PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

THE MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

University of Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculty of Sciences and Technology

Department Electronic

Memory

Presented for the fulfillment of the MASTER degree

FILIERE: Telecommunication

Specialty: Systheme de Telecommunication

By

Khelfa Hadil

Belgroune Bouchra

Entitled

Breast Tumor detection in mamogramme image using Yolo models

Supported on: 12/06/2025

Before the jury composed of:

Name	*Grade*	*Quality*	*Establishment*
SID AHMED Soumia	Dr	Supervisors	Univ-BBA
MESSALI Zoubeida	Pr	Sub-supervisor	Univ-BBA
BENREDOUANE Abdellah	Dr	Président	Univ-BBA
BELGUIDOUME Khaoula	Dr	Examiner	Univ-BBA

College Year 2024/2025

Acknowledgments

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

We extend our sincere gratitude to Allah, who blessed us with health, strength, determination, and patience to complete this work. We thank our supervisor, ***Dr. SID AHMED Soumia*** *for her invaluable guidance and support throughout our academic journey. Special thanks to our sub-supervisor,* ***Pr. MESSALI Zoubeid.***

We express our deep appreciation to the honorable members of the examination committee for their valuable time and constructive feedback. We are also grateful to all the respected professors who enriched our academic path with their knowledge and guidance.

We cannot overlook the contribution of all those who directly or indirectly supported us in this endeavor. Particularly, we would like to thank our friends, for their significant support and encouragement.

This achievement wouldn't have been possible without the efforts and dedication of everyone involved. We pray that Allah rewards every one of you abundantly and places this work among our good deeds.

Peace be upon you all.

Dedication

To my dearest family and friends,

Today is a milestone in my life, and I attribute this achievement to the grace of God and the incredible support and love from all of you.

***Mom** and **Dad**, your unconditional love, sacrifices, and guidance have been the cornerstone of my success. I dedicate this achievement to you with deep gratitude and love.*

My siblings, love, support, and companionship have been my source of strength. Every one of you has played a vital role in this journey, and I am grateful to all of you.

To all my uncles and aunts, thank you for your constant support and love. You have all contributed to my growth and success in countless ways.

*Special thanks to my partner, **Belgroune bouchra***

This achievement is a reflection of the collective love, support, and guidance from each of you. I am forever grateful to all of you.

With all my love and appreciation.

Khelfa Hadil

Dedication

*Dear **my parents**, without their support and encouragement, I would not be where I am today. This achievement is the fruit of your hard work and sacrifices, so you have all my love and gratitude. I dedicate this work to you with pride and honor*

dear my sisters.

I dedicate this work to you with love and gratitude, as an expression of how much I love and appreciate you as my dear sisters

***dear my family**, the source of love, tenderness and unlimited support, and the strong support in all stages of my life*

***Dear my friends** who have shared moments of joy and sorrow, stood by me in times of difficulty and success, and have been my help and support, never skimping on their advice and encouragement.*

*Special thanks to my partner **Khelfa Hadil***

In the end, I thank God Almighty for his success and help, and I thank everyone who supported me and supported me in my life from near and far.

Belgroune Bouchra

Table of Contents

List of Figures

List of Tables

List of Equation

List of Acronyms

AI: Artificial Intelligence

DL: Deep Learning

ML: Machine Learning

CNN: Convolution Neural Network

ConvNets: Convolutional Networks

FC: Fully Connected

IOU: Intersection over Union

Fast R-CNN: Fast Region-based Convolutional Neural Network

Faster R-CNN: Faster Region-based Convolutional Neural Network

CPU: Central Processing Units

GPU: Graphic Processor Unit

CSV: Comma-separated values

MAP: Mean Average Precession

Abstract

Computer vision is one of the main applications of artificial intelligence, and object detection is among its most important uses, especially in the medical field. This research focused on two techniques, YOLO and CNN, applied to breast cancer detection. A CNN model was tested on our dataset, but due to its insufficient accuracy, the focus was shifted to YOLO-based techniques. We used several models from the YOLO family, including YOLOv5, YOLOv8, and YOLOv12, and compared their performance to evaluate their effectiveness in detecting breast tumors and analyzing the obtained results.

Résumés

La vision par ordinateur est l'une des principales applications de l'intelligence artificielle, et la détection d'objets représente l'un de ses usages les plus importants, notamment dans le domaine médical. Ce mémoire port sur deux techniques : YOLO et CNN, appliquées à la détection du cancer du sein. Un modèle CNN a été testé sur notre ensemble de données, mais en raison d'une précision insuffisante, l'effort a été concentré sur les techniques de la famille YOLO. Nous avons utilisé plusieurs modèles, notamment YOLOv5, YOLOv8 et YOLOv12, et comparé leurs performances afin d'évaluer leur efficacité dans la détection des tumeurs mammaires et d'analyser les résultats obtenus.

ملخص

تُعدّ الرؤية الحاسوبية من التطبيقات الرئيسية في مجال الذكاء الاصطناعي، ويُعدّ كشف الأشياء من أهم استخداماتها، خصوصًا في المجال الطبي. تناولت هذه الأطروحة تقنيتين هما YOLOوCNN، وذلك بهدف الكشف عن سرطان الثدي .تم تطبيق نموذج CNN على مجموعة البيانات الخاصة بنا ولكن نظرًا لعدم تحقيقه دقة كافية تم تركيز الجهد على تقنيات YOLO استخدمنا عدة نماذج من عائلةYOLO ، بما في ذلك YOLOv5، YOLOv8، وYOLOv12، وقمنا بمقارنة أدائها لتقييم فعاليتها في اكتشاف أورام الثدي وتحليل النتائج المتحصّل عليها.

Key words: Artificial intelligence, Deep learning, Machine learning, CNN, YOLO

General Introduction

General Introduction

Artificial intelligence (AI) is merely the beginning of a profound technological transformation. As AI continues to evolve, it offers unprecedented advancements in efficiency, automation, and problem-solving capabilities across industries. From healthcare to transportation, finance to education, breakthroughs in machine learning algorithms and neural networks have revolutionized fields like computer vision, speech recognition, and decision-making systems. These advancements enable AI to analyze vast amounts of data with remarkable speed and accuracy.

A prime example of AI's transformative potential is object detection, which leverages advanced algorithms and deep learning techniques to analyze images or videos by identifying and localizing objects of interest. This capability has broad applications across industries. In healthcare, for instance, object detection aids in diagnosing diseases through medical image analysis, enabling earlier detection and improved patient outcomes.

Our research thesis focuses on training and comparing multiple YOLO models specifically YOLOv5, YOLOv8, and YOLOv12 on a breast cancer dataset. We trained these models on mammography images annotated for breast cancer detection and then conducted a comparative analysis of their performance. This approach allows us to evaluate the advancements across different YOLO versions in detecting breast cancer lesions, our work is structured into three chapters:

1. Chapter 1: An overview of deep learning with a focus on CNN architecture.
2. Chapter 2: An in-depth exploration of object detection techniques.
3. Chapter 3: A comparative result's implementation of Yolo models.

Chapter 1: Deep Learning

Summary

This chapter provides a presentation to the fundamental concepts of deep learning and neural network , which are key components of modern Artificial Intelligence (AI). Additionally, it explores Deep Learning, focusing specifically on the Convolutional Neural Networks (CNNs).

Table of Contents

1.1 Introduction

Artificial intelligence (AI) is the scientific discipline of creating computer programs that perform operations comparable to those of the human mind, it has become among the most interesting fields of computer science, due to its efficiency in solving very complex problems and automating day-to-day tasks that were once considered hard, rather impossible for a machine to comprehend. A lot of the credit for that goes to ML and DL.

In this chapter, we will talk about Artificial Intelligence (AI) then we will explore Deep Learning, including neural networks and convolutional neural networks (CNNs).

1.2 Artificial intelligence

Artificial Intelligence (AI) refers to the capability of robots or computer programs to perform tasks that typically require human intelligence, such as learning, problem-solving, and decision-making. AI is often used as a broad term that encompasses Machine Learning (ML) and Deep Learning (DL), although these terms are not interchangeable.

Machine Learning is a subset of AI that uses statistical methods to enable machines to learn from data without explicit programming. It involves teaching machines to recognize patterns and predict outcomes based on data. Deep Learning, a subset of ML, utilizes multi-layer artificial neural networks to learn from data. Deep Learning algorithms can analyze vast amounts of data and automatically identify patterns and features that are beyond human capabilities.

To simplify, Deep Learning is a specialized form of Machine Learning within the broader field of AI. All Deep Learning models are Machine Learning models, but not all Machine Learning models are Deep Learning models. Similarly, while all Machine Learning models are part of AI not all AI systems use Machine Learning[1].

The connection between AI and the fields of machine learning, and deep learning is shown in Figure 1.1

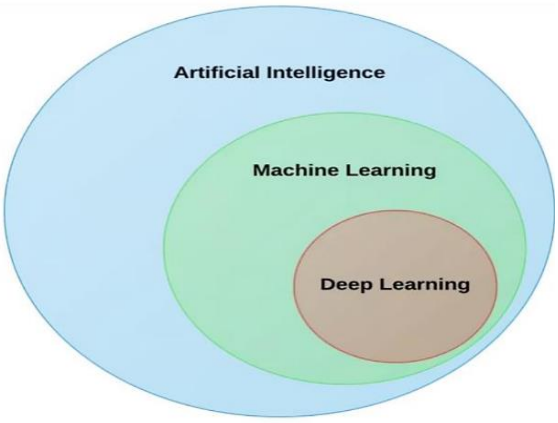


Figure1.1:Relation between AI ,ML ,deep learning [2]

1.3 Deep learning

Deep learning, a specialized branch of machine learning, focuses on training complex neural networks with multiple layers to identify and interpret patterns within data. The term "deep" highlights the extensive layering of these networks, which enables them to capture intricate data representations. Currently, deep learning is pivotal in creating groundbreaking technologies, from voice assistants to medical imaging, that profoundly influence various aspects of our daily lives, including how we live, work, and interact.

1.4 Neural Network

Neural Networks (NN) are a set of algorithms designed to identify underlying patterns in data by simulating the way the human brain operates, it is made up of layers that work together to process information. Here's what each layer does:

1. **Input Layer**: This is where the data first enters the network. Each neuron in this layer represents a single piece of information, like a pixel in an image.
2. **Hidden Layers**: These layers are in the middle and do the complex work. They take the input data, process it, and pass it on to the next layer. The number of hidden layers depends on how complicated the task is. More layers are needed for harder tasks.
3. **Output Layer**: This is the final layer that gives you the results. Each neuron here represents a possible answer or category. For example, in image recognition, these neurons might represent categories like "dog," "cat," or "car."[3] As shown in Figure 1.2

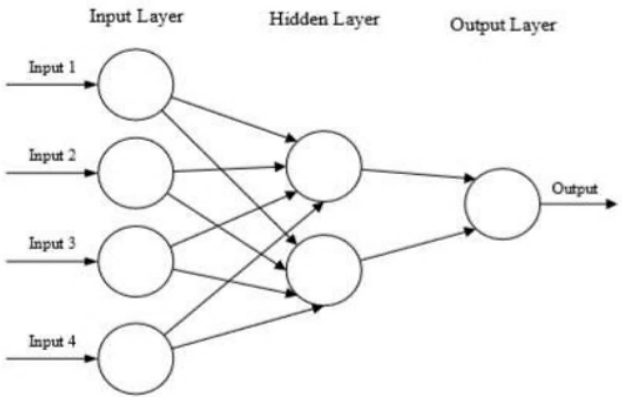


Figure1.2:Basic structure of neural Networks [4]

1.4.1 perceptron

A perceptron, which is one type of artificial neuron, serves as the fundamental unit of neural networks. A perceptron takes a fixed number of inputs and produces a single output. The way of computing the result essentially has several steps, such as taking an input, calculating the weighted sum by introducing weights, bias term, and applying the activation function. The process described above can be formalized as follows:

The perceptron has n inputs represented as an input vector $x = (x1, x2, \ldots\ldots x_n)$. Each input has an assigned weight that is defined by a vector of weights $w = (w1, w2, \ldots\ldots w_n)$ Consequently, the weighted input values are combined, which gives the weighted sum:

$$\varepsilon = w.x = \sum_{i=1}^{n} w_i \,.\, x_i \qquad (1.1)$$

At this step, the activation function is applied to the weighted sum to calculate the output, and the weighted sum is compared with a threshold θ to produce an output y that is either 0 or 1, depending on whether or not it exceeds the threshold. Thus. As shown in Figure1.3.

$$y = \sigma(\varepsilon) = \begin{cases} 1, \varepsilon \geq 0 \\ 0, \varepsilon < 0 \end{cases} \quad (1.2)$$

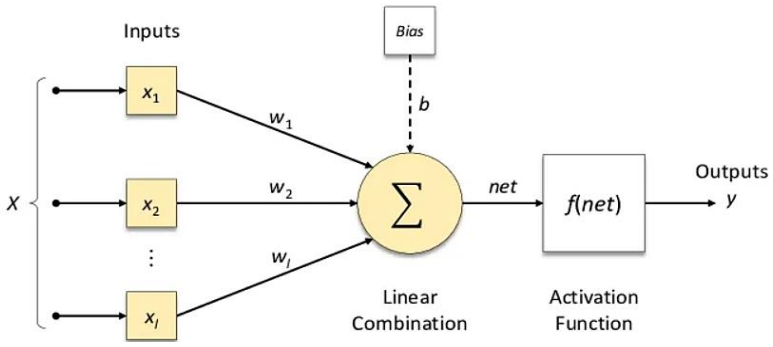


Figure1.3:The perceptron [5]

1.5 Neural Networks Vocabulary

Familiarizing ourselves with the terminology associated with neural network architecture is a crucial step to better comprehending the processes involved in building and optimizing deep learning models. Strengthening this knowledge base will enable a thorough understanding of how deep learning architectures are designed and optimized for diverse tasks**.**

1.5.1 Gradient Descent

The most widely used optimization method in Deep Learning is gradient descent. To minimize the loss, it includes computing the gradient of the loss function with respect to the parameters of the model and updating them in the opposite direction of the gradient. As shown in Figure 1.4.

$$\vartheta k \; = \; \vartheta k \; - \; \gamma \; (d(cost)) \; / (d(\vartheta k) \quad (1.3)$$

Where ϑk are the actual weights and γ is the learning rate.

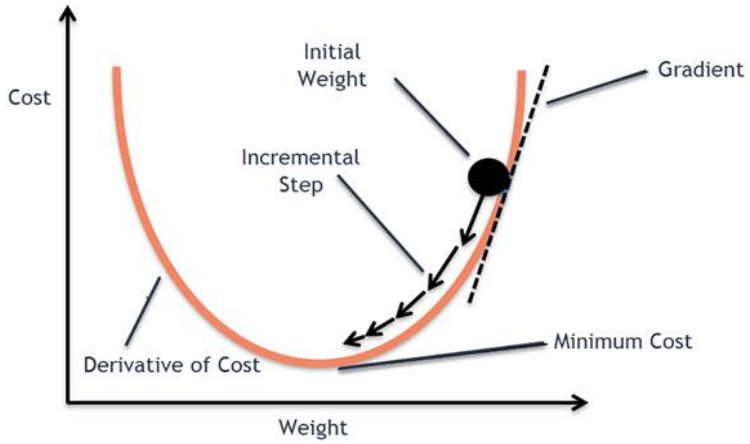


Figure1.4:Weight update by gradient descent in the cost function [6]

1.5.2 Backpropagation

In neural networks, the critical process of updating weights and biases is achieved through the backpropagation algorithm. This method begins after computing the predicted outputs and assessing the error. Backpropagation works by propagating the error gradient backward through the network layers, revealing each layer's contribution to the overall error. This backward flow of gradients enables the calculation of gradients for weights and biases at each layer, forming the basis for iterative parameter updates.

Using optimization algorithms like gradient descent, these updates are carefully performed to explore the parameter space and minimize the cost function. This systematic approach not only enhances the model's accuracy but also strengthens its ability to generalize to unseen data, encapsulating the principles of continual learning and adaptation in neural networks. As shown in Figure 1.5.

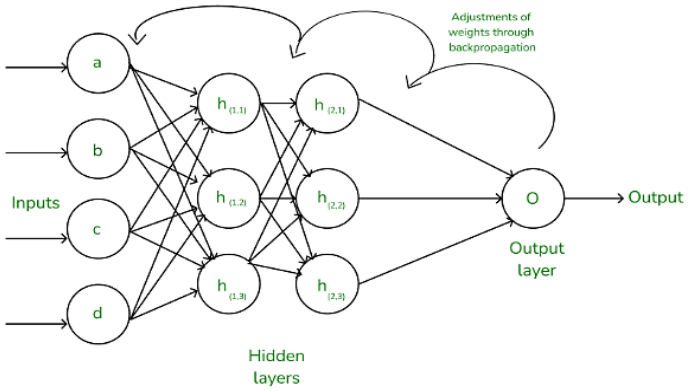


Figure1.5:Simple Illustration of how the backpropagation works by adjustement of weight [7]

1.5.3 Learning Rate

The learning rate is a crucial hyperparameter that determines how much the cost function is minimized in each iteration. It essentially controls the step size of the gradient descent algorithm, which is used to update the neural network weights. If the learning rate is too low, the model might converge very slowly. On the other hand, if it's too high, the algorithm might overshoot the optimal minimum. Typically, the learning rate is set to a small value, such as 0.1 or 0.01, but the exact value can vary depending on the specific problem.[8]

1.5.4 Batch

Batch size is another important hyperparameter that decides how many training examples are used in each iteration of the gradient descent algorithm. In simpler terms, it specifies how many instances are analyzed before updating the weights. Using small batch sizes can lead to faster convergence and less memory usage, but it can also make the training process more noisy. Conversely, larger batch sizes reduce noise during training but require more memory and may take longer to converge.[8]

1.5.5 Epoch

An epoch in neural network training refers to a complete cycle of forward propagation and backpropagation, representing the number of iterations the algorithm performs[9]. During an epoch, batches of data pass through the network, enabling the model to learn and adjust its parameters[10]. Monitoring the progression of training data through mini-batches and updating weights provides valuable insights into the learning process, helping determine when to stop training effectively. This process also validates the model's ability to generalize on test data.

For example, if a training set contains 500 images and a batch size of 25 is chosen, one epoch would consist of 20 iterations. These iterations involve both forward and backward propagation, resulting in 20 updates to the model's parameters.

Neural networks rely on two types of adjustable elements:

- **Hyperparameters:** These are predefined settings that control the overall structure and behavior of the network. Examples include the number of layers, neurons per layer, learning rate, batch size, number of epochs, regularization strength, and optimization algorithms like gradient descent[11]
- **Parameters:** These are internal values that are learned during training. They include weights and biases, activation functions, and feature extractors.

1.6 Convolution Neural Networks (CNN)

Convolutional Neural Network (CNN), also called ConvNet, is a particular type of feed-forward neural network. CNN's are models that are most frequently used for image processing and computer vision. They are designed in such a way as to imitate the structure of the animal visual cortex[12]. It is designed to work with grid-structured inputs, that have strong spatial dependencies in local regions of the grid [13]. This type of network relies on a linear arithmetic process called the convolution, from here the name convolutional neural network. The primary benefit of CNN compared to its predecessors is that it automatically determines the relevant features without the need for human intervention [14].

The most important feature that distinguishes them from neural networks is the convolutional layer, which extracts features. In CNNs first stage, there are many stages such as the convolution and pooling layers. In the final stage, there are the Fully-Connected layer and the Classification layer. Following these multiple consecutive trainable layers, the Deep Learning structure proceeds with a training layer. CNN gives an output to Verification with veritable/correct results. This comparison gives an error rate which is the difference between the generated output and the targeted result. CNN can Utilize different input files such as images, audio video, or other signals. As shown in Figure1.6

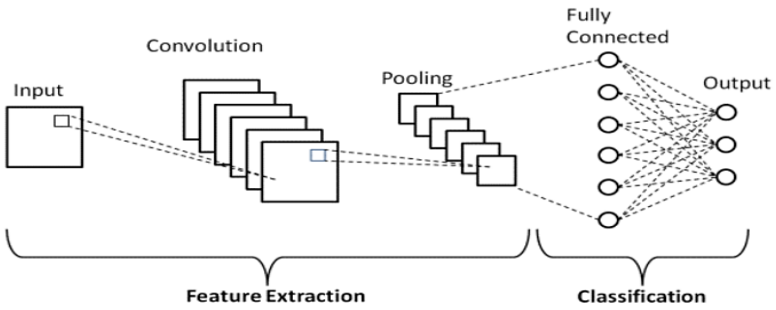


Figure1.6: Structure of a convolutional neural network [15]

1.6.1 Convolution Layer

The convolutional layer is the most important component. This layer's role is to analyze the images supplied as input and detect the existence of a set of features. So that the convolutional layer grabs the input and applies a filter to each position when outputting this layer we get a set of feature maps. This convolution process creates feature maps where each feature map is a convoluted result of different individual feature detectors which is shown in Figure1.7 Rectified linear unit (ReLu) activation function is then applied to increase the non linearity in the resulting feature maps to distinguish adjacent pixels of the maps more accurately [16].

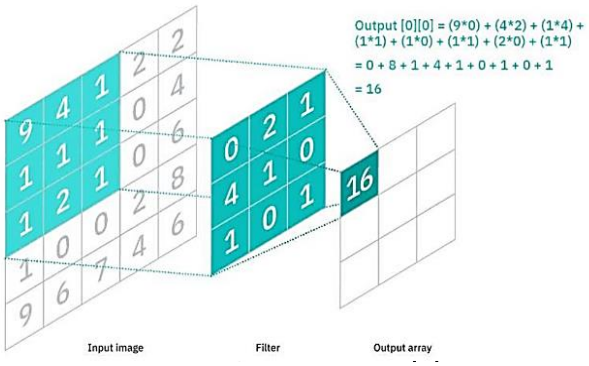


Figure1.7:convolutional layers [17]

1.6.2 Relu Function

After each convolution operation, a CNN applies a transformation known as the Rectified Linear Unit (ReLu). As shown in Figure 1.8 This step is crucial as it introduces nonlinearity into the model. By using ReLu, the network can learn more complex and nuanced representations of the input data, moving beyond simple linear relationships and enabling it to capture a wider range of patterns and features.

$$f(x) = max\ (0, x) \tag{1.4}$$

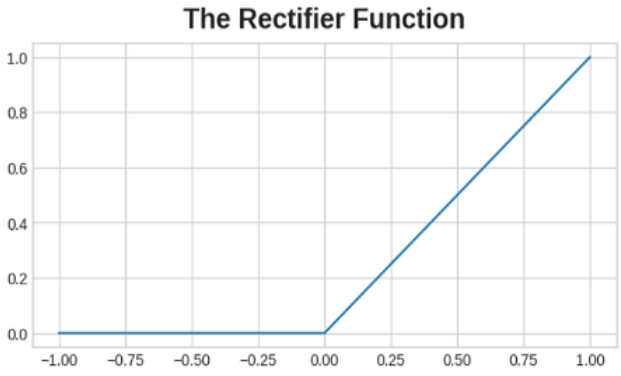


Figure1.8:Relu activation function [18]

1.6.3 Pooling Layer

After applying the ReLU transformation, CNNs use a technique called downsampling to reduce the size of the convolved feature. This process, often achieved through max pooling, decreases the number of dimensions in the feature map while preserving the most important information. This helps reduce processing time and focus on key features As shown in Figure 1.9.

Max pooling operates similarly to convolution but serves a different purpose. Instead of applying filters, you extract tiles of a specific size from the feature map. For each tile, the maximum value is selected and added to a new feature map, while all other values are discarded. Two key parameters control max pooling:

- **Filter Size**: Typically, this is a 2x2 pixel tile. It determines the area from which the maximum value is extracted.
- **Stride**: This is the distance, in pixels, between each tile extraction. Unlike convolution, where filters move pixel by pixel, the stride in max pooling determines where each tile is extracted. For example, with a 2x2 filter and a stride of 2, the operation extracts all non-overlapping 2x2 tiles from the features map.

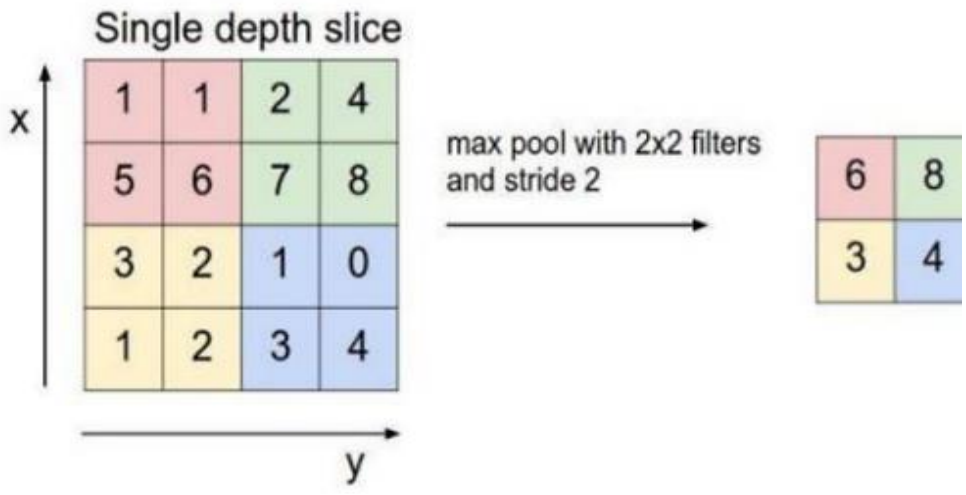


Figure1.9: Example of pooling principle [19]

1.6.4 Fully Connected Layer

At the end of a convolutional neural network, one or more fully connected layers are typically found. In a fully connected layer, every node in one layer is connected to every node in the next layer. These layers are responsible for using the features extracted by the convolutional and pooling operations to classify the data. The final fully connected layer often uses a softmax activation function. This function outputs a probability value between 0 and 1 for each of the classification labels that the model is trying to predict. This means that the output represents the likelihood of each possible classification, allowing the model to make a prediction based on the highest probability. For feature extraction, this CNN employs two convolution modules, each consisting of a convolution operation followed by ReLU and pooling. For classification, it uses two fully connected layers. However, other CNNs might have a different number of convolution modules and fully connected layers, depending on the specific requirements of the task. Engineers often conduct experiments to find the optimal configuration for their model, tailoring it to achieve the best performance for their particular application.[20] As shown in Figure 1.10**.**

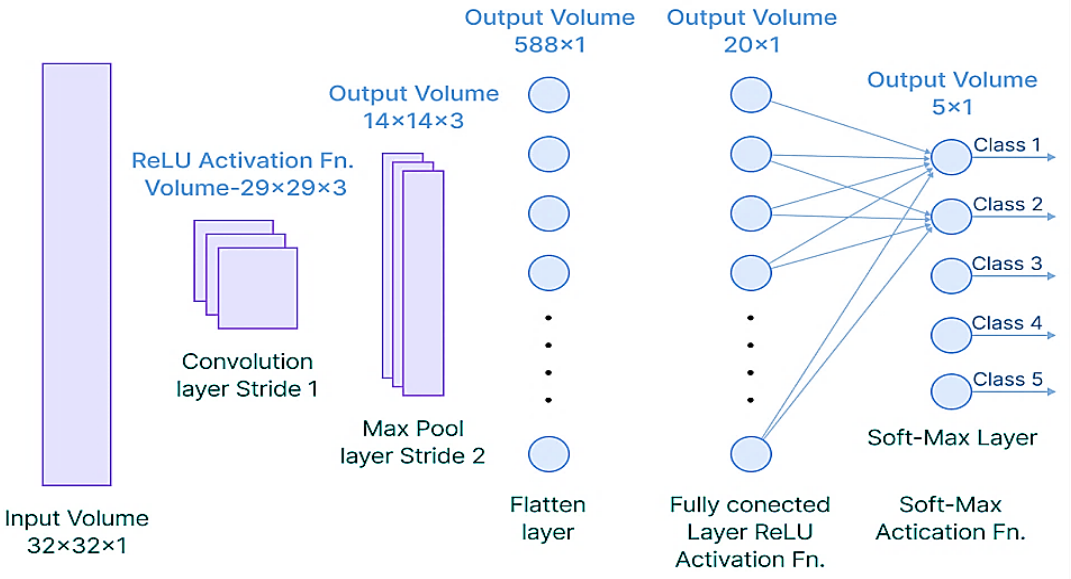


Figure1.10:fully connected layers[21]

1.7 Conclusion

In this chapter, we presented a brief approach to Deep Learning, which will serve as a foundation for our exploration of object detection. Then, we dealt with the fundamental studies of Deep Learning including neural network concepts, and how to improve a neural network. Also, we covered one of the most commonly used deep neural networks, CNN, and provided an overview of each CNN component and layer, from the convolutional layer to the final layer known as the fully connected layer (FC).

All these notions let us proceed towards the next chapters, where we will put them into practice by implementing these theoretical concepts, we will gain a better understanding of how deep learning architectures work and will be able to build our deep neural network.

Chapter 2: Object detection

Summary

This chapter provides an overview of the background and core focus of our thesis. It explores the categories of object detection algorithms, with a particular emphasis on the YOLO algorithm, and includes a detailed discussion of the evaluation metrics.

Table of Contents

2.1 Introduction

Object detection is a crucial yet challenging task in the field of computer vision. It plays a fundamental role in many applications, including robot navigation, as well as medical and biological fields. In this chapter of our project, we focus on object detection. We begin by presenting the main object detection algorithms, then delve deeper into the YOLO (You Only Look Once) algorithm, which is widely used for its speed and accuracy.

Finally, we will discuss the evaluation metrics used to assess the performance of object detection models.

2.2 Object Detection

Object detection is a profound computer vision technique that identifies and locates objects within images, videos, and even live footage. Models used for object detection are trained on a large dataset of annotated visuals, allowing them to process new input data effectively. A key component of object detection is the bounding box, which identifies the edges of the object tagged with a clear-cut quadrilateral typically a rectangle. Each bounding box is accompanied by a label indicating the object's type, such as a person, a car, or a dog. [22]

2.2.1 Image classification

Image classification is the process of predicting of the class of an item in an image. For example, when you carry out a reverse image search on Google, you are likely to receive a note that says “might include ‘x’, with the ‘x’ being anything that the technology detects as the primary object of the image. Image classification can show that a particular object exists in the image, but it involves one primary object and does not provide the object’s location within the visual, Figue2.1 shows the Image classification.[22]

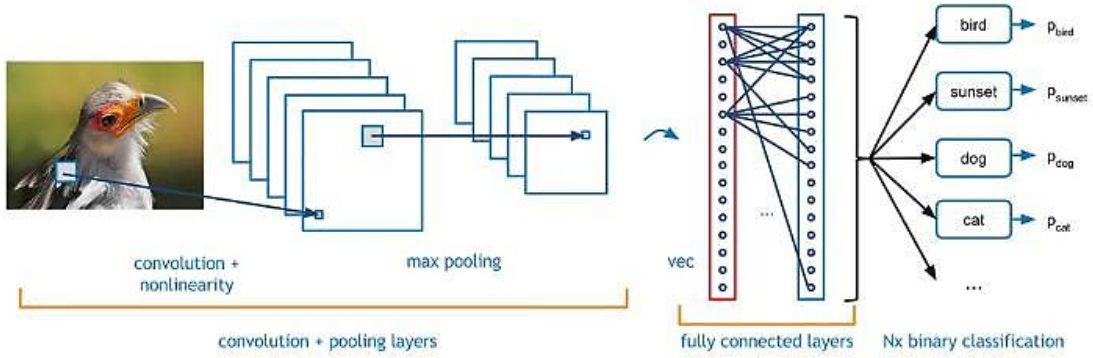


Figure 2.1: Image classification

2.2.2 Segmentation

Semantic segmentation is the task of grouping pixels with comparable properties together instead of bounding boxes to identify objects [22] Figure 2.2 illustrates an example of semantic segmentation, where each distinct object or region in the image is categorized separately and color-coded to indicate its class.

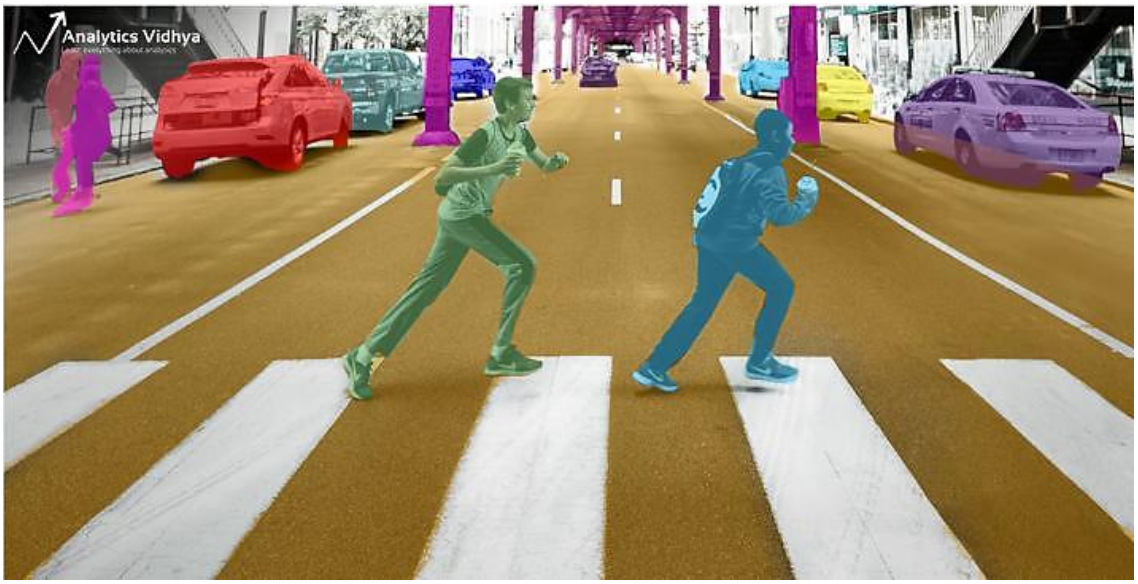


Figure 2.2: Object Segmentation

2.2.3 Object localization

Object localization and object detection are closely related but have a key difference. Object localization focuses on finding the exact location of one or more objects in an image, usually by marking them with bounding boxes. It's concerned with identifying where an object is. Object detection goes a step further it not only finds the objects but also identifies what they are, typically classifying them while drawing bounding boxes around them. In simple terms, localization is about pinpointing positions, while detection includes both locating and recognizing objects. Figure 2.3 shows an example of the output of a trained model to detect men and women, the model identifies and classifies search within the image, providing probabilities for both categories.

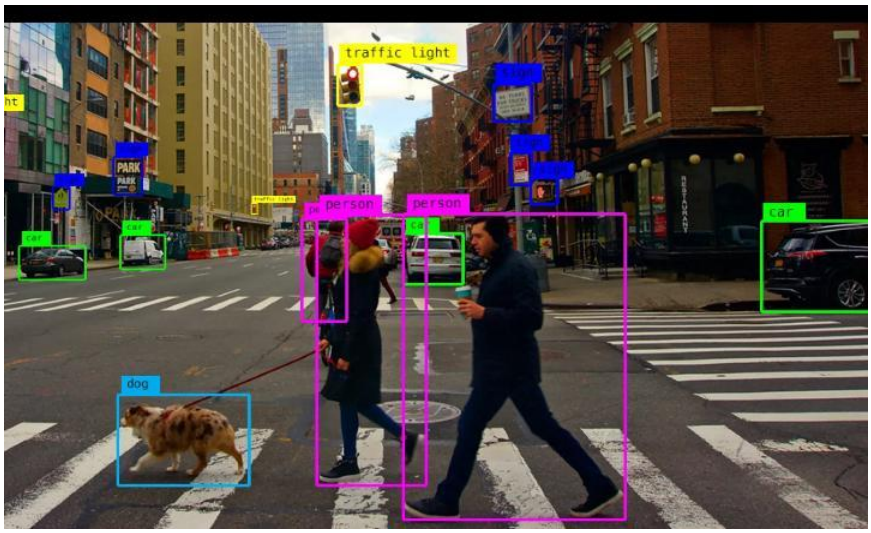


Figure 2.3:localisating object (Image source)

When trained to detect men and women, the model will identify and classify each within the image providing probabilities for both categories.

2.3 Object Detection Algorithms

After 2014, object detection algorithms were mainly categorized into two types: two-stage and one-stage detectors [23] The primary distinction between the two lies in the use of region proposals. Two-stage detectors generate region proposals as potential object locations, whereas one-stage detectors skip this step.

2.3.1 One-Stage Detectors (Single-Stage Object Detectors)

One-stage object detectors perform object detection in a single step, bypassing the need for region proposals. Instead, they divide the input image into a grid and directly predict both bounding boxes and class probabilities for each region. This streamlined approach allows for much faster processing compared to two-stage detectors, making one-stage models like YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector) highly suitable for real-time applications. However, this speed advantage may come at the cost of slightly reduced accuracy, especially in complex scenes where precise localization is critical [23]. For instance, YOLO divides the image into a grid, predicts bounding boxes along with confidence scores, and then applies Non-Maximum Suppression (NMS) to remove overlapping detections and refine the final results.

2.3.1.1 Single Shot Detector (SSD)

SSD is a single-pass object detector, meaning it does not rely on a separate region proposal network. Instead, it directly predicts both the bounding boxes and object classes from feature maps in a single forward pass. To enhance accuracy, SSD The SSD is shown in Figure 2.5, it introduces :

Small convolutional filters to predict object classes and refine default bounding boxes.

Separate filters for default boxes to better handle different aspect ratios.

Multi-scale feature maps to improve detection across various object sizes [24].

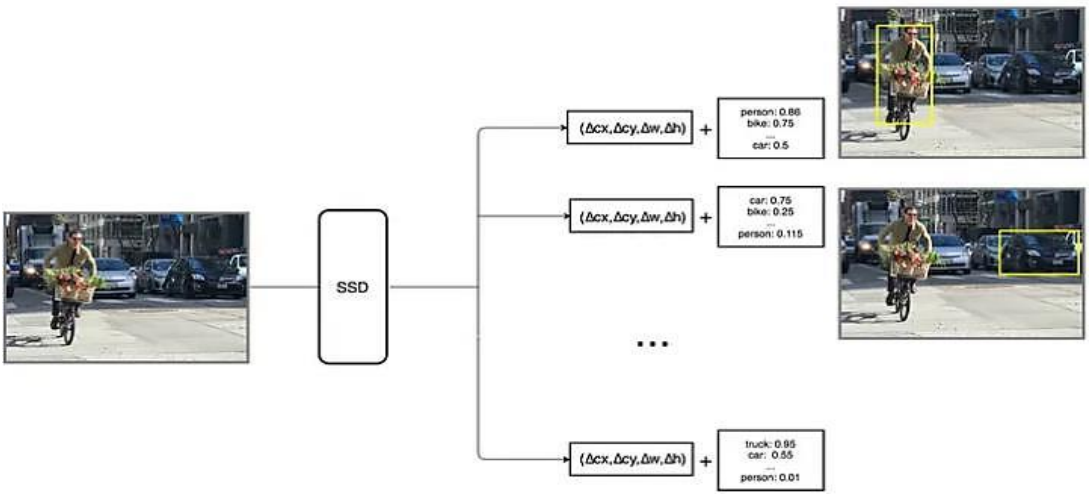


Figure 2.4:Single shot Detector

2.3.2 Two-Stage Object Detection Algorithm

Two-stage detectors operate in two steps: first, generating region proposals using methods like Selective Search or Region Proposal Networks (RPN); second, classifying the objects and refining bounding boxes for each proposed region. These methods generally offer higher accuracy due to the refinement step but are slower compared to one-stage detectors. Notable examples include the R-CNN series: R-CNN (2014), Fast R-CNN (2015), and Faster R-CNN (2015). For instance, R-CNN extracts region proposals from an image and classifies each separately, achieving high accuracy at the cost of increased processing time Figure 2.4 shows R-CNN with CNN features.[23]

2.3.2.1 Faster R-CNN

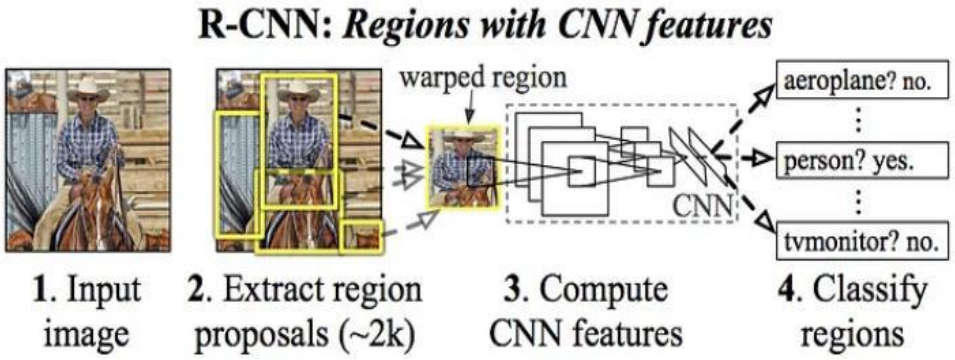


Figure 2.5:R-CNN with CNN features

Faster R-CNN (short for « Faster Region-Convolutional Neural Network ») is an advanced object detection architecture from the R-CNN family, introduced by Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun in 2015.

The main objective of Faster R-CNN is to create a unified framework that not only identifies objects in an image but also accurately determines their locations. It integrates the strengths of deep learning, convolutional neural networks (CNNs), and region proposal networks (RPNs) into a single, cohesive system. This integration significantly enhances both the speed and accuracy of object detection. Figure 2.6 shows Faster R-CNN.[25]

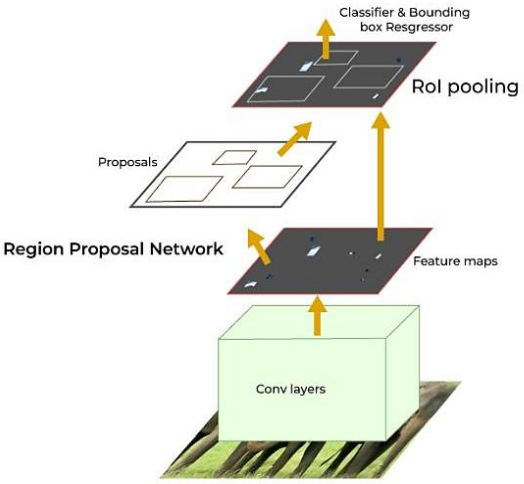


Figure 2.6:Faster R-CNN

.

The Fast R-CNN detector is a crucial component of the Faster R-CNN architecture, tasked with identifying objects within the proposed regions generated by the Region Proposal Network (RPN). The process begins with Region of Interest (RoI) Pooling, where the variable-sized region proposals from the RPN are converted into fixed-size feature maps. This step is essential because it standardizes the input dimensions, allowing the subsequent layers of the network to effectively analyze and classify the regions while also refining their bounding box coordinates. Figure 2.7 shows Fast R-CNN Architecture. [25]

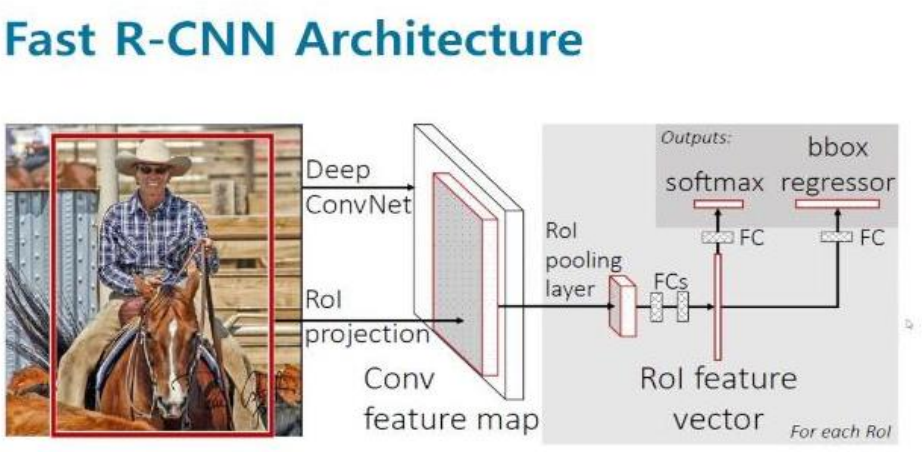


Figure 2.7: Fast R-CNN Architecture

2.4 YOLO(YouOnly Look Once)

YOLO (You Only Look Once) is a technique/approach for object detection. It is the algorithm that determines how the code will identify objects in an image. YOLO processes the entire image in a single pass and predicts bounding box coordinates along with class probabilities for detected objects.

The biggest advantage of using YOLO is its incredible speed it is extremely fast and can process 45 frames per second. YOLO also understands generalized object representation. However, this approach is somewhat moderate in accuracy and can sometimes be inefficient. YOLO adopts a unique strategy.

YOLO examines the entire image just once, passes through the neural network once, and identifies objects hence the name « You Only Look Once. » It is known for its speed, which is why it has gained such popularity. Other well-known object detection frameworks, like Faster R-CNN and SSD, are also widely used.[26]

2.4.1 Overview of the YOLOv5 Architecture

YOLOv5 (You Only Look Once, version 5) is a single-stage object detection model, meaning it performs both object localization and classification simultaneously in a single forward pass through the network. This makes it highly efficient for real-time applications.[27]

Key Components of YOLOv5 Architecture :

- Backbone: Extracts feature maps from the input image.
- Neck: Aggregates and refines features at different scales.
- Head: Predicts bounding boxes and class probabilities for detected objects.

Backbone: The backbone of YOLOv5 is responsible for extracting hierarchical feature maps from the input image. It is based on the Cross Stage Partial Networks (CSPNet) architecture, which enhances computational efficiency by reducing the number of gradient updates required during training[27].

The backbone consists of a series of convolutional layers and CSP modules that progressively downsample the image while extracting increasingly abstract features.

Neck: The neck in the YOLOv5 model combines feature maps from different stages of the backbone to construct a feature pyramid. This enables the model to detect objects at multiple scales, which is especially important for identifying small objects like COTS.

The neck utilizes a structure known as the Path Aggregation Network (PAN), which improves the model's ability to capture features at different resolutions, enhancing its overall detection performance.[27]

1) **Head**: The head of the YOLOv5 model is responsible for predicting bounding boxes and class probabilities for the detected objects. It consists of multiple convolutional layers that generate predictions at different scales, corresponding to the various feature maps produced by the neck.[27]

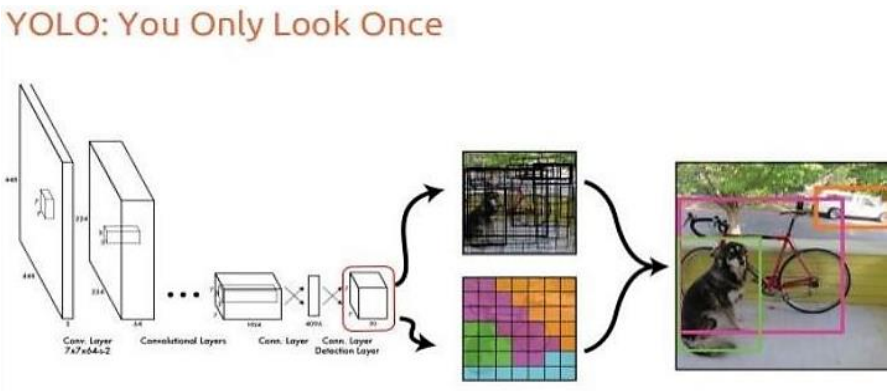


Figure 2.8:Yolo architecture [A Practice for Object]

The YOLOv5 architecture consists of five different models, ranging from the highly efficient YOLOv5n to the high-precision YOLOv5x [27].

YOLOv5n: This is the smallest and fastest model in the YOLOv5 series, designed for environments with limited resources. Its compact size less than 2.5 MB in INT8 format and around 4 MB in FP32 format—makes it ideal for deployment on edge devices and IoT platforms.

Additionally, its compatibility with OpenCV DNN improves its usability in mobile applications.[27]

YOLOv5s: This is the baseline model in the series, containing approximately 7.2 million parameters. Due to its balance between efficiency and performance, it is well-suited for CPU-based inference tasks.[27]

YOLOv5m: This is a mid-sized model with 21.2 million parameters, offering a good balance between speed and accuracy. It is considered a versatile choice for various object detection applications and datasets.[27]

YOLOv5l: With 46.5 million parameters, this model is designed for scenarios that require higher precision, especially when detecting smaller objects in images.[27]

YOLOv5x: As the largest and most complex model in the series, YOLOv5x achieves the highest mean Average Precision (mAP) among its counterparts. However, this high performance comes with increased computational requirements due to its 86.7 million parameters.[27]

2.5 Main Evaluation Metrics

In object detection, several key metrics are used to evaluate the performance of a model. The most fundamental ones are[28] :

True Positive (TP): A correctly detected bounding box that matches a ground-truth object.

False Negative (FN): A ground-truth bounding box that the model failed to detect.

False Positive (FP): An incorrect detection, either detecting a nonexistent object or placing a bounding box incorrectly.

Unlike classification problems, True Negatives (TN) are not applicable in object detection since there are infinitely many possible bounding boxes that should not be detected in an image.

Intersection over Union (IoU)

To determine whether a detection is correct or incorrect, a common approach is to use Intersection over Union (IoU), which measures the overlap between the predicted bounding box

(Bp) and the ground-truth bounding box (Bgt). It is calculated using the Jaccard Index formula [29] :

$$J(Bp, Bgt) = IOU = area\,(Bp \cap Bgt) \,/\, Area\,(Bp \cup Bgt) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2.1)$$

Where : $(Bp \cap Bgt)$: Is the area of overlap between the predicted and ground-truth bounding boxes.

$(Bp \cup Bgt)$: Is the total combined area of both bounding boxes.

The Intersection over Union (IoU) metric ranges from 0 to 1, where:

0 means no overlap between the predicted and ground truth regions.

1 means perfect overlap the predicted region exactly matches the ground truth.

In practice:

IoU < 0.5 is often considered a poor prediction.

IoU ≥ 0.5 is usually the minimum threshold for a detection to be considered acceptable.

IoU ≥ 0.75 is considered a strong or high-quality detection in many applications.

Average Precision (AP) and its Importance :

The most widely used metric for evaluating object detection models is Average Precision (AP). AP considers both precision (how many of the detected objects are correct) and recall (how many of the actual objects were detected).

Before exploring AP variations, it's crucial to understand these fundamental concepts, as they form the basis for measuring the accuracy of object detection models in scientific research and challenges.[28]

2.6 Conclusion

This chapter provided an overview of object detection, a fundamental task in computer vision. We explored two main approaches used in object detection: One-stage object detection algorithms, such as SSD and YOLO, which are known for their speed and real-time performance. Two-stage object detection algorithms, exemplified by Fast R-CNN and Faster R-CNN, prioritize accuracy through an additional refinement stage. Additionally, we highlighted the key evaluation metrics used to assess the performance of the object detection model.

Chapter 3: Models Implementation and Experimental Results

Summary

This chapter presents the practical implementation of a breast cancer detection system using the YOLO object detection models. The goal is to develop an automated method to detect breast tumors in mammography images.

Table of Contents

3.1 Introduction

Breast cancer incidence has been rising steadily during the past few decades. It is the second leading cause of death in women. If it is diagnosed early, there is a good possibility of recovery. Mammography is proven to be an excellent screening technique for breast tumor diagnosis, but its detection and classification in mammograms remain a significant challenge.

In this chapter, we present the practical implementation of our breast tumor detection project using YOLO object detection models. The purpose of this stage is to translate the theoretical foundations discussed in previous chapters into a functional system capable of detecting breast tumors in mammogram images. We will detail the software tools, libraries, dataset preparation, model training process, and the interpretation of the obtained results, which collectively contribute to evaluating the system's ability to detect breast tumors.

3.2 Software and libraries used in the implementation

3.2.1 Software

In our project, we utilized Google Colab as the primary software tool because it provides free access to GPUs.

Google Colab, short for Collaboratory, is a cloud-based service by Google that enables users to run Python scripts and perform machine learning tasks on CPUs, GPUs, and TPUs. It incorporates the following key features:

- **Virtual machines:** Google Colab runs code on cloud-hosted virtual machines preconfigured with popular machine learning libraries and frameworks such as TensorFlow and PyTorch, allowing seamless execution of scripts and computations.
- **Jupyter Notebooks**: Built on the Jupyter Notebook interface, Colab offers an interactive environment for writing and running code, visualizing data, and documenting workflows. Users can create, edit, and share notebooks easily.
- **GPU and TPU Acceleration**: Colab provides free access to hardware accelerators like GPUs and Tensor Processing Units (TPUs), significantly speeding up machine learning operations. This enables efficient handling of large datasets and faster model training.

- **Integration with Google Drive**: Colab integrates directly with Google Drive, allowing users to save, access, and manage their notebooks and data from anywhere. Users can mount their Drive within Colab to work with stored files seamlessly.
- **Collaboration Features:** The platform supports real-time collaboration by enabling users to share notebooks and code, comment on work, and co-edit projects simultaneously.
- **Code Snippets Library**: Google Colab includes a rich collection of code snippets and examples that facilitate quick implementation of common machine learning tasks and techniques.

Overall, Google Colab is a powerful and flexible platform for machine learning and data science, offering a wide range of tools and features that support diverse workflows and projects with minimal setup and free access to advanced computing resources.

3.2.2 Programming language

Within the Google Colab environment. Python is widely supported in Google Colab and is known as a popular high-level programming language. It is extensively used across various fields such as web development, data analysis, machine learning, artificial intelligence, and many others.

3.2.3 Frameworks

- **TensorFlow:** TensorFlow is an open-source machine learning framework developed by the Google Brain team and released publicly in 2015. It enables developers to build, train, and deploy machine learning models across multiple platforms. TensorFlow offers a comprehensive suite of tools, libraries, and resources for model development and testing, supporting a broad range of applications including image and audio recognition, natural language processing, and time series analysis.
- **Keras:** Keras is a high-level neural network API written in Python that can run on top of TensorFlow, Microsoft Cognitive Toolkit (CNTK), Theano, or PlaidML. Created by François Chollet, now a Google employee, Keras is distributed under the

permissive MIT license. It simplifies the process of building and experimenting with deep learning models.

- **PyTorch:** PyTorch is an open-source machine learning library for Python, primarily developed by Facebook's AI Research (FAIR) team. It provides a flexible and efficient framework for designing deep learning models. PyTorch supports dynamic computational graphs and automatic differentiation, enabling rapid model prototyping and training.

1.2.4 Libraries

- **OpenCV:** OpenCV (Open Source Computer Vision Library) is a free and open-source software library designed for computer vision and machine learning applications. It offers a wide range of functions for image and video processing, including object detection, facial recognition, feature extraction, and real-time video analysis. OpenCV supports various advanced techniques such as image segmentation, motion tracking, and augmented reality, making it a versatile tool for numerous computer vision tasks.
- **NumPy:** NumPy (Numerical Python) is a fundamental Python library that provides support for large, multi-dimensional arrays and matrices. It also includes a comprehensive collection of high-level mathematical functions to efficiently perform operations on these arrays, serving as a core component for scientific computing and data manipulation in Python.
- **Matplolib:** Matplotlib is a Python library used for data visualization. It enables the creation of static, animated, and interactive plots and charts, making it a valuable tool for visually representing data and analyzing results.
- **Pandas:** Pandas is an open-source Python library for data manipulation and analysis. It offers powerful data structures and functions to clean, explore, and transform large datasets effectively, facilitating efficient data handling and preparation for further analysis or modeling.

3.3 Data description

The Curated Breast Imaging Subset of DDSM (CBIS-DDSM) dataset originally contains 10,239 images, but not all of them were suitable for our study, as shown in Figure 3.2. We selected 1424 images where the tumors were visible, as illustrated in Figure 3.1. After applying data augmentation, our dataset expanded to a total of 2365 images. We then divided the data into three subsets: 60% for training, 30% for validation and 10% for testing. In our study, we experimented with different versions of YOLO models, including YOLOv5 (small, large, and nano), YOLOv8, and YOLOv12.

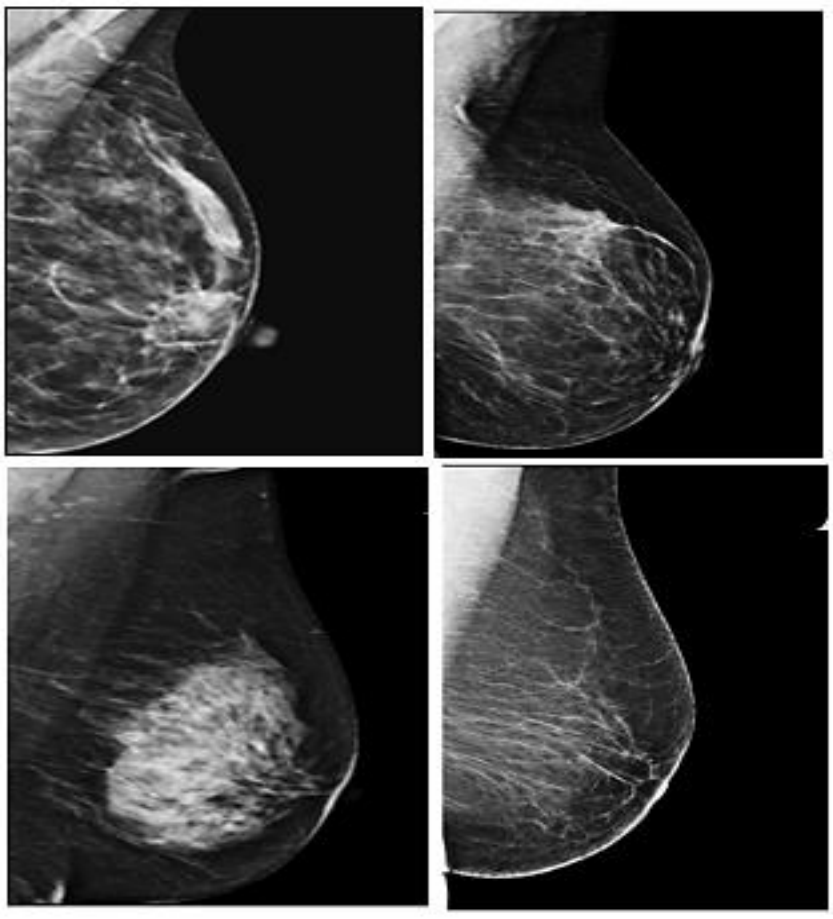


Figure 3.1: Breast cancer images from the CBIC-DDSM datasets

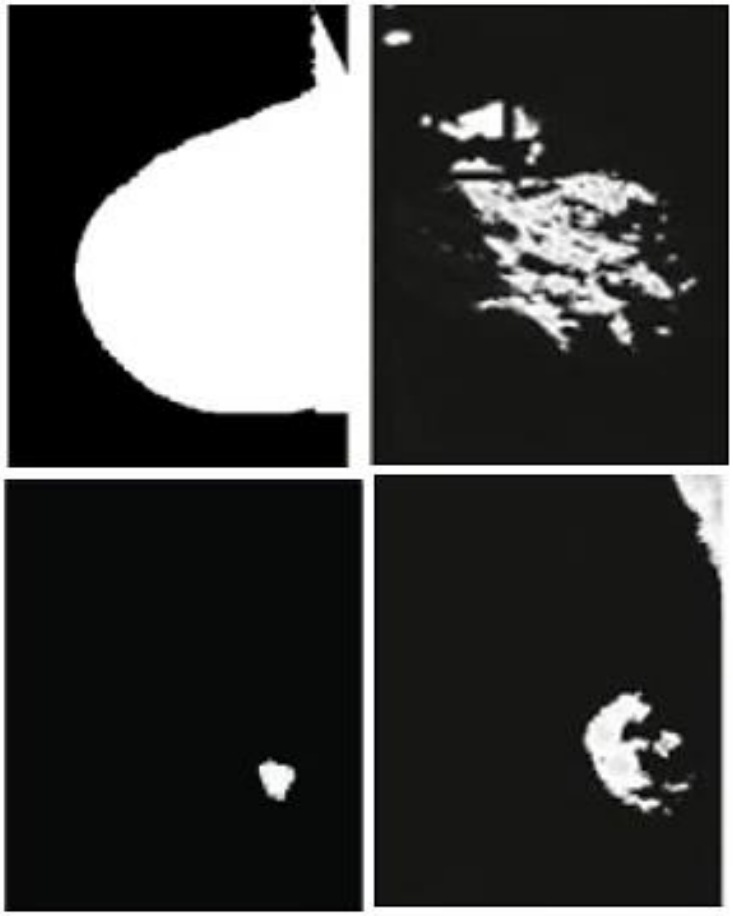


Figure 3.2:The inapplicable breast cancer images

We found that these images were challenging to process because they consisted of a mixture of masks and original images, which made it difficult for us to start our preprocessing techniques directly. Additionally, the poor contrast and lack of clear boundaries further reduced their usability for model training or analysis. Eventually, we decided to exclude the masks and poor-quality images from the dataset, as they are unsuitable for deep learning tasks.

3.4 The preprocessing steps

Preprocessing is a crucial step in medical image analysis, especially in mammography, as it significantly improves image quality and prepares the data for more accurate model training and evaluation. To achieve better results, we found it necessary to carefully apply a series of preprocessing steps starting from noise removal, contrast enhancement, and artifact suppression, to standardizing image sizes. These steps helped us highlight important features such as tumors and remove irrelevant parts like labels and backgrounds. As shown in Diagramme 3.3, our

thorough preprocessing pipeline was essential to ensure that the input images were clean, consistent, and suitable for our proposed model, ultimately leading to improved detection performance.

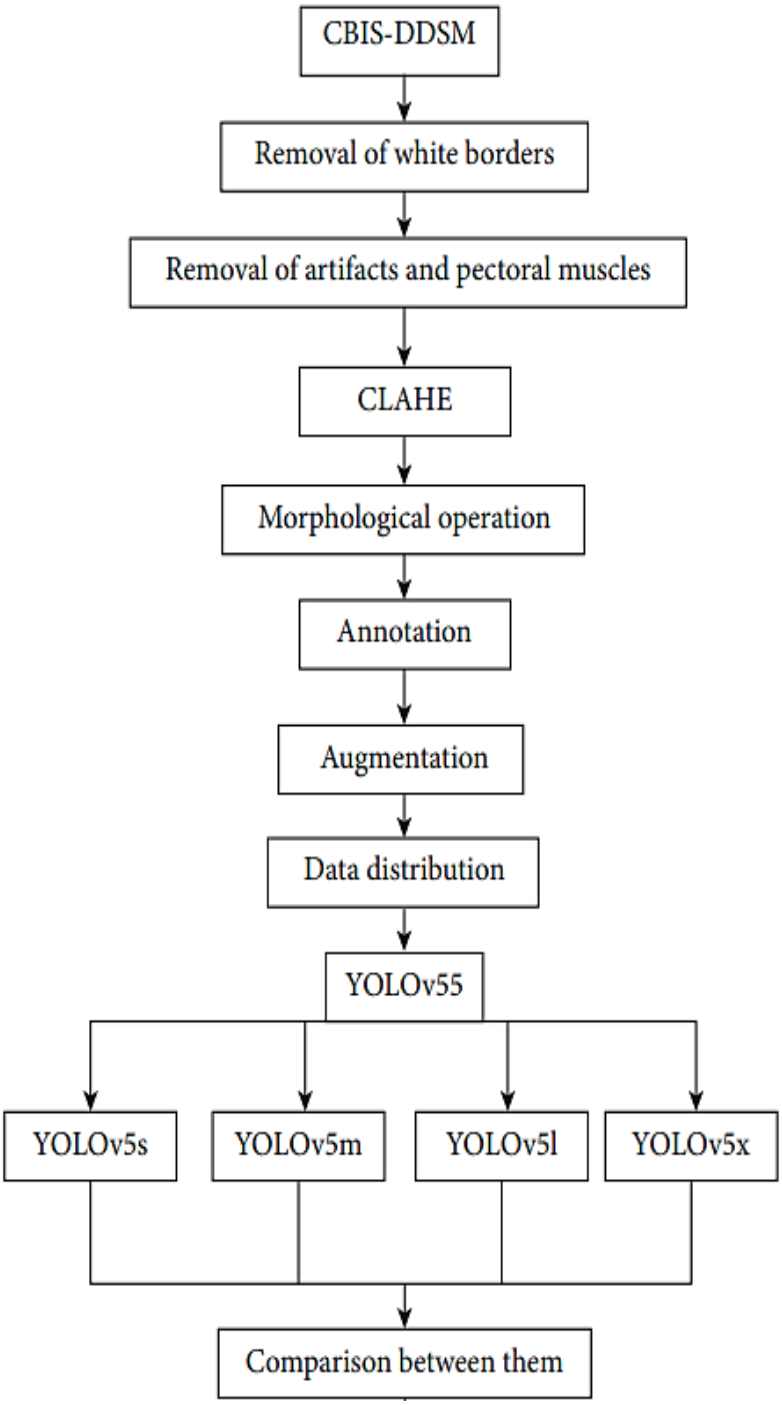


Figure 3.3: proposed wrokflow

3.4.1 Removal of white lines

Before training our model, we thoroughly cleaned the images using several techniques. We have noticed that the original images contain rough outer borders with white lines. Since white areas in mammograms can also indicate tumors, these white borderlines risk causing misdiagnosis. To prevent such misclassification, we remove the white lines by cropping the image borders slightly using a cropping function. The resulting images, after this preprocessing step, are shown In Figure 3.4

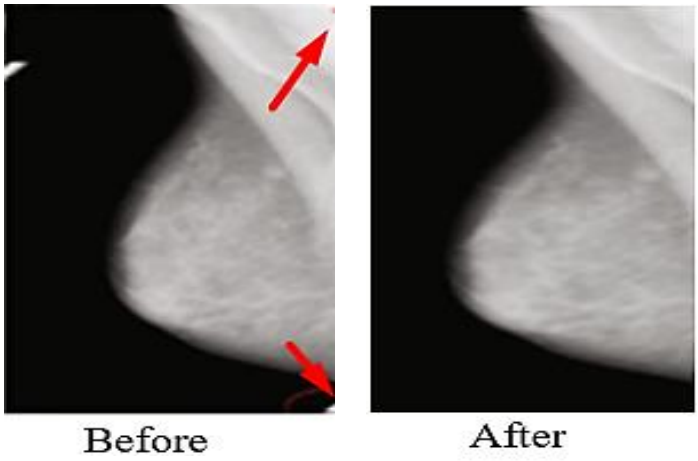


Figure 3.4:image with and without lines

3.4.2 Removal of Artifacts and Pectoral Muscles

Because the pectoral muscles and artifacts exhibit intensity levels similar to tumor regions, removing them from mammograms poses a significant challenge. To accurately isolate the tumor area, we found it essential to eliminate both the pectoral muscles and artifacts. After cropping the white border lines, we removed these elements based on a visual inspection of specific columns in the mammograms, as demonstrated in Figure 3.5. We handled the removal of the left pectoral muscles and their associated labels separately from the right pectoral muscles and their labels to ensure precise preprocessing.

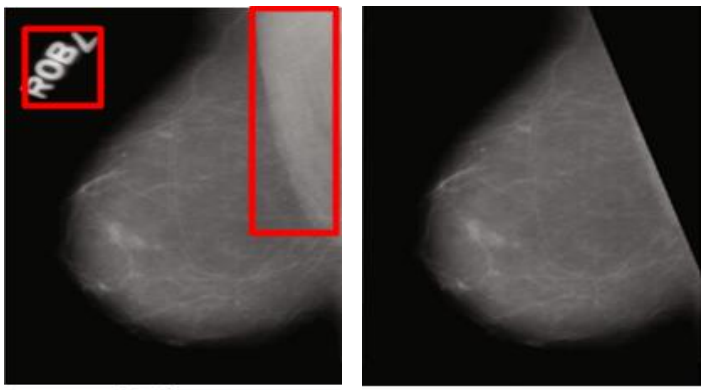


Figure 3.5: Left Image with Pectoral Muscle and Right Image after Pectoral Muscle Removal.

3.4.3 Contrast Limited Adaptive Histogram Equalization (CLAHE)

After removing the pectoral muscles and the labels, we applied Contrast Limited Adaptive Histogram Equalization (CLAHE) to enhance the mammograms. Image enhancement is crucial in medical imaging because it reveals hidden features within the images. In our study, we applied the CLAHE technique to improve the visibility of the tumor regions. CLAHE operates by dividing the image into small sections called tiles rather than processing the entire image at once. To avoid artificial boundaries between these tiles, we blended adjacent tiles using bilinear interpolation. When applying CLAHE, we considered two important parameters: the clip limit, which controls the contrast threshold and was set to 40 by default, and the tile grid size, which defines the number of tiles in rows and columns, with a default value of 8×8. Figure 3.6 illustrates an image before and after applying CLAHE.

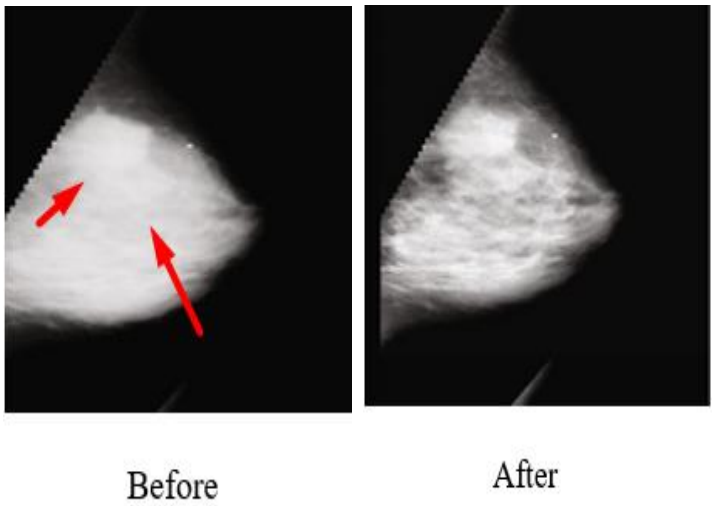


Figure 3.6:Before and after applying CLAHE, respectively

3.4.4 Morphological Operation

Morphological image processing comprises a set of nonlinear techniques that focus on the shape and structure of image features. These operations are commonly used to remove small, irrelevant details while preserving larger, important structures. In our study, we applied morphological operations to eliminate surrounding tissues and highlight the tumor region. Specifically, we employed erosion to enhance the tumor's prominence by removing small unnecessary tissue elements. Figure 3.7 illustrates the image before and after applying the morphological erosion operation.

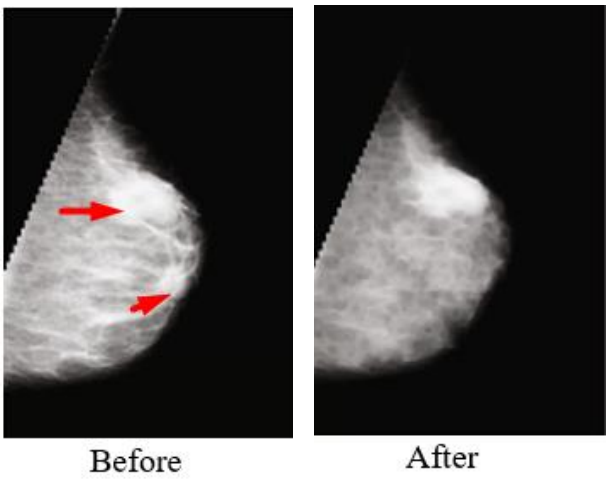


Figure 3.7:Before and after applying erosion morphological operation, respectively

3.4.5 Annotation

After applying the erosion operation, we prepared the dataset for tumor detection. In this study, we utilized YOLO models to detect breast tumors, which required annotated data. To create these annotations, we used Roboflow which is an online platform. We annotated the images by drawing square bounding boxes around the tumors. The annotated dataset is presented in Figure 3.8. Each annotation consists of two file types: an image file and a corresponding text (txt) file. The text file contains the coordinates and dimensions of the bounding box surrounding the tumor.

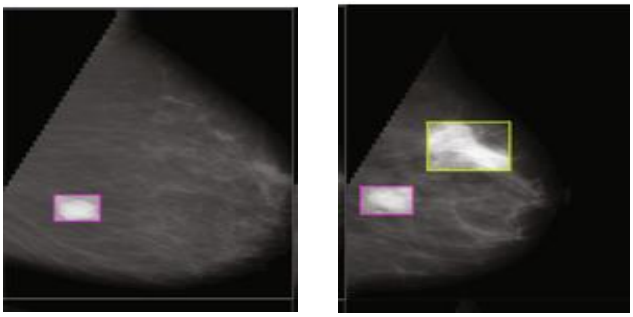


Figure 3.8:Annotated images

After the labeling process, we got the information about the object desired which is typically represented by four values: the coordinates of the top-left corner ($Xmin, Ymin$) and the coordinates of the bottom-right corner ($Xmax, Ymax$) and save them in a CSV (comma-separated values) file. This is a sample of the CSV file structure:

filename	width	height	class	xmin	ymin	xmax	ymax
BC001_pr	300	300	tumor	112	60	220	180
BC002_pr	300	300	tumor	130	70	240	200
BC003_pr	300	300	tumor	125	55	215	185
BC004_pr	300	300	tumor	120	80	210	210
BC005_pr	300	300	tumor	135	65	225	195
BC006_pr	300	300	tumor	128	75	218	205
BC007_pr	300	300	tumor	140	60	230	190
BC008_pr	300	300	tumor	115	85	205	215
BC009_pr	300	300	tumor	132	78	222	202

Figure 3.9:Segement from a CSV file.

3.4.6 Data augmentation

Data augmentation is a technique that we used to artificially increase the size and diversity of our dataset by applying various transformations to the original images. In this project, we employed data augmentation to enhance the performance of the YOLO model by reducing overfitting and improving its ability to generalize. We applied horizontal and vertical flipping, rotation, scaling, and brightness adjustment, These transformations simulate real-world

variations, making our model more robust in detecting breast tumors in mammogram images. Figure 3.9 demonstrates how data augmentation increases the training sample size and boosts model performance.

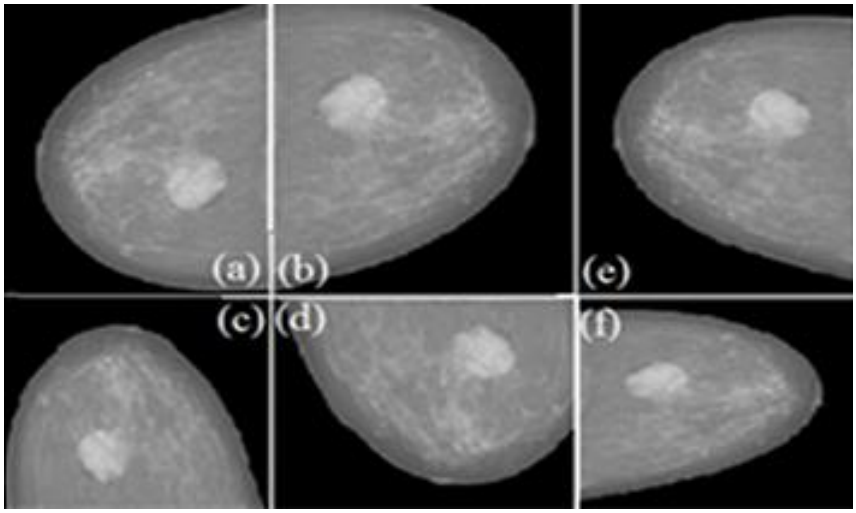


Figure 3.10:Augmentation of data

3.4.7 Data distribution

Data partitioning, or data distribution, is a crucial step in preparing the dataset for training deep learning models such as YOLO. After completing the preprocessing steps and augmenting the data, we divided the dataset into three subsets: training, validation, and testing. We allocated 60% of the data for training the model, 30% for validating and fine-tuning its parameters during training, and the remaining 10% for evaluating the final model's performance. We ensured that each subset contained a balanced number of tumor cases to enable the model to learn effectively and generalize well. This careful distribution helped us prevent bias and overfitting, leading to more reliable and accurate detection of breast tumors in unseen mammogram images**.**

3.5 Convolution Neural Network Implementation

To achieve our goal of "object detection": first, we tried to build a convolution neural network model, we decided to use Google Colab which allows us to use its free GPU.

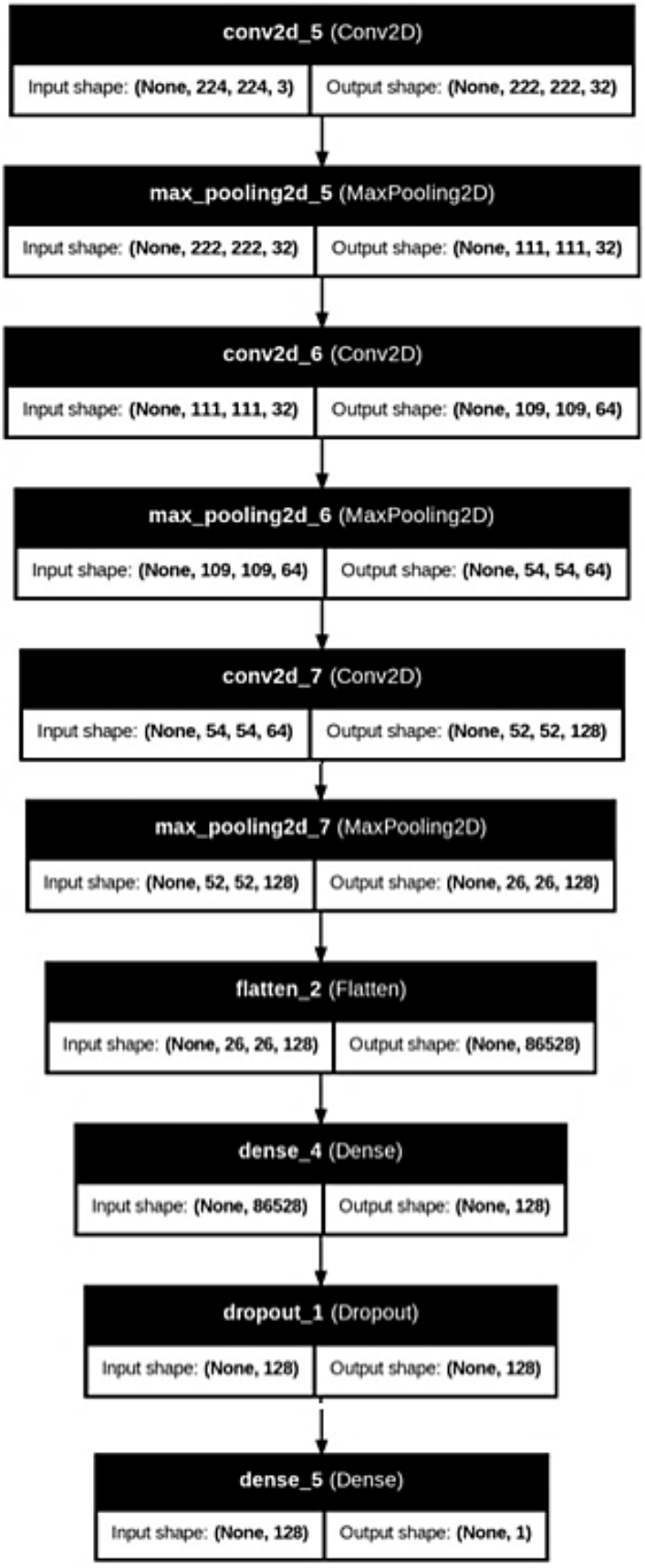


Figure 3.11:CNN Architecture.

3.5.1 Training CNN model

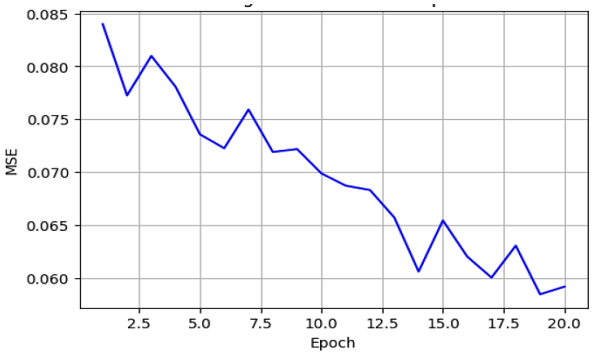
We trained a Convolutional Neural Network (CNN) model on breast cancer datasets to develop an AI system capable of detecting tumors from mammogram images. Our process began with collecting and preparing labeled images annotated for tumor presence and location. We preprocessed the images by resizing, normalizing, and enhancing them to improve feature learning. During training, the CNN learned to extract crucial features such as shapes, textures, and edges that help identify tumors. However, despite these efforts, the CNN model did not achieve sufficiently accurate results for reliable tumor detection. Consequently, we decided not to rely on this model and instead transitioned to using the YOLO (You Only Look Once) architecture, which demonstrated superior accuracy and faster detection capabilities.

3.5.1.1 Result using Breast Cancer Dataset

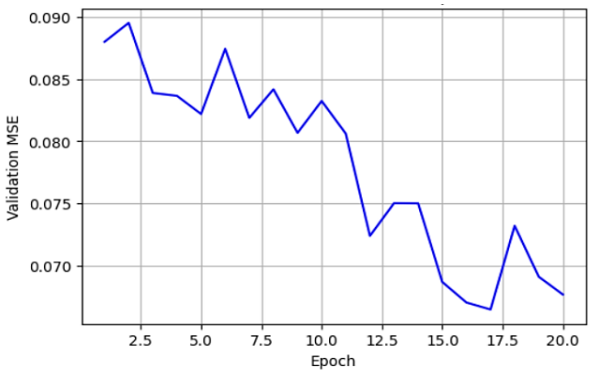
Table 3.1 presents an example of the IoU outcomes obtained from 6 test images, selected from a larger dataset, following the training of the network for 20, 60, and 100 epochs, respectively.

	IOU		
Images	**Epoch 20**	**Epoch 60**	**Epoch 100**
1	**0.41**	**0.53**	**0.45**
2	0.39	0.51	0.42
3	0.37	0.49	0.40
4	**0.35**	0.48	0.39
5	0.38	0.50	0.41
6	0.38	**0.47**	**0.38**

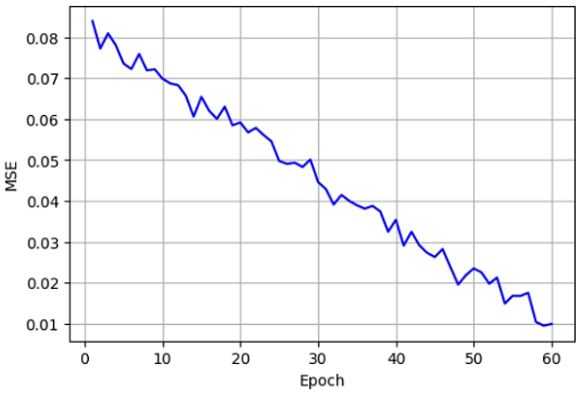
Table 3.1: IOU results, 6 samples.



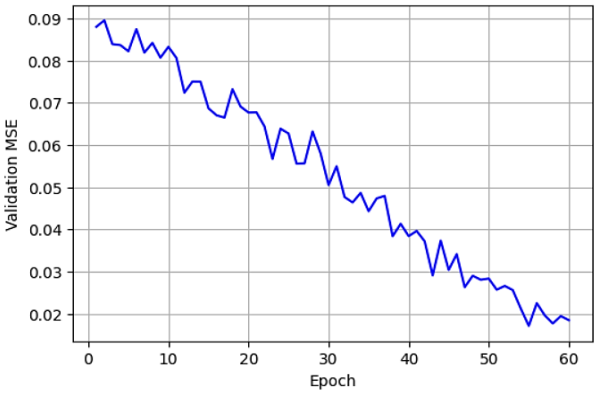
Training curve for 20 epochs



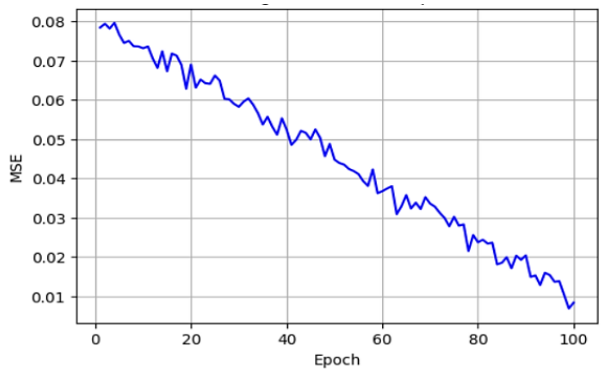
Validation curve for 20 epochs



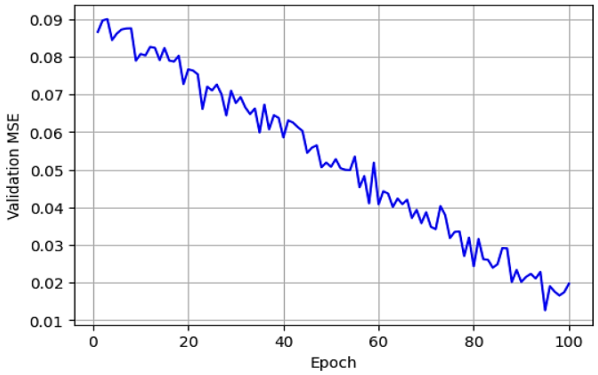
Training curve for 60 epochs



Validation curve for 60 epochs



Training curve for 100 epochs



Validation curve for 100 epochs

Figure 3.12:Breast cancer performance metrics

The results we obtained from training a Convolutional Neural Network (CNN) on the breast cancer dataset revealed several limitations. Although our model was able to learn some relevant features from the mammogram images, the overall detection performance remained unsatisfactory. Specifically, the Intersection over Union (IoU) values were consistently low across different epochs, indicating that our CNN struggled to accurately localize tumors.

This suggests that the classical CNN architecture is not sufficiently robust for precise tumor detection in mammographic images. As a result, these findings led us to consider more advanced object detection models, such as YOLO, to achieve better accuracy and reliability in breast cancer detection tasks.

3.6 YOLO implementation

We decided to switch to YOLO due to its superior performance and extensive training on large datasets. We implemented YOLOv5, YOLOv8, and the latest version, YOLOv12 (released on February 18th, 2025), and conducted a thorough comparison among them.

3.6.1 The difference between YOLOv5, YOLOv8 and YOLOv12

When it comes to object detection, there are many models available, but YOLOv5, YOLOv8, and YOLOv12 stand out as three of the most popular and advanced models developed by Ultralytics.

YOLOv5 is recognized for its speed, simplicity, and accuracy, making it a reliable choice for real-time applications and resource-constrained environments due to its PyTorch-based framework that facilitates easy integration and deployment.

YOLOv8 builds upon these strengths by introducing architectural improvements that enhance both accuracy and processing speed, as evidenced by its superior performance in benchmarks, making it more suitable for demanding real-time detection tasks.

The latest iteration, YOLOv12, incorporates advanced techniques such as area attention to better capture contextual information, achieving higher accuracy than previous versions however, this comes with a trade-off of slightly reduced speed, positioning YOLOv12 as the preferred model when precision is critical over raw processing speed.

Choosing the best object detection model depends on several key factors, including speed, accuracy, and ease of use. Both YOLOv8 and YOLOv5 offer fast processing speeds, with YOLOv8 being significantly faster, while YOLOv12 pushes the boundaries of detection accuracy and intelligent feature extraction.

In summary, YOLOv5 is best for those prioritizing simplicity and fast deployment, YOLOv8 offers superior speed and flexibility, especially for real-time applications, and YOLOv12 leads in accuracy and advanced features, suitable for tasks demanding the highest precision. The choice between these models depends on the specific requirements and constraints of the intended application. In our study, we chose to implement all three models to evaluate their performance in the highly critical task of breast cancer detection.

Models	Params (Million)	Accuracy (mAP)	CPU Time(ms)	GPU Time(ms)
YOLOv5n	1.9	45.7	**45**	**6.3**
YOLOv8n	**3.2**	**37.3**	8.4	0.99
YOLOv12n	2.6	40.6	20	1.64

Table 3.2: Total Parameters of YOLOv5n and YOLOv8n and YOLOv12n models.

3.6.2 Yolov5

YOLOv5 is widely used for detecting and classifying breast tumors, and it includes four different model variants: YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. These variants mainly differ in the number of feature extraction modules and the size of convolutional kernels at specific points within the network. Consequently, the model size and the number of parameters increase progressively from YOLOv5s to YOLOv5x, as detailed in Table 3.3. This hierarchical design allows users to select a model that best balances accuracy and computational efficiency for their specific application.

Model	Parameters	Model Size (MB)	Inference Speed (ms)	Input Size
YOLOv5n (Nano)	1.9 M	~3.9	7.0	640×640
YOLOv5s (Small)	7.2 M	~14.0	6.4	640×640
YOLOv5m (Medium)	21.2 M	~41.0	8.2	640×640
YOLOv5l (Large)	46.5 M	~90.0	10.1	640×640

Table 3.3: Difference between four versions of Yolov5

3.6.2.1 Yolov5n Result

The first model we used was YOLOv5n (Nano), and we implemented it in Google Colab. This setup allowed us to efficiently run the lightweight YOLOv5n model using Colab's cloud-based GPU resources, enabling fast experimentation without the need for high-end local hardware. Through this implementation, we were able to test the model's performance on object detection tasks and obtained the results shown in Figures 3.13, 3.14, and 3.15.

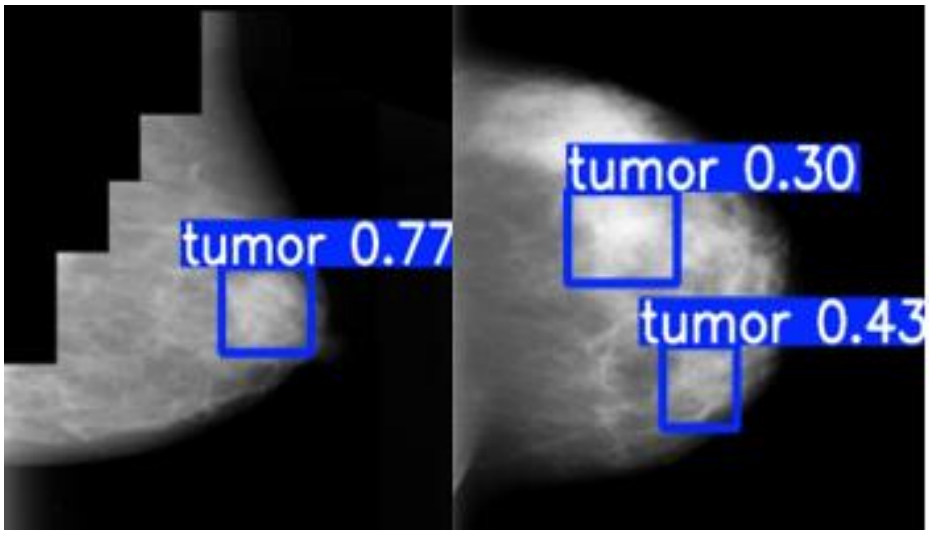


Figure 3.13: Breast cancer detection using yolov5n.

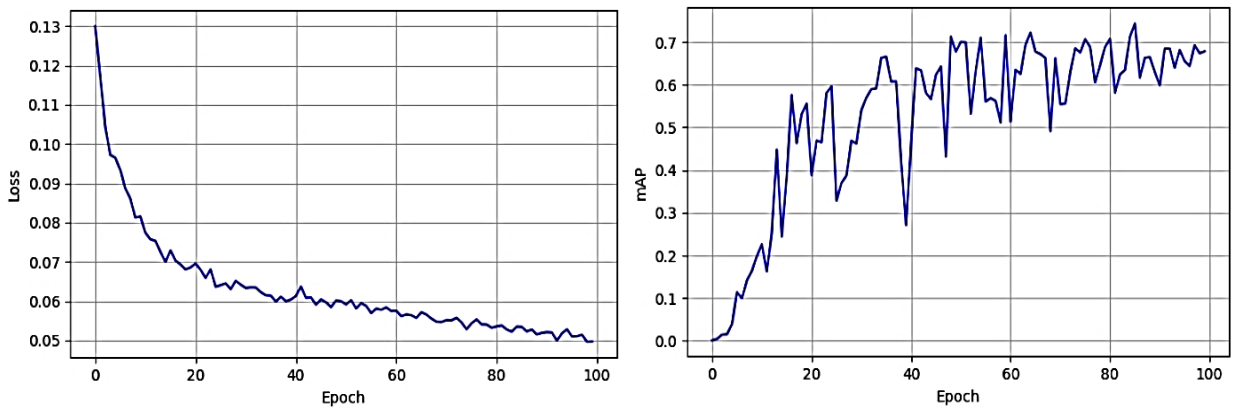


Figure 3.14: MAP and loss plots after training the YOLOv5 Nano on the breast cancer detection Dataset.

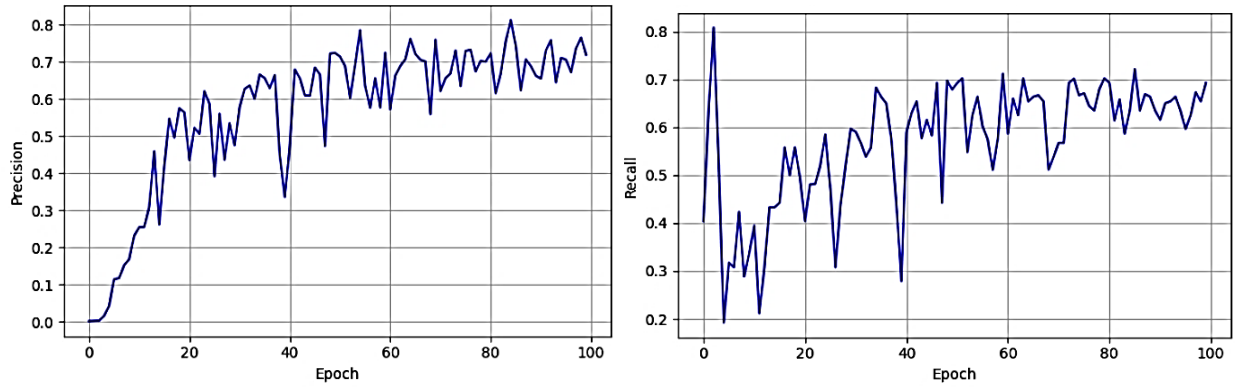


Figure 3.15: Precision and Recall plots after training the YOLOv5 Nano on the Breast cancer

As shown in Figure 3.14, the YOLOv5n model demonstrates stable learning behavior, with a smooth decrease in the loss curve from approximately 0.13 to 0.05, indicating consistent convergence throughout training. On the other hand, the mAP (mean Average Precision), despite some fluctuations, improves significantly and stabilizes around 0.7 after approximately 50 epochs, suggesting good detection accuracy. As illustrated in Figure 3.15, the precision and recall curves further support these observations: precision increases steadily and recall shows a positive trend, both confirming the model's improving ability to accurately detect tumors while minimizing false positives and false negatives. Despite its lightweight nature, YOLOv5n achieves reasonable performance, making it a good option for scenarios with limited computational resources while offering remarkable reliability in complex medical tasks.

In conclusion, the results demonstrate that YOLOv5n, despite being a lightweight model, is capable of delivering strong performance in tumor detection:

- The **loss is low**, indicating effective learning.
- The **mAP is relatively high**, reflecting good overall detection performance.
- The **precision is excellent**, ensuring few false alarms.
- The **recall is reasonable**, meaning most tumors are successfully detected.

3.6.2.2 Yolov5s Result

Next, we tested the YOLOv5s (Small) model on Google Colab, taking advantage of the platform's resources to efficiently run the model. We then evaluated its object detection performance and obtained the following results.

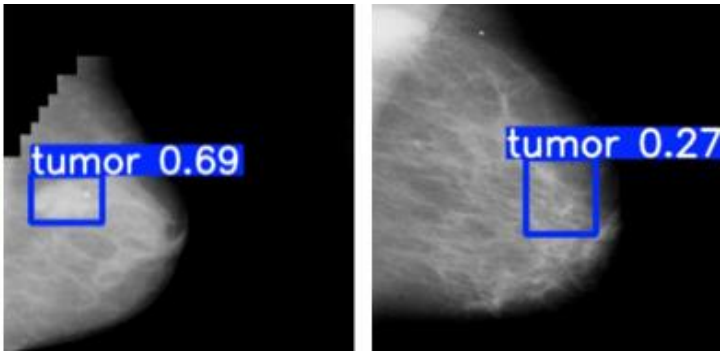


Figure 3.16:Breast cancer detection using yolov5s

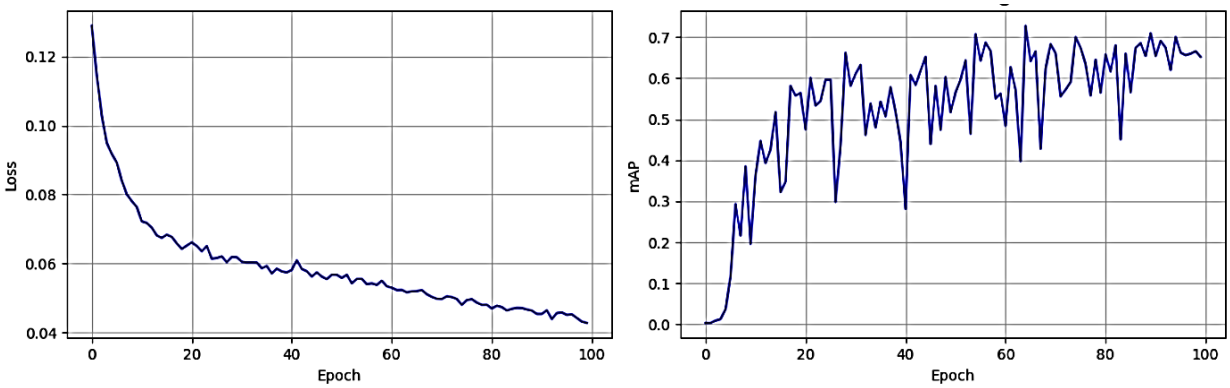


Figure 3.17:MAP and loss plots after training the YOLOv5 Small on the breast cancer.

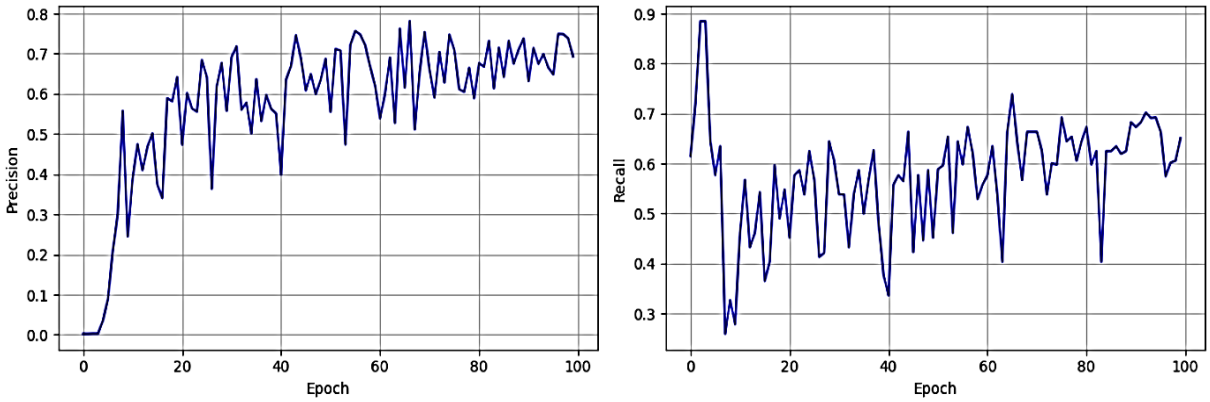


Figure 3.18:Precision and Recall plots after training the YOLOv5 small on the Breast cancer.

As shown in Figure 3.16, The training results of the YOLOv5s model demonstrate its effectiveness in breast tumor detection. In Figure 3.17, we can see that the loss consistently decreases from 0.12 to 0.04, indicating effective learning and stable convergence, while the mAP (mean Average Precision) rapidly improves and stabilizes around 0.7, showing that the model achieves a good balance between precision and recall. Figure 3.18 shows that the precision reaches up to 0.8, meaning the model makes few false-positive predictions. Although the recall curve is somewhat fluctuating, it stabilizes around 0.7, indicating that most tumors are correctly detected. Overall, the YOLOv5s model demonstrates strong and reliable performance.

3.6.2.3 Yolov5l Result

Finally, we used the YOLOv5l (Large) model, which is recognized for its enhanced detection accuracy but comes with increased computational requirements. We evaluated the model's performance on object detection tasks and obtained the following results.

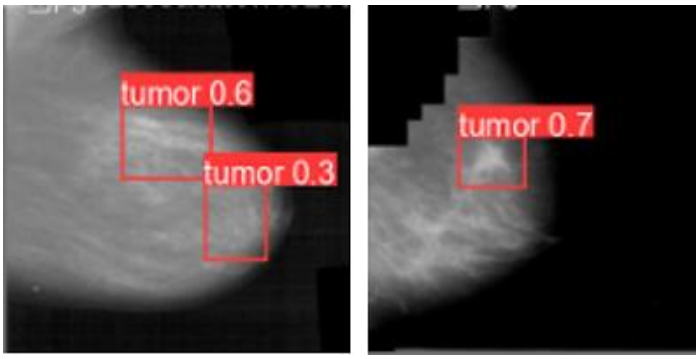


Figure 3.19:Breast cancer detection using yolov5l

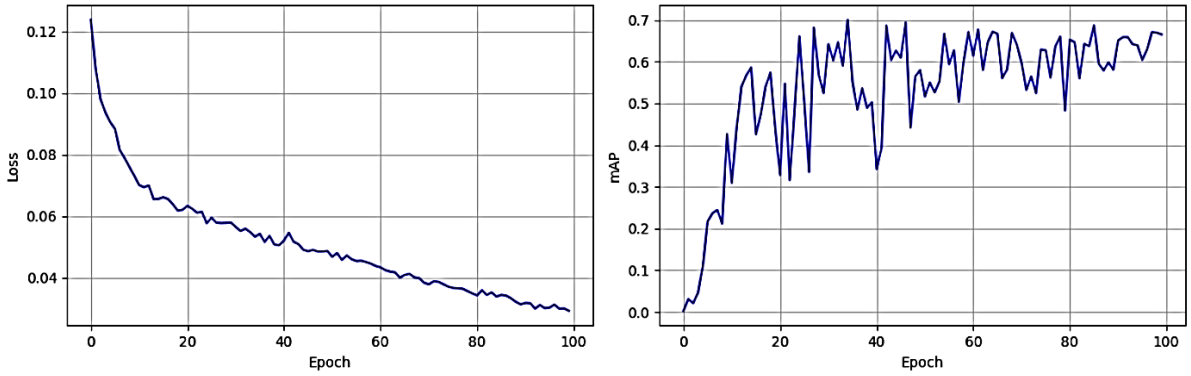


Figure 3.21:MAP and loss plots after training the YOLOv5 Large on the breast cancer

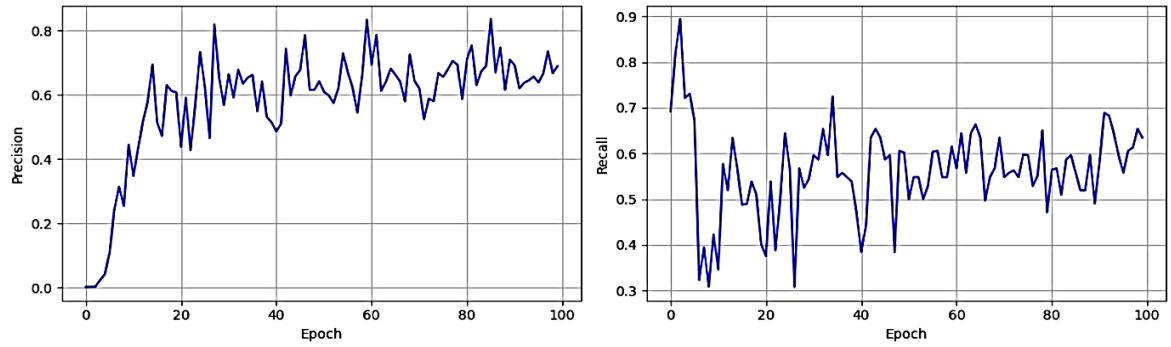


Figure 3.20:Precision and Recall plots after training the YOLOv5 large on the Breast tumor detection dataset.

As shown in Figure 3.20, the YOLOv5l model shows excellent learning performance over 100 epochs for breast tumor detection. The loss curve decreases steadily from around 0.12 to below 0.035, indicating effective model convergence. The mAP rapidly increases during the early epochs and stabilizes at approximately 0.6-0.7, showing that the model consistently achieves a high level of detection accuracy.

Figure 3.21 shows that the Precision improves significantly, reaching up to 0.8 and remaining stable, which means the model generates few false positives. Although the recall values show some fluctuations, they remain mostly between 0.6 and 0.7, indicating a generally good ability to identify actual tumors. Overall, YOLOv5l performs reliably and with slightly improved stability compared to smaller versions, making it suitable for detailed medical image analysis where accuracy is critical.

3.6.3 Comparaison between Yolov5 version (small, large, nano)

The results of the YOLOv5n, YOLOv5s, and YOLOv5l models show clear differences in performance. The YOLOv5n (Nano) model, being the smallest and fastest, achieves the lowest precision, recall, and mAP values among the three, indicating that it sacrifices accuracy for speed and efficiency.

The YOLOv5s (Small) model demonstrates improved results compared to YOLOv5n, with higher precision, recall, and mAP, making it a better balance between speed and accuracy. The YOLOv5l (Large) model outperforms both YOLOv5n and YOLOv5s, achieving the highest values in all key metrics. This shows that increasing the model size and complexity leads to better detection performance but at the cost of higher computational requirements. Overall, YOLOv5l is the most accurate, while YOLOv5n is the fastest and most lightweight, and YOLOv5s offers a compromise between the two.

3.6.4 Yolov8n Result

YOLOv8n is the smallest and fastest model in the YOLOv8 family, designed specifically for environments with limited computational resources like mobile devices and edge hardware. Despite its compact size of around 2 MB, it delivers impressive real-time object detection performance with a mean Average Precision (mAP) of about 47.2% on the COCO dataset. YOLOv8n achieves this by using an efficient backbone and neck architecture, including CSPDarknet53 and PANet, along with an anchor-free detection head that simplifies predictions and speeds up processing. This makes YOLOv8n ideal for applications where speed and

efficiency are critical, such as IoT devices or embedded systems, without sacrificing too much accuracy.

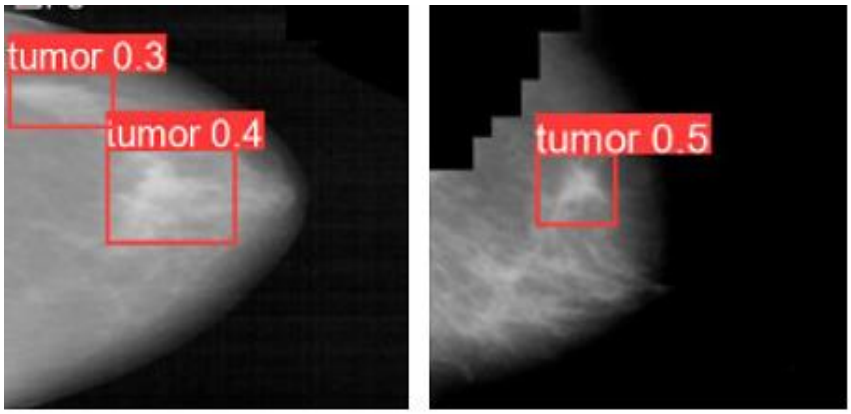


Figure 3.22:breast cancer detection using YOLOv8.

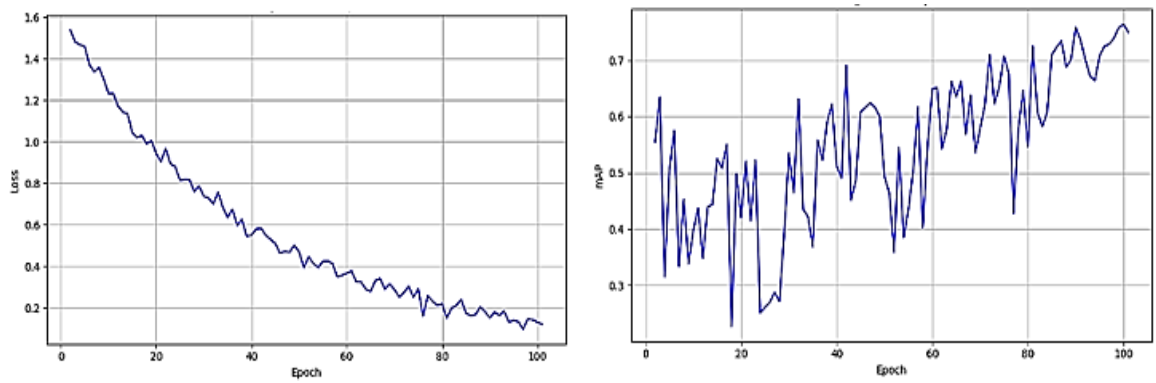


Figure 3.23:MAP and loss plots after training the YOLOv8 Nano on the breast cancer

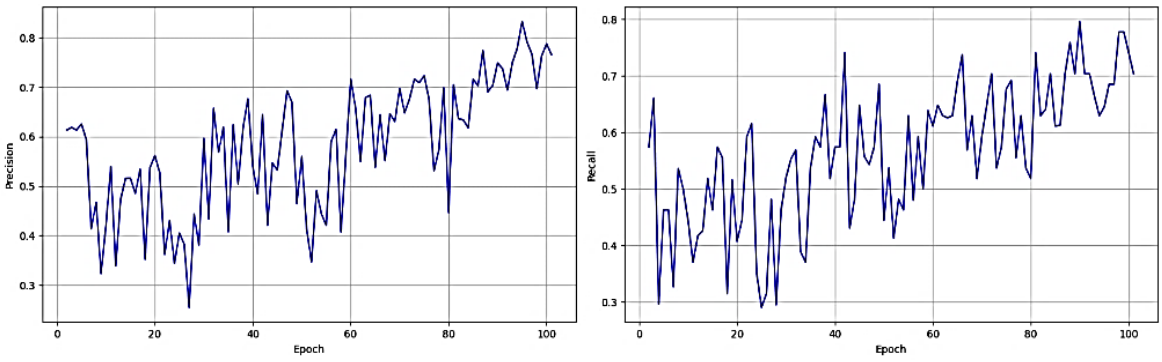


Figure 3.24:Precision and Recall plots after training the YOLOv8 Nano on the breast cancer detection Datasets.

The results presented for the YOLOv8n model in Figure 3.23 over 100 epochs show clear signs of effective training and convergence. The loss decreases steadily throughout the training, dropping from above 1.2 to below 0.3, which reflects improved model performance and reduced prediction errors. Meanwhile, the mean Average Precision (mAP) also rises significantly, stabilizing around 0.8 after 60 epochs, demonstrating that the model achieves strong overall detection accuracy. Figure 3.24 shows that the precision and recall metrics both increase rapidly during the initial epochs, with precision stabilizing around 0.75–0.8 and recall reaching similar high values after about 40 epochs, indicating that the model is learning to correctly identify and retrieve relevant objects.

These trends collectively suggest that the YOLOv8n model has trained successfully, with metrics indicating good generalization and reliable object detection performance.

3.6.5 YOLOv12n Result

YOLOv12n is the smallest and fastest variant of the latest YOLOv12 object detection models, designed for real-time applications where speed and efficiency are crucial. It features an attention-centric architecture that uses a novel Area Attention mechanism to focus on important regions of an image more efficiently, reducing computational cost while maintaining a large receptive field. YOLOv12n achieves impressive accuracy, especially for small objects, while running with very low latency, making it ideal for deployment on edge devices and scenarios requiring quick, accurate object detection. This balance of speed and precision marks a significant step forward in lightweight, real-time computer vision models.

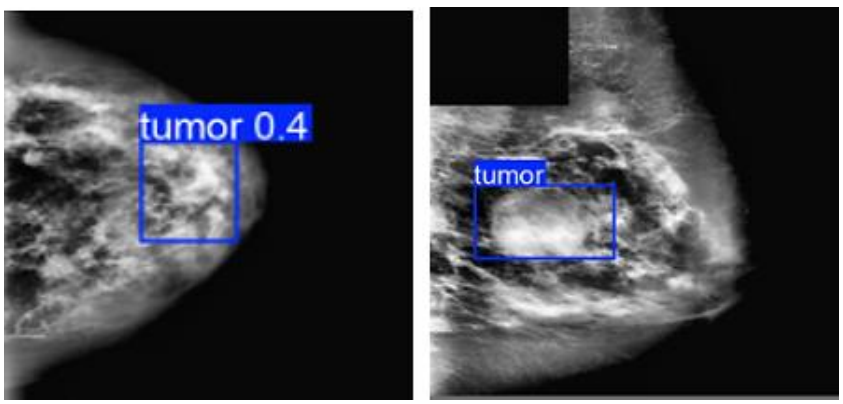


Figure 3.25:breast cancer using YOLOv12.

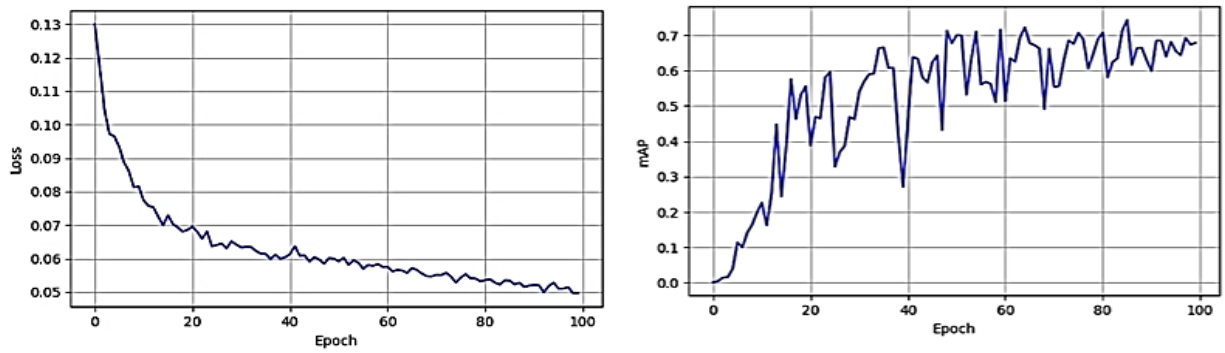


Figure 3.27:MAP and loss plots after training the YOLOv12 Nano on the breast cancer detection Dataset

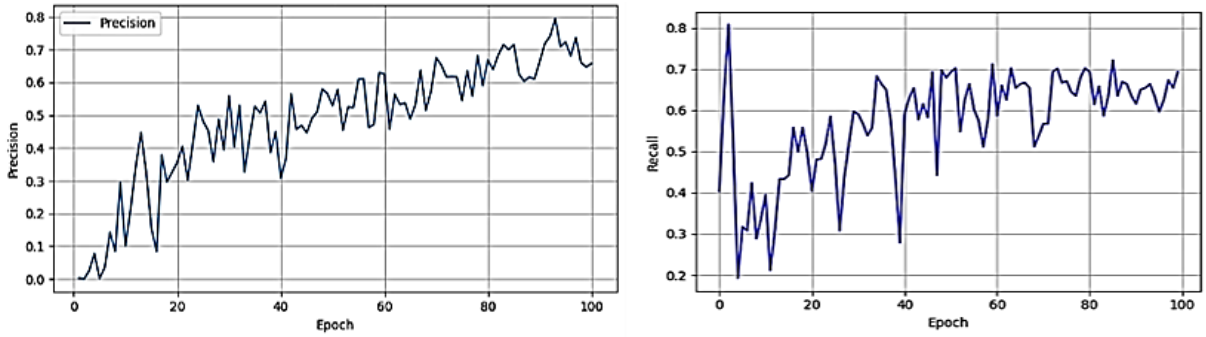


Figure 3.26:Precision and Recall plots after training the YOLOv12 Nano on the breast cancer detection Dataset

In Figure 3.26, we see that the loss curve exhibits a consistent decrease, dropping from approximately 0.13 to below 0.02, which reflects a significant reduction in prediction errors and effective learning by the model. Additionally, the mean Average Precision (mAP) increases rapidly during the initial epochs and stabilizes at values above 0.75, suggesting that the model achieves robust overall performance in object detection. Figure 3.27 shows that both the precision and recall metrics steadily increase, with precision rising from below 0.1 to around 0.8, and recall following a similar trend. This indicates that the model becomes more accurate and reliable in detecting relevant cases as training progresses.

Overall, these results confirm that the YOLOv12 Nano model successfully learns from the dataset and generalizes well, making it suitable for breast cancer detection tasks.

3.6.6 YOLOv5 ,YOLOv8 and YOLOv12 Results Interpretation

The results presented for the YOLOv5, YOLOv8, and YOLOv12 models highlight the evolution and improvement of object detection performance across different versions of the YOLO architecture. The YOLOv5 models, including Nano (n), Small (s), and Large (l), demonstrate a clear trend where larger models achieve higher precision, recall, and mean Average Precision (mAP), but require more computational resources. YOLOv5n offers fast inference with lower accuracy, while YOLOv5l achieves the best detection results among the YOLOv5 variants.

When comparing YOLOv5n with YOLOv8n and YOLOv12n, both YOLOv8n and YOLOv12n show noticeable improvements in detection metrics. YOLOv8n outperforms YOLOv5n in terms of precision, recall, and mAP, indicating better generalization and more reliable object detection. YOLOv12n further builds on these improvements, achieving the highest values among the three models. This suggests that the latest YOLOv12 architecture incorporates significant advancements in model design and training, resulting in superior accuracy and robustness.

In summary, as we progress from YOLOv5 to YOLOv8 and then to YOLOv12, there is a consistent enhancement in detection performance. YOLOv12n stands out as the most effective model for accurate object detection, while YOLOv5n remains the fastest and most lightweight. These results demonstrate the continuous development and optimization of the YOLO family of models for various object detection applications

3.7 Conclusion

In this experimental part, we have thoroughly explored the fundamental components required to implement a robust object detection system using YOLO (You Only Look Once) models. We began by outlining the overall architecture and describing the essential software tools and libraries that supported the development process, such as Python, PyTorch, and OpenCV. A detailed overview of the dataset, along with the preprocessing steps including border removal, pectoral muscle elimination, CLAHE, and annotation emphasized the critical role of data quality and preparation in ensuring accurate and consistent results.

The implementation phase demonstrated the application of three YOLO versions YOLOv5, YOLOv8, and YOLOv12 highlighting the evolution in performance, detection precision, and computational efficiency across successive model generations. Through

comparative analysis, we were able to evaluate the strengths of each model and their suitability for medical imaging tasks, particularly in detecting breast tumors in mammograms.

Overall, this chapter provides a solid foundation for understanding the practical aspects of object detection using YOLO. It sets the stage for more advanced experimentation, such as hyperparameter tuning, deployment on edge devices, and integration with real-time diagnostic tools in future chapters.

General Conclusion

General Conclusion

Since the beginning of this research, we have made significant progress toward achieving our main objective of effective object detection. Our work involved implementing and evaluating advanced YOLO models on specialized datasets, primarily focusing on breast cancer detection through mammography images.

In the theoretical part of our thesis, presented in Chapter 1, we explored the fundamentals of deep learning with a particular focus on Convolutional Neural Networks (CNNs), which are essential for image-based tasks. Chapter 2 provided an in-depth survey of object detection techniques, tracing their evolution and identifying the most promising approaches for real-time and accurate detection. Finally, Chapter 3 detailed the practical implementation and comparative analysis of YOLO models, specifically YOLOv5, YOLOv8, and YOLOv12. Our experiments demonstrated that the latest versions, especially YOLOv8 and YOLOv12, offer superior accuracy and speed, making them highly suitable for real-world applications.

Throughout this project, we have not only deepened our understanding of artificial intelligence and deep learning principles but also gained valuable hands-on experience with programming tools such as Python and platforms like Google Colab. This journey has broadened our perspective on how AI can transform various sectors by providing efficient, reliable, and scalable solutions.

In summary, this thesis bridges theoretical knowledge and practical application, highlighting the promising future of AI-driven object detection in healthcare.

Bibliography

Bibliography

[1] Gharaibeh, M., Alzu'bi, D., Abdullah, M., Hmeidi, I., Al Nasar, M. R., Abualigah, L., & Gandomi, A. H. (2022). Radiology imaging scans for early diagnosis of kidney tumors: a review of data analytics-based machine learning and deep learning approaches. Big Data and Cognitive Computing, 6(1), 29

[2] Dhande, M. (2017). What is the difference between AI, machine learning and deep learning. Geospatial World.

[3] Bento, C. (2021). Multilayer perceptron explained with a real-life example and python code: Sentiment analysis. Towards Data Science.

[4] BEGGARI, S., & KHAMRA, K. (2017). Système de reconnaissance de visage par un réseau de neurone convolutionnel (CNN).

[6] Heaton, J. (2018). Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618. Genetic programming and evolvable machines, 19(1), 305-307.

[8] Gaspar, A., Oliva, D., Cuevas, E., Zaldívar, D., Pérez, M., & Pajares, G. (2021). Hyperparameter optimization in a convolutional neural network using metaheuristic algorithms. In Metaheuristics in machine learning: Theory and applications (pp. 37-59). Cham: Springer International Publishing.

[9] Kinsley, H., & Kukieła, D. (2020). Neural networks from scratch in Python. Harrison Kinsley.

[10] Schilling, F. (2016). The effect of batch normalization on deep convolutional neural networks.

[11] Heckman, J. J., Pinto, R., & Savelyev, P. A. (1967). Artificial Intelligence, IOT and Machine Learning. Angew. Chemie Int. Ed. 6 (11), 951, 952..

[12] Beysolow II, T. (2017). Introduction to deep learning using R: A step-by-step guide to learning and implementing deep learning models using R. Apress.

[13] Aggarwal, C. C. (2018). Neural networks and deep learning (Vol. 10, No. 978, p. 3). Cham: Springer.

[14] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. Pattern recognition, 77, 354-377.

[15] Courbariaux, M., Bengio, Y., & David, J. P. (2015). Binaryconnect: Training deep neural networks with binary weights during propagations. Advances in neural information processing systems, 28.

[16] CHACHOUA, R., & BENSAHA, I. (2021). Radio signal classification using deep learning (Doctoral dissertation, université Ghardaia).

[17] Sha, R. (2022). Pneumonia Classification Model by Convolutional Neural Network.

[18] Bega, D., Gramaglia, M., Fiore, M., Banchs, A., & Costa-Perez, X. (2019, April). DeepCog: Cognitive network management in sliced 5G networks with deep learning. In IEEE INFOCOM 2019-IEEE conference on computer communications (pp. 280-288). IEEE.

[19] Vasilenko, V., Korolchenko, D., & Thanh, P. N. (2018). Definition of the inspection criteria for personal protective equipment (for work at heights) on example of full body harnesses. In MATEC Web of Conferences (Vol. 251, p. 02042). EDP Sciences.

[20] Li, P., Wang, B., & Zhang, L. (2021). Virtual fully-connected layer: Training a large-scale face recognition dataset with limited computational resources. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 13315-13324).

[21] Wani, M. A., Bhat, F. A., Afzal, S., Khan, A. I., Wani, M. A., Bhat, F. A., ... & Khan, A. I. (2020). Basics of supervised deep learning. Advances in Deep Learning, 13-29..

[23] [Zou, Z., Chen, K., Shi, Z., Guo, Y., & Ye, J. (2023). Object detection in 20 years: A survey. Proceedings of the IEEE, 111(3), 257-276.

[24] Hossain, J., & Momtaz, M. (2023). Follow the Soldiers with Optimized Single-Shot Multibox Detection and Reinforcement Learning. arXiv preprint arXiv:2308.01389.

[25] A-143, 7th Floor, Sovereign Corporate Tower, Sector- 136, Noida, Uttar Pradesh (201305) K 061, Tower K, Gulshan Vivante Apartment, Sector 137. Noida, Gautam Buddh Nagar, Uttar Pradesh, 201305 https://www.geeksforgeeks.org/faster-r-cnn-ml/

[26] Mittal, N., Vaidya, A., & Kapoor, S. (2019). Object detection and classification using Yolo. Int. J. Sci. Res. Eng. Trends, 5(2), 871-875.

[27] Xu, G., Xie, Y., Luo, Y., Yin, Y., Li, Z., & Wei, Z. (2024). Advancing Automated Surveillance: Real-Time Detection of Crown-of-Thorns Starfish via YOLOv5 Deep Learning. Journal of Theory and Practice of Engineering Science, 4(06), 1-10.

[28] : Padilla, R., Netto, S. L., & Da Silva, E. A. (2020, July). A survey on performance metrics for object-detection algorithms. In 2020 international conference on systems, signals and image processing (IWSSIP) (pp. 237-242). IEEE.

Weboghraphie

[5] https://medium.com/codex/what-are-neural-networks-3a0965e2ebfb

[7] https://www.geeksforgeeks.org/backpropagation-in-neural-network/

[22] https://www.superannotate.com/blog/object-detection-with-deep-learning