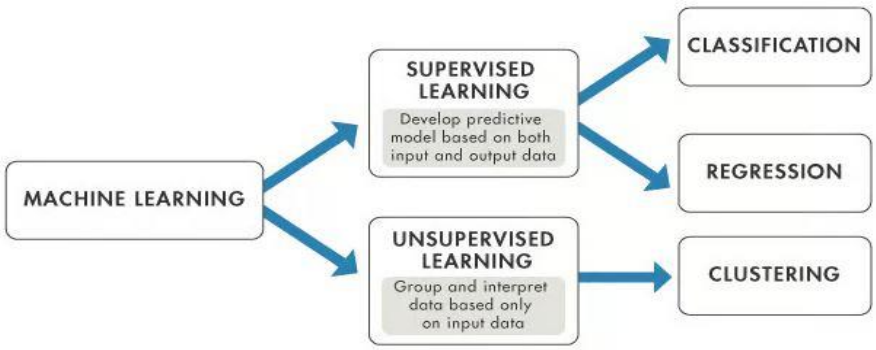


Figure 1. 9:Comparing supervised and unsupervised learning [22]

  - **Reinforcement learning**: relies on learning through interacting within an environment. The model, or agent, receives reward or penalty based on its action and learns to choose actions in order to maximize cumulative reward. It's commonly used in robotics and playing games.

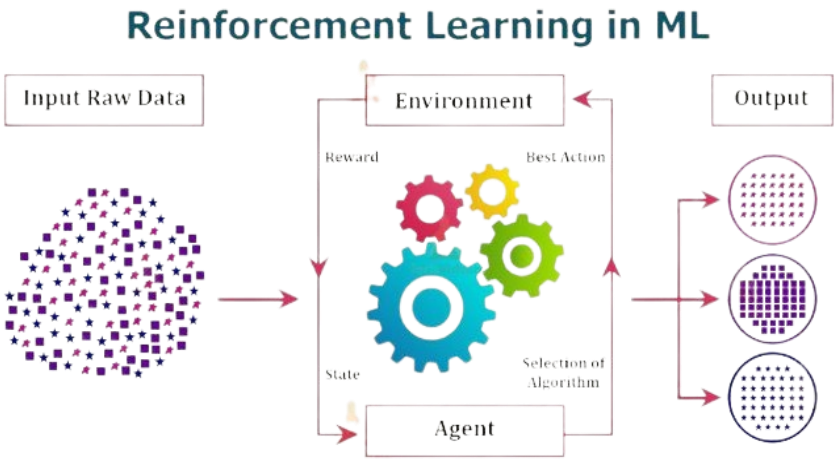


Figure 1. 10:Reinforcement learning processing [23]

1.5.3 Deep Learning (DL)

DL is a specialized field of ML that enables artificial neural networks, multiply layers to handle complicated tasks such as face recognition and autonomous driving in vehicles, generate images, create videos, craft creative things, produce music and many other applications. Deep Learning requires much greater computational power and custom hardware to learn hierarchical and sophisticate features automatically from data compared to other machine learning models, which require human supervision for feature identification.

- **How Does Deep Learning Work?**

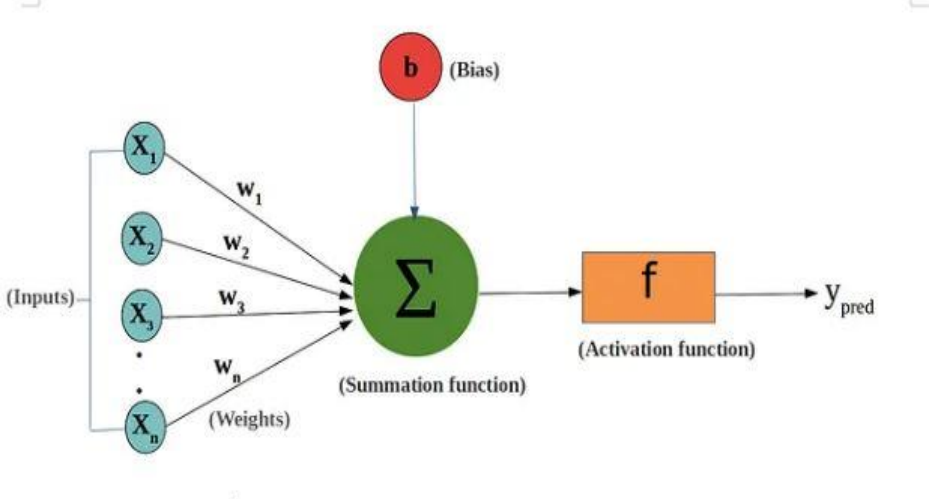


Figure 1. 11:The main components of Deep Learning [24]

- **Input Layer and Weighted Sums:** The input layer of the deep learning model takes the input data. Each input has a weight, which signifies the importance of the input in reaching the decision. The weights have typically random values assigned to them at the beginning of the training. The inputs are multiplied by their respective weights, and the products are summed up to provide each neuron with a weighted sum.

- **Activation Function**: The weighted sum is fed to the activation function subsequently. The activation function's job is to bring in non-linearity into the network so that the network learns to learn complicated patterns in data. The activation function also includes a bias in the weighted sum before it moves to the next layer or the output layer. The bias enables the model to shift the activation function so that it can make more accurate predictions. The function then determines if the neuron is to fire or not, i.e., if the output should activate. The sigmoid, ReLU (Rectified Linear Unit), and tanh (hyperbolic tangent) functions are some of the common activation functions.

- **Output Layer** : After processing by the hidden layers (which can have multiple neurons and activation functions), the model ends up at the output layer. The output layer produces the predicted output of the model based on what has been processed by the network.

- **Comparison with Actual Output** : After the network has made a prediction, the predicted output is compared to the actual output (the actual value). This enables the network to compute the error (or loss) of the prediction. The smaller the error, the better the model.

- **Back propagation and Weight Adjustment** : Once the error has been computed, the model then utilizes back propagation to alter the weights. Back propagation is accomplished by feeding the error back through the network, beginning with the output layer and moving toward the input layer. In this process, the model moves the weights so that the error is minimized overall. Gradient descent is typically

utilized for this purpose, where the weights are slowly moved by calculating the gradient (or slope) of the loss function.

- **Cost Function and Error Minimization**: The cost function (or loss function) is utilized to determine how far away the model's predictions are from the true results. A common cost function is mean squared error (MSE) for a regression task or cross-entropy loss for a classification task. The objective of model training is to minimize the cost function by back propagating the update in the weights. As the weights are updated over numerous iterations (epochs), the network improves in generating the correct output, decreasing the error rate.

- **Machine learning VS deep learning**

Machine learning	Deep learning
A subset of AI	A subset of machine learning
Can train on smaller data sets	Requires large amounts of data
Requires more human intervention to correct and learn	Learns on its own from environment and past mistakes
Shorter training and lower accuracy	Longer training and higher accuracy
Makes simple, linear correlations	Makes non-linear, complex correlations
Can train on a CPU (central processing unit)	Needs a specialized GPU (graphics processing unit) to train

Table 1. 3:Comparison between ML and DL [25]

- **Types of Deep Neural Networks**

  - **Convolutional Neural Network (CNN)** : is one of the deepest neural networks that are most commonly utilized for image processing. CNN is made of four main parts. They help the CNNs mimic how the human brain operates to recognize patterns and features in images:

    - Convolutional layers
    - Rectified Linear Unit (ReLU for short)
    - Pooling layers
    - Fully connected layers

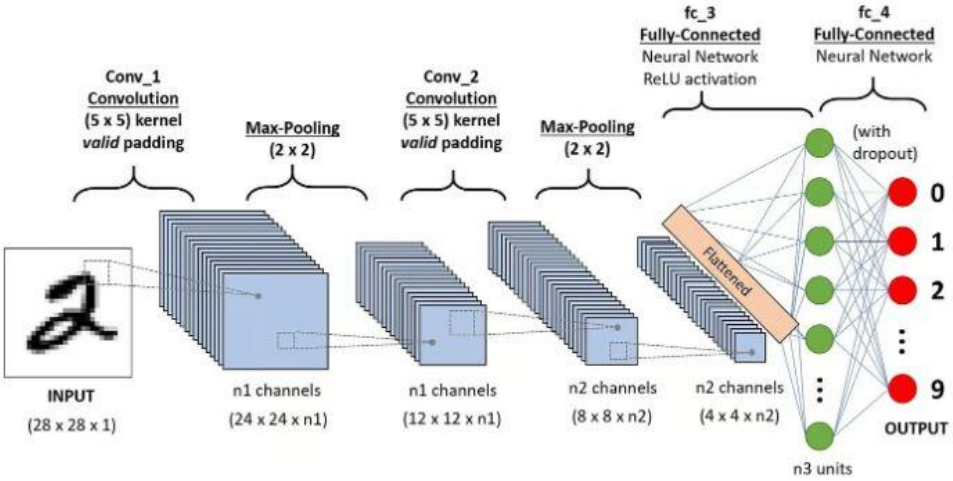


Figure 1. 12:Architecture of the CNNs applied to digit recognition [26]

  - **Recurrent Neural Network (RNN)**: Similar to the regular neural networks, including feed forward neural networks and convolutional neural networks (CNNs), recurrent neural networks are trained from training data. They differ from the rest because they possess "memory" since they draw from past inputs to determine the current input and output.

    While standard deep networks make predictions that inputs and outputs are independent, recurrent neural network output relies on the previous elements of the sequence. While future events would also be helpful when deciding on the output of a particular sequence, unidirectional recurrent neural networks cannot incorporate such events into their predictions.

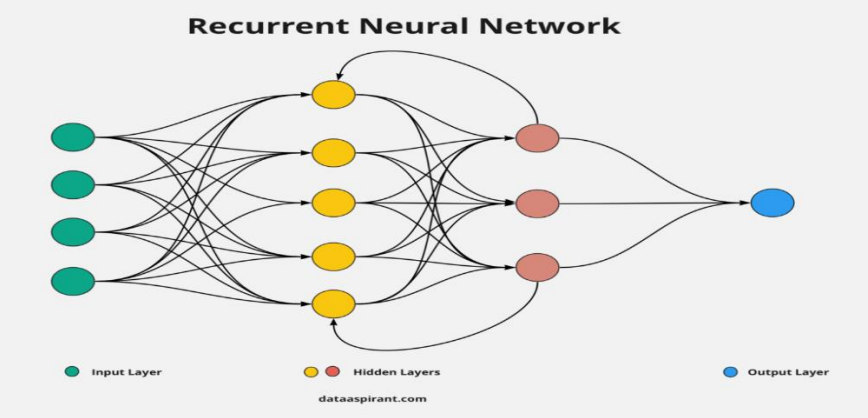


Figure 1. 13:Simple Recurrent Neural Network architecture [27]

- **Generative Adversarial Network (GAN)** : are synthetic models that use two neural networks to create novel, synthetic data samples that are copies of existing data. A GAN using a photograph can be utilized to create new pictures that seem superficially real to the human observer. A Generative Adversarial Network (GAN) has two neural networks, the Discriminator and the Generator, and they are both trained at the same time under adversarial training.

  - **Generator**: It accepts random noise as input and generates data (e.g., image). Its objective is to generate data as real as possible.

  - **Discriminator**: This network uses the actual data and the data created by the Generator as inputs and tries to discriminate between both. It produces the probability that the provided data is real.

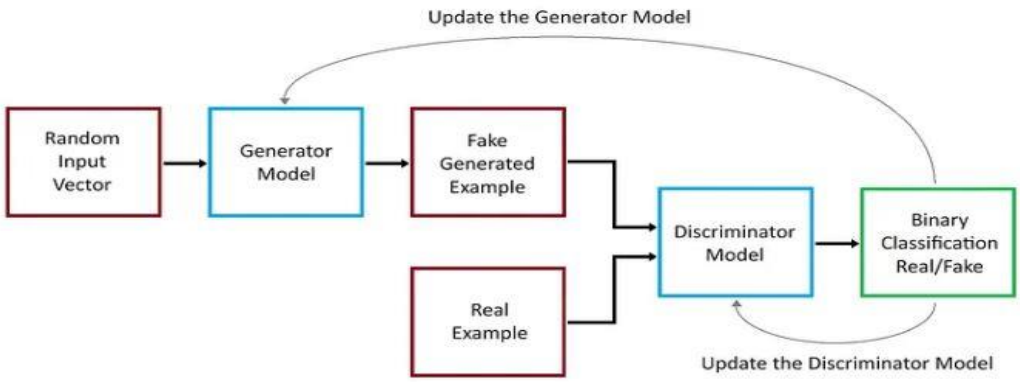


Figure 1. 14: GAN architecture [28]

1.6 Conclusion

This Chapter outlined the necessity for monitoring of forest health, referring to the ecological impact of Thaumetopoea pityocampa. It offered the potential of artificial intelligence, and specifically computer vision, to enhance the efficiency and accuracy of monitoring. Principles of basic AI, machine learning, and deep learning were also addressed as background requirements for the techniques developed in subsequent chapters.

Chapter 2

Deep Learning Approaches for PPM nest detection

2.1 Introduction

Mediterranean and European forests are increasingly under threat from the pine processionary moth (Thaumetopoea pityocampa), an exotic pest with severe defoliation impact and allergenic action in humans and animals. Early and accurate detection of its nests is crucial in order to guarantee effective pest control and forest protection. However, traditional detection methods i.e. surveying by hand and remote sensing are often limited by time, cost, and accuracy, particularly where nests are hidden or in the top canopy.

Recent advances in deep learning have opened up new potential for automated nest detection through high accuracy image processing. Convolutional neural network (CNN) architectures have proved most successful in object detection tasks, with the potential to be achieved at faster, more scalable implementations. Among these, architectures like YOLO, Faster R-CNN, and SSD have been successful in real-time detection scenarios. While several new versions of YOLO, this study focuses on YOLOv11 due to its balance between speed, accuracy, and accessibility at the time of implementation.

2.2 Deep Learning for Object Detection

Object detection using deep learning provides a fast and accurate method to estimate the object location in an image. Deep learning is a strong machine learning technique in which the object detector learns automatically the image features required for object detection tasks. Computer Vision Toolbox provides several object detection using deep learning techniques, such as you only look once YOLOv2, YOLOv3, YOLOv4, YOLOX, RTMDet, and single shot detection (SSD) [29].

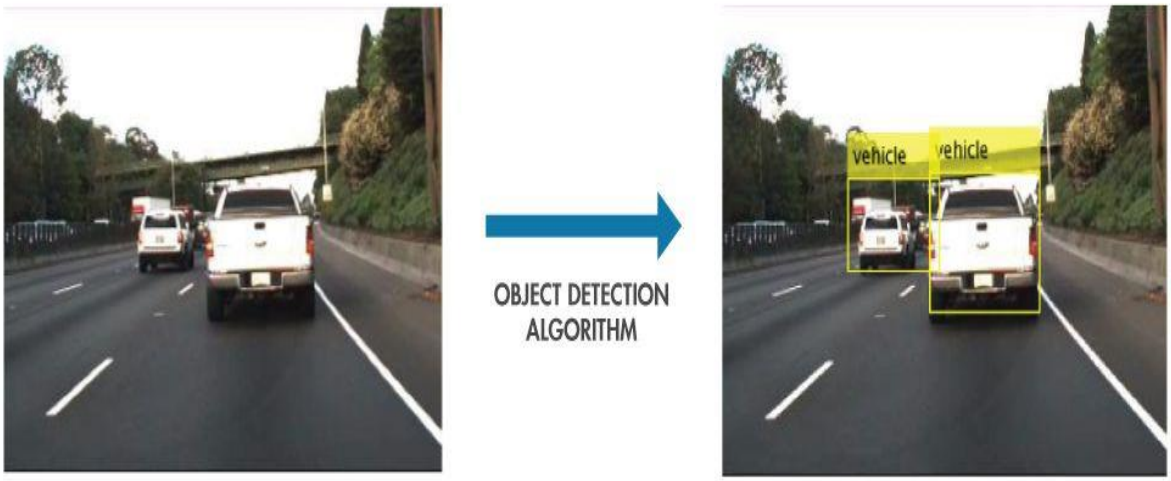


Figure 2. 1:Using object detection to identify and locate vehicles. [30]

Two approaches exist for performing object detection using deep learning techniques:

- **Use pretrained object detectors:** This method involves utilizing pretrained object detectors. Having been trained on vast datasets capable of recognizing common objects people, vehicles or text in images without needing retraining. The approach is particularly ideal for generic use cases with the requirement for rapid deployment.

- **Custom object detector:** The second method is to create a custom object detector. This is typically done by transfer learning, where the pre-trained network is customized to perform for specific detection tasks. Through the fine-tuning of pre-trained models, this method allows for the creation of highly specialized detectors with reduced computation cost and time, as the underlying network is already trained on large-scale image datasets. The method is particularly beneficial while dealing with domain-specific objects that lie outside the domains of typical datasets.

2.3 CNN Based Detection Architectures

Convolutional Neural Networks (CNNs) are now a necessity in modern object detection applications due to their exceptional ability to learn spatial hierarchies and extract meaningful features from images. For pine processionary moth (PPM) nests detection ,

CNN-based architectures offer the ability to automatically detect patterns and structures in nests, such as complex forest conditions.

Some of the most popular and utilized CNN-based object detectors include YOLO (You Only Look Once), Faster R-CNN, and SSD (Single Shot MultiBox Detector), each offering different strengths in terms of speed, accuracy and architecture.

2.3.1 YOLO (You Only Look Once)

You Only Look Once (YOLO) is one of the most recent, real-time object detection algorithms that was introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in their famous paper entitled You Only Look Once: Unified, Real-Time Object Detection. The object detection problem is framed by the authors as a regression problem rather than as a classification problem by dividing the bounding boxes in the spatial manner and providing a probability to every detected image utilizing a single CNN [31].

- **Mechanism of the YOLO Algorithm**

  Yolo divides the image into a grid. For each grid, some values like class probabilities and the bounding box parameters are calculated. The model works by first dividing the input image into a grid of cells, and each cell will predict a bounding box when the center of a bounding box falls within the cell. Each cell in the grid estimates a bounding box with the x, y coordinate and width and height and the confidence. Each class prediction also relies on each cell.

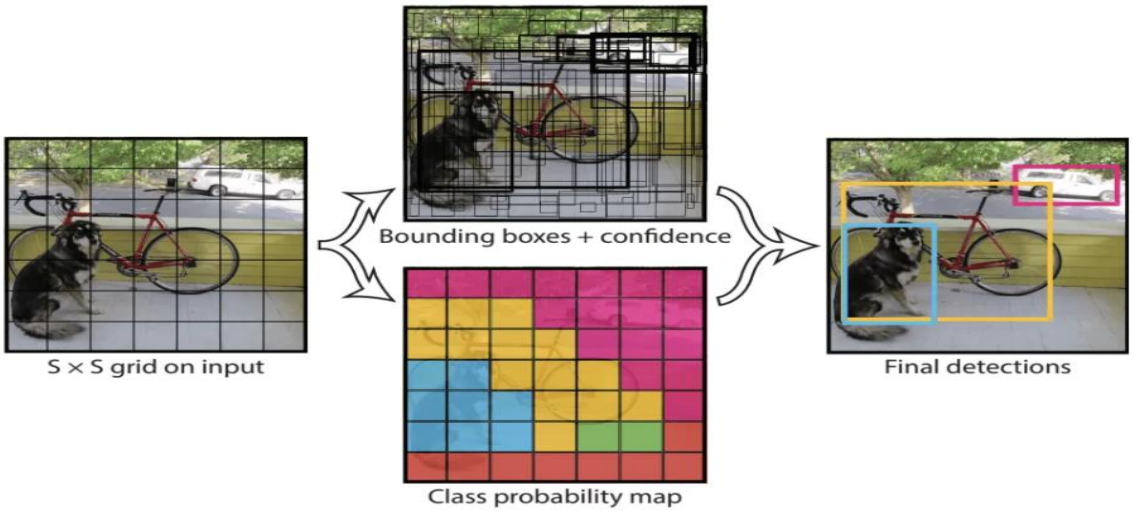
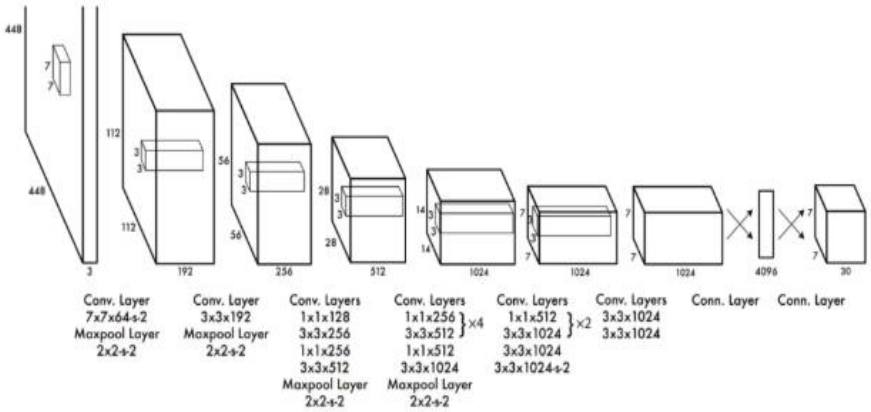


Figure 2. 2:YOLO object detection mechanism [32]

- **YOLO architecture**

YOLO resizes the input image to 448×448 before passing it through the convolutional network. It starts with a 1×1 convolution that downsamples the number of channels and subsequently utilizes a 3×3 convolution to yield a cuboidal output. The activation function used throughout the network is ReLU, except for the final layer, which uses a linear activation function. Additional techniques, including batch normalization and dropout, are used to regularize the model and prevent overfitting.



The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224 × 224 input image) and then double the resolution for detection.

Figure 2. 3:YOLO Architecture [33]

- **The Evolution of YOLO: From 2015 to 2024**

The YOLO (You Only Look Once) series has undergone significant evolution since its introduction in 2015, continuously improving in terms of speed, accuracy, and architecture design. Each new version has addressed the limitations of its predecessors and adapted to new object detection challenges.

1. **YOLOv1**: Introduced a new approach to object detection by dividing an image into an S × S grid. Each grid cell was responsible for detecting an object if the center of the object fell within that cell. Each grid cell predicted B bounding boxes with a confidence score that indicated how probable the existence of the object is and to what extent the predicted box correctly describes the object (using IoU – Intersection over Union).

   YOLOv1 handled overlapping boxes using Non-Maximum Suppression (NMS) to eliminate less accurate predictions. It used a custom loss function for location, size, confidence, and class probability to improve training performance.The model demonstrated competitive results in terms of both accuracy and speed, as shown in the following benchmarks:

   - Normal YOLO: 63.4% mAP at 45 FPS
   - Fast YOLO: 52.7% mAP at 155 FPS

   While fast, YOLOv1 was afflicted with drawbacks like low recall and localization errors, which prompted subsequent versions upgrades.

2. **YOLOv2 (YOLO9000):** Was proposed in 2016, which is a significant enhancement of YOLOv1. The name comes from its capability to recognize more than 9000 classes of objects, a major improvement in accuracy and generalization. It combines an existing detection dataset with a classification dataset, running joint training based on the hierarchical structure of the two datasets, which is one of the significant innovations of YOLO9000. The detection images do teach the model to localize objects as well, but the classification images also make it learn a more diverse set of words with more robustness [34].

Unlike two-stage detectors like Faster R-CNN, YOLOv2 uses single-stage detection, which is much faster with not too much loss in accuracy. YOLOv2 passes an input image through a deep convolutional neural network (CNN) and outputs predictions decoded into bounding boxes.

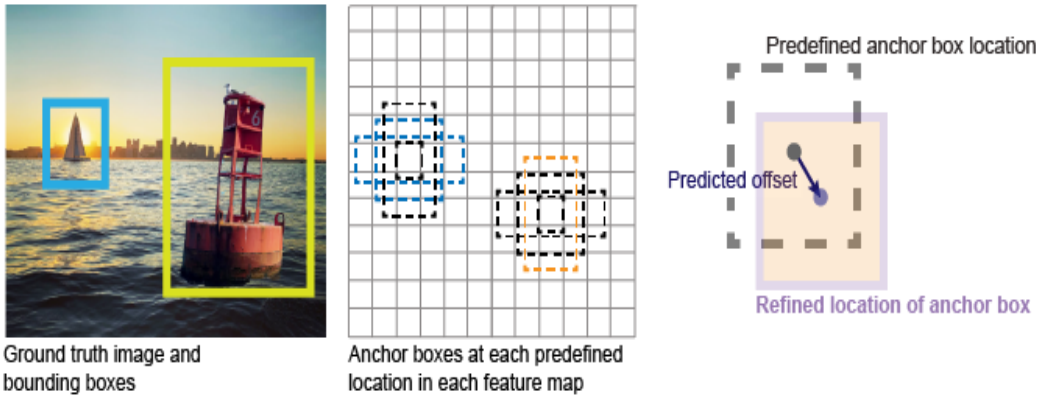


Figure 2. 4:predefined anchor boxes [35]

The figure shows predefined anchor boxes (the dotted lines) at each location in a feature map and the refined location after offsets are applied. Matched boxes with a class are in color.

YOLOv2 predicts these three attributes for each anchor box:

- Intersection over union (IoU): Predicts the objectness score of each anchor box.
- Anchor box offsets: Refine the anchor box position.
- Class probability: Predicts the class label assigned to each anchor box [36].

3. **YOLOv3:** Small changes, big impact. YOLOv3 released in 2018 by Joseph Redmon et al[37]**,** brought a series of significant improvements over YOLOv2 (YOLO9000). While YOLOv2 was already a very successful model, YOLOv3 has further improved accuracy, speed and versatility, consolidating YOLO's position as one of the most successful object detectors.

- **Principal Architectural improvements**: YOLOv3 uses Darknet-53, which

is a deeper and more powerful network comprising 53 convolutional layers, residual connections, and oversampling layers. This replaces the Darknet-19 network used in YOLOv2 and offers improved feature extraction performance, particularly for detecting small objects. Despite being more complex, it retains real-time processing capability.

- **Better Bounding Box Predictions**: YOLOv3 improves bounding box prediction by using logistic regression to estimate an objectness score for each anchor box. This score indicates whether an anchor box overlaps a ground truth box as much as possible (score = 1) or not (score = 0). Unlike Faster R-CNN, which can assign several anchor boxes to a single object, YOLOv3 assigns a single anchor box per object, which simplifies calculations (with no loss of localization or confidence in the absence of a match). [38]
- **Smarter Class Prediction**:Unlike YOLOv2, which uses the softmax function, YOLOv3 introduces independent logistic classifiers for each class. This is useful for multi-label classification, where a box may belong to several classes (for example, 'Person' and 'Football Player'), reflecting complex real-world scenarios.
- **Multi-Scale Detection**: YOLOv3 performs predictions at three different scales in the network. This enables the model to more effectively detect objects of various sizes large, medium and small by exploiting feature maps from different layers of the network.

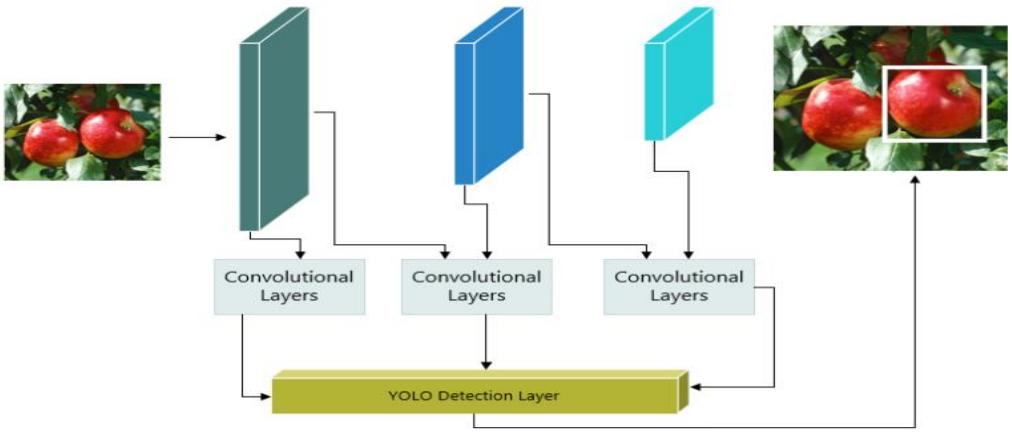


Figure 2. 5:Multi-scale Detection Architecture[37]

4. **YOLOv4**: was unveiled in April 2020 by Alexey Bochkovskiy and his team, representing a major breakthrough for the YOLO family of object detection models [37]. The released version included significant architectural changes, while continuing to operate in real-time detection.
    - **Architecture**: YOLOv4 employed a three-part architecture of backbone, neck, and head:

        - Backbone: CSPDarknet53 , a convolutional neural network using Cross Stage Partial Network (CSPNet) to improve gradient flow and feature learning. [39]
        - Neck: Used Spatial Pyramid Pooling (SPP) and Path Aggregation Network (PANet), adaptations designed to improve multi-scale feature extraction. [40]
        - Head: Used YOLOv3's anchor-based detection mechanism for final prediction.

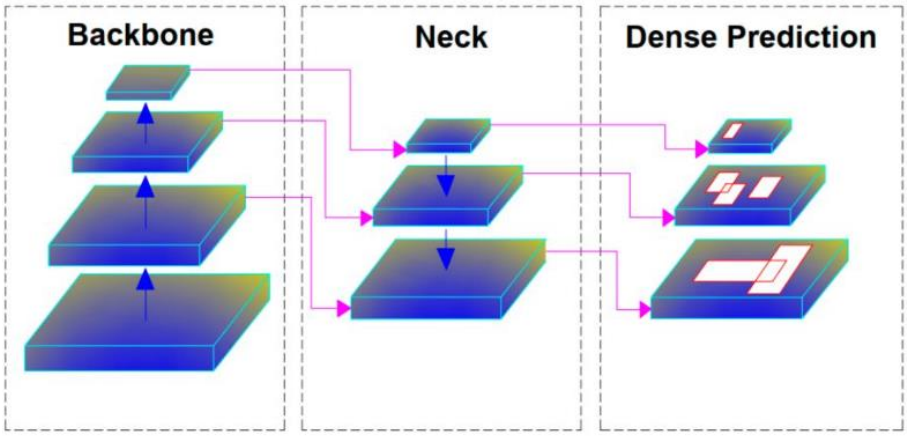


Figure 2. 6:Yolov4 Architecture [41].

5. **YOLOv5**: In June 2020, Glenn Jocher released YOLOv5, marking a significant milestone in the evolution of the YOLO family. Unlike its predecessors, which were developed on the Darknet platform, YOLOv5 has been implemented in PyTorch, which is more versatile and popular, making it more accessible to researchers and developers [37]. Ultralytics maintains YOLOv5, but there are no associated

academic papers.

- **Architecture**: YOLOv5's architecture retains the fundamental design elements of previous YOLO versions while incorporating innovations that improve efficiency and accuracy. The architecture consists of three main parts:

  - Backbone: YOLOv5 uses CSPDarknet53, a Cross Stage Partial (CSP) version of the Darknet-53 backbone introduced in YOLOv4. This architecture reduces computations by partially sharing layer gradients, which increases learning potential while reducing overlearning.

  - Neck: is composed of Path Aggregation Network (PAN) and Spatial Pyramid Pooling (SPP) blocks, to improve multi-scale feature extraction and detection of objects of various sizes.

  - Head: uses convolutional layers to predict objectivity scores, class probabilities and bounding boxes, thanks to an anchor-based approach.

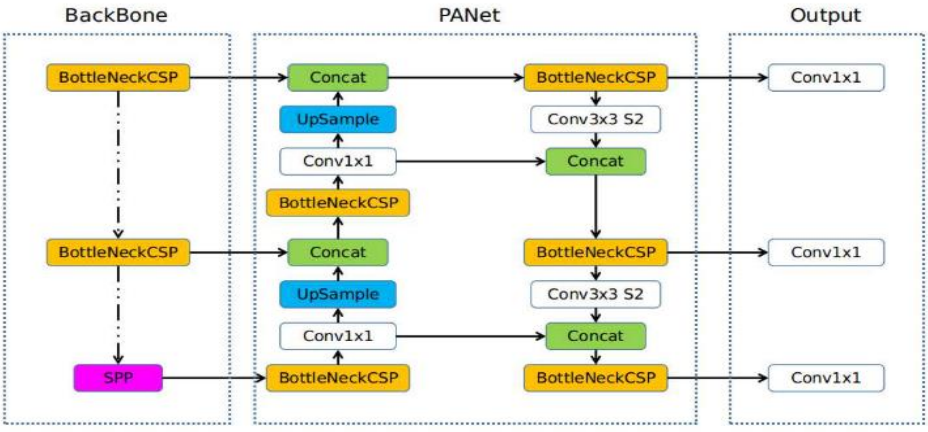


Figure 2. 7:Yolov5 Architecture[42]

6. **YOLOv6**: is a single-stage, industry-oriented object detection system based on PyTorch. Key improvements over YOLOv5 in this version include a hardware-optimized backbone and neck architecture, a refined decoupled head, and an improved training strategy. YOLOv6 outperforms previous models in the YOLO

series in terms of accuracy and speed, as confirmed by the COCO dataset.

While YOLOv6-N achieved 1,234 FPS and 35.9% AP on an NVIDIA Tesla, YOLOv6-S set a new record with 43.3% AP at 869 FPS. Even higher accuracy was achieved by YOLOv6-M and YOLOv6-L, with 49.5% and 52.3% AP respectively, without compromising speed. [43] The architecture consists of three main parts:

- Backbone: uses a hardware-optimised architecture, often based on EfficientRep, to improve speed and compactness while retaining good feature extraction capabilities.

- The Neck is composed of a Rep-PAN (Replicated Path Aggregation Network) and enables better multi-scale feature fusion, enhancing the detection of objects of different sizes.

- The head of YOLOv6 is decoupled, separating the classification and regression branches to improve accuracy and convergence during training.

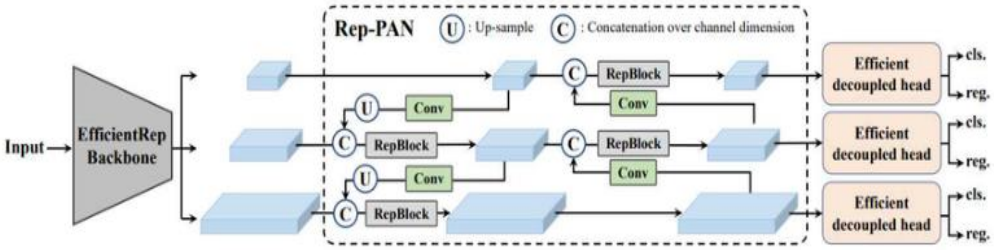


Figure 2. 8:YOLOv6 network architecture[44]

7. **YOLOv7:** Is one of the latest model in the sequence of YOLO models. YOLO models are one-stage object detectors. Image frames get featurized inside a YOLO model through a backbone. The features are blended and merged within the neck and further forwarded to the head of the network YOLO predicts locations and classes of objects where bounding boxes are to be drawn.YOLO does a post-processing via non-maximum suppression (NMS) to come up with its final prediction.

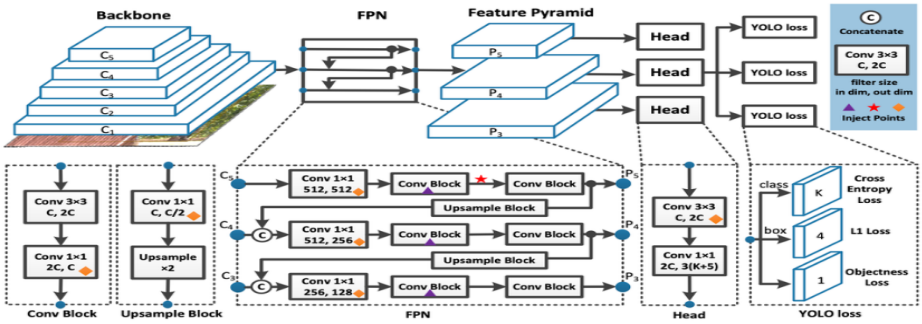


Figure 2. 9:YOLOv6 network architecture[45]

8. **YOLOv8:** Was released by Ultralytics on January 10th, 2023, with cutting-edge performance in speed and accuracy. Building on the advancements of previous YOLO releases, YOLOv8 introduced new features and optimizations that make it an excellent choice for many object detection tasks across a wide range of tasks.

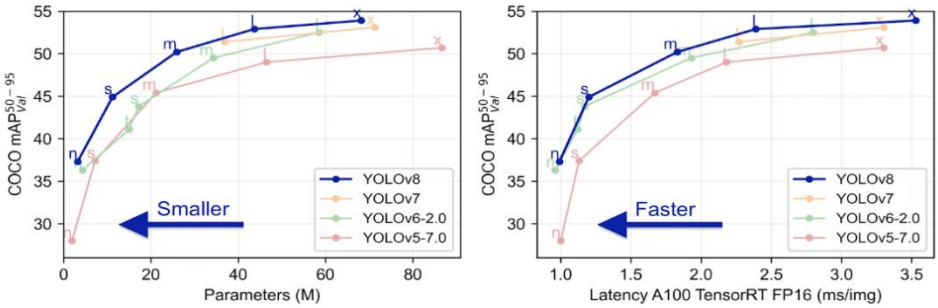


Figure 2. 10:YOLOv8 Comparison with Other Versions verions [46]

- **Key Features of YOLOv8**:

  - High-Level Backbone and Neck Architectures: YOLOv8 employs state-of-the-art backbone and neck architectures, resulting in improved feature extraction and object detection performance.
  - Anchor-free Split Ultralytics Head: YOLOv8 employs an anchor-free split Ultralytics head, thereby providing better accuracy and a more efficient detection process than anchor-based techniques.
  - Optimized Speed-Accuracy Tradeoff: Focusing on attaining a best-case speed-accuracy tradeoff, YOLOv8 is suitable for real-time object detection tasks in many areas of application.

- Variety of Pre-trained Models: YOLOv8 offers a range of pre-trained models to meet various tasks and performance requirements, and it's simple to locate the model that best suits your specific application.

- **Architecture**: the YOLOv8 architecture consists of three primary components:

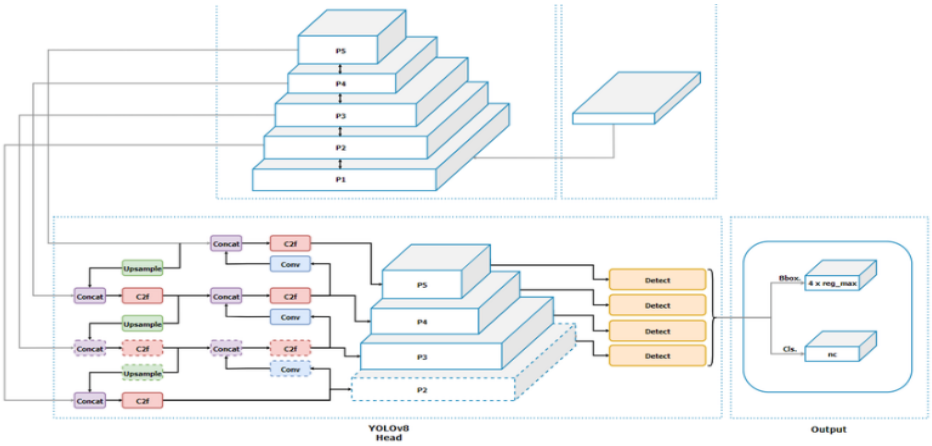


Figure 2. 11:YOLOv8 architecture [47]

- Backbone: This is a convolutional neural network (CNN) module whose task is to extract important features from the input image. YOLOv8 employs an adapted version of CSPDarknet53, which incorporates Cross-Stage Partial (CSP) connections to enhance feature propagation while reducing computational complexity, ultimately increasing model accuracy.

- Neck: Also referred to as the feature aggregation layer, the neck combines feature maps of various stages within the backbone to allow for efficient incorporation of multi-scale information. Unlike traditional YOLO models of the Feature Pyramid Network (FPN) type, YOLOv8 introduces the C2f module which effectively merges high-level semantic features and low-level spatial information. The model has improved performance in detecting small and densely packed objects using this architecture.

- Head: YOLOv8 head is tasked with generating final predictions. It consists of a number of detection layers that yield outputs in the shape of bounding box coordinates, objectness scores, and class probabilities for each grid cell in the feature map. These are then improved upon by post-processing steps in an attempt to obtain the final detection outputs[48].

9. **YOLOv11**: At the YOLO Vision 2024 (YV24) conference, the newest advancement in the YOLO (You Only Look Once) line of object detection was unveiled: YOLOv11. It expands on YOLOv1's original concepts by refining training protocols and architectural design to increase precision, speed, and effectiveness. YOLOv11 can accomplish a variety of computer vision tasks, including object detection, classification, instance segmentation, pose estimation, and oriented object detection, with enhanced feature extraction and reduced parameters, all while preserving high computational efficiency and real-time performance[49].

- **Architecture:** YOLOv11 boasts a simplified architecture targeting high accuracy and real-time processing. There are three essential components:

  - Backbone: accepts features from the input image with enhanced convolutional layers. It introduces the C3k2 block, which is an enhanced C3 and CSP bottleneck, and includes a C2PSA (Cross Stage Partial with Spatial Attention) module for boosting spatial attention and detection capability.

  - Neck: Merges multi-scale features from the backbone via upsampling and concatenation. Feature fusion is optimized by applying C3k2 blocks and the C2PSA attention mechanism, especially for small and occluded object detection.

  - Head: Generates the final prediction (class labels and bounding boxes) via a number of C3k2 and C3k blocks, which scale based on kernel size and depth to optimize precision without compromising computational

efficiency.

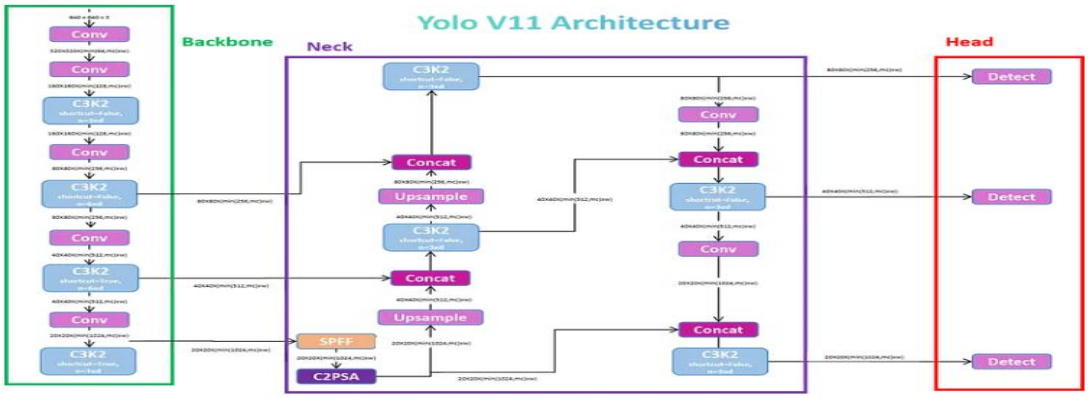


Figure 2. 12:YOLOv11 architecture[50]

Version	Year	Key Features	Performance	Impact
YOLOv1	2015	Unified architecture for real-time object detection	63.4% mAP at 45 FPS on PASCAL VOC 2007	Pioneered real-time object detection with a single neural network
YOLOv2	2016	Introduced batch normalization, high-resolution classifiers, and anchor boxes	76.8% mAP at 67 FPS on PASCAL VOC 2007	Improved accuracy and speed; expanded applicability
YOLOv3	2018	Used Darknet-53 backbone; multi-scale predictions; feature pyramid networks	57.9% AP on COCO dataset	Enhanced detection of small objects and improved accuracy
YOLOv4	2020	CSPDarknet53 backbone; mosaic data augmentation; self-adversarial training	43.5% AP at 65 FPS on COCO dataset	Balanced speed and accuracy; widely adopted in industry
YOLOv5	2020	Focused on ease of use; implemented auto-learning bounding box anchors	50.4% AP at 140 FPS on COCO dataset	User-friendly; facilitated deployment in various applications
YOLOv6	2022	Optimized for mobile devices; introduced efficient backbone and neck designs	43.1% AP at 120 FPS on COCO dataset	Enabled real-time detection on edge devices

Version	Year	Key Features	Performance	Impact
YOLOv7	2022	Extended efficient layer aggregation networks; model scaling techniques	51.4% AP at 150 FPS on COCO dataset	Achieved state-of-the-art performance; efficient for various tasks
YOLOv8	2023	Incorporated transformer layers; adaptive computation for dynamic scenes	53.9% AP at 160 FPS on COCO dataset	Improved handling of complex scenes and occlusions
YOLOv9	2024	Introduced Generalized Efficient Layer Aggregation Network (GELAN) and Programmable Gradient Information (PGI)	YOLOv9e variant achieved 55.6% mAP with 58.1M parameters	Enhanced accuracy and efficiency; suitable for diverse applications
YOLOv10	2024	Advanced loss function; variants from nano to extra-large models	YOLOv10-S achieved 46.3% APval with 2.49ms latency	Reduced latency and parameter count; adaptable to various computational needs
YOLOv11	2024	Transformer-based backbone; dynamic head design; NMS-free training	61.5% mAP at 60 FPS with 40M parameters	Improved speed and accuracy; efficient for real-time applications
YOLOv12	2025	Area Attention Module (A2); Residual Efficient Layer Aggregation Networks (R-ELAN); Flash Attention	YOLOv12-Nano achieved 40.6% mAP with 1.64ms latency	Combined attention mechanisms with speed; effective in real-time scenarios

Table 2. 1:YOLO Series – Comparison [51]

2.3.2 Faster R-CNN

Most of the latest models available today stem from the work produced by the Faster R-CNN model. Faster R-CNN is an object detection model that classifies objects within an image, draws bounding boxes around them, and identifies what those objects are. It processes in two stages:

- Stage 1: Proposes possible areas in the image that might contain objects, governed by the Region Proposal Network (RPN).

- Stage 2: Uses these proposed regions to predict the class of the object and refines the bounding box to better fit the object.

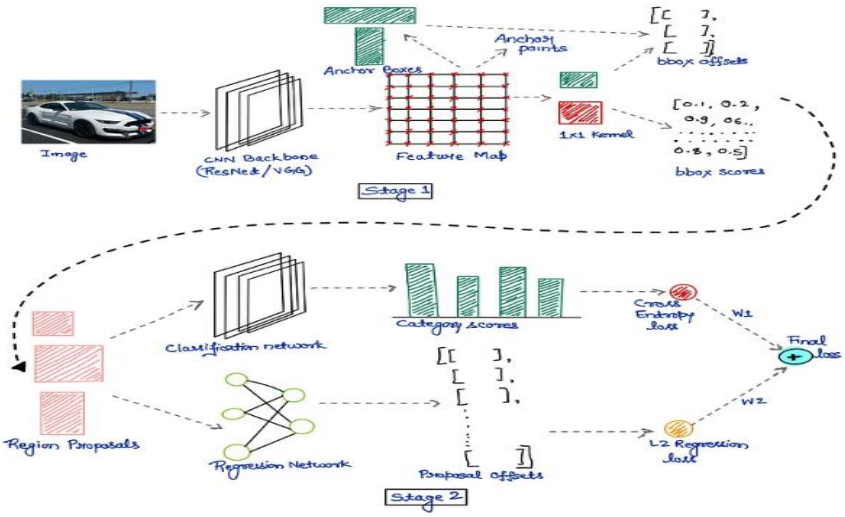


Figure 2. 13:Faster R-CNN Architechture [52]

2.3.3 SSD (Single Shot MultiBox Detector)

SSD is composed of two components: SSD head and backbone model. Backbone model is usually a pre-trained image classification network as a feature extractor. It is usually a network like ResNet trained on ImageNet with the last fully connected classification layer discarded. This leaves a deep neural network that can capture semantic meaning of the input image and also preserve the spatial organization of the image but at a lower resolution. For ResNet34, the backbone produces a 256-channel 7×7 feature map for an input image. The definitions of feature and feature map will be provided later. The SSD head is just one or several convolutional layers added on top of this backbone and the outputs are interpreted as the object classes and bounding boxes in the spatial location of the final layers' activations.

In the figure below, the early layers (white boxes) are the backbone, the later layers (blue boxes) are the SSD head

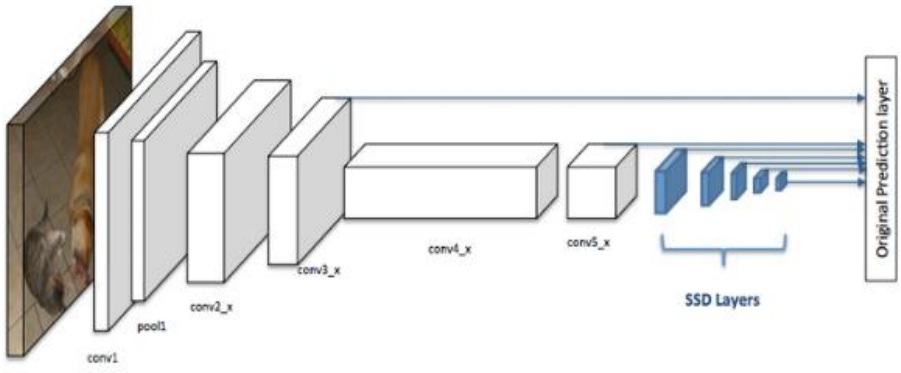


Figure 2. 14:Architecture of a convolutional neural network with a SSD detector [53]

2.4 YOLOv11 for PPM Nest Detection

After studying several object detection architectures based on deep learning, YOLOv11 was selected as the starting model for the detection of Thaumetopea pityocampa (the pine processionary caterpillar) nests. This choice is explained by YOLOv11's advanced capabilities for real-time detection and identification of small objects, as well as its efficient anchorage-free architecture, which perfectly meets the specific challenges of ecological image analysis.

Nests of Thaumetopea pityocampa (the pine processionary caterpillar) often appear as small, dense and irregular formations on pine trees, frequently blending into the natural texture of the environment. Variability in terms of size, shape and position requires an architecture capable of efficiently extracting features at different scales, while being robust in the face of visual complexity. YOLOv11's improved backbone, combined with its neck based on the C2f structure, enables it to capture both high-level and low-level features, making it particularly suited to the detection of these nests.

YOLOv11 also offers a good compromise between accuracy and inference speed, enabling its potential use in real-time forest monitoring systems or devices deployed in the field. Support for several variants of the model (YOLOv11, YOLOv11s, for example) enables adaptation to suit available computing resources and performance requirements.

Furthermore, YOLOv11 is both a technically powerful and practical tool for environmental monitoring research thanks to its ability to generalize across different datasets and the ongoing support of the Ultralytics ecosystem in terms of ease of training, testing and integration into real-life use cases.

To ensure the effectiveness of the YOLOv11-based detection system, the choice of dataset plays a crucial role. The model's performance heavily depends on the quality, diversity, and annotation accuracy of the data used for training and evaluation. In the context of detecting pine processionary moth nests, it is essential to provide images that reflect real-world conditions, including various lighting, angles, and environmental complexity. A carefully constructed dataset tailored to these challenges significantly enhances the model’s ability to generalize and perform reliably in field applications.

2.5 Dataset

The dataset used in this study consists of two parts. The first data was accessed through the Roboflow platform, which provides pre-labeled image datasets for computer vision tasks. It contains images of pine processionary moth (Thaumetopea pityocampa) nests captured in various forms of forests capes. They were taken from both aerial and ground-level perspectives, under varying lighting conditions, and with varying backgrounds. The dataset also includes both negative and positive examples images with and without nests visible such that the model can distinguish real nests from similar visual features like pine cones and branches. Annotations are provided in YOLO format, making the dataset directly compatible with the YOLOv11 training pipeline.

The second dataset is a custom dataset captured by our team using a standard digital camera. These images were taken during field visits in local forested areas, primarily for real-world scenario-based testing and model validation. Further information about the data collection process as well as field conditions are provided in Chapter 3.

Figure 2. 15:Labeled image showing bounding boxes around PPM nests

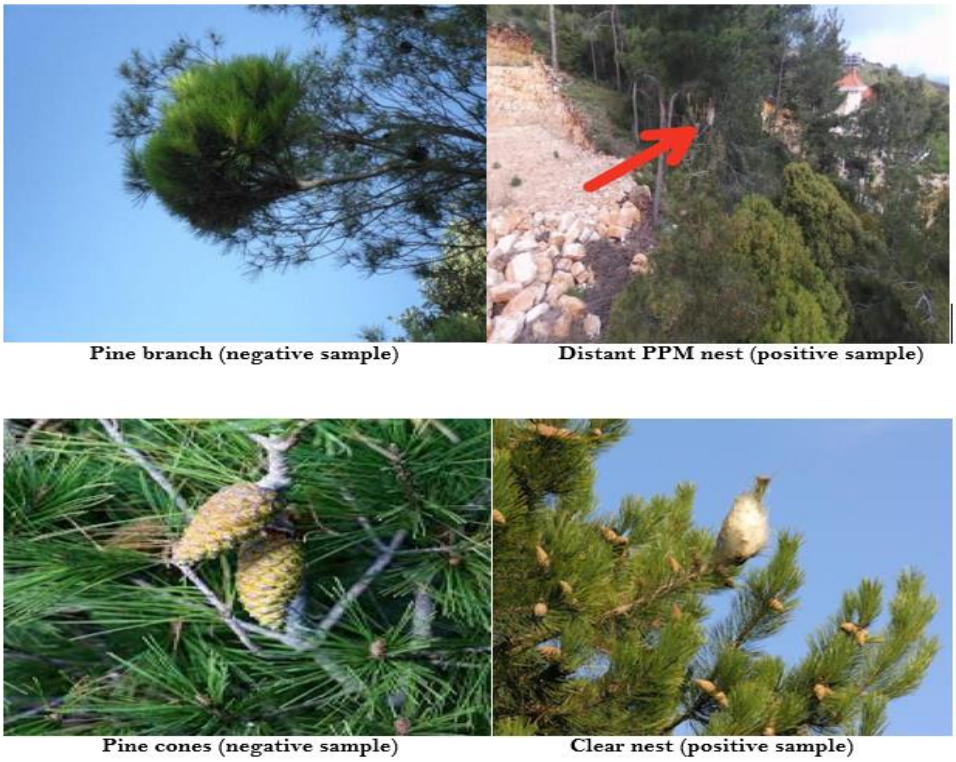


Figure 2. 16:The diversity of scenes and nest appearances in the dataset

2.6 Challenges in PPM Nest Detection

Detection of pine processionary moth (PPM) nests by deep learning methods is confronted with several challenges, primarily due to the complexity of forest environments and the nests themselves. These pose challenges for both model training and real-world model application.

- **Limited Availability of Large Data Sets**

  One of the largest challenges to develop accurate PPM nest detection models is the absence of large, high-quality datasets. While there are sites that offer annotated and curated image datasets for computer vision tasks, they do not contain the quantity and diversity required to effectively train deep learning models. This limitation prevents the model from being capable of generalizing well to different forest conditions and types of nests.

- **Visual Similarity with Natural Elements**

  PPM nests tend to appear very similar to natural objects such as clumps of pine needles, pine cones, or bright spots caused by sunlight. Same appearance regarding the way they appear increases the chances that there will be false positives or false negatives, especially in heavy forest cover.

- **Variation in Nest Appearance**

  Nests could be distinctly varying in shape, size, texture, and visibility depending upon infestation age, light level, and the angle from which the image is captured. Such variation makes it difficult for models to learn to generalize in all possible cases..

- **Occlusion and Background Clutter**

  Nests in the majority of cases are partly occluded by leaves, branches, or other forest elements. This occlusion, along with the variability and cluttered background, prevents the model from easily identifying nest structures.

- **Small Object Detection**

  Some nests are captured from a distance either by drone or camera making them appear very small in the picture. The detection of such small objects is a known limitation for most object detection models, especially outdoors.

- **Lighting and Weather Conditions**

  Environmental condition variability such as shadows, fog, rain, or harsh contrast sunlight may cause both nests and surrounding trees to appear irregularly. These inconsistencies make the detection even more difficult.

2.7 Existing Research and Research Gaps

Advancements in deep learning have enabled researchers to develop pine processionary moth (PPM) nest automatic detection systems. Several methodologies were explored in various studies,whose results were found to be positive:

- Multi-Stream Convolutional Neural Networks: Jaber et al. (2021)[15] had put forward a system that integrated RGB and thermal images, which were analyzed through a two-channeled deep convolutional neural network. The system achieved a mean accuracy of 97% in detection and allowed geo-localization of the nests to the centimeter order.

- UAV-Based Forest Detection Using Deep Learning: Garcia et al. (2023)[12] experimented with the UAV use and deep learning models, YOLO and Faster R-CNN, on various types of forests in south Europe. They discovered that YOLO outperforms Faster R-CNN with F1-measures 0.826 for presence/absence and 0.696 for single nest detection.

- Semantic Segmentation Methods: Akıncı and Göktoğan (2019) [13] utilized semantic segmentation on UAV images for mapping and detection of PPM nests in pine plantations. The method enabled the generation of spatiotemporal maps, thus making it easier to carry out strategic planning in the management of PPM infestation.

2.8 Conclusion

This chapter provided a general description of deep learning techniques employed in detecting pine processionary moth (PPM) nests, i.e., the CNN-based architectures such as YOLO, Faster R-CNN, and SSD. YOLOv11 was accorded special attention since it possesses a newer architecture and was observed to be very proficient in real-time object detection tasks. The datasets employed in this work were also described, commenting on their range and relevance to the task.

By going through previous work, it was evident that while deep learning-based models have achieved promising results, there are issues primarily environmental variance, small and distant targets, and insufficient availability of large, diverse datasets. Research limitations were identified as far as model generalization, real-time performance, and insufficient testing of recent model versions like YOLOv11 are concerned.

These observations lay the groundwork for the experimental method outlined in the following chapter, where model training, data preprocessing, and measurement of performance are elaborated on.

Chapter 3

System Design and Model Development

3.1 Introduction

This chapter presents the applied design and development process of the intelligent system for Thaumetopoea pityocampa nest detection. The focus is placed on the selection and configuration of the most suitable deep learning model for the application, data preparation, solving of specific detection issues, and integration of supporting technologies such as GPS-based localization. The deployment leverages advanced computer vision and data preprocessing techniques to enhance accuracy of detection, especially under adverse conditions such as distant or small-sized nests. The decision made at this point was governed by the goal of having a robust, scalable, and feasible solution for real-world forest health monitoring use cases.

3.2 System Objectives

The main objective of the system is the automatic detection and localization of Thaumetopoea pityocampa nests in forest environments using deep learning. To achieve this, the system is designed with the following specific goals:

- High Detection Accuracy: Be able to identify nests of different shapes and sizes, partially hidden or far away from the camera.
- Precise Localization: Use GPS metadata from the captured image to correlate every detected nest with its geographical location
- Enhanced Detection Performance: Improve accuracy through preprocessing by using Multi-Scale Patch Analysis (MPA), in case of small or faraway nests.

3.3 YOLOv11 Model Selection

YOLOv11 was selected for detecting Thaumetopoea pityocampa nests due to its advanced architecture, real-time performance, and adaptability to complex detection tasks. Its design offers several key advantages:

- **Best precision/efficiency compromise**: YOLOv11 offers higher precision than YOLOv8m, while reducing the number of parameters by 22%, making it more compact and faster. [54]

- **Optimized real-time performance:** with an inference time of just ≈2.4 ms on a TensorRT FP16 GPU, YOLOv11n is among the fastest models in the series, well ahead of YOLOv8, v9 and v10.[49]

- **Enriched architecture:** YOLOv11 incorporates new blocks such as C3k2, SPPF and C2PSA, which enhance multi-layer feature extraction essential for the detection of small and complex objects.
- **Mature ecosystem for conservation:** YOLOv11 benefits from a powerful Ultralytics ecosystem, including experimental monitoring tools (e.g. DVCLive), tutorials, multi-platform deployment and support for environmental tasks such as species monitoring or pollution detection.[55]

3.4 Components of YOLOv11

As part of our application dedicated to the detection of Thaumetopoea pityocampa nests, YOLOv11 offers a series of powerful, modular functionalities that perfectly meet the requirements of precision, flexibility and integration. Available tools include:

- Oriented bounding boxes (-obb), useful for detecting nests that are tilted or partially visible at non-standard angles.
- Pose estimation (-pose): in future versions, this feature could be used to isolate the precise contours of nests instead of simple boxes.
- Instance segmentation (-seg): this functionality, which will be integrated in future versions, will enable the precise contours of nests to be isolated instead of simple boxes.
- Classical object detection is at the heart of our current approach to locating nests with precision.

- Classification (-cls): can be used to distinguish nest types or development stages.

YOLOv11 offers these tools in several sizes (nano, small, medium, large, x-large), allowing adaptation to the resources available. Thanks to its easy integration with the Ultralytics library and the Ultralytics HUB platform, the model can be trained, exported and then efficiently deployed in our pipeline [56].

Model	Size (pixels)	mAPval (50-95)	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	Params (M)	FLOPs (B)
YOLOv11n	640	39.5	56.1 ± 0.8	1.5 ± 0.0	2.6	6.5
YOLOv11s	640	47.0	90.0 ± 1.2	2.5 ± 0.0	9.4	21.5
YOLOv11m	640	51.5	183.2 ± 2.0	4.7 ± 0.1	20.1	68.0
YOLOv11l	640	53.4	238.6 ± 1.4	6.2 ± 0.1	25.3	86.9
YOLOv11x	640	54.7	462.8 ± 6.7	11.3 ± 0.2	56.9	194.9

Table 3. 1:YOLOv11 performance on COCO Object Detection

It provides better feature extraction with more accurate detail capture, higher accuracy with fewer parameters, and faster processing rates (better real-time performance).

3.5 Overall System Architecture

The model follows a modular pipeline designed to detect and localize Thaumetopoea pityocampa nests in field images. The main components are:

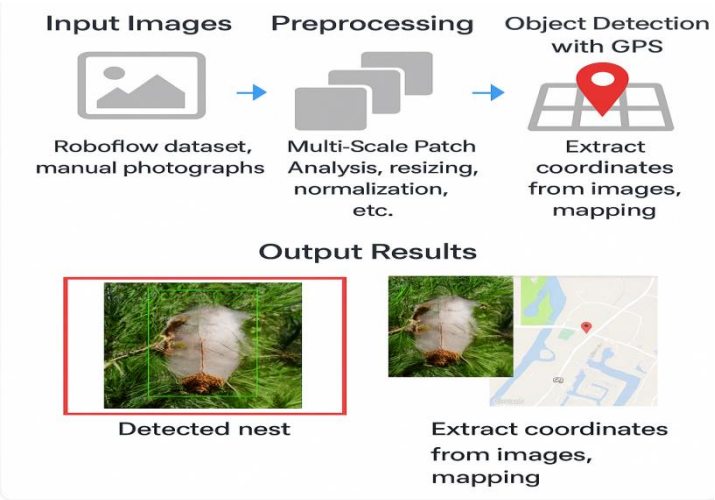


Figure 3. 1:System pipeline for the detection and localization of PPM nests (Source: Authors)

1. **Input Images**: There is two main sources a public dataset from Roboflow and photos taken by hand which have GPS metadata.

2. **Preprocessing**: Before detecting nests, the images are preprocessed with techniques like Multi-Scale Patch Analysis. This helps make small or distant nests easier to see. it also includes other techniques likes resizing .

3. **Nest Detection**: Once the images are ready, they are passed through a YOLOv11-based detection model, which detects the nests and draws boxes around them.

4. **Visualization with GPS**: If GPS metadata is there, the system grabs the location data from the photos and maps where each nest is.

3.6 Dataset Description

The dataset used for this project consists of two primary sources, each selected to support different stages of the system development process:

1. Training Dataset Roboflow:

The model was trained using a dataset sourced and annotated through the Roboflow platform[61].This dataset includes a wide range of images of Thaumetopoea pityocampa nests captured under varying environmental conditions. Roboflow's preprocessing, augmentation, and export tools (e.g., for YOLO format) facilitated efficient preparation of the training data.

2. Testing Dataset Field Images (Personal Collection):

Field testing was conducted using images collected manually across three visits to forested areas in the region of Guelma, specifically in Aïn Ksoub, Roknia, and Bouhmedene.

- During the first two visits, high-resolution images and videos were captured using a professional camera to ensure image clarity for detection evaluation.

Figure 3. 2:Field Images Collection: Nest Samples and On-Site Work

- In the third visit, smartphones were used to collect images containing GPS metadata, enabling geolocation of detections and validating the model's application in spatial mapping.

3. YOLOv11 Annotation Format :

YOLOv11 uses a straightforward annotation format that is an extension of what was used with YOLOv11. For each image, there's a corresponding .txt file where each line represents one object in the image. [58]

```
<class> <center_x> <center_y> <width> <height>

0 0.45859375 0.4921875 0.2109375 0.428125
```

Figure 3. 3:YOLOv11 Annotation Format

Each line contains:

- The class ID (a number starting from 0)
- center_x: the horizontal center of the box
- center_y: the vertical center
- width: how wide the box is
- height: how tall the box is

3.7 Dataset Preparation

The precision and preprocessing of the dataset play a critical role for high detection accuracy. Several preprocessing and enhancement techniques were employed in this project to address detection problems and improve model performance.

3.7.1 Small and Distant Nest Problems

Initial experiments with YOLOv11 on the raw data were disappointing for detecting small or distant nests. These nests were not being detected due to their low resolution in wide-angle forest scenes. The model achieved a mAP@0.5 of approximately 88.9% and a precision of 90.7%,before applying MSPA, but nearly all the false negatives were in such challenging cases.

3.7.2 Multi-Scale Patch Analysis (MSPA)

To overcome the challenge of identifying small, distant nests, we adopted the Multi-Scale Patch Analysis (MSPA) approach, which draws inspiration from techniques employed in smart farming, as outlined in the article 'Smart Farming Solutions': Automated Crop and Plantation Disease Detection' (ResearchGate, 2024) [59].

This method involves generating several image patches at different scales from each original image and focusing on areas with a high probability of nest occurrence (e.g. tree branches). Dividing the original high-resolution image into enlarged regions of interest enables even very small or partially hidden nests to be captured with sufficient resolution for reliable detection.

Patch generation

To create relevant patches from annotated images, we used a custom Python script that follows several key steps. The main function responsible for this process is the 'generate_multi_scale_patches' function, which:

- loads images and their associated annotations in YOLO format using the load_images_and_labels function.
- Analyzes the size of each annotated nest (bounding box) using the calculate_nest_size function.
- Each nest is categorized as very small, small, medium, or large based on the area it occupies relative to the image size.
- Extracts a square patch centered on each nest, whose size depends on the nest category:
  - 320 × 320 pixels for medium-sized nests.
  - 448 × 448 pixels for large nests.
- Extremely small nests, which occupy less than 0.5% of the image area, are ignored in order to reduce noise and avoid introducing information that is of little use to the model.

Re-annotation of Nests in Patches

After extracting the patches centred on the nests, it is crucial to adjust the annotations (or 'bounding boxes') of the visible objects within each patch. This is because the original coordinates are no longer valid for these partially extracted images. The adjust_bounding_boxes function is used for this purpose.

Therefore, all nest annotations visible within a patch must be recalculated relative to the new patch size using a coordinate transformation, to ensure that the bounding boxes retain their accuracy when the model is trained.

The formula used to convert the absolute coordinates of objects in the original image to the normalized coordinates of the patch is as follows:

```
x_new = (x_original - x1) / patch_width

y_new = (y_original - y1) / patch_height

w_new = w_original / patch_width

h_new = h_original / patch_height
```

Figure 3. 4:YOLO Coordinate Transformation for Image Patch

- (x1, y1, x2, y2) are the top-left and bottom-right coordinates of the patch in the original image.
- The width and height of the patch:
  - patch_w = x2 - x1
  - patch_h = y2 - y1
- The original object (nest) has absolute center (x_original, y_original) and size (w_original, h_original) in pixels.

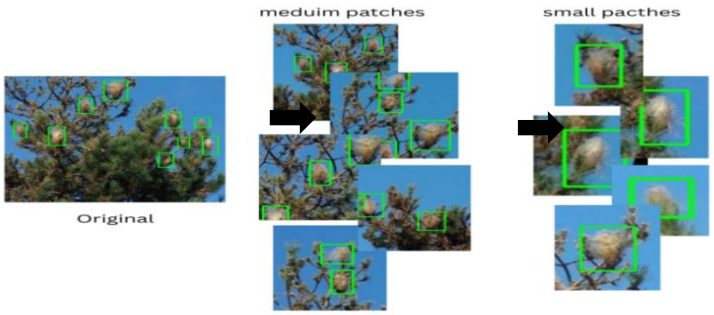


Figure 3. 5:Example of Multi-Scale Patch Analysis (from personal dataset created)

By breaking the large images into overlapping and multi-scale windows, MSPA ensures that even the smallest nests are presented to the model at sufficient resolution. This significantly improves detection accuracy, especially for low-resolution targets or partially

occulted nests in tall vegetation.

After applying MSPA, the model's mAP@0.5 improved from approximately 88.9% to 98.4%, and precision increased from 90.7% to 94.9%, demonstrating the method's potential to mitigate false negative.

Figure 3. 6:Illustration of Multi-Scale Patch Analysis (MSPA): Original Image and Corresponding Zoomed Patches Author, from personal dataset created

3.7.3 Dataset Splitting and Augmentation

The initial dataset was sourced from Roboflow, where it was automatically divided into training, validation, and testing sets in accordance with standard deep learning practices. To enhance the model’s robustness to real-world conditions, data augmentation was performed directly on Roboflow using its built-in parameters.

These adjustments (Figure 3.7) helped in simulating different real-life situations such as changing lightings, positions, and sizes of objects.

Figure 3. 7:Data Augmentation Parameters Used in Roboflow [60]

Rebalancing after MSPA

However, after applying the Multi-Scale Patch Analysis (MSPA) method, implemented via the generate_multi_scale_patches function, a new version of the dataset was generated, consisting of multi-scale patches extracted around the nests. This process led to an imbalance: images containing multiple nests (or large nests) produced a large number of patches, while others generated few. To correct this imbalance, a set of specific pre-processing steps was applied:

- normalize_patch_distribution, which was used to limit the number of patches per image by selecting only useful patches, in order to avoid over-representation of dense scenes.
- The filter_useless_patches function was used to remove blurry, empty, or poorly framed patches, i.e., those that do not contain any visible nests or cannot be used for learning.
- augment_underrepresented_patches: to compensate for the under-representation of certain images after the MSPA step, this function applied targeted local augmentations (rotation, flip, brightness modification, etc.) only to patches from underrepresented images. This restored a certain balance to the dataset in terms of image volume and visual diversity.

Final division of the dataset

After the multi-scale patch generation (MSPA), filtering, and targeted augmentation steps, the final dataset was divided into two subsets: 80% for training and 20% for validation. This division was performed randomly using a Python script. Unlike these sets, the test set was not extracted from the main dataset, as the model testing phase was carried out on a set of real images captured in the field (personal collection).

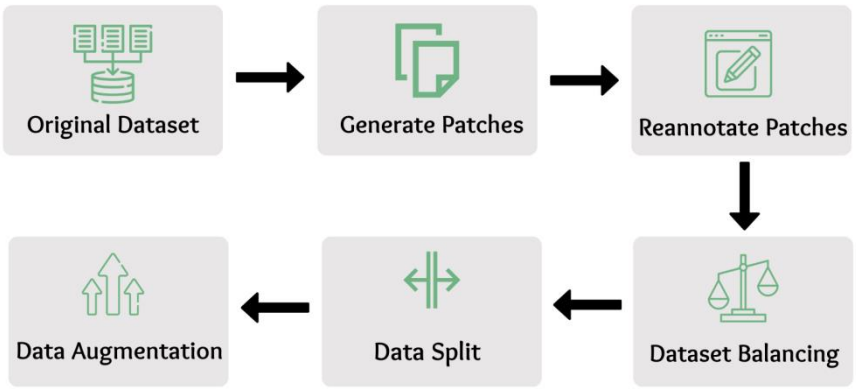


Figure 3. 8:MSPA dataset preparation pipeline

3.8 Model Configuration and Training

3.8.1 Training Strategy

After the preparation of the dataset by merging normal images and multi-scale patches the training is done by employing the YOLOv11 model optimized for small and distant nest detection. The training process employed top techniques to achieve maximum performance:

- **Multi-Resolution Training**

  To improve generalization across images of varied quality and nest sizes, multi-resolution training was used. This technique randomly changes the resolution of input images during training, allowing the model to learn more effectively for nests of different sizes and visibility.

- **Mosaic Augmentation During Training**

  Although core data augmentation was done initially via Roboflow, mosaic augmentation was also utilized dynamically during training time. Mosaicing four training images together, this technique augments context diversity and allows the model to generalize to varied environments, such as cluttered forest settings.

3.8.2 Hyperparameters and Loss Function

- **Hyperparameters**

The selection of hyperparameters played a crucial role in the adjustment of performance of the YOLOv11 model for nest detection of Thaumetopoea pityocampa. Rather than fixed values, the values of significant hyperparameters such as the number of epochs, batch size, and learning rate were selected experimentally. Multiple training experiments using different settings were conducted to allow us to view their impact on performance parameters and convergence behavior. The final setup was selected based on the best accuracy, training time, and regulation of overfitting.

The final configuration that yielded the optimal performance is the following:

Parameter	Value	Justification
Epochs	50	Moderate value providing enough learning cycles for the enlarged dataset.
Image Size	640 × 640	Balanced resolution ensuring small nest detection with reasonable training speed.
Batch Size	16	Fits within GPU memory while maintaining stable gradient updates.
Device	CUDA (GPU)	Accelerated training performance and ability to handle larger data efficiently.
Model Variant	YOLOv11n	Lightweight model enabling faster experimentation with competitive accuracy.

Table 3. 2:Final YOLOv11 Training Configuration and Hyperparameters

- **Loss Functions**

During training, the model converges to a composite loss function internally, as is standard in YOLOv11 but critical to mention:

➢ **CIoU (Complete Intersection over Union) Loss**

Used for bounding box regression, this loss enhances accuracy by considering not only the overlap area but also the center distance and aspect ratio between predicted and ground truth boxes.

- train/box_loss: Decreased from approximately 2.05 to 0.55
- val/box_loss: Decreased from around 1.80 to 0.78

➢ **Binary Cross Entropy (BCE) Loss**

This loss is applied both to the objectness score and the classification of detected objects. It performs effectively in binary and multi-label classification tasks, which are typical in object detection.

- train/cls_loss: Dropped from over 2.0 to around 0.35
- val/cls_loss: Dropped from 1.3 to approximately 0.4

➢ **Distribution Focal Loss (DFL)**

Introduced in recent versions of YOLO, DFL improves the accuracy of predicted bounding box coordinates by learning a distribution over discrete distance bins. It improves the accuracy of localisation by refining how the model interprets object boundaries.

- Train/DFL loss: Decreased from 1.40 to 0.80.
- Val/DFL_Loss: Decreased from 1.35 to 0.95.

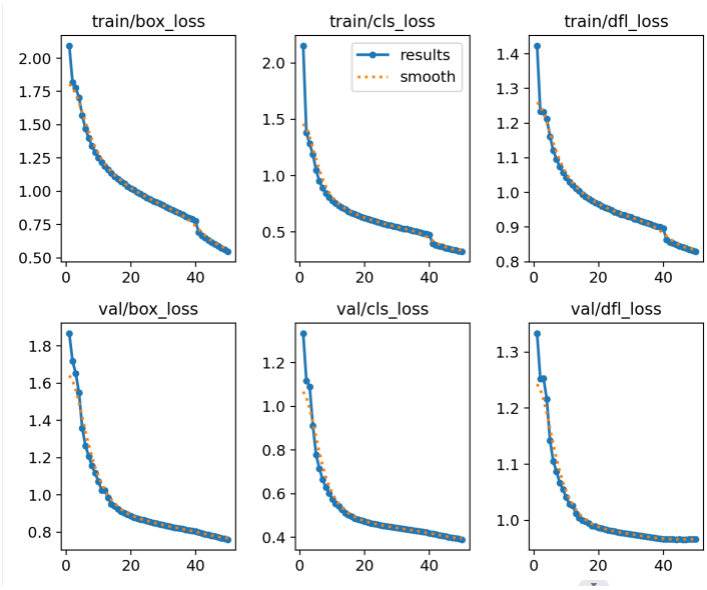


Figure 3. 9:Loss Function Curves (Box, Cls, DFL) – Training and Validation

The steady and consistent decrease in the three loss functions (CIoU, BCE, and DFL), both for training and validation, indicates that the model is learning effectively. The proximity of the values between the training and validation curves rules out any overfitting, and the low final values indicate good generalization as well as accurate localization of Thaumetopoea pityocampa nests. These results confirm the stability and robustness of the training process with YOLOv11 for this task.

3.8.3 Challenges Faced During Training

- **Limited Dataset Size:** Before applying the Multi-Scale Patch Analysis (MSPA) strategy, the available dataset was relatively small. The small volume of data limited the model's exposure to diverse nest conditions, with the potential for overfitting and reduced generalization performance on new samples.

- **Camouflaged Nests in Natural Backgrounds:** Thaumetopoea pityocampa nests merge with the environment due to the same textures and colors. Camouflage made the model difficult to learn discriminative features. The issue was addressed by increasing the resolution and diversity of training examples and adding augmentations to familiarize the model with varying lighting, textures, and angles.

- **Detection of Small and Far away Nests:** One of the main issues was accurately detecting nests that were either too far away or too small from the camera. These nests lacked sufficient visual salience in full-resolution images. This was resolved by utilizing the MSPA approach to generate high-resolution patches centered around the potential nest areas, significantly improving detection rates.

Before MSPA After MSPA

Figure 3. 10:Impact of MSPA on Detection Performance

The left image shows the detection result without applying Multi-Scale Patch Analysis (MSPA), the model failed to detect the distant and small nest. The right image is the detection result after MSPA was applied, which the nest is detected successfully. MSPA enhances detection by generating focused high-resolution patches, especially for small or camouflaged objects.

- **Overfitting on Early Training:** Due to the small size of the dataset in the initial stages of training, the model was plagued by overfitting good performance on training data but poor on validation or test data. This was countered with increasing the size of the dataset with MSPA, applying data augmentation (e.g., Mosaic), and controlling through dropout and early stopping.

- **Lack of Geolocation Data:** None of the original photos included GPS metadata, limiting the geolocation of recognized nests. Fieldwork was consequently conducted to capture new photos using GPS enabled devices manually. While these images were not used for training, they enriched the dataset for evaluation purposes and enabled

geospatial analysis and visualization in the final phases of the project.

3.9 Geolocation and Mapping Integration

- **GPS Metadata Extraction:** The images captured in the field on mobile phones and cameras also come with inherent GPS coordinates. Such metadata were downloaded using EXIF parsing tools and stored along with detection outputs.

Figure 3. 11:GPS Metadata Extraction Example (bouhamdane Guelma)

Screenshot of an image taken with a GPS-enabled smartphone, showing geolocation data in Google Photos. This information was crucial for associating detected nests with their real-world coordinates.

- **Real-Time Mapping Potential:** Inclusion of GPS coordinates allows geolocation of nests discovered by the system. This will allow for future potential establishment of real-time monitoring platforms where users can upload images and see the results mapped on an interactive map instantly facilitating early warning and intervention.

3.10 Conclusion

This chapter explained the design, setup, and training process for the YOLOv11 based model to detect Thaumetopoea pityocampa nests. The training pipeline was designed with consideration for the specific difficulties of this task, e.g., the initially small dataset, the camouflage or distant appearance of nests, and lack of GPS metadata.

To enrich the dataset, the Multi-Scale Patch Analysis (MSPA) method was utilized, producing zoomed patches of images around potential nesting areas. This significantly increased the quantity and diversity of the training data. Hyperparameters were experimentally tuned across multiple training iterations, with final values chosen . Along with this, fieldwork was conducted to collect GPS tagged photos for geospatial analysis and to provide a basis for future integration of IoT and real-time infestation mapping.

Together, this chapter set the technical foundations of the system, setting stage for performance testing and results analysis in the next chapter.

Chapter 4

Implementation and Results

4.1 Introduction

This chapter presents the practical implementation and results of the intelligent detection system developed for monitoring Thaumetopoea pityocampa (pine processionary caterpillar) nests using deep learning techniques. The goal of this system is to provide an effective and accessible tool for the detection and management of infestations

The implementation was carried out using modern frameworks and tools. In addition to the integration of the server-side model, a user-friendly platform has been developed in the form of a website and mobile application. This platform enables users be they experts, forestry officers or ordinary citizens to upload images or videos, view detection results, and even display the geographical location of detected nests when GPS metadata is available.

The chapter begins with the development environment, detailing the hardware and software tools used. Next, the system's complete workflow is presented. Particular attention is paid to the impact of using the Multi-Scale Patch Analysis (MSPA) method, which has significantly improved model performance, especially for detecting small or distant nests.

The system was evaluated using several metrics before and after the application of MSPA. Finally, the results are discussed in detail. Tests were also carried out on real images captured under natural conditions. Visual examples of detection results to provide a comprehensive overview of the effectiveness and suitability of the proposed system.

4.2 Development Environment

4.2.1 Hardware Environment

At the start of the project, model training was carried out on the Kaggle platform, which offered sufficient resources to work with a small dataset (prior to the application of MSPA). However, after the significant increase in data volume following the application of the MSPA method, a more powerful hardware environment was required. Thus the training was transferred to a high-performance machine which is located in the LabSTIC laboratory at the University 8 May 1945 of Guelma. This machine is equipped with a high-end GPU (NVIDIA), a multi-core processor and a large RAM capacity, enabling faster and more efficient training.

4.2.2 Software Environment

- **Python**

Python is a high-level, interpreted, object-oriented programming language with dynamic semantics. Its high-level data structures, combined with dynamic typing and dynamic linking, make it very attractive for rapid application development, as well as for use as a scripting or glue language to link existing components together. Python's simple, easy-to-learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, promoting program modularity and code reuse. The Python interpreter and the extensive standard library are available free of charge in source or binary form for all major platforms, and can be freely distributed.

Programmers often fall in love with Python because of the increased productivity it offers. As there is no compile step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation error. Instead, when the interpreter discovers an error, it throws an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source-level debugger allows you to inspect local and global variables, evaluate arbitrary expressions, set breakpoints, browse code line by line, etc. The debugger is written in Python. The debugger is written in Python itself, which testifies to Python's power of introspection. On the other hand, the quickest way to debug a program is often to add a few print instructions to the source code: the fast edit-test-debug cycle makes this simple approach very effective.[61]

Figure 4. 1:Python logo [62]

In this project, Python served as the primary language for all implementation aspects, including data preprocessing, model training, integration with YOLOv8, and interfacing with other software tools like Roboflow.

- **Ultralytics**

Ultralytics is a technology company that specialises in advanced computer vision solutions, particularly real-time object detection using artificial intelligence. Founded by Glenn Jocher, the company is best known for developing the popular YOLO (You Only Look Once) series of object detection models, including YOLOv5 and YOLOv8. These models have become the industry standard and are widely used in academic research and practical applications involving real-time detection tasks.

Ultralytics has significantly improved the accessibility and reproducibility of deep learning–based object detection by providing an open-source, PyTorch-based framework alongside comprehensive documentation and tools. YOLOv11 introduces advanced capabilities, including instance segmentation, pose estimation and improved data handling, as well as enhanced training performance. This makes it suitable for a broad range of computer vision problems.[63]

Figure 4. 2:Ultralytics logo [64]

This thesis used Ultralytics YOLOv11 as the main detection engine to identify Thaumetopoea pityocampa nests. This is due to its high accuracy, its compatibility with Roboflow datasets and its efficiency in processing high-resolution pre-processed images generated using the MSPA (Multi-Scale Patch Analysis) technique.

- **Roboflow**

Roboflow is an easy-to-use cloud-based platform aimed at streamlining dataset creation and preparation for computer vision projects. It facilitates efficient image structuring, annotation and pre-processing as an individual or part of a team. Through integrated data augmentation features (image rotation, flipping and color modification), Roboflow helps to increase the diversity of datasets, which is central to training robust deep learning models. It also exports data in other formats compatible with top frameworks such as YOLO, TensorFlow and COCO.[65]

Figure 4. 3:Roboflow logo[66]

in this project Roboflow is used in preparing the dataset used to train the YOLOv11 object detection model for this project. It orchestrated the whole process of data preparation from human annotation through augmentation to get images ready for efficient training and testing within the YOLOv11 pipeline.

- **Kaggle**

Kaggle is an interactive web platform that offers machine learning competitions in data science. The platform provides free datasets, notebooks and tutorials that data scientists need to carry out their machine learning projects. [67]

Figure 4. 4:kaggle logo [68]

For this project, Kaggle was used during the early stages of model training, particularly before applying the Multi-Scale Patch Analysis (MSPA) method.

4.3 System Workflow and Platform

- **Overall Pipeline**

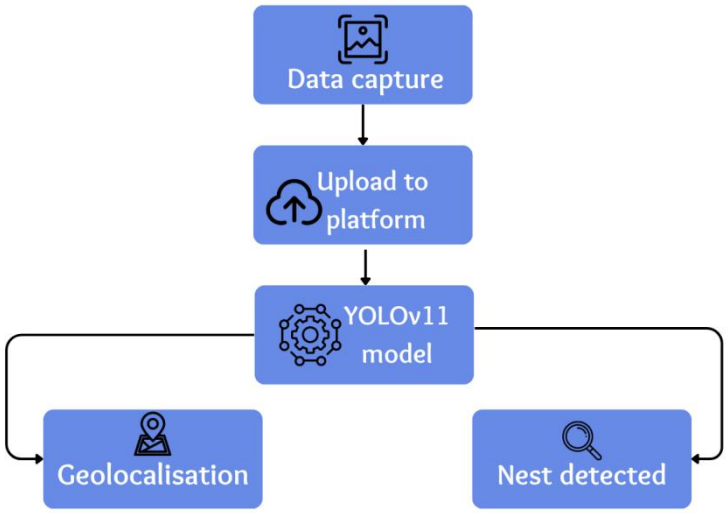


Figure 4. 5:Workflow of the PPM nests Detection System

1. **Data capture:** Images or videos are captured in the field using drones, smartphones or digital cameras. This media may include GPS metadata, which allows each capture to be located.

2. Uploading to the platform: The captured files are then uploaded to the GreenGuard platform. This step prepares the data for automatic processing.

3. Processing by the YOLOv11 model: The uploaded data is analysed by a YOLOv11 artificial intelligence model that has been specifically trained to detect pine processionary caterpillar nests. The model identifies the nests and generates bounding boxes accompanied by confidence scores.

4. Geolocation: If the images contain embedded GPS data, the system automatically extracts it. This allows each detection to be associated with actual geographic coordinates.

5. Nest detection: The system displays the detection results, including:

- images annotated with the detected nests.
- confidence scores.

- **User Platform**

The system GreenGuard provides a user-friendly interface that can be accessed via a web platform or a mobile application. Users upload images or videos captured by drones, smartphones or cameras. Next, the system automatically detects nests and extracts GPS coordinates if available to display the results . The interface includes visual overlays of detected nests and interactive map views for geolocation.

Below are screenshots showing the home page of the platform and the step-by-step process of how users interact with the system :

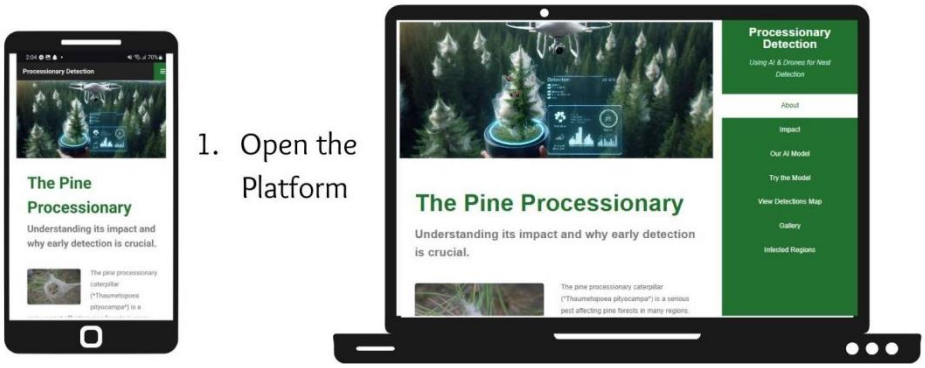


Figure 4. 6:Step 1- Access the Interface

This screen shows the initial interface of the GreenGuard platform. The user accesses the home page where they are welcomed with options to explore features or test the nest detection model.

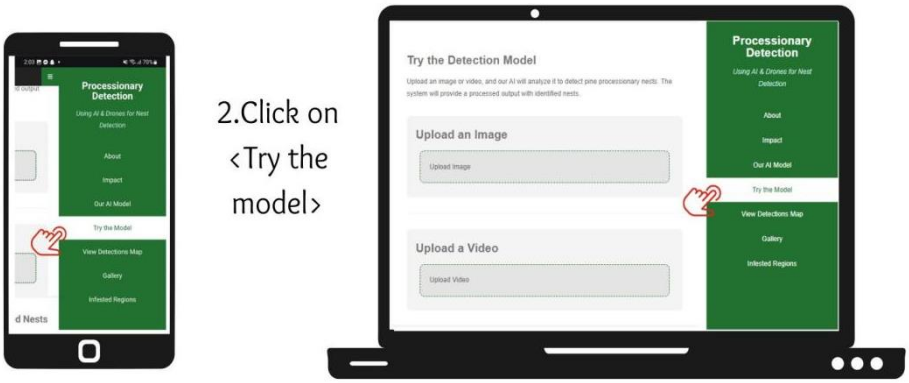


Figure 4. 7:Step 2 - Select the Option "Try the Model"

In this step, the user clicks on the “Try the Model” button to begin testing the YOLOv11-based detection system. This option allows them to upload an image or video for analysis.

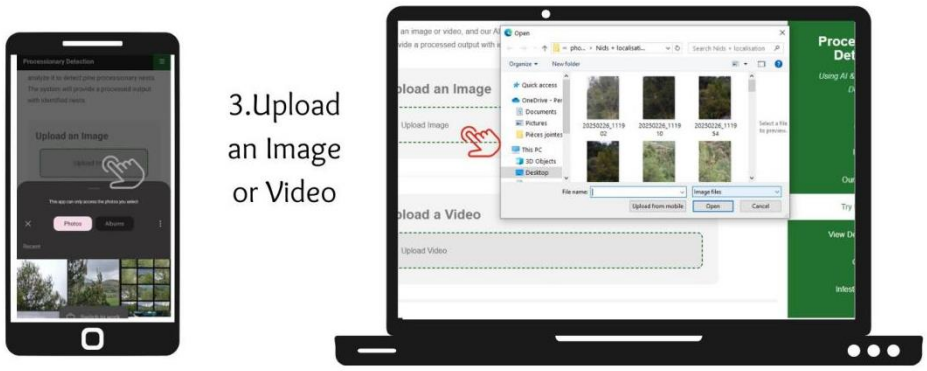


Figure 4. 8:Step 3 - Upload an Image or Video

The user uploads a media file (image or video) captured via drone, smartphone, or camera. The uploaded file may contain GPS metadata, which will be extracted automatically during processing.

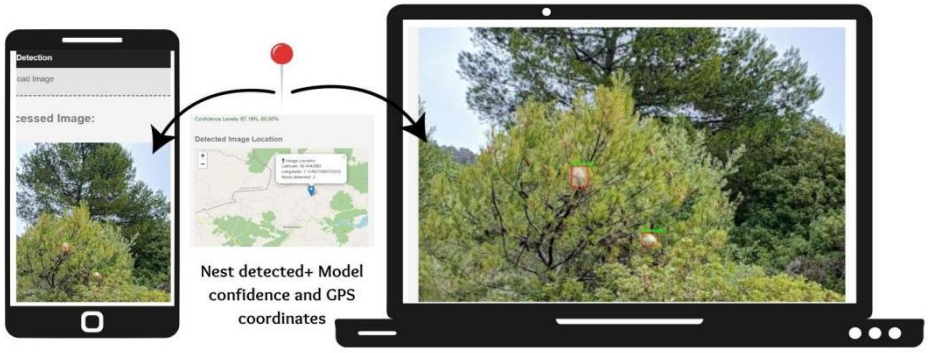


Figure 4. 9:Step 4 – View Detection Results (Image + GPS Coordinates)

Once the media is uploaded and processed, the system displays the results directly on the image. The detected nests are highlighted using bounding boxes, and the associated GPS coordinates (if available) are shown.

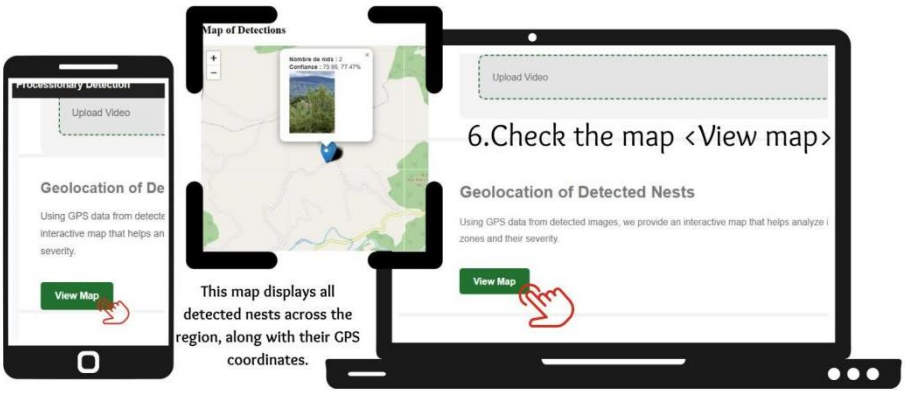


Figure 4. 10:Step 5 - View Interactive Map of All Detections

In this step, the user accesses a dynamic map displaying all the processed images and their corresponding detection coordinates. Each marker on the map represents a detection, and clicking on a marker displays the associated image and details.

4.4 Model Training and Validation

4.4.1 Evaluation Metrics

For YOLO models evaluation metrics, the mAP50 measures are key indicators of accuracy, showing how well the model is able to detect objects. Precision and recall measures provide a more accurate assessment of the model's effectiveness, balancing out false positives and missed detections. Consistently high values for these measures indicate strong model performance in object detection tasks, and we explain them below.

- **True Positive (TP):**
  A correct detection. The model predicts an object, and it actually exists in that location.
- **False Positive (FP):**
  An incorrect detection. The model predicts an object where there is none (a wrong detection).
- **False Negative (FN):**
  A missed detection. An object exists in the image, but the model fails to detect it.

Precision : Measures how often a deep learning model like YOLO correctly predicts positive instances. It is calculated by dividing the number of true positives (correct detections) by the total number of predicted positives, which includes both true and false

positives. In object detection, this is often referred to as Precision (B), where “B” stands for bounding boxes. A high precision indicates that the model effectively identifies relevant objects while minimizing false detections.[73]

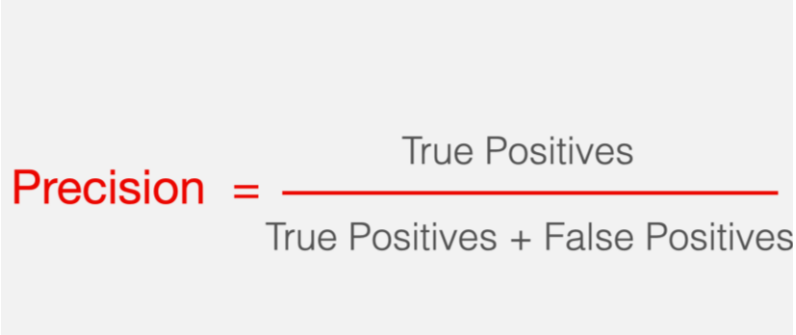


Figure 4. 11:Illustration of Precision in Object Detection [70]

Recall : Measures the ability of a deep learning model like YOLO to detect all relevant objects in an image [71]. It reflects the completeness with which the model identifies instances of the target class. Calculated as the ratio of true positives to the sum of true positives and false negatives, it aims to minimize missed detections. In object detection, recall (B) stands for bounding boxes. High recall indicates that the model is effective in capturing most objects of interest, even if it also makes some incorrect predictions (false positives).[72]

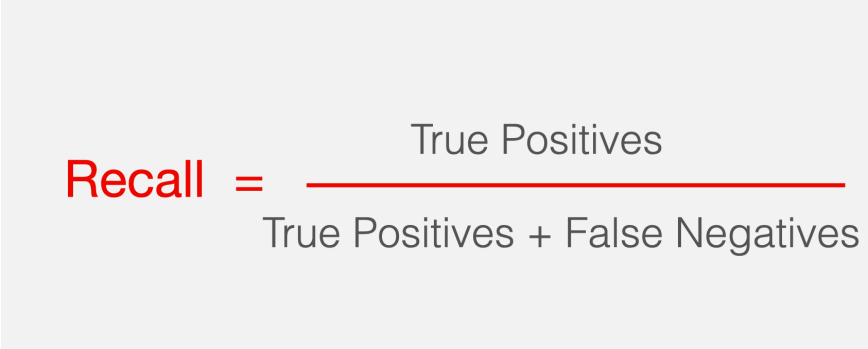


Figure 4. 12:Illustration of recall in Object Detection [73]

The mAP50 : (Mean Average Precision at a 0.5 Intersection over Union threshold) is a key measure for evaluating the performance of an object detector model,“B” stands for bounding boxes. It describes how the model succeeds in detecting and locating objects correctly to the extent that the overlap between the predicted box and the actual box is at least 50%

complete.[74]
In the context of YOLO, the mAP50 is often referred to as the model's “accuracy”, as it broadly reflects its ability to accurately predict the presence and position of target objects.

4.4.2 YOLOv11 Architecture and Baseline Comparison

Prior to applying the Multi-Scale Patch Analysis (MSPA), we evaluated three variants of the YOLOv11 architecture: YOLOv11n, YOLOv11s and YOLOv11m. The aim was to identify the most promising baseline in terms of accuracy, generalisation and detection performance.

Despite being the lightest model, YOLOv11n outperformed YOLOv11s and YOLOv11m in all major areas, including precision, recall, and mAP@50. It also demonstrated better generalisation and handling of small targets. Although larger in architecture and computational cost, YOLOv11s and YOLOv11m failed to achieve better results, particularly struggling to detect small or distant nests.

Metric	YOLOv11n	YOLOv11s	YOLOv11m
Precision	**0.907**	0.902	0.907
Recall	**0.801**	0.799	0.798
mAP@50	**0.889**	0.864	0.868
mAP@50–95	**0.430**	0.410	0.410
Fitness Score	**0.471**	0.455	0.455

Table 4. 1:Comparison of YOLOv11 Architectures (n, s, m) Prior to MSPA

Based on these results, we selected YOLOv11n as the optimal foundation for further enhancement using MSPA.

4.4.3 Impact of Multi-Scale Patch Analysis (MSPA)

The Multi-Scale Patch Analysis (MSPA) technique was applied as a pre-processing step to improve the model's ability to detect small or distant objects and for this case to detect pine processionary caterpillar nests in high-resolution forest images. This method consists in splitting images into overlapping patches at different scales before training, ensuring that even small or distant nests become more prominent in the model's receptive field.

Using MSPA enriched the dataset with localized image segments, enabling the model to focus on finer details that are often overlooked in a global analysis. This approach significantly improved detection accuracy, particularly in cases where nests were small, far away, or difficult to distinguish from the background. A comparison of performance before and after the application of MSPA revealed a significant improvement on several metrics:

- Precision increased , indicating fewer false positives.
- Recall improved, indicating better detection of true nests.
- MAP50 increased, which is often considered an indicator of accuracy in YOLO, indicating better localization and classification.

Metric	Before MSPA	After MSPA	Improvement
Images	143	10,946	+10,803
Instances	208	39,915	+39,707
Precision	0.907	0.949	+0.042
Recall	0.801	0.948	+0.147
mAP50	0.889	0.984	+0.095
mAP50–95	0.430	0.824	+0.394

Table 4. 2:Comparison of YOLOv11 Performance Before and After Applying MSPA

4.4.4 Performance Visualization

- **Confusion Matrix**

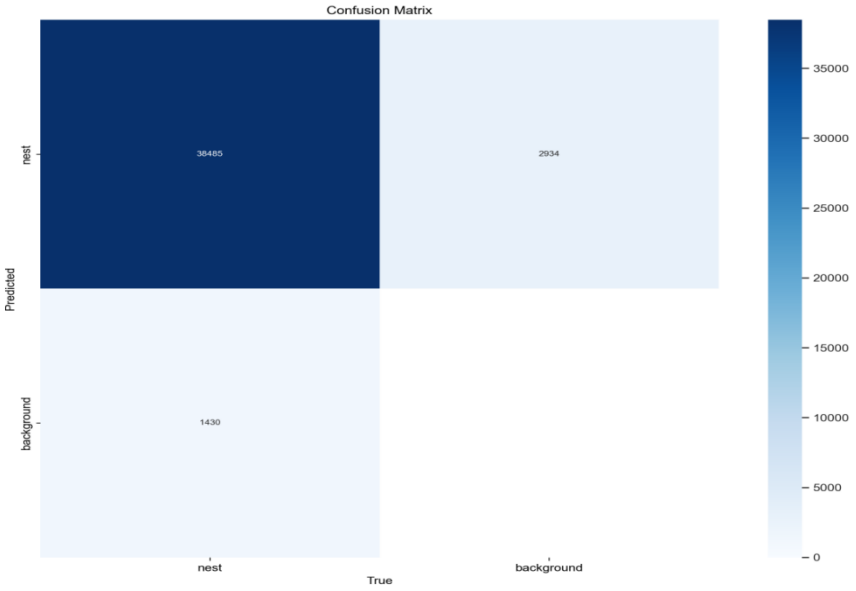


Figure 4. 13:Confusion Matrix Analysis

The confusion matrix shows how the model's predictions relate to the actual annotations. It shows the following:

- True positives (TP): 38,487 instances of nests that were correctly detected.
- False positives (FP): 2,914 instances where the model predicted a nest, but it was actually background.
- False negatives (FN): 1,428 real nests that were not detected by the model (predicted as background).

The total number of errors (FN + FP = 4,342) is low compared to the total number of real objects (39,915), indicating good overall performance.

The model correctly predicted 38,487 out of 39,915 real instances, showing high precision and recall.

- **Performance Curves**

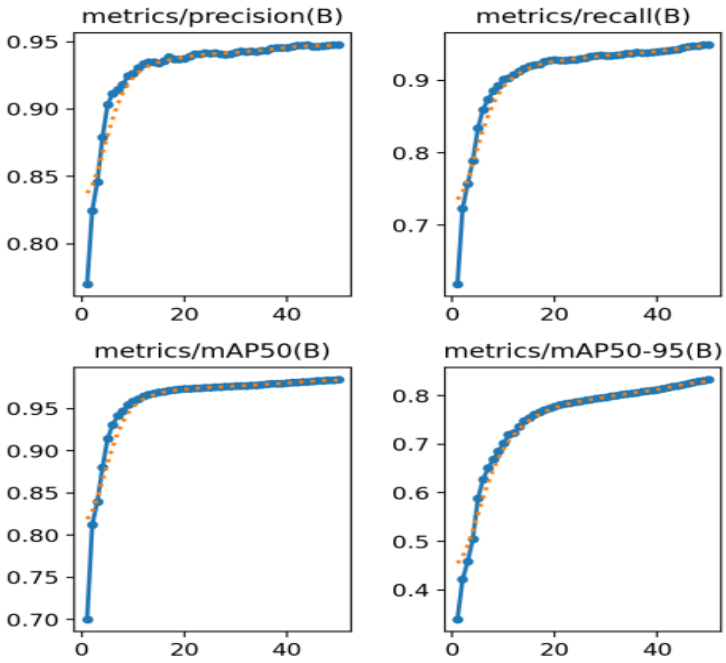


Figure 4. 14:Performance Metrics Across Epochs

- **Precision (B):** The precision curve starts at around 0.78, increases sharply during the first 15–20 epochs and then stabilizes at approximately 0.95. This indicates that, over time, the model reduces false positives, learning to make more accurate and confident detections.
- **Recall (B):** Recall starts at around 0.65 and improves consistently throughout the training process. It then plateaus just above 0.94, reflecting the model's growing capacity to identify the majority of actual nest instances with fewer missed detections as training progresses.
- **MAP50(B):** The mAP@50 curve rises from around 0.72 to approximately 0.98. This suggests that the model achieves very high detection accuracy when considering moderately overlapping predictions.
- **MAP50-95(B)**

  The mAP@50–95 curve starts at 0.40 and gradually increases to around 0.82. This indicates strong performance even under stricter IoU thresholds, confirming the model's generalization capability.

All measures show significant improvement over the first 20–30 epochs and then reach a stable point. This suggests that learning and convergence are effective and that there are no visible signs of overfitting. Although training was extended to 143 epochs, the performance measures stabilized at the 50 epochs, suggesting that this number of epochs is sufficient for optimal learning. Earlier experiments involving 100 epochs revealed that the model began to plateau without notable gains, which supports the idea that 50 epochs strike a balance between learning time and performance.

- **Analysis of labels**

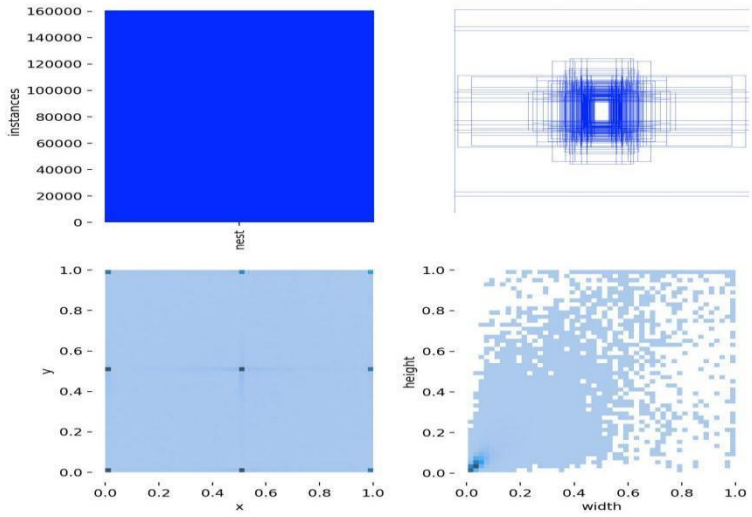


Figure 4. 15:Label Analysis: Instance Count, Position, and Box Dimensions

This figure groups together several statistical visualizations of the annotations in the dataset used after MSPA:

- Histogram of instances by class (top left): we can see that all annotated instances belong to the “nest” class, which is consistent with our goal of targeted detection.

- Bounding box distribution (top right): the blue frames represent the relative positions of the annotated boxes in the images. A concentration in the center suggests that nests appear frequently in this region, which can be exploited by the model to improve its predictions.

- Spatial distribution (bottom left): the density graph shows the location of the centers of the annotated boxes in the images. A central concentration is also visible.

- Width/height distribution (bottom right): this density map shows that the majority of boxes have low widths and heights, confirming that nests are often small and require a method like MSPA to detect them effectively.

4.5 Results and Comparison

4.5.1 Testing with Personal Images

In order to thoroughly evaluate the robustness and practical applicability of the YOLOv11 + MSPA model, tests were conducted on two categories of image.

- Field images taken with a professional camera
- Smartphone images containing GPS metadata.

1. **Detection on Personal Camera Images**

   These images were taken manually in the field using a professional camera, introducing real-world variations such as lighting changes and natural obstacles .

Image	Description
 **Figure 4. 16:Detection of Multiple Nests with Varying Sizes**	This image demonstrates the model's ability to detect multiple nests of different sizes, validating the effectiveness of the MSPA-enhanced YOLOv11.
	Shows the model’s robustness

Image	Description
 **Figure 4. 17:Long-Range Detection of Isolated Tiny Nest**	in detecting very small and remote nests that would typically be missed by standard detection systems.
 **Figure 4. 18:Clear Detection of Visible Nest**	image, captured with a camera in natural light, shows a clearly visible nest successfully detected by the model.

2. Detection on Smartphone Images with GPS Metadata

These images were captured using a smartphone equipped with GPS. The objective was to validate both the nest detection accuracy and the extraction of geolocation data for mapping purposes.

Figure 4. 19:Example of a test image with embedded GPS coordinates – Test 1

Detection of a clearly visible nests. Coordinates: 36.49446° N, 7.13500° E. The detection was accurate and the metadata was correctly parsed for visualization.

Figure 4. 20:Example of a test image with embedded GPS coordinates – Test 2

Detection of a small, distant nest. Coordinates: 36.49462° N, 7.13471° E. The result confirms the model's robustness and GPS metadata integration.

4.5.2 Comparison with Related Work

Ref.	Year	Method	Image Type	Model	Results
[11]	2023	YOLOv5, Faster R-CNN	RGB UAV (forests in Catalonia)	Object Detection	YOLOv5: mAP = 0.826 (per image), 0.696 (per nest)
[14]	2023	CNN with RGB + thermal fusion	Multimodal UAV imagery	Custom Deep CNN	Precision: 97%
Our Model	2025	YOLOv11 + MSPA	RGB UAV	YOLOv11	Precision: 95%, Recall: 94%, mAP50: 98%, mAP50–95: 82%

Table 4. 3:Comparison of Object Detection Models for PPM Nest Detection

In the field of object detection, three major deep learning models are commonly used: YOLO (You Only Look Once), Faster R-CNN, and SSD (Single Shot MultiBox Detector). YOLO stands out among these for its ability to perform real-time detection while maintaining a good balance between speed and accuracy. For our study, we opted for a model from the YOLO family rather than Faster R-CNN, based on experimental results and conclusions drawn from the literature.

In particular, article [11] demonstrated that YOLOv5 outperformed Faster R-CNN in detecting pine processionary nests, achieving an mAP of 0.826 per image compared to Faster R-CNN's 0.696. This observation led us to select YOLO models.
Our improved model, based on YOLOv11 combined with MSPA (Multi-Scale Patch Analysis), achieved excellent results using only RGB images from UAVs. It achieved a precision of 95%, a recall of 94% and an mAP50 of 98%, demonstrating good generalisability, even for small or distant nests. For comparison, article[14] used a more advanced model that combined thermal and RGB images, achieving a level of accuracy of

97%. However, this difference is negligible, particularly since our model did not utilise thermal data. Had we had access to multimodal images (RGB and thermal), our approach could potentially have outperformed current models in terms of accuracy and robustness.

4.5.3 YOLOv11 vs YOLOv8–YOLOv12

- **Comparison of YOLO Versions Before MSPA Application**

Before applying the Multi-Scale Patch Analysis (MSPA) method, we carried out a series of comparative tests to evaluate the performance of different versions of YOLO (from v8 to v12) on our original dataset. The table below summarizes the results obtained:

Model	Precision (P)	Recall (R)	mAP50	mAP50–95	Inference Time per Image
YOLOv8	0.890	0.779	0.834	0.411	2.1 ms
YOLOv9	0.840	0.808	0.832	0.408	16.6 ms
YOLOv10	0.835	0.760	0.826	0.402	4.5 ms
YOLOv11	**0.907**	**0.801**	**0.889**	**0.430**	**2.3 ms**
YOLOv12	0.858	**0.822**	0.850	0.426	4.2 ms

Table 4. 4:Comparison of YOLO Versions Before Applying MSPA

- YOLOv11 has the best accuracy (0.907) and mAP50 (0.889), while maintaining a fast inference speed (2.3 ms).
- Although YOLOv12 has a slightly higher recall, its mAP is still lower than that of YOLOv11 making it less efficient overall.
- YOLOv9 has the longest inference time, which is not optimal for real-time applications.
- Based on these findings, YOLOv11 offers the best balance between accuracy, performance and speed, justifying its initial choice.

- **Post-MSPA: Final Model Selection for Training**

Following the application of the MSPA method, our dataset has increased considerably in size and quality, particularly with regard to the detection of small, distant nests.

In view of this development, we decided to focus the training exclusively on YOLOv11 for the following reasons:

- The large volume of data generated by MSPA requires a substantial amount of training time.
- We had access to high-performance machines (at LabSTIC university of Guelma – university resources), but efficient use of time was essential.
- YOLOv11 had already proved superior to other YOLO versions in preliminary tests, so there was no need to re-train all versions with MSPA.

We continued training with YOLOv11, adjusting the hyperparameters to obtain the following final results:
Accuracy = 95%, recall = 94%, mAP50 = 98%, mAP50-95 = 82% and this only from RGB images.

In comparison, work such as [14] using RGB + thermal fusion achieved an accuracy of 97%, demonstrating the competitiveness of our model. It is likely that with access to thermal images, our model could have achieved even better results.

4.6 Discussion

The final results obtained using the improved YOLOv11 model combined with Multi-Scale Patch Analysis (MSPA) method confirm the effectiveness of our detection pipeline. The model performed excellently, achieving a mAP@50 of 0.984, a precision of 0.95, a recall of 0.948 and an mAP@50–95 of 0.824. These scores demonstrate the model's high reliability in detecting Thaumetopoea pityocampa nests, including the smallest or partially hidden ones, which are often overlooked by conventional models.

Compared to previous versions of YOLO (v8 to v12), YOLOv11 demonstrated the most consistent and stable performance across all evaluation criteria. Although YOLOv12 achieves a similar mAP@50, it exhibits poorer generalisation and requires more training time. YOLOv8 and v9, on the other hand, struggled to detect small or distant nests, even after data augmentation.

Tests carried out on personal images taken in the field validated the model's robustness in real-world conditions. Nests were successfully detected despite environmental variations, such as shadows, changes in lighting, and complex backgrounds.

It is also important to note that training beyond 50 epochs did not result in any significant improvements. In fact, initial tests with 100 epochs led to slight overfitting and inconsistent results. Thus, 50 epochs were selected as the optimal point, ensuring a good balance between training time, convergence, and generalization.

In summary, combining YOLOv11 with MSPA proved very effective and generalisable. This approach provides a solid foundation for future enhancements, including real-time integration on drones and integration with attention-based or segmentation modules to enable even more precise detection in complex environments.

4.7 Perspectives

As a follow-up to this work, the GreenGuard prototype was presented to the Forest Conservation Department. The system was well received by the managers and engineers in attendance, who showed a keen interest in the proposed solution.

Figure 4. 21:Discussion Session with Forestry Engineers[75]

The meeting provided an opportunity to open a direct dialogue with stakeholders in the field, gather relevant feedback and consider how the system could be adjusted to meet the specific needs of the forestry sector. Among the prospects discussed were:

- Extending the approach to the detection of other environmental threats such as pests,

diseases, and fires.

- As well as the integration of connected sensors (IoT) to improve the accuracy and richness of the data collected in the field.

In the medium term, GreenGuard could evolve into a comprehensive, operational solution, supported by local institutions and capable of being deployed on a large scale in Algerian forests. This prototype thus provides a solid basis for further development, both technologically (e.g real-time integration on drones) and structurally (partnerships, calls for projects, fundraising).

4.8 Conclusion

The integration of YOLOv11 with the Multi-Scale Patch Analysis (MSPA) method has enabled us to design a high-performance, robust solution for the detection of Thaumetopoea pityocampa nests. The results obtained in terms of precision, recall and mAP testify to the maturity of the system, capable of effectively identifying even small or partially concealed nests, generally ignored by conventional approaches. Experiments carried out on personal images, captured in real-life conditions, have confirmed the system's ability to generalize to complex and varied natural environments.

Beyond these encouraging performances, this approach opens up some interesting prospects. A natural evolution of the system would be to integrate it into embedded platforms, such as drones, for automated, real-time forest monitoring. In the longer term, adapting the model to the detection of other types of biological threat, in both forest and agricultural environments, could lead to a global solution for intelligent and sustainable environmental monitoring.

General conclusion

This thesis provided an integrative solution to forest health monitoring,paying special attention to the identification of pine processionary moth (PPM) nests. It began by emphasising the importance of forest ecosystems and the threats they face. Traditional surveillance methods proved inadequate in scalability and efficiency, thus emphasizing the need for smart, automated systems.

To overcome these limitations, a system was developed that utilises advanced visual analysis techniques by combining the YOLOv11 detection model with the Multi-Scale Patch Analysis (MSPA) technique. These developments significantly improved the detection of small and distant nests at high accuracy and recall scores. The system was deployed as a web and mobile platform offering detection and geolocation features to support researchers, forest managers, and environmental monitoring officers.
Extensive tests and comparisons with existing models showing the solution's reliability, speed, and suitability for practical field use.

In the near future, further work will focus on improving the system's adaptation capabilities, including drone-based surveillance, enriching the dataset with even more diverse examples, and rendering it offline-capable with minimal deployment.These future enhancements will allow more efficient and sustainable strategies in forest surveillance and ecosystem preservation.
This methodology can be extended beyond the detection of Thaumetopoea pityocampa to include other ecological threats, such as different types of insect nests (e.g. Lymantria dispar caterpillar tent nests) and signs of forest diseases (e.g. fungal infections, bark beetle damage and needle discolouration).

More broadly, this research contributes to the growing use of artificial intelligence to protect biodiversity and natural habitats. Combining data-driven approaches with real-time monitoring tools makes it possible to develop early warning systems for forest health, enabling faster, more sustainable responses to ecological threats.

Bibliography

[4] ScienceDirect, “Factors affecting pine processionary moth (Thaumetopoea pityocampa) incidence in Mediterranean pine stands: A multiscale approach,”. Available: https://www.sciencedirect.com/science/article/pii/S0378112722007228.

[8] ScienceDirect, “When insect pests build their own thermal niche: The hot nest of the pine processionary moth,”. Available: https://www.sciencedirect.com/science/article/abs/pii/S0306456521001145.

[9] ResearchGate, “Testing early detection of pine processionary moth Thaumetopoea pityocampa nests using UAV-based methods,”. Available: https://www.researchgate.net/publication/370883294_Testing_early_detection_of_pine_processionary_moth_Thaumetopoea_pityocampa_nests_using_UAV-based_methods.

[11] University of Pennsylvania, “Weighted Training for Cross-Task Learning,”. Available: https://cogcomp.seas.upenn.edu/files/presentations/CCHRS22_presentation.pdf.

[12] ScienceDirect, “Testing early detection of pine processionary moth Thaumetopoea pityocampa nests using UAV-based methods,”. Available: https://www.sciencedirect.com/org/science/article/pii/S1619003323000208.

[13] ResearchGate, “Detection and Mapping of Pine Processionary Moth Nests in UAV Imagery of Pine Forests Using Semantic Segmentation,”. Available: https://www.researchgate.net/publication/337272857_Detection_and_Mapping_of_Pine_Processionary_Moth_Nests_in_UAV_Imagery_of_Pine_Forests_Using_Semantic_Segmentation.

[14] IEEE Xplore, “An Eco-Friendly Fight Against Thaumetopoea Pityocampa Infestations in Pine Forests Using Deep Learning on UAV Imagery,”. Available: https://ieeexplore.ieee.org/document/9925556.

[15] ResearchGate, “Deep-Pest-Detector: Automated Detection and Localization of Processionary Moth Nests on Pine Trees via Aerial Drones and Deep Neural Networks,”. Available: https://www.researchgate.net/publication/359880854_Deep-Pest-

Detector_Automated_Detection_and_Localization_of_Processionary_Moth_Nests_on_Pine_Trees_via_Aerial_Drones_and_Deep_Neural_Networks.

[16] PubMed Central, “Palm tree disease detection and classification using residual network and transfer learning of inception ResNet,”. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC9980777/.

[17] MDPI, “VddNet: Vine Disease Detection Network Based on Multispectral Images and Depth Map,”. Available: https://www.mdpi.com/2072-4292/12/20/3305.

[18] TechScience, “Deep Learning-Based Trees Disease Recognition and Classification Using Hyperspectral Data,”. Available: https://www.techscience.com/cmc/v77n1/54449/html.

[19] MDPI, “MobiRes-Net: A Hybrid Deep Learning Model for Detecting and Classifying Olive Leaf Diseases,”. Available: https://www.mdpi.com/2076-3417/12/20/10278.

[20] PubMed Central, “Precision Detection and Assessment of Ash Death and Decline Caused by the Emerald Ash Borer Using Drones and Deep Learning,”. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC9964414/.

[37] arXiv, “YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain,”. Available: https://arxiv.org/html/2406.10139v1.

[46]43 arXiv, “YOLOv6 Preprint,”. Available: https://arxiv.org/html/2412.13006v1#:~:text=YOLOv6%20is%20a%20PyTorch%2Dbased.

[52]49 arXiv, “YOLOv11: An Overview of the Key Architectural Enhancements,”. Available: https://arxiv.org/html/2410.17725v1.

[63]59 ResearchGate, “Smart Farming Solutions: Automated Crop and Plantation Disease Detection,”. Available: https://www.researchgate.net/publication/385554956_Smart_Farming_Solutions_Automated_Crop_and_Plantation_Disease_Detection.

Webography

[1] Dryad.net, “Forest health monitoring explained,”. Available: https://www.dryad.net/post/forest-health-monitoring-explained. [Accessed: **march. 20, 2025**].

[2] U.S. Forest Service, “Forest Health Protection Monitoring,”. Available: https://www.fs.usda.gov/science-technology/forest-health-protection/monitoring. [Accessed**: march. 20, 2025**].

[3] The Independent, “Import controls reviewed after highly destructive pine tree pest found in UK,”. Available: https://www.independent.co.uk/climate-change/news/defra-woodland-trust-england-france-europe-b2060824.html. [Accessed: **march. 20, 2025**].

[5] Jardin Sostenible, “La lucha contra la procesionaria del pino,”. Available: https://jardinsostenible.eu/en/la-lucha-contra-la-procesionaria-del-pino. [Accessed: **march. 21, 2025**].

[6] Pharma-Domicile, “Urticaire à cause des chenilles processionnaires,”. Available: https://www.pharma-domicile.be/actualites/urticaire-a-cause-des-chenilles-processionnaires-que-faire-pour-me-soulager. [Accessed: **march. 21, 2025**].

[7]Forest Research, “Pine processionary moth (Thaumetopoea pityocampa)”. Available: https://www.forestresearch.gov.uk/tools-and-resources/fthr/pest-and-disease-resources/pine-processionary-moth-thaumetopoea-pityocampa/?utm_source=chatgpt.com [Accessed: **march. 21, 2025**].

[10] V7 Labs, “What is computer vision,”. Available: https://www.v7labs.com/blog/what-is-computer-vision. [Accessed: **march. 22, 2025**].

[21] Simplilearn, “https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/ai-vs-machine-learning-vs-deep-learning,”. Available: https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/ai-vs-machine-learning-vs-deep-learning. [Accessed: **march. 27, 2025**].

[22] DataCamp, “What is Machine Learning,”. Available: https://www.datacamp.com/blog/what-is-machine-learning. [Accessed: **march. 28, 2025**].

[23] TechVidvan, “Reinforcement Learning Tutorial,”. Available: https://techvidvan.com/tutorials/reinforcement-learning/. [Accessed: **march. 28, 2025**].

[24] Towards AI, “Unboxing Weights & Biases,”. Available: https://pub.towardsai.net/unboxing-weights-biases-loss-hone-in-on-deep-learning-523e659eac80. [Accessed: **march. 28, 2025**].

[25] Coursera, “Deep Learning vs. Machine Learning,”. Available: https://www.coursera.org/articles/ai-vs-deep-learning-vs-machine-learning-beginners-guide. [Accessed: **march. 29, 2025**].

[26] DataCamp, “An Introduction to Convolutional Neural Networks (CNNs),”. Available: https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns. [Accessed: **march. 30, 2025**].

[27] DataAspirant, “How RNN Works,”. Available: https://dataaspirant.com/how-recurrent-neural-network-rnn-works/. [Accessed: **march. 30, 2025**].

[28] Medium, “Generative Adversarial Networks,”. Available: https://medium.com/@marcodelpra/generative-adversarial-networks-dba10e1b4424. [Accessed: **march. 30, 2025**].

[29] MathWorks, “Getting Started with Object Detection Using Deep Learning,”. Available: https://www.mathworks.com/help/vision/ug/getting-started-with-object-detection-using-deep-learning.html. [Accessed: **march. 30, 2025**].

[30] MathWorks, “Object Detection Overview,”. Available: https://www.mathworks.com/discovery/object-detection.html. [Accessed: **april. 5, 2025**].

[31] DataCamp, “YOLO Object Detection Explained,”. Available: https://www.datacamp.com/blog/yolo-object-detection-explained. [Accessed: **april. 5, 2025**].

[32] Medium, “Faster R-CNN vs YOLO vs SSD Object Detection Algorithms,”. Available: https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc. [Accessed: **april. 5, 2025**].

[33] V7 Labs, “YOLO Object Detection,”. Available: https://www.v7labs.com/blog/yolo-object-detection. [Accessed: **april. 6, 2025**].

[34] Viso.ai, “YOLO Explained,”. Available: https://viso.ai/computer-vision/yolo-explained/. [Accessed: **april. 7, 2025**].

[35] Ultralytics, “Benefits of Ultralytics YOLO11 Being Anchor-Free,”. Available: https://www.ultralytics.com/ar/blog/benefits-ultralytics-yolo11-being-anchor-free-detector. [Accessed: **april. 7, 2025**].

[36] MathWorks, “Getting Started with YOLO v2,”. Available: https://www.mathworks.com/help/vision/ug/getting-started-with-yolo-v2.html. [Accessed: **april. 7, 2025**].

[38] DataCamp, “YOLO Object Detection Explained,”. Available: https://www.datacamp.com/blog/yolo-object-detection-explained. [Accessed: **april. 9, 2025**].

[39] Viso.ai, “YOLO Explained,”. Available: https://viso.ai/computer-vision/yolo-explained/. [Accessed: **april. 9, 2025**].

[40] Ultralytics, “Benefits of Ultralytics YOLO11 Being Anchor-Free,”. Available: https://www.ultralytics.com/ar/blog/benefits-ultralytics-yolo11-being-anchor-free-detector. [Accessed: **april. 10, 2025**].

[41] Viso.ai, “YOLOv4 Overview,”. Available: https://viso.ai/deep-learning/yolov4/. [Accessed: **april. 10, 2025**].

[42] GitHub, “Ultralytics YOLOv5 Issues,”. Available: https://github.com/ultralytics/yolov5/issues/280. [Accessed: **april. 11, 2025**].

[44] ResearchGate, “YOLOv6 Network Architecture,”. Available: https://www.researchgate.net/figure/YOLOv6-network-architecture_fig2_374632617. [Accessed: **april. 12, 2025**].

[45] Roboflow, “YOLOv7 Breakdown,”. Available: https://blog.roboflow.com/yolov7-breakdown/. [Accessed: **april. 13, 2025**].

[46] Ultralytics, “YOLOv8 Documentation,”. Available: https://docs.ultralytics.com/models/yolov8/#overview. [Accessed: **april. 15, 2025**].

[47] ResearchGate, “YOLOv8 Detailed Architecture,”. Available: https://www.researchgate.net/figure/Detailed-architecture-of-YOLOv8-showcasing-the-backbone-networks-multiple-convolutional_fig3_385510373. [Accessed: **april. 15, 2025**].

[48] YOLOv8.org, “YOLOv8 Architecture,”. Available: https://yolov8.org/yolov8-architecture/. [Accessed: **april. 15, 2025**].

[50] Medium, “YOLOv11 Architecture Explained: Next-Level Object Detection with Enhanced Speed and Accuracy,”. Available: https://medium.com/@nikhil-rao-20/yolov11-explained-next-level-object-detection-with-enhanced-speed-and-accuracy-2dbe2d376f71. [Accessed: **april. 15,2025**].

[51] Lightly.ai, “YOLO and Active Learning,”. Available: https://www.lightly.ai/blog/yolo. [Accessed: **april. 16, 2025**].

[52] Medium, “Understanding and Implementing Faster R-CNN,”. Available: https://medium.com/@RobuRishabh/understanding-and-implementing-faster-r-cnn-248f7b25ff96. [Accessed: **april. 17, 2025**].

[53] ArcGIS Developers, “How SSD Works,”. Available: https://developers.arcgis.com/python/latest/guide/how-ssd-works/. [Accessed: **april. 18, 2025**].

[54] Ultralytics, “Ultralytics YOLO11 Has Arrived,”. Available: https://www.ultralytics.com/blog/ultralytics-yolo11-has-arrived-redefine-whats-possible-in-ai. [Accessed: **april. 25, 2025**].

[55] Ultralytics, “YOLO11 and Computer Vision for Environmental Conservation,”. Available: https://www.ultralytics.com/blog/ultralytics-yolo11-and-computer-vision-for-environmental-conservation. [Accessed: **april. 26, 2025**].

[56] Viso.ai, “YOLOv11: Components and Explanation,”. Available: https://viso.ai/computer-vision/yolov11/#components-of-yolov11. [Accessed: **april. 27, 2025**].

[57] Roboflow. “Nids de chenilles”. Available: https://universe.roboflow.com/caterpillar-ifizj/nids-de-chenilles/dataset/5/download. [Accessed: **april. 27, 2025**].

[58] Roboflow, “YOLOv11 PyTorch Format,”. Available: https://roboflow.com/formats/yolov11-pytorch-txt. [Accessed: **april. 28, 2025**].

[60] Roboflow, “Nids de Chenilles Dataset,”. Available: https://universe.roboflow.com/caterpillar-ifizj/nids-de-chenilles/dataset/5. [Accessed: **april. 29, 2025**].

[61] Python.org, “Welcome to Python.org,”. Available: https://www.python.org. [Accessed: **april. 29, 2025**].

[62] Logos-world.net, “Python Logo,”. Available: https://logos-world.net/python-logo/. [Accessed: **april. 29, 2025**].

[63] Ultralytics, “Documentation en Français,”. Available: https://docs.ultralytics.com/fr/. [Accessed: **may. 4, 2025**].

[64] MWC Barcelona, “Ultralytics Exhibitor Page,”. Available: https://www.mwcbarcelona.com/exhibitors/30643-ultralytics. [Accessed: **may. 5, 2025**].

[65] Roboflow Docs, “Roboflow Documentation,”. Available: https://docs.roboflow.com/. [Accessed: **may. 5, 2025**].

[66] Roboflow Security, “Security Portal,”. Available: https://security.roboflow.com/. [Accessed: **may. 6, 2025**].

[67] DataScientest, “Kaggle: Tout ce qu’il faut savoir,”. Available: https://datascientest.com/kaggle-tout-ce-quil-a-savoir-sur-cette-plateforme. [Accessed: **may. 6, 2025**].

[68] Kaggle, “Brand Guidelines,”. Available: https://www.kaggle.com/brand-guidelines. [Accessed: **may. 7, 2025**].

[69] Kaggle, “Getting Started Discussion,”. Available: https://www.kaggle.com/discussions/getting-started/523677. [Accessed: **may. 8, 2025**].

[70] EvidentlyAI, “Precision, Recall, Accuracy Explained,”. Available: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall#what-is-precision. [Accessed: **may. 15, 2025**].

[71] Ultralytics, “YOLO Performance Metrics: Class-wise Metrics,”. Available: https://docs.ultralytics.com/fr/guides/yolo-performance-metrics/#class-wise-metrics. [Accessed: **may. 20, 2025**].

[72] Kaggle, “Getting Started Discussion,”. Available: https://www.kaggle.com/discussions/getting-started/523677. [Accessed: **may. 20, 2025**].

[73] EvidentlyAI, “Precision, Recall, Accuracy Explained,”. Available: https://www.evidentlyai.com/classification-metrics/accuracy-precision-recall#what-is-recall. [Accessed: **june. 1, 2025**].

[74] Ultralytics, “YOLO Performance Metrics: Class-wise Metrics,”. Available: https://docs.ultralytics.com/fr/guides/yolo-performance-metrics/#class-wise-metrics. [Accessed: **june. 1, 2025**].

[75] Facebook, Available: https://www.facebook.com/100064700888209/posts/pfbid03462W8R8VZX1SvJDqv8oVJjXsa7YKzbUNmAGQTDbJ2pK98DwBYBAjL6YHh6h68yRkl/?app=fbl. [Accessed: **june. 10, 2025**].