People's Democratic Republic of Algeria

The Ministry of Higher Education and Scientific Research

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

Institute of Electronics and Telecommunications (IET)

Department of Electronic and Embedded Systems ESE

Thesis

Presented for the

The Master's degree

Faculty: Science and Technology

Specialty: Embedded Systems

By

- **SOUIHER Mohammed EL Amine**
- **NOUFEL Zakaria**

Entitled

Object Detection Using YOLO: Applications to Medical Images

Defended on: 26/09/2026

Before a jury composed of:

Last name & First name	Grade	Role	Institution
Dr. SARRA Mustapha	Dr.	President	Univ-BBA
Dr. SID AHMED Soumia	Dr.	Supervisor	Univ-BBA
Dr. BELGUIDOUM Khaoula	Dr.	Examiner	Univ-BBA

Academic year 2025/2026

Acknowledgements

First and foremost, we praise and thank Allah, the Almighty, for granting us the strength, patience, and perseverance to complete our studies and this thesis .

We would like to express our sincere gratitude to our supervisor, **Dr. SID AHMED Soumia**, for her valuable guidance, constructive feedback, patience, and continuous support throughout the preparation of this work. Her advice and recommendations were greatly appreciated and contributed significantly to the completion and improvement of this thesis.

We would also like to extend our sincere thanks to all the teachers of the **University of Mohamed El-Bachir El-Ibrahimi**, whose knowledge, guidance, and dedication have contributed to our academic development throughout our university studies.

Finally, we would like to express our deepest gratitude to our families for their unconditional love, encouragement, patience, and support throughout our academic journey. Their presence and encouragement have been an essential source of motivation, and we are truly grateful for everything they have done for us.

"بعض الطرق ليست سهلة"
وإنما وُجدت لتظهر لنا
ما نحن قادرون على أن نُصبحه".

إهداء



أولاً وأقبل كل شيء، أشكر الله عز وجل على منحي القوة والصبر والإصرار للوصول إلى هذه اللحظة، فكل ما كان بيدٍ بعيدٍ أصبح اليوم حقيقة، وأنا ممتن لكل درس وكل تحدٍ وكل خطوة قادتني إلى هنا.



إلى أمي، التي كانت دائما مصدر قوتي وطمأنينتي، إلى المرأة التي آمنت بي حتى عندما شككت في نفسي، والتي استمعت إليّ حين احتجت إلى الحديث، ودُعمت لي في كل لحظة صعبة، صبرها وحنانها ودعمها لما يعبر عن شكري وامتناني لها.



إلى أبي، الذي الذي علمني، أن لا شيء يحقق دون جهد، وأن أتحلى بالصبر عندما لا تسير الأمور كما أريد، وأن أواصل رغم الصعوبات؛ دعمك وثقتك بي كانا من أهم أساس ما وصلت إليه، وهذا الإنجاز يحمل جزءاً من تضحياتك وتوجيهاتك وإيمانك بيَ.



إلى إخوتي وأخواني، لأنكم كنتم دائما إلى جانبي، في لحظات الفرح وفي أصعب المواقف. وجودكم جعل الطريق أسهل وأجمل.



إلى كل من وقف بجانبي، بكلمة طيبة أو نصيحة أو دعم أو صداقة، أقدم لكم شكري العميق. فبعض الأشخاص لا يذركون كم يمكن لخضورهم أن يعني لنا في الأوقات الصعبة، لكن أثرهم بيقى طويلاً بعد أن تمر اللحظة.



إلى أستاذتي الدكتورة سيد أحمد سمية، على إشرافها وتوجيهها وصبرها وكل ما قدمته من نصائح قيمة لإنجاز هذا العمل. دعمها كان له دور كبير في إكمال هذه المذكرة، وأشكرها من قلبي على وقتها وجهدها.



وأخيراً، أهدي هذا العمل إلَى نَفسي، إلى النسخة التي لم تستسلم، رغم الصعوبات، وإلى كل ليلة سهر وكل عقبة تجاوزتها. لم يكن هذا الإنجاز مجرد شهادة أو مذكرة، بل هو اختبار للصبر والانضباط والإرادة والإيمان، فكل ما مررت به كان جزءاً من رحلتي التي صنعت مني ما أنا عليه اليوم.

وكذلك... إلى من تضيء أيامي بوجودهم، إلي يناني



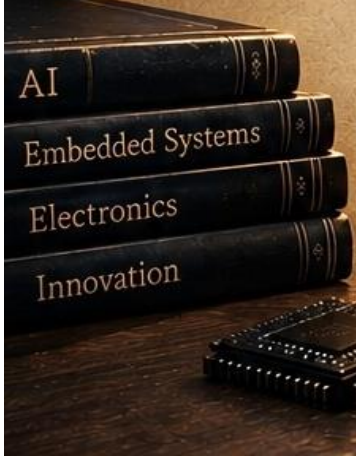
إلى تلك الروح الجميلة التي كانت دائماً إلى جانبي
بصمت، تشجعني بكلماتها، وتمنحني طاقة أمل لا تنتهي،
إلى كل لحظة جميلة تجمعنا، وإلى كل حلم نخطط له معًا...
شكراً لكونك جزءاً من رحلتي، وسبباً في أن أكون أفضل.

هذا العملُ أكثر من مجرد إنجاز أكاديمي، إنه دليل على أن المثابرة
يمكن أن تحول الحلم البعيد إلى حقيقة.

إلى والدي، وإلى عائلتي، ومعلميني، وأصدقائي، وكل من ساهم في هذه الرحلة
ولو بكلمة طيبة... هذا الإنجاز يحمل جزءاً منكم جميعاً.

مع خالص الامتنان، والتقدير، والأمل لما هو قادم..

نوفل زكرياء





"Some roads are not meant to be easy; they are meant to show us what we are capable of becoming."

DEDICATION

First and foremost, I thank **Allah** for giving me the strength, patience, and determination to reach this moment. What once seemed like a distant goal has finally become a reality, and I am deeply grateful for every lesson, every challenge, and every step that brought me here.

I dedicate this work to **my mother**, whose love has always been a quiet source of strength in my life. To the woman who believed in me even when I doubted myself, who listened when I needed to speak, and whose prayers followed me through every difficult moment. Her patience, kindness, and unconditional support have accompanied me far beyond what words can express. Whatever I achieve in life, a part of it will always belong to her.

To **my father**, whose example taught me that nothing meaningful comes without effort. You taught me to remain patient when things do not go as planned, to keep moving forward when the road becomes difficult, and to face challenges without giving up. Your support and confidence in me have been among the foundations of everything I have accomplished. This achievement is not mine alone; it carries a part of your sacrifices, your guidance, and your belief in me.

To **my brothers and sisters**, thank you for being part of this journey. For the moments of laughter during stressful days, for the encouragement when I needed it, and simply for being there. Your presence made many difficult moments easier to overcome.

To **the people** who stood beside me throughout this journey, whether through advice, encouragement, friendship, or a simple word at the right time, I offer my sincere gratitude. Some people may never realize how much their presence can mean during difficult periods, but their impact remains long after the moment has passed.

I also express my deepest appreciation to my supervisor, **Dr. Sid Ahmed Soumia,** for her guidance, patience, and valuable advice throughout the preparation of this work. Her support and direction contributed greatly to the completion of this thesis, and I am sincerely grateful for the time and effort she devoted to this work.

Finally, I dedicate this work to **myself.**
To the version of me who kept going when things became difficult. To every late night, every setback, every moment of doubt, and every time I had to start again. This journey was never only about completing a degree or writing a thesis. It was a test of patience, discipline, resilience, and faith.

There were moments when the easiest choice would have been to stop, but I chose to continue. And today, standing at the end of this chapter, I realize that the difficulties were not simply obstacles on the way; they became part of what made the achievement meaningful.

This thesis is therefore more than an academic accomplishment. It is a reminder that even when the path becomes uncertain, persistence can turn a distant dream into something real.

To my parents, my family, my teachers, my friends, and everyone who contributed to this journey in one way or another this achievement carries a part of all of you.

With gratitude, pride, and hope for what comes next.



Souiher Mohammed Amine

MASTER'S THESIS

EMBEDDED SYSTEMS & ELECTRONICS



Table Of Contents

List of Figures VII

List of Tables IX

List of Abbreviations X

Introduction XI

Chapter 1: Foundations of Artificial Intelligence and 1

Deep Learning 1

1.1 Introduction 2
1.2 Artificial Intelligence 2
1.2.1 Definition 2
1.2.2 Artificial Intelligence, Machine Learning and Deep Learning 2
1.3 Machine Learning Paradigms 4
1.3.1 Supervised Learning 4
1.3.2 Unsupervised Learning 4
1.3.3 Reinforcement Learning 5
1.3.4 Comparing the Three Paradigms 5
1.4 Deep Learning 6
1.4.1 Machine Learning vs Deep Learning 6
1.4.2 The Artificial Neuron 7
1.4.3 Neural Network Architecture 8
1.4.4 Convolutional Neural Networks (CNN) 10
1.4.5 Key Parameters and Hyperparameters in Neural Network Training 13
1.4.6 Regularization Techniques 14
1.4.7 Optimization Algorithms 15
1.5 Conclusion 16
Chapter 2: Object Detection and the YOLO Architecture 18

2.1 Introduction 19
2.2 From Image Classification to Object Detection 19
2.3 One-Stage versus Two-Stage Detectors 21
2.3.1 Two-Stage Detectors 21
2.3.2 One-Stage Detectors 21
2.4 YOLO 22
2.4.1 Evolution Toward YOLOv8, YOLOv11, and YOLOv26 22
2.5 Loss Functions and Non-Maximum Suppression 26
2.6 Model Scaling 27
2.7 Evaluation Metrics for Object Detection 27
2.7.1 The Confusion Matrix in Object Detection 28

2.8 Training Environment and Explored Hyperparameters....29
2.9 Conclusion....30
Chapter 3: Model Implementations and Experimental Results....31

3.1 Introduction....32
3.2 Tools and Development Environment....32
3.3 Datasets....33
3.3.1 The BUSI Dataset....33
3.3.2 The Figshare Brain Tumor Dataset....33
3.4 Data Preprocessing Pipeline....35
3.5 Annotation Preparation and Verification....36
3.6 Experimental Results....37
3.6.1 Convolutional Neural Network Implementation....38
3.6.2 Transfer Learning Implementation....42
3.6.3 YOLO Object Detection Results and Comparison....44
3.7 Conclusion....57
Conclusion....59

References....60

Abstract....61

Résumé....61

الملخص....61

List of Figures

Figure 1 - Relationship between Artificial Intelligence, Machine Learning, and Deep Learning. 3
Figure 2 – Comparison of the three Machine Learning paradigms. 6
Figure 3 – Difference between ML and DL. 7
Figure 4 – Structure of an artificial neuron. 8
Figure 5 – A fully connected (dense) neural network. 9
Figure 6 – Example of Convolution Operation. 11
Figure 7 – Rectified Linear Unit (Relu) activation function. 11
Figure 8 – Graph of Sigmoid activation. 12
Figure 9 – MaxPooling principle. 12
Figure 10 – General architecture of a Convolutional Neural Network. 13
Figure 11 – Data Augmentation. 15
Figure 12 – Illustration of gradient descent minimizing the cost function. 16
Figure 13 – Image Classification vs Object Detection. 20
Figure 14 – Intersection Over Union. 20
Figure 15 – Types of Object detectors. 22
Figure 16 – General architecture of YOLOv8. 24
Figure 17 – General architecture of YOLO11. 25
Figure 18 – General architecture of YOLO26. 26
Figure 19 – Structure of a confusion matrix for a two-class detection problem. 29
Figure 20 – Sample BUSI ultrasound images with their corresponding lesion masks. 33
Figure 21 – Samples from Figshare brain MRI images with their corresponding tumor masks. 34
Figure 22 – The BUSI dataset imported into Roboflow, with algorithm-generated bounding boxes 36
Figure 23 – Mask-versus-box overlay comparison. 37
Figure 24 – CNN Architecture. 39
Figure 25 – Localization results for two BUSI test images at 20, 40, and 60 training epochs. 41
Figure 26 – VGG16 Localization Results on Two BUSI Test Images Across Different Training Epochs. 43
Figure 27 – Evolution of YOLOv8n Training Metrics on the BUSI Dataset. 45
Figure 28 – Comparison of Ground-Truth Annotations and YOLOv8n Predictions on BUSI Test Images. 46

Figure 29 – Evolution of YOLOv11n Training Metrics on the BUSI Dataset. 47
Figure 30 – Comparison of Ground-Truth Annotations and YOLOv11n Predictions on BUSI Test Images. 48
Figure 31 – Evolution of YOLOv26n Training Metrics on the BUSI Dataset. 49
Figure 32 – Comparison of Ground-Truth Annotations and YOLOv26n Predictions on BUSI Test Images. 50
Figure 33 – Evolution of YOLOv8n Training Metrics on the Figshare Dataset..... 52
Figure 34 – Ground-Truth and YOLOv8n Predictions for Representative Meningioma, Glioma, ... 52
Figure 35 – Evolution of YOLOv11n Training Metrics on the Figshare Dataset..... 53
Figure 36 – Ground-Truth and YOLO11n Predictions for Representative Glioma, Pituitary Tumor, ... 54
Figure 37 – Evolution of YOLOv26n Training Metrics on the Figshare Dataset..... 55
Figure 38 – Ground-Truth and YOLO26n Predictions for Representative Glioma, Pituitary Tumor, ... 56

List of Tables

Table 1 – Comparison between the BUSI and Figshare datasets. 35
Table 2 – Main training configurations used in the experiments. 38
Table 3 – IOU results for the Busi dataset, 4 samples. 40
Table 4 – IOU results for Busi dataset using VGG16. 42
Table 5 – Comparison of Model Complexity in Terms of Number of Parameters. 44
Table 6 – Performance comparison of YOLO models on the BUSI test set. 50
Table 7 – Overall performance comparison of YOLO models on the Figshare test set. 56

List of Abbreviations

Abbreviation	Meaning
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
YOLO	You Only Look Once
IoU	Intersection over Union
NMS	Non-Maximum Suppression
mAP	mean Average Precision
mAP50	mean Average Precision at an IoU threshold of 0.5
mAP50-95	mean Average Precision averaged over IoU thresholds from 0.5 to 0.95
TP / FP	True Positive / False Positive
FN / TN	False Negative / True Negative
DFL	Distribution Focal Loss
CIoU	Complete Intersection over Union
BCE	Binary Cross-Entropy
SGD	Stochastic Gradient Descent
Adam / AdamW	Adaptive Moment Estimation / Adam with decoupled Weight decay
ReLU	Rectified Linear Unit
PAN-FPN	Path Aggregation Network – Feature Pyramid Network
SPPF	Spatial Pyramid Pooling – Fast
C2PSA	Cross-Stage Partial block with Parallel Spatial Attention
DWConv	Depth-Wise Separable Convolution
GPU	Graphics Processing Unit
GFLOPs	Giga Floating-Point Operations per Second
BUSI	Breast Ultrasound Images (dataset)
DICOM	Digital Imaging and Communications in Medicine
ROI	Region of Interest
BI-RADS	Breast Imaging Reporting and Data System
MRI	Magnetic Resonance Imaging

Introduction

Artificial Intelligence (AI) has become an important field in modern technology, offering advanced methods for data analysis, automation, and decision support. The emergence of deep learning has substantially enhanced machine capabilities to learn complex patterns from extensive datasets, resulting in significant progress in computer vision and medical image analysis.

Within computer vision, object detection is a fundamental task that aims to identify objects in an image and determine their location. Deep learning-based detection models autonomously extract visual features, classify objects, and estimate spatial positions. In medical imaging, these models facilitate the detection and localization of abnormal regions, including lesions and tumors.

Among the available object detection approaches, YOLO (You Only Look Once) has gained significant attention due to its ability to perform detection and localization within a single framework. Its lightweight variants are particularly interesting for embedded systems, where computational resources, memory, and execution time are important constraints.

Accordingly,this thesis evaluates and compares lightweight object detection models for medical image analysis.Two medical imaging datasets, BUSI and Figshare, are used to assess the detection and localization of breast lesions and brain tumors. Before evaluating the YOLO models, conventional CNN-based approaches, including a custom CNN and VGG16, are first considered for lesion localization on the BUSI dataset. These preliminary experimentsprovide a basis for comparison and help highlight the trade-off between localization performance and model complexity. The study then evaluates three lightweight YOLO architectures, YOLOv8n, YOLO11n, and YOLO26n, across the selected datasets.

To achieve these objectives, the thesis is organized into three chapters.

- **Chapter 1** presents the main concepts of Artificial Intelligence, Machine Learning, Deep Learning, and Convolutional Neural Networks.
- **Chapter 2** introduces object detection principles and focuses on the YOLO architecture and its evolution.
- **Chapter 3** presents the dataset preparation, model implementation, experimental evaluation, and comparison of the obtained results.

Chapter 1: Foundations of Artificial Intelligence and Deep Learning

Abstract

This chapter presents the fundamental concepts related to Artificial Intelligence, Machine Learning, and Deep Learning. It introduces the main learning paradigms and provides the theoretical background required to understand Convolutional Neural Networks and the object detection methods discussed in the following chapters.

1.1 Introduction

1.2 Artificial Intelligence

1.3 Machine Learning

1.4 Deep Learning

1.5 Conclusion

1.1 Introduction

Artificial Intelligence (AI) has become an important technology in many fields, including computer vision and medical image analysis. The development of Deep Learning has further improved computer systems' ability to analyze complex medical images and identify relevant patterns. These techniques can be applied to different imaging modalities, such as ultrasound and magnetic resonance imaging (MRI), to support the analysis and localization of abnormal regions.

Before introducing the YOLO-based object detection approach used in this study, it is important to understand the main concepts behind Artificial Intelligence and its related techniques. This chapter therefore follows a progressive approach, starting with Artificial Intelligence and Machine Learning, then introducing their main learning paradigms and Deep Learning. The chapter also presents the fundamental concepts of neural networks and Convolutional Neural Networks, which provide the theoretical basis for the object detection methods discussed in Chapter Two.

1.2 Artificial Intelligence

1.2.1 Definition

Artificial Intelligence (AI) refers to the field of computer science concerned with developing systems capable of performing tasks that normally require human intelligence, such as learning, reasoning, perception, and decision-making. AI includes a wide range of methods and approaches for processing information and solving complex problems.

One of the main approaches within AI is Machine Learning, which allows systems to learn patterns from data instead of relying entirely on predefined rules. This data-driven approach is particularly important in medical image analysis, where large collections of images and their annotations can be used to train models to recognize and localize specific regions of interest. In this thesis, AI and Machine Learning concepts provide the theoretical foundation for the Deep Learning and object detection techniques applied to the BUSI and Figshare datasets.

1.2.2 Artificial Intelligence, Machine Learning and Deep Learning

Artificial Intelligence (AI), Machine Learning (ML), and Deep Learning (DL) are closely related concepts, but they do not refer to the same thing. Artificial Intelligence is the

broader field concerned with developing systems capable of performing tasks that normally require human intelligence, such as learning, reasoning, perception, and decision-making. Machine Learning is a subfield of AI that allows systems to learn patterns from data and improve their performance without requiring explicit rules for every task. Deep Learning is a further subfield of Machine Learning that uses artificial neural networks with multiple layers to automatically learn complex features and representations from large amounts of data.

These concepts can therefore be viewed as different levels within the same hierarchy, where Machine Learning is a branch of Artificial Intelligence and Deep Learning is a branch of Machine Learning. Traditional Machine Learning methods such as decision trees and linear regression, can learn from data without using deep neural networks. Deep Learning, however, is particularly effective for complex data such as images, because the network can progressively learn features from the input data, from simple patterns to more complex structures.

In medical image analysis, Deep Learning has become an important approach for tasks such as image classification, segmentation, and object detection. The YOLO approach used in this thesis is a Deep Learning–based method that uses neural networks to detect and localize objects within images. The relationship between Artificial Intelligence, Machine Learning, and Deep Learning is illustrated in Figure 1.

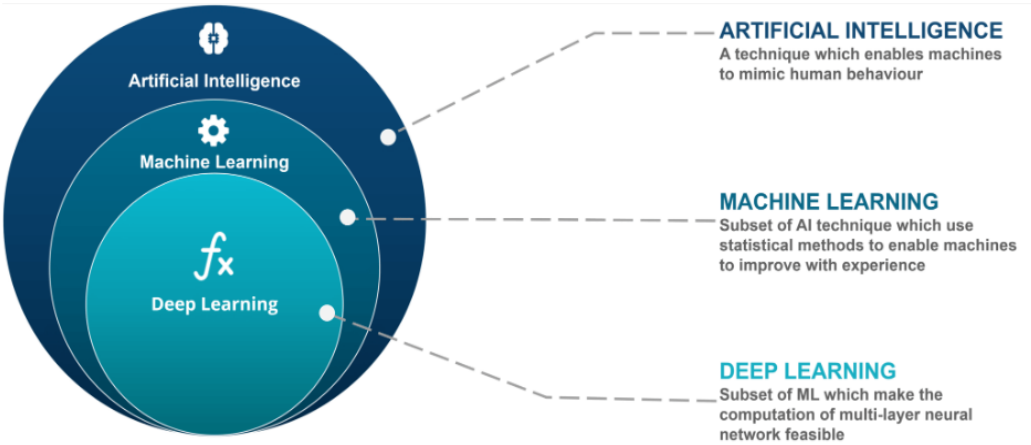


Figure 1 - Relationship between Artificial Intelligence, Machine Learning, and Deep Learning.

1.3 Machine Learning Paradigms

Machine Learning is generally divided into three main learning paradigms, supervised learning, unsupervised learning, and reinforcement learning. The main difference between these paradigms is the type of data available during training and the way the model receives feedback. Understanding these three paradigms is important before introducing Deep Learning. Deep Learning is not considered a separate learning paradigm; rather, it is a group of neural network architectures that can be trained using different learning paradigms depending on the application.

1.3.1 Supervised Learning

In supervised learning, the model is trained using a labeled dataset. This means that each input is associated with a correct output, usually referred to as the ground truth or target. During training, the model makes a prediction, compares it with the corresponding true label, and adjusts its parameters to reduce the difference between the prediction and the expected result. After training, the model should be able to apply what it has learned to new and unseen data.

Supervised learning can generally be divided into two main types of problems:

- **Classification**: where the output is a discrete class. For example, a model can be used to determine whether a region in an ultrasound image is benign or malignant.

- **Regression**: where the output is a continuous value. For example, regression can be used to predict the coordinates of a bounding box around a tumor.

Object detection, including YOLO, combines both classification and regression. Classification is used to determine what the detected object is, while regression is used to determine its location through the coordinates of a bounding box. For this reason, the datasets used in this work need to be annotated before training. Each relevant image must contain information about the object class and its bounding-box coordinates so that the model can learn both what the object is and where it is located.

1.3.2 Unsupervised Learning

Unsupervised learning is used when the available data does not have predefined labels or target values. Instead of learning a direct relationship between inputs and known outputs,

the model tries to identify patterns and structures within the data. Common examples include clustering, where similar samples are grouped, and dimensionality reduction, where the number of variables is reduced while trying to preserve the important information in the data. Another example is the use of autoencoders, which learn to reconstruct an input from a compressed internal representation.

Unsupervised learning is not the main learning paradigm used to train YOLO, but it can still be useful in some stages of medical image analysis. For example, it can help explore an unlabeled dataset, identify unusual samples or duplicate images, or learn useful representations before training a model with labeled data. Generative Adversarial Networks (GANs) can also be used to generate synthetic images for data augmentation, particularly when the available medical dataset is limited.

1.3.3 Reinforcement Learning

Reinforcement Learning (RL) is different from supervised and unsupervised learning. In this paradigm, an agent interacts with an environment by taking actions and receiving rewards or penalties based on the results of these actions. The objective is to learn a policy, which defines how the agent should choose its actions in order to maximize the total reward over time.

Reinforcement learning is based mainly on learning through interaction and trial and error. It has been used in applications such as game-playing systems, including chess and Go, as well as in some robotic control systems. However, it is generally not used for object detection in static medical images. In this type of application, the model receives an image and directly predicts the object classes and their bounding boxes rather than making a sequence of actions over time. Reinforcement learning is therefore presented here mainly to complete the overview of the three main Machine Learning paradigms and to show why the object detection task considered in this work belongs to supervised learning.

1.3.4 Comparing the Three Paradigms

The three Machine Learning paradigms differ mainly in the type of feedback provided during training. In supervised learning, the model learns from labeled data where the correct output is known for each training example. In unsupervised learning, no predefined labels are provided, so the model identifies patterns or structures within the data on its own. In rein-

forcement learning, the model learns through interaction with its environment by receiving rewards or penalties according to its actions.

The task addressed in this work belongs to supervised learning because the datasets contain annotated images with information about the class and location of the target regions. The three main Machine Learning paradigms are compared in Figure 2.

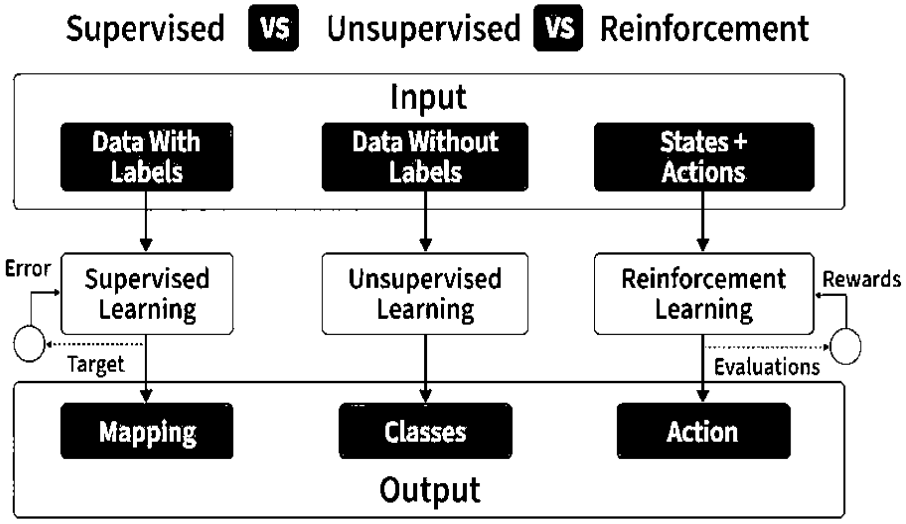


Figure 2 – Comparison of the three Machine Learning paradigms.

1.4 Deep Learning

1.4.1 Machine Learning vs Deep Learning

Classical Machine Learning methods usually rely on features that are selected and extracted from the input data before training. For image analysis, these features may include edges, textures, intensity, contrast, or shape descriptors. This process, known as feature engineering, requires domain knowledge and may require different features depending on the dataset or application.

Deep Learning reduces the need for manual feature extraction by allowing the model to learn relevant features directly from the input data during training **[1].** This is achieved using artificial neural networks composed of multiple layers, where each layer learns increasingly complex representations. For image data, early layers can learn simple patterns such as

edges and gradients, while deeper layers combine these patterns to represent more complex shapes and structures.

This hierarchical feature learning is one of the main characteristics of Deep Learning and makes it well suited to image-based tasks such as classification, detection, and localization. The main differences between Machine Learning and Deep Learning are highlighted in Figure 3.

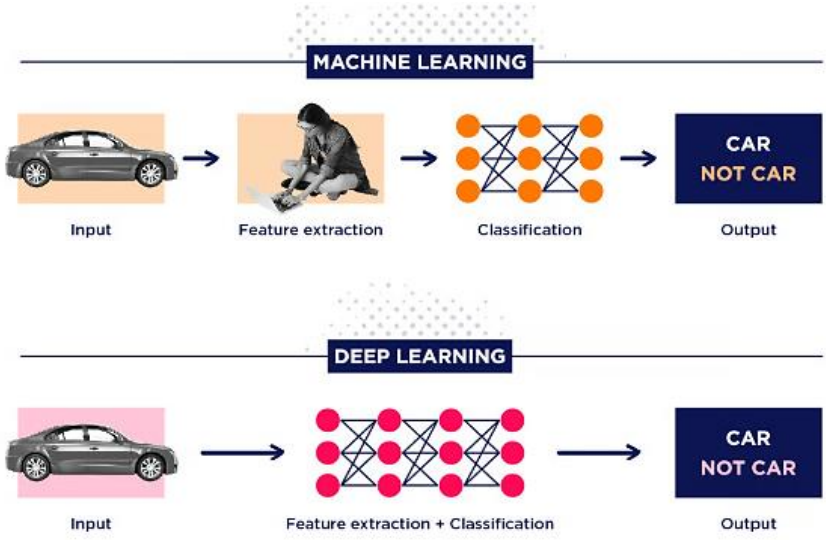


Figure 3 –Difference between ML and DL.

1.4.2 The Artificial Neuron

The artificial neuron is one of the basic elements of a Deep Learning model. It is a simplified mathematical model inspired by the way biological neurons process information. An artificial neuron receives one or more numerical inputs. Each input is multiplied by a weight, which determines its contribution to the output. The weighted inputs are then added together with another parameter called the bias. The resulting value is passed through an activation function to produce the output of the neuron.

The weights and biases are the parameters that the neural network learns during training. The learning process consists of adjusting these parameters so that the predictions produced by the network become closer to the ground-truth values. A network can contain a large number of neurons, each with its own weights and biases, allowing it to learn increasingly complex patterns from the training data.

A single neuron can only represent relatively simple relationships between its inputs and output. The main strength of Deep Learning comes from combining many neurons into layers and connecting several layers together. In this way, the network can learn more complex representations of the input data. For image-based applications, Convolutional Neural Networks (CNNs) are particularly important because they are designed to process spatial information in images. Their structure and their role in YOLO are discussed in the following sections, along with the main mechanisms used to train neural networks, such as loss functions, gradient descent, and optimizers. The basic structure of an artificial neuron is presented in Figure 4.

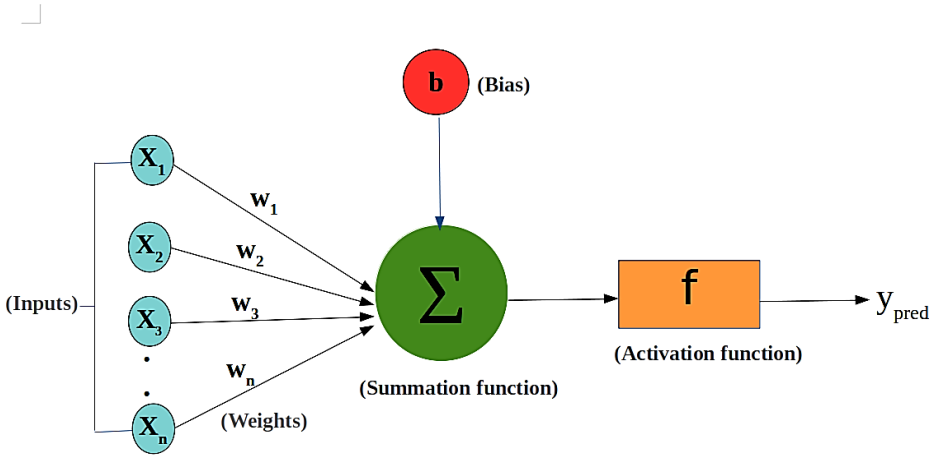


Figure 4 –Structure of an artificial neuron.

1.4.3 Neural Network Architecture

A neural network is formed by connecting artificial neurons and organizing them into different layers. The first layer, called the input layer, receives the input data. In this work, the input consists of the pixel values of MRI or ultrasound images. Between the input and output layers, there can be one or more hidden layers where most of the computations are performed.

The final layer, known as the output layer, produces the prediction of the network. For example, it can provide the probability that a detected region belongs to a particular class.

One common type of neural network is the fully connected, or dense, network. In this architecture, each neuron in one layer is connected to all neurons in the previous layer. Although this structure can be useful for relatively small inputs, it becomes inefficient when working with images. An image can contain thousands of pixels, and connecting all of these pixels to the neurons in the next layer would require a very large number of weights. In addition, this approach does not take advantage of the spatial relationship between neighboring pixels.

This limitation led to the development of specialized architectures for image processing, particularly Convolutional Neural Networks (CNNs). CNNs use convolutional layers to extract and learn spatial features from images while using fewer parameters than a fully connected network. Figure 5 presents the structure of a fully connected neural network.

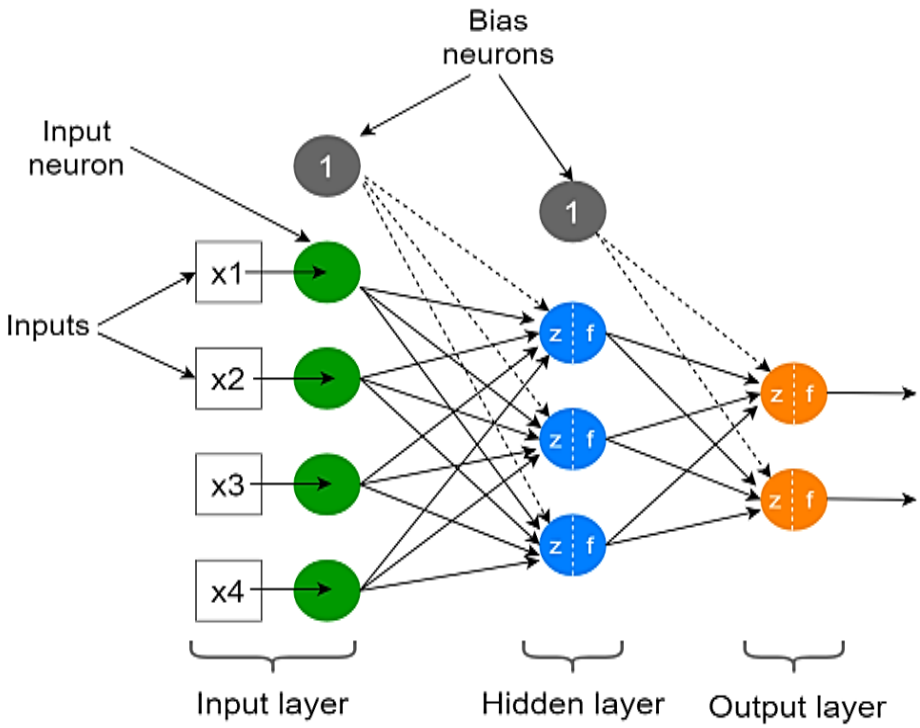


Figure 5 –A fully connected (dense) neural network.

1.4.4 Convolutional Neural Networks (CNN)

A Convolutional Neural Network (CNN, or ConvNet) is a neural network designed specifically to process data with a grid-like structure, such as an image **[2].** Instead of connecting every input pixel to every neuron, a CNN relies on small, learnable filters that slide across the image and are reused at every position, a property called weight sharing. This drastically reduces the number of parameters compared to a fully connected network of the same depth, and it also makes the network naturally sensitive to local patterns, such as edges or textures, regardless of where exactly they appear in the image. A typical CNN is built by stacking four kinds of layers:

- Convolutional layer
- Activation function layer
- Pooling layer
- Fully connected layer

The convolutional layer is the core building block of a CNN. It applies a set of filters (also called kernels) to the input through the convolution operation, each filter slides across the image and produces a feature map that highlights where a particular pattern is present. Two hyperparameters mainly control this operation: the stride, which is the step size the filter moves by each position, and zero padding, which adds a border of zeros around the input to control the spatial size of the output. Stacking several convolutional layers lets the network build up from very simple, local patterns in the first layers (edges, contrast changes) to increasingly abstract and global patterns in the deeper layers, such as the irregular shape typically associated with a malignant mass.

After each convolution, a non-linear activation function is applied element-wise to the resulting feature map. An example of the convolution operation is shown in Figure 6.

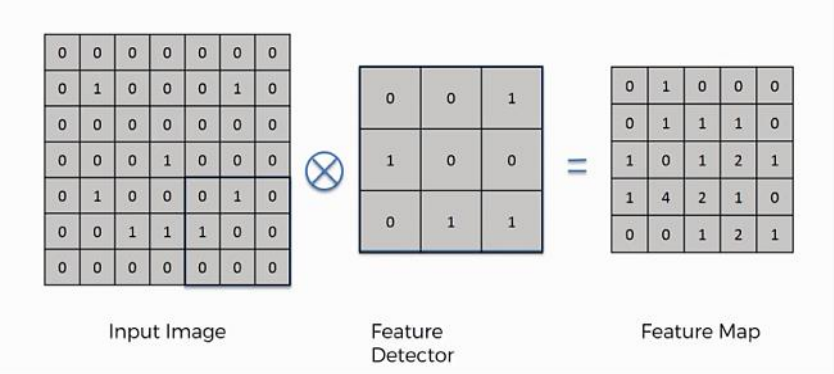


Figure 6 –Example of Convolution Operation.

The Rectified Linear Unit (ReLU) is by far the most common choice in modern CNNs **[3],** defined simply as ReLU(x) equals max(x, 0); it keeps positive values unchanged and sets every negative value to zero. This simple operation is what allows the network to model non-linear relationships without it, stacking multiple layers would mathematically collapse into a single linear transformation, no matter how many layers were added. Figure 7 illustrates the Rectified Linear Unit (ReLU) activation function.

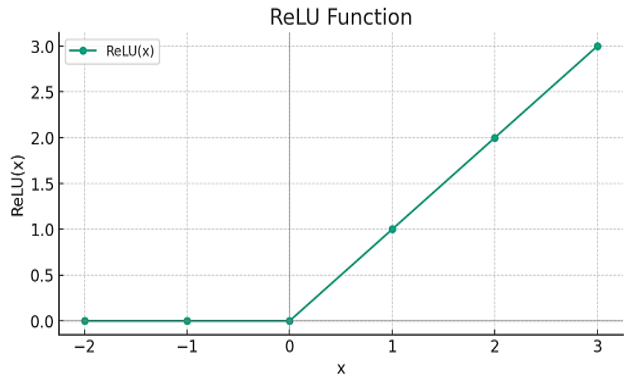


Figure 7 –Rectified Linear Unit (Relu) activation function.

The sigmoid function is another classical activation function, more commonly used at the output of a network for binary classification tasks, such as predicting whether a detected region is malignant, since it squashes its input into a value between 0 and 1 that can be read as a probability. The sigmoid activation function is illustrated in Figure 8.

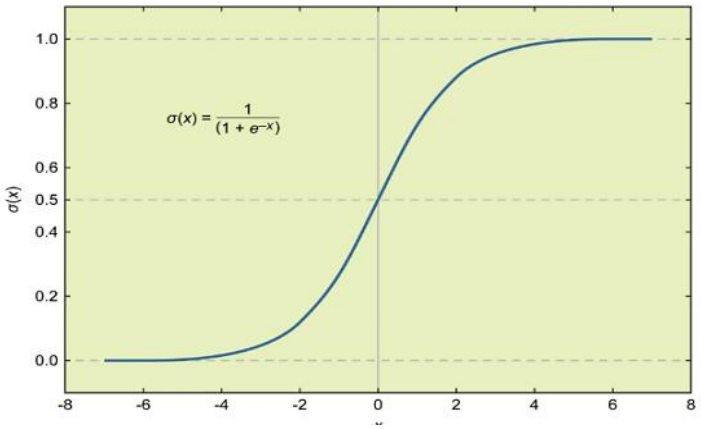


Figure 8 – Graph of Sigmoid activation.

Figure 9 illustrates the Max Pooling operation.

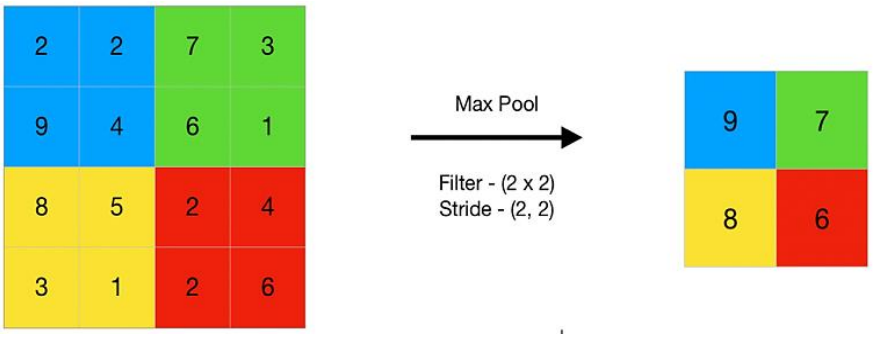


Figure 9– Max Pooling principle.

Finally, once the feature maps have passed through enough convolutional and pooling layers, they are flattened into a single vector and fed into one or more fully connected layers, exactly as in a classical dense network. These final layers combine all the features extracted earlier to produce the network's actual output: class scores in a classification setting, or, in the case of an object detector such as YOLO, both the class scores and the coordinates of the predicted bounding box. The general architecture of a Convolutional Neural Network is presented in Figure 10.

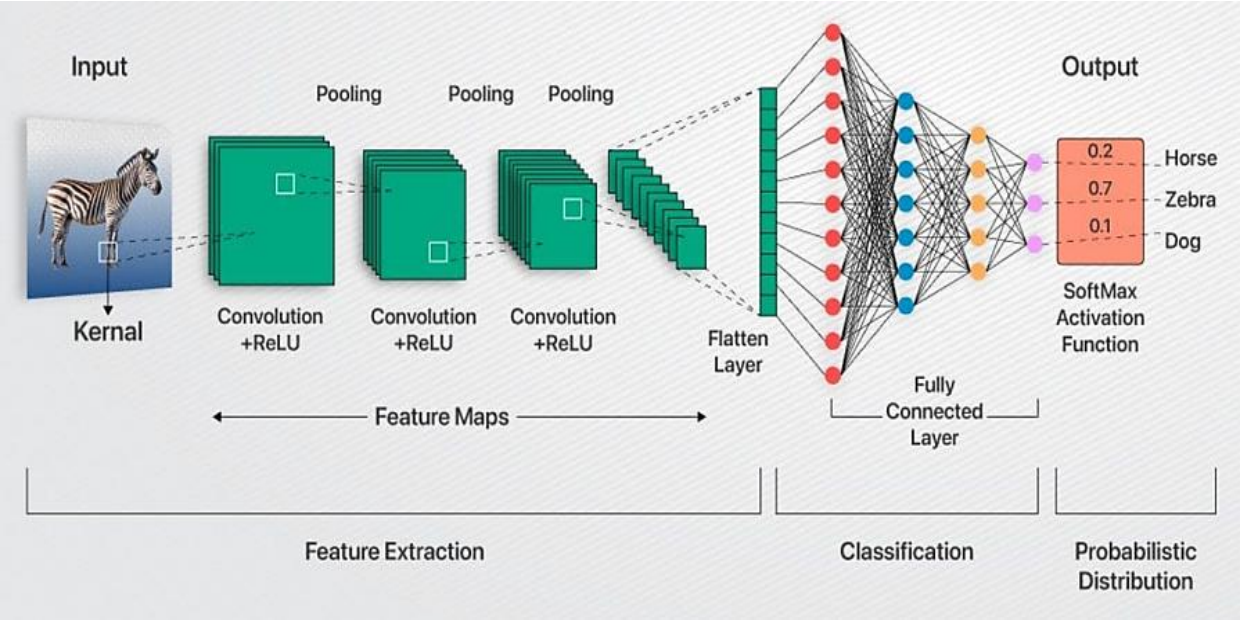


Figure 10–General architecture of a Convolutional Neural Network.

1.4.5 Key Parameters and Hyperparameters in Neural Network Training

Training a neural network can be viewed as a search process in which the network starts with randomly initialized weights and biases and gradually adjusts them to make its predictions closer to the ground truth. This process depends on different quantities. Some are learned automatically during training and are known as parameters, while others are defined before training by the model designer and are known as hyperparameters. To understand how this training process works in practice, it is useful to introduce the main quantities that influence learning, including the loss function, learning rate, batch size, and number of epochs.

- The cost function, also called the loss function, measures the difference between the model predictions and the ground-truth values. During training, the objective is to minimize this loss. A high loss value indicates that the predictions are far from the expected results, while a low loss value indicates that they are closer to the ground truth.

- As introduced earlier in this chapter, the weights and biases of the network are the main parameters adjusted during training to reduce the loss function. The type of loss used depends on the task. For example, classification and bounding-box regression generally use different loss functions because they represent different learning objectives.

- The learning rate is a hyperparameter that controls how large a step is taken, at each training iteration, in the direction that reduces the cost function. If it is set too high, training can become unstable and overshoot a good set of weights; if it is set too low, training becomes very slow and may get stuck before reaching a good solution. In practice, the learning rate is usually one of the first hyperparameters that needs to be tuned for a given dataset.

- Instead of updating the weights after looking at the entire dataset at once, or after every single image, training is usually carried out on small groups of images called batches. The batch size is the number of images processed together before the weights are updated once. Small batches make training noisier but require less memory, while large batches produce more stable updates at the cost of more memory and, often, more computation time.

- An epoch corresponds to one complete pass of the training algorithm over the entire training dataset. A model is typically trained over many epochs, and monitoring how the loss evolves from one epoch to the next both on the training data and on a separate validation set is the main way of checking whether the model is genuinely learning or, on the contrary, starting to overfit.

1.4.6 Regularization Techniques

As a network becomes deeper and its number of parameters grows, it becomes increasingly capable of simply memorizing the training data instead of learning patterns that generalize to new, unseen images, a problem known as overfitting. Regularization groups together the techniques used to limit this risk.

- Dropout randomly deactivates a fraction of the neurons of a layer at each training iteration, which forces the remaining neurons to learn features that do not depend on any single specific neuron always being present **[4].** All neurons are used again once training is finished.

- Data augmentation increases the diversity of a training dataset by applying controlled transformations to existing images, such as rotations, flips, cropping, or changes in brightness and contrast, while preserving their original labels. This can be particularly useful for smaller medical imaging datasets such as BUSI, where the number of available images is more limited. However, augmentation in medical imaging must be carefully selected because inappropriate transformations may alter important anatomical or pathological features and negatively

affect model performance. Therefore, augmentation techniques should be chosen according to the characteristics of the images and the medical task being considered.

Figure 11 shows examples of common data augmentation techniques.

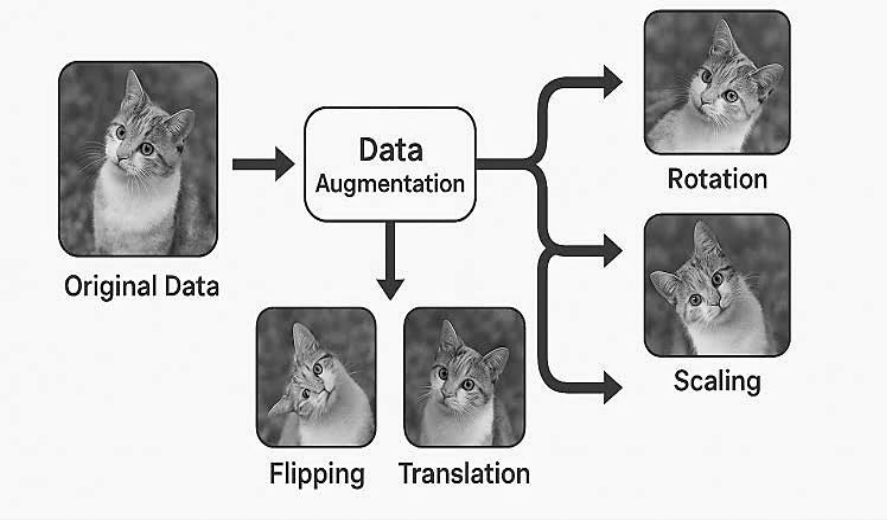


Figure 11– Data Augmentation.

1.4.7 Optimization Algorithms

The optimizer is the algorithm responsible for actually updating the weights after each training batch, based on the gradient of the cost function. The most fundamental optimization algorithm is gradient descent, which updates each **trainable parameter**, including weights and biases, by taking a step in the direction opposite to the gradient of the cost function with respect to that parameter:

$$\theta_k^{(t+1)} = \theta_k^{(t)} - \gamma \times \left(\frac{\partial L}{\partial \theta_k}\right) \quad \text{.....(1)}$$

Where t represents the current training step and L is the loss function.$\theta_k$represents a trainable parameter of the network,γis the learning rate, and$\frac{\partial L}{\partial \theta_k}$represents the gradient of the loss with respect to that parameter.

In practice, neural networks are commonly trained using more advanced gradient-based optimizers such as Adam **[5]** rather than basic gradient descent. Adam keeps track of the moving average of past gradients and their squared values, allowing it to adapt the learn-

ing rate of each parameter during training. This adaptive behavior can help the model converge faster and make the training process less sensitive to the initial learning rate.

Adam was therefore used as the optimizer for the model training conducted in this work.

The gradient descent process for minimizing the cost function is illustrated in Figure 12.

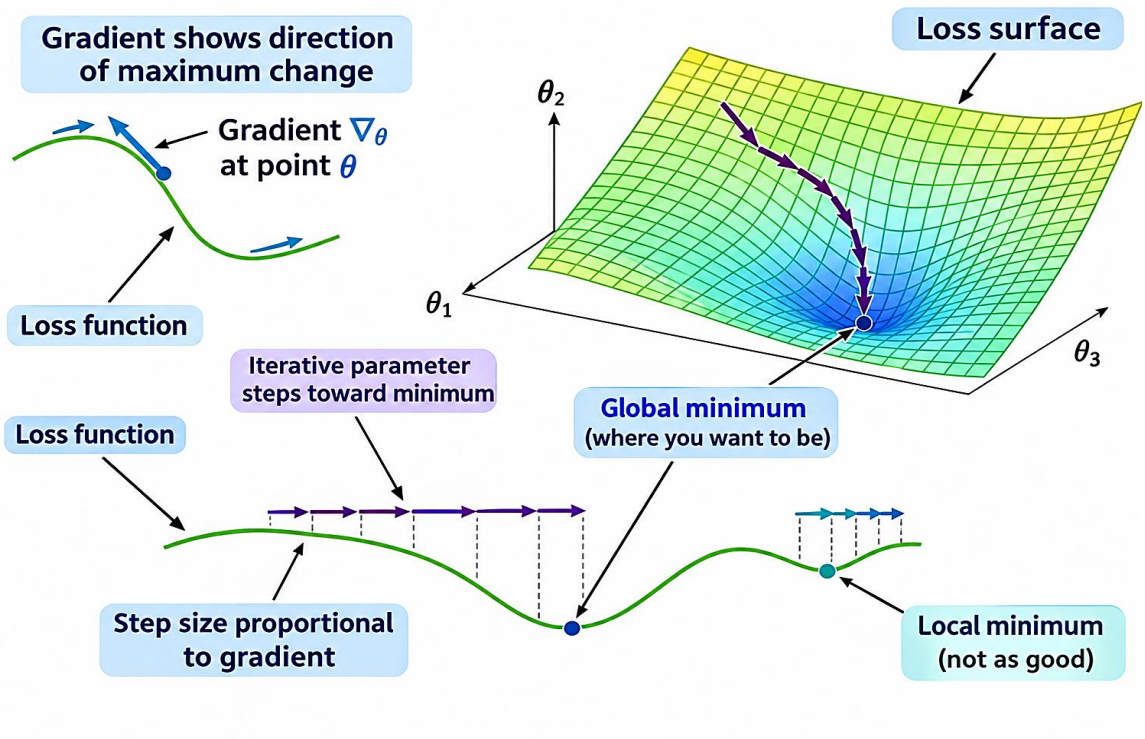


Figure 12 – Illustration of gradient descent minimizing the cost function.

1.5 Conclusion

This chapter introduced the main concepts needed to understand the technical part of this thesis. It first presented Artificial Intelligence, Machine Learning, and Deep Learning, explaining their relationship and the differences between them. The three main Machine Learning paradigms were then presented. Since the objective of this work is to detect and localize suspicious regions in breast images, the problem was identified as a supervised learning task that combines classification and regression.

Then, we introduced the basic concepts of Deep Learning, starting with the artificial neuron and the structure of neural networks. Convolutional Neural Networks (CNNs) were also introduced as an important architecture for image processing. Finally, the main parame-

ters and training concepts, including the loss function, learning rate, batch size, epochs, regularization, and optimization algorithms, were discussed.

These concepts provide the necessary background for the next chapter, which focuses on object detection and the YOLO architecture. The next chapter will introduce the main principles of object detection and follow the evolution of the YOLO family, leading to the models considered in this work.

Chapter 2: Object Detection and the YOLO Architecture

Abstract

This chapter presents the main concepts related to object detection. It introduces the difference between image classification and object detection, followed by the main types of object detection methods and the YOLO approach. The chapter also presents the YOLO architectures considered in this work and the main evaluation metrics used to assess their performance. These concepts provide the theoretical background for the implementation and experiments presented in the following chapter.

2.1 Introduction

2.2 From Image Classification to Object Detection

2.3 One-Stage versus Two-Stage Detectors

2.4 YOLO

2.5 Loss Functions and Non-Maximum Suppression

2.6 Model Scaling

2.7 Evaluation Metrics for Object Detection

2.8 Training Environment and Explored Hyperparameters

2.9 Conclusion

2.1 Introduction

Object detection is an important task in computer vision that aims to identify and locate objects or regions of interest within an image. Unlike image classification, which assigns one or more categories to an image, object detection provides both the class of the target and its spatial location.This capability makes object detection useful for applications where identifying the presence of an object is not sufficient, and its position must also be determined. In medical image analysis, for example, detection models can be used to identify and localize abnormal regions such as lesions and tumors.

This chapter presents the main concepts of object detection and introduces the different approaches used to perform this task. It then focuses on the You Only Look Once (YOLO) family, which is the main detection approach used in this work. The chapter also presents the YOLO architectures considered in the experiments and the evaluation metrics used to assess their performance.

2.2 From Image Classification to Object Detection

Image classification and object detection are two related but different computer vision tasks. Image classification assigns a class label to an entire image, while object detection identifies the objects present and determines their locations using bounding boxes. Object localization is similar to detection but is generally limited to identifying the location of a single object.

Object detection therefore combines classification and localization, allowing multiple objects to be identified and located within the same image. This distinction is illustrated in Figure 13.



Figure 13– Image Classification vs Object Detection.

Comparing two candidate boxes, whether during training or at evaluation time, requires a way to measure how well a predicted box overlaps the ground-truth box drawn by the annotator. This is the role of the Intersection over Union (IoU), defined as the area of the overlap between the two boxes divided by the area of their union:

$$\mathbf{IoU} = \frac{\mathbf{Area\ (prediction \cap ground\ truth)}}{\mathbf{Area\ (prediction \cup ground\ truth)}} \quad \ldots\ldots(2)$$

An IoU of one means the two boxes coincide perfectly, while an IoU of zero means they do not overlap at all. A prediction is usually only counted as correct if its IoU with a ground-truth box exceeds a chosen threshold, most commonly zero point five.

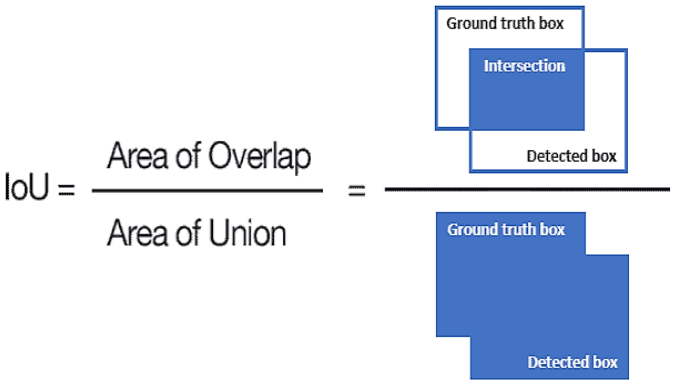


Figure 14– Intersection Over Union.

2.3 One-Stage versus Two-Stage Detectors

Historically,deep-learning-based object detectors have followed one of two design philosophies, distinguished by whether they locate and classify objects in a single pass over the image or in two successive passes.

2.3.1 Two-Stage Detectors

Two-stage detectors, such as the R-CNN family (R-CNN, Fast R-CNN **[8],** and Faster R-CNN), perform object detection in two main stages. In the first stage, potential regions containing objects are identified in the image. In Faster R-CNN, this step is performed by a Region Proposal Network (RPN) **[7],** which generates candidate regions without determining their class. In the second stage, these regions are classified, and their bounding boxes are further refined.

This approach can provide high detection and localization accuracy because the second stage focuses on a smaller number of candidate regions. However, the additional processing required by the two-stage structure makes these models more computationally demanding than one-stage detectors. This can limit their suitability for applications where fast or real-time detection is required.

2.3.2 One-Stage Detectors

One-stage detectors, such as SSD and the YOLO family, perform object detection in a single pass through the network. Instead of first generating separate region proposals, the model directly predicts the class and location of objects from the input image. The bounding-box coordinates and class information are therefore obtained within the same detection process, which makes these models faster than most two-stage approaches.

The main advantage of one-stage detectors is their speed, which makes them suitable for applications that require fast inference. Earlier one-stage detectors generally had lower detection accuracy than two-stage models, but this difference has been reduced with improvements in modern architectures. Recent versions of YOLO provide a good balance between detection accuracy and computational efficiency.Figure 15 presents the main types of object detectors

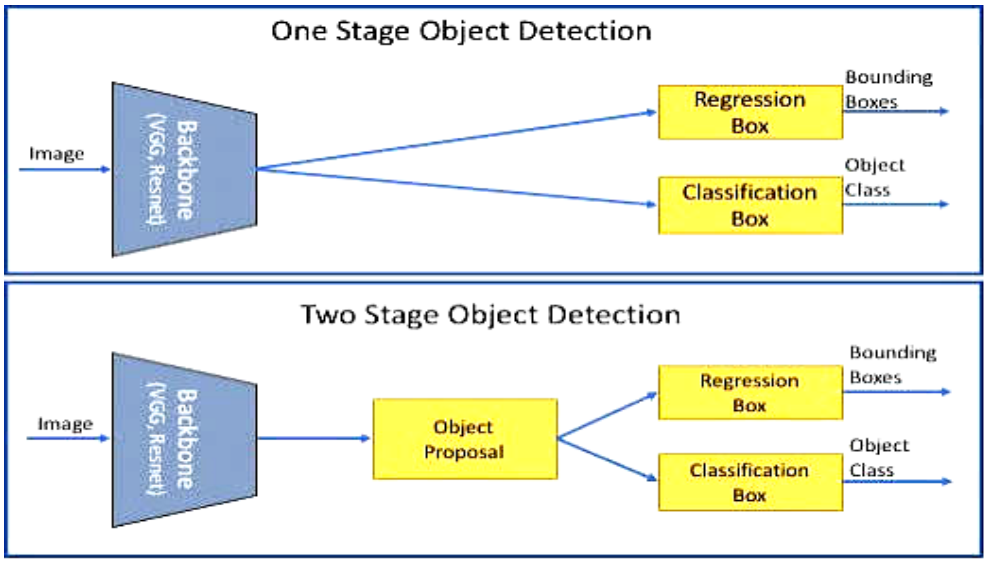


Figure 15 - Types of Object detectors.

2.4 YOLO

YOLO is an object detection algorithm based on convolutional neural networks that identifies and locates objects in an image in a single pass, rather than scanning the image multiple times or using separate steps to propose and then classify regions. This single-pass design is what gives YOLO its name and also what makes it fast enough for real-time use think video feeds, live camera systems, or time-sensitive applications like medical image analysis. The main trade-off is a slight dip in accuracy compared to slower, multi-stage detectors, particularly with small or densely packed objects, but this gap has narrowed significantly across newer versions of the architecture. It's this speed-accuracy balance that's made YOLO one of the most widely used detection frameworks in both academic research and real-world applications.

2.4.1 Evolution Toward YOLOv8, YOLOv11, and YOLOv26

The original YOLO, introduced by Redmon et al. in 2016 **[6],** treated detection as one regression problem the image gets split into a grid, and each cell predicts a set number of bounding boxes plus class probabilities, all in a single forward pass. Later versions (YOLOv2 through YOLOv5) built on this with stronger backbones, anchor boxes for handling different object shapes, and more efficient training. YOLOv8, from Ultralytics, went a step further by dropping anchor boxes entirely in favor of an anchor-free approach, along with a decoupled

head that separates classification from box regression instead of mixing them. YOLO26, the most recent of the three, continues down that anchor-free path with additional architectural improvements.

2.4.1.1YOLOv8n Architecture

YOLOv8n, like most modern one-stage detectors, is built around three main parts: a backbone, a neck, and a head **[9].** The backbone is a CSPDarknet-style network that pulls features out of the image at different scales, using C2f blocks (an updated Cross-Stage Partial design) to keep gradients flowing well during training without piling on extra parameters. The neck uses a PAN-FPN structure to combine features from different depths of the backbone, so the model can pick up on both small and large objects a tiny tumor in an ultrasound image or a larger one, using the same weights. The head is anchor-free and decoupled, meaning class prediction and box-coordinate prediction happen on separate branches instead of being fused. For training, YOLOv8n combines binary cross-entropy for classification with a box loss that blends CIoU and Distribution Focal Loss (DFL), where DFL treats box coordinates as a learned distribution rather than a single fixed value. Figure 16 illustrates the general architecture of YOLOv8n.

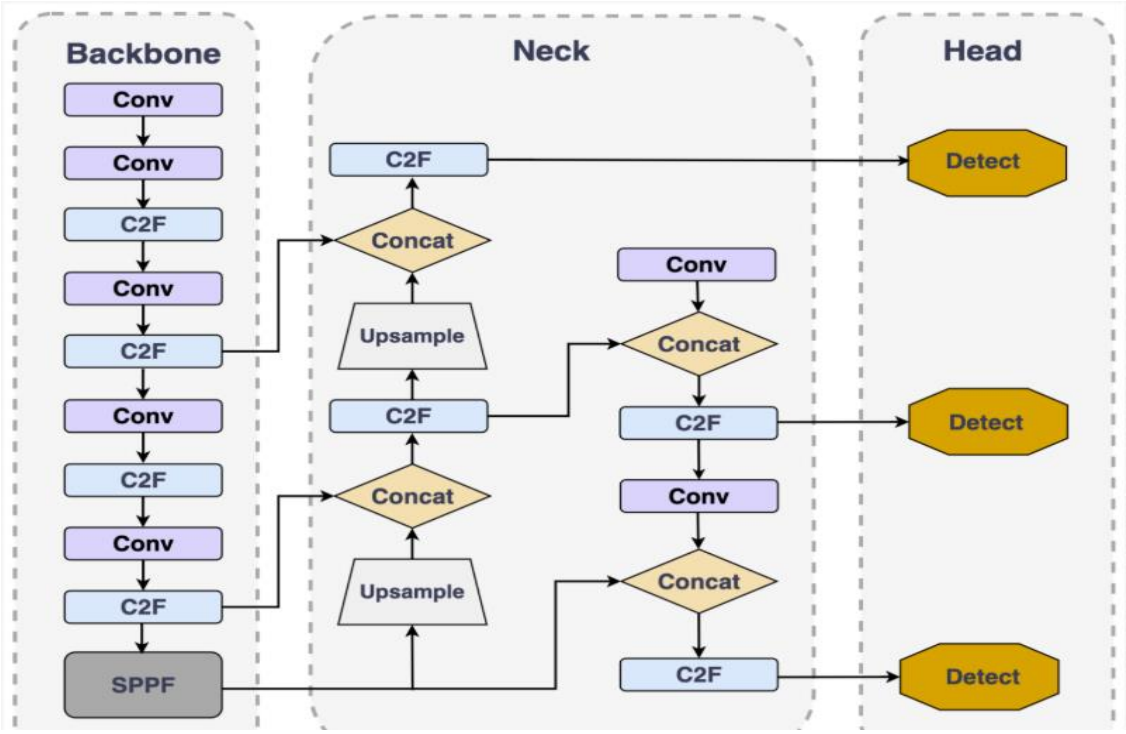


Figure 16- General architecture of YOLOv8n.

2.4.1.2YOLOv11n Architecture

YOLO11n, released by Ultralytics between YOLOv8n and YOLOv26n **[10]**, sits as the intermediate step in the evolution and is the second of the three architectures compared experimentally in this work. It keeps the same backbone-neck-head philosophy, but brings in two changes that YOLOv26n later builds on further. First, it swaps out YOLOv8n's C2f backbone blocks for C3k2 blocks, the same lightweight block where this design actually originates. Second, YOLOv11n adds a C2PSA attention block after the SPPF layer, giving the network an early form of the spatial self-attention that YOLOv26n later relies on more heavily. Unlike YOLOv26n, though, YOLOv11n still uses a conventional decoupled head, which means it still needs Non-Maximum Suppression at inference time rather than the end-to-end "one-to-one" head YOLOv26n introduces.

In terms of raw size, the nano variant of YOLOv11n is close enough to YOLOv8n and YOLO26n that it stays fair; all three are evaluated at their smallest scale, under the same training budget and dataset. Including YOLOv11n alongside YOLOv8n and YOLOv26n makes it possible to see not just the final jump from YOLOv8n to YOLOv26n, but the intermediate step in

between, which turns out to matter for the actual results on the BUSI dataset.Figure 17 presents the general architecture of YOLOv11n

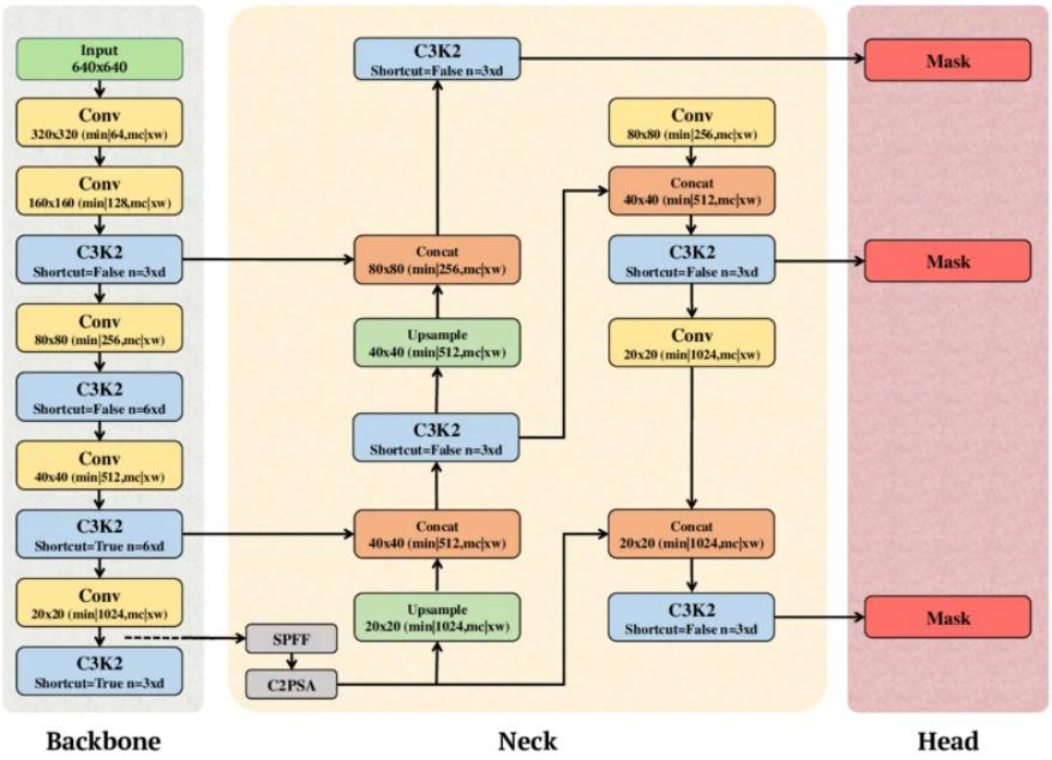


Figure 17– General architecture of YOLOv11n.

2.4.1.3YOLOv26n

YOLOv26n follows the same general backbone, neck, and detection head structure used in YOLOv8n, but introduces several changes to improve efficiency and detection performance **[11].**

First, YOLOv26n replaces the C2f blocks used in YOLOv8n with C3k2 blocks in the backbone. These blocks provide a different balance between computational efficiency and feature extraction. Second, a C2PSA block is added after the SPPF layer. This block uses spatial attention to help the network focus on important regions of the feature maps.

The detection head is also modified by adding an end-to-end one-to-one prediction branch. This branch is designed to produce a single prediction for each object, which allows YOLOv26n to avoid the use of Non-Maximum Suppression (NMS) during inference. In contrast, YOLOv8n uses NMS to remove redundant overlapping predictions.

These modifications allow YOLOv26n to maintain a lightweight architecture while increasing its depth and introducing attention-based feature processing. The nano version contains 122 layers, compared with 73 layers in YOLOv8n. The general architecture of YOLOv26n is illustrated in Figure 18.

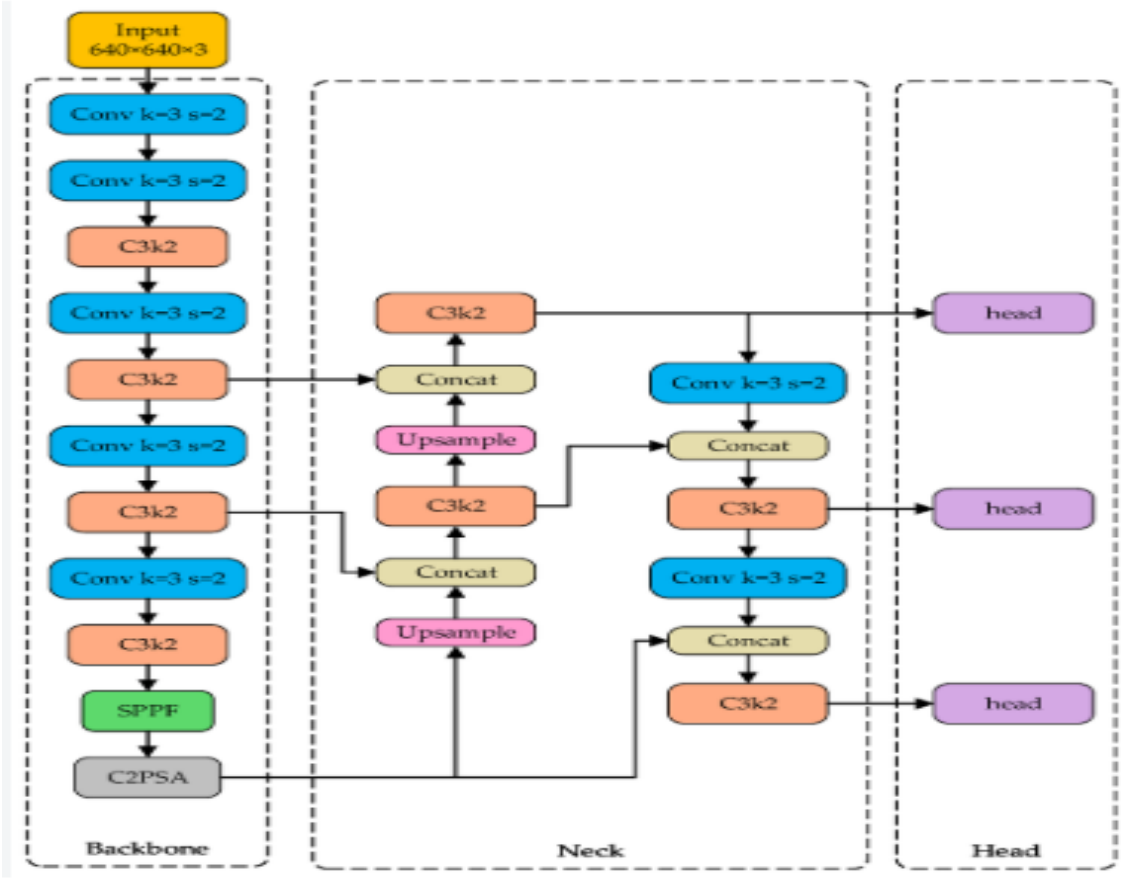


Figure 18 – General architecture of YOLOv26n.

2.5 Loss Functions and Non-Maximum Suppression

YOLO-based object detection uses a combined loss function to measure classification and localization errors during training. The classification loss is based on Binary Cross-Entropy (BCE), while the bounding-box loss combines Complete IoU (CIoU) loss and Distribution Focal Loss (DFL). CIoU considers the overlap, center distance, and aspect ratio between predicted and ground-truth boxes, whereas DFL improves the precision of box-coordinate prediction.

During inference, overlapping predictions may be produced for the same object. YOLOv8n and YOLOv11n use Non-Maximum Suppression (NMS) to remove redundant boxes by keep-

ing the most confident prediction. YOLOv26n introduces an end-to-end one-to-one detection head, allowing it to produce a single prediction per object and eliminate the need for NMS.

2.6 Model Scaling

YOLOv8n, YOLOv11n, and YOLOv26n are available in different model sizes, including nano (n), small (s), and medium (m) variants. These variants use the same general architecture but differ in depth and width, which affects the number of parameters, computational cost, and model capacity.

In this study, the nano variants were selected because the main objective is to evaluate lightweight models suitable for Embedded Systems applications, where computational resources and memory are limited. For YOLOv26n, the nano variant contains 2.41 million parameters and requires 5.2 GFLOPs, while the small variant contains 9.47 million parameters and requires 20.5 GFLOPs at the same input resolution. The small model is therefore considerably heavier while providing greater model capacity.

The nano configuration was retained as the primary choice to maintain a balance between detection performance, computational efficiency, and the size of the available datasets. Larger variants were not investigated experimentally because their higher computational requirements are less suitable for the embedded systems focus of this work.

2.7 Evaluation Metrics for Object Detection

All the metrics used here are built on top of four elementary outcomes, evaluated for a given class and a given IoU threshold (typically 0.5):

- **True Positive (TP):** a predicted box that correctly matches a ground-truth object of the right class with sufficient IoU.
- **False Positive (FP):** a predicted box for which no matching ground-truth object exists, i.e., the model detected something that is not there.
- **False Negative (FN):** a ground-truth object that the model failed to detect at all.
- **True Negative (TN):** although rarely used directly in detection, corresponds to background regions correctly left unpredicted.

From these four quantities, Precision and Recall are defined as:

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \quad \text{.....(3)}$$

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \quad \text{.....(4)}$$

- **Precision:** measures how trustworthy a positive prediction is of every box the model claims contain a tumor; what fraction actually does.
- **Recall:** measures completeness of every real tumor present in the dataset, what fraction the model actually found.

Recall is important in medical image detection because a False Negative means that a relevant region was not detected. A higher Recall indicates that the model is better at identifying the regions of interest in the images.

These two quantities are combined into the following aggregate metrics:

- **Mean Average Precision (mAP):**Summarizes Precision and Recall into a single number by measuring the area under the Precision-Recall curve, averaged over all classes.
- **mAP50:**Reports this value using a single, relatively lenient IoU threshold of 0.5, meaning a prediction only needs to overlap the ground-truth box by half to be counted as correct.
- **mAP50-95:**A stricter and more informative variant, averages the same computation over ten IoU thresholds ranging from 0.5 to 0.95, so that a model is only rewarded for boxes that are both correctly classified and precisely localized, not merely "roughly right."
- **Instances:**As reported in every results table, refers to the number of individually annotated ground-truth objects present in a given evaluation split, for example benign or malignant masses, which is distinct from the number of Images, since a single image can contain zero, one, or occasionally more than one annotated instance.

2.7.1 The Confusion Matrix in Object Detection

While Precision, Recall, mAP50, and mAP50-95 provide overall performance measures, the confusion matrix gives a more detailed view of the model's classification errors. It compares the actual and predicted classes, with diagonal cells representing correct classifications and off-diagonal cells indicating misclassifications.

For object detection, the Ultralytics confusion matrix also includes a background row and column. These represent missed detections and false positive predictions, respectively. A high number of missed malignant regions indicates False Negatives, which are particularly

important in breast lesion detection because suspicious regions may remain undetected. False Positives occur when the model predicts a region where no annotated object is present and can also affect the reliability of the detection system. The structure of a two-class detection confusion matrix is presented in Figure 19.

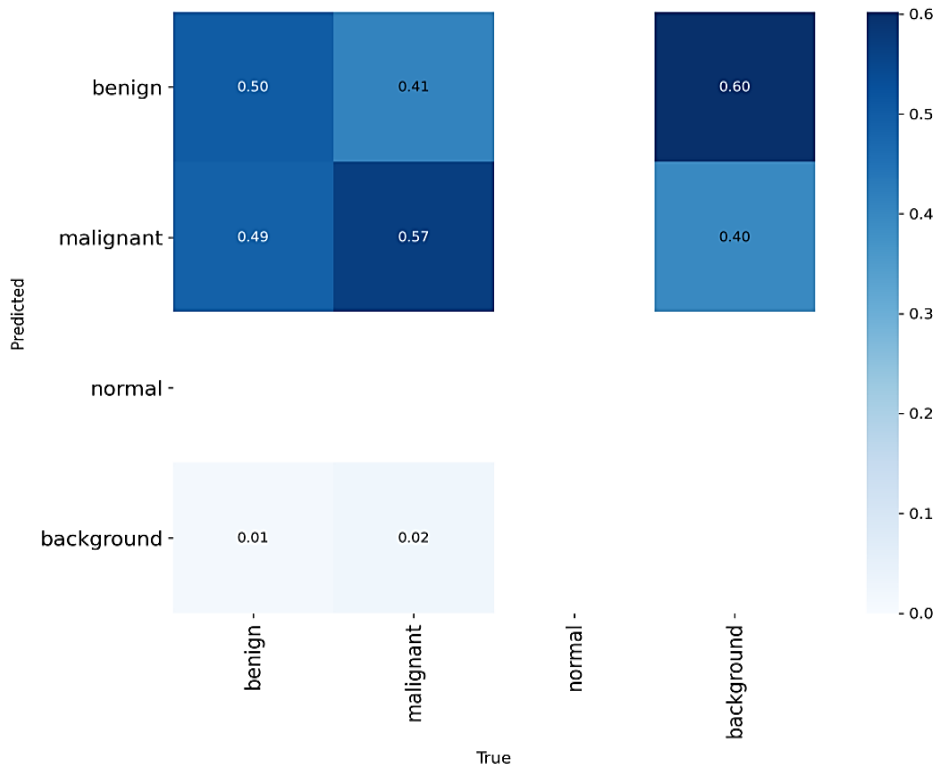


Figure 19 – Structure of a confusion matrix for a two-class detection problem.

2.8 Training Environment and Explored Hyperparameters

All experiments were conducted using Kaggle Notebooks **[12]** with a Python environment and an NVIDIA Tesla T4 GPU. The YOLO models were implemented using the Ultralytics framework with PyTorch. The datasets, converted annotations, and trained models were stored within the Kaggle environment.

To ensure consistent evaluation, the train, validation, and test partitions were fixed throughout all experiments. This allowed the different models and configurations to be compared under the same data conditions.

In addition to the model architectures and sizes, several training hyperparameters were considered during the experimental process. These included the optimizer, learning rate,

weight decay, data augmentation, early-stopping patience, and class-balancing strategy. Their definitions are provided below :

- **Optimizer:** The method used to update the model weights during training. Different optimizers, such as SGD and AdamW, use different strategies for adjusting the weights.
- **Initial learning rate (lr0):** The initial step size used to update the model weights during training.
- **Weight decay:** A regularization method that limits the growth of model weights and helps reduce overfitting.
- **Data augmentation:** Techniques used to increase training data diversity through transformations such as mosaic, mixup, flipping, and color adjustments.
- **Early-stopping patience:** The number of consecutive epochs without improvement in validation performance before training is stopped.
- **Class balancing:** A training strategy used to reduce the effect of unequal class distributions, for example by giving greater representation to minority classes.

The detailed outcome of each of these experiments, together with the resulting Precision, Recall, mAP50, and mAP50-95 values for YOLOv8n, YOLOv11n, and YOLOv26n.

2.9 Conclusion

This chapter moved from the general foundations of neural networks to the specific problem of object detection. It distinguished object detection from image classification and placed it within the family of one-stage detectors to which the YOLO architectures belong.

We presented the architectures of YOLOv8n, YOLOv11n, and YOLOv26n, the three models compared in this thesis. We also highlighted the main architectural changes introduced by YOLOv26n, including the C3k2 blocks, the C2PSA attention mechanism, and the end-to-end detection head. The smallest nano variants were selected because they were better suited to the size of the datasets used in this work.

We also introduced the evaluation metrics used throughout this study, including precision, recall, mAP50, and mAP50–95, and described the training environment and the main hyperparameters considered during the experiments. With this theoretical and architectural background established, the next chapter focuses on the datasets used for breast lesion detection and explains the selection of the data used in the experiments.

Chapter 3: Model Implementations and Experimental Results

Abstract

This chapter presents the experimental methodology employed to investigate deep learning-based image analysis in medical imaging. The study utilizes two datasets: the BUSI dataset, which contains breast ultrasound images, and the Figshare dataset, which includes brain MRI images. The chapter details the tools, preprocessing and annotation procedures, experimental setup, as well as the implementation and evaluation of the selected models. The results are subsequently analyzed to compare the various approaches and to assess their effectiveness in medical image analysis.

3.1 Introduction

3.2 Tools and Development Environment

3.3 Datasets

3.4 Data Preprocessing Pipeline

3.5 Annotation Preparation and Verification

3.6 Experimental Results

3.7 Conclusion

3.1 Introduction

This third and final chapter presents and discusses the experimental framework and results of this study. The quality of the training data and its preparation are important for the performance of medical image object detection models. Therefore, particular attention is given to the datasets used in the experiments and their preparation.

The chapter introduces the two medical imaging datasets used in this study, namely the Breast Ultrasound Images (BUSI) dataset and the Figshare brain tumor dataset. BUSI contains a relatively small collection of breast ultrasound images, while Figshare contains a larger collection of contrast-enhanced brain MRI images. Their main characteristics and preparation for object detection are presented before the experimental evaluation.

The experiments evaluate three lightweight YOLO architectures, YOLOv8n, YOLOv11n, and YOLO26n, on both datasets. CNN and VGG16 approaches are also evaluated for lesion localization on BUSI and compared with the YOLO models. The results are then presented and discussed using detection and localization metrics, followed by the main findings, limitations, and conclusion of the study.

3.2 Tools and Development Environment

The experimental pipeline was implemented using a set of software tools for data preparation, annotation processing, model training, evaluation, and visualization. The main tools used in this work are summarized below:

- **Python:** Used for data preprocessing, mask-to-bounding-box conversion, dataset organization, and model training and evaluation scripts.
- **Kaggle Notebooks:** Provided the cloud-based computational environment and NVIDIA Tesla T4 GPU used for the experiments.
- **Ultralytics:** Used to implement and train the YOLOv8n, YOLOv11n, and YOLO26n models and to perform model validation and evaluation.
- **OpenCV (cv2):** Used for image and mask processing, including reading images, processing segmentation masks, extracting tumor or lesion regions, and generating bounding-box annotations.
- **Roboflow:** Used as an annotation and annotation-verification platform. BUSI images were annotated using bounding boxes, while the bounding-box annotations generated from the Figshare tumor masks were imported and checked in Roboflow to verify their correspondence with the tumor regions.

3.3 Datasets

3.3.1 The BUSI Dataset

The Breast Ultrasound Images (BUSI) dataset was collected at Baheya Hospital for Early Detection and Treatment of Women's Cancer in Cairo, Egypt **[13].** This data is organized into three categories that directly reflect the diagnostic outcome of each scan,including normal cases with no visible mass, benign lesions, and malignant lesions. Each non-normal image is accompanied by one or more binary segmentation masks, produced by radiologists, delineating the exact boundary of the lesion, from which a bounding box can be derived automatically. After filtering out images without a usable mask, the working subset totals 647 annotated benign and malignant images, later complemented with the normal images used as negative, background-only examples during training, for a combined dataset of 778 images.Figure 20 shows representative BUSI images and their corresponding lesion masks.

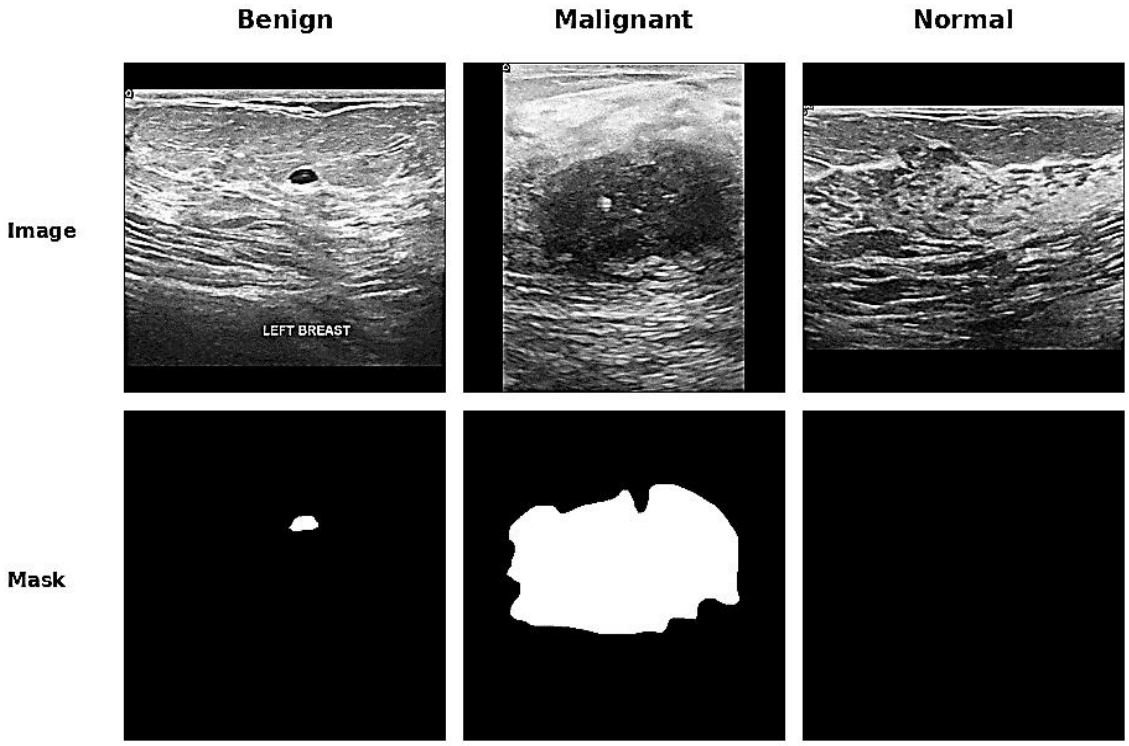


Figure 20 - Sample BUSI ultrasound images with their corresponding lesion masks.

3.3.2 The Figshare Brain Tumor Dataset

The Figshare brain tumor dataset is a publicly available medical imaging dataset originally introduced by Cheng et al. **[14]** and hosted on the Figshare research data repository..

The images were collected from Nanfang Hospital in Guangzhou and the General Hospital of Tianjin Medical University in China. The dataset contains 3,064 T1-weighted contrast-enhanced MR images from 233 patients and includes three types of brain tumors: meningioma, glioma, and pituitary tumor.

Compared with BUSI, Figshare provides a larger number of images and tumor regions that can be used for object localization. The dataset also contains variations in tumor size, shape, and location, making it suitable for training and evaluating object detection models. The available tumor masks were used to generate the bounding-box annotations required by the YOLO models. Examples of the original MR images and their corresponding tumor masks are shown in Figure 21.

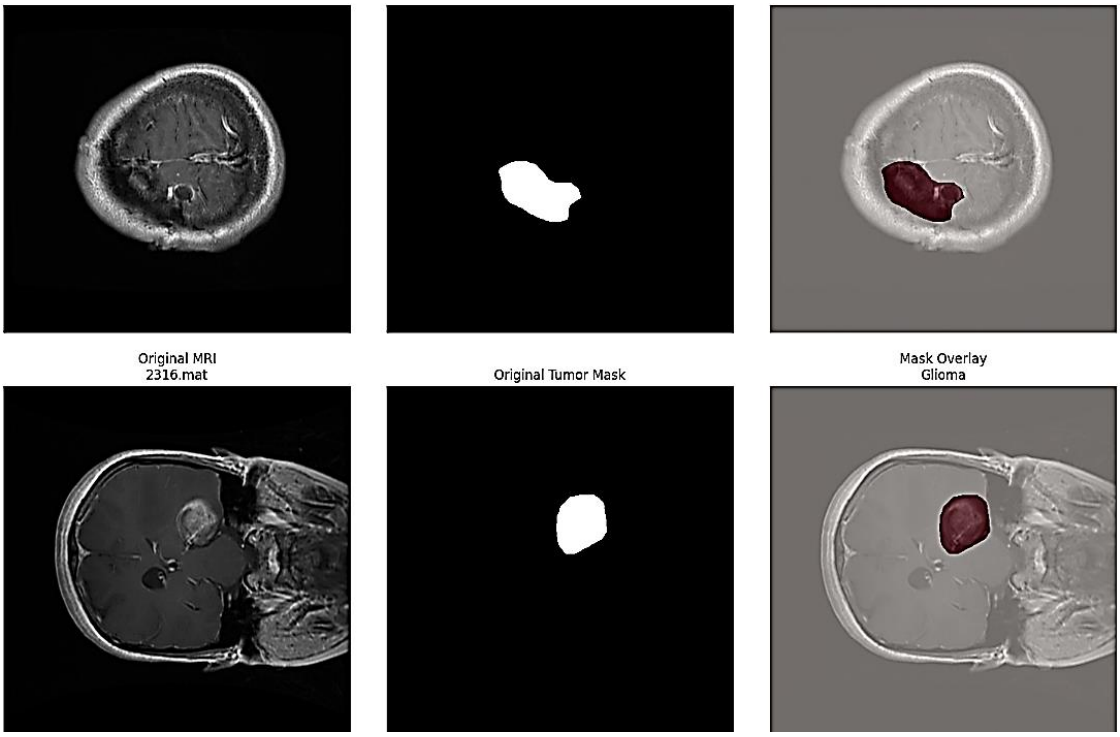


Figure 21– Samples from Figshare brain MRI images with their corresponding tumor masks.

Before discussing the results, it is important to highlight the key characteristics of the datasets used in the experiments. Table 1 summarizes the main differences between the two datasets as actually used.

Aspect	BUSI	Figshare
Imaging modality	Breast ultrasound	T1-weighted contrast-enhanced brain
Usable images (after processing)	778 (647 lesions + 131 normal)	3064 images
Classes	Benign, Malignant, Normal	Meningioma, Glioma, Pituitary-Tumor
Class balance	437 benign / 210 malignant	708 meningioma, 1426 glioma, 930 pituitarytumor
Annotation type	Radiologist-drawn binary lesion masks	Tumor-region annotations
Lesion/background contrast	Comparatively high	Generally favorable

Table 1 - Comparison between the BUSI and Figshare datasets.

3.4 Data Preprocessing Pipeline

Before training, both datasets were prepared in a common format compatible with the YOLO object detection framework. The available region annotations were converted into YOLO-compatible bounding boxes, with the required coordinates normalized according to the corresponding image dimensions.

For BUSI, the provided lesion masks were used to generate bounding-box annotations for images containing lesions. Images without lesion annotations were not included in the final object detection dataset. For Figshare, an additional preprocessing step was required because the original data were stored in MATLAB .mat files. The required image, tumor mask, patient identifier, and tumor class information were extracted from each file. The images were converted to PNG format after normalization, while the tumor masks were used to generate the corresponding bounding boxes. The original tumor classes were mapped to YOLO class identifiers: meningioma (0), glioma (1), and pituitary tumor (2).

After preprocessing, the images and annotation files were organized into training, validation, and test sets, together with the corresponding data.yaml configuration file. For Figshare, the split was performed at the patient level to prevent images from the same patient from appearing in different subsets. Patients were randomly divided using a fixed seed of 42,

with approximately 70% assigned to training, 15% to validation, and 15% to testing. These partitions were kept unchanged throughout the experiments to ensure a consistent comparison between the different YOLO models.

3.5 Annotation Preparation and Verification

After generating the bounding-box annotations, we imported both datasets into Roboflow for annotation management and visual inspection. For the BUSI dataset, we derived the bounding boxes from the provided lesion masks, whereas for the Figshare dataset, we programmatically converted the available tumor masks into bounding-box annotations. We then overlaid the generated annotations on their corresponding ultrasound images to verify that each bounding box accurately covered the intended lesion region.

The annotations for both datasets were reviewed in Roboflow, and a small number of boxes requiring adjustment were corrected before training. This final verification ensured that the images and label files were properly prepared for the YOLO training stage. Figure 22 presents examples of the annotated BUSI images in Roboflow.

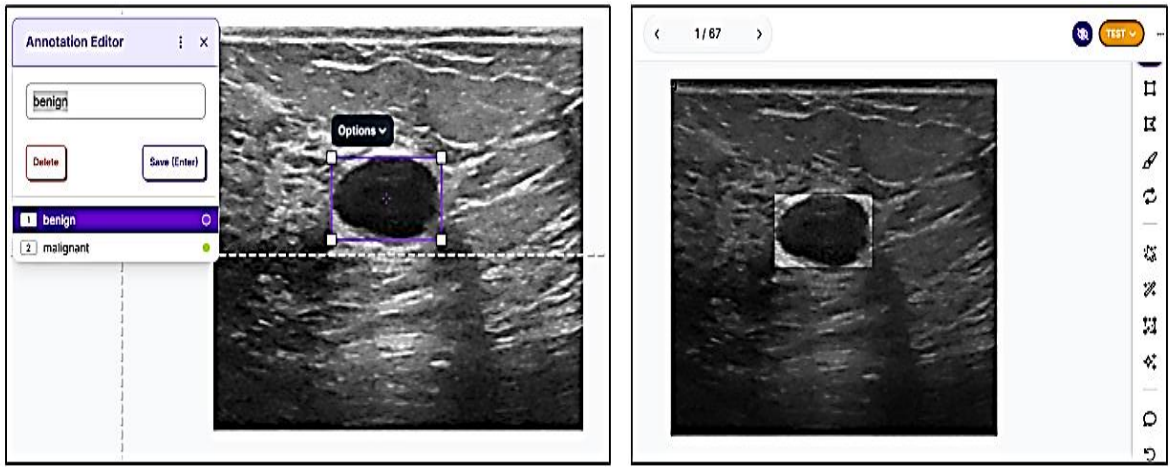


Figure 22– The BUSI dataset imported into Roboflow, with algorithm-generated bounding boxes.

The review process combined two complementary checks, summarized below:

– **Roboflow visual review:** every image in the dataset was inspected inside Roboflow to confirm the box enclosed the visible lesion without being noticeably too loose or too tight, and without missing part of the mass.

– **Mask-versus-box overlay:** for a representative sample of images, the original expert mask outline was drawn together with the generated box on the same image, so that the box could be checked directly against the exact boundary it was derived from, rather than against a visual approximation of where the lesion appears to be.

The annotation and review process is illustrated in Figure 23.

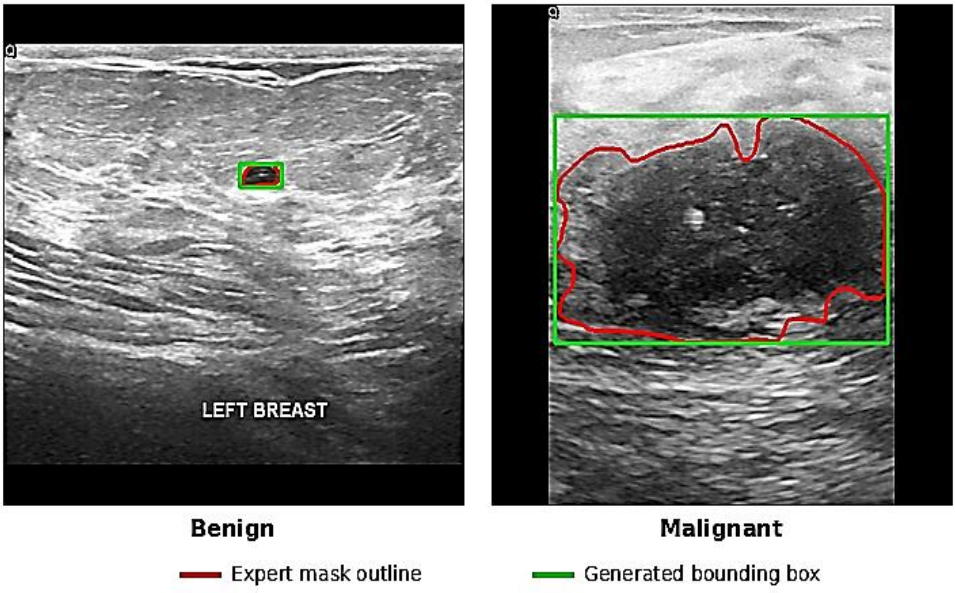


Figure 23–Mask-versus-box overlay comparison.

This two-step review, first at the level of the individual image inside Roboflow, then geometrically against the source mask, confirmed that the generated bounding boxes consistently enclosed the visible lesion, with the box boundary closely following the mask outline on all sides across the reviewed sample. Where a box appeared visibly misaligned during review, it could be corrected directly within Roboflow's annotation interface and re-exported, so that the validation step doubled as a correction mechanism rather than a purely observational one.

3.6 Experimental Results

The experimental section comprised two stages. First, we tested conventional localization methods using a custom convolutional neural network (CNN) and a pretrained VGG16 model to set a baseline. The outcomes of these methods were analyzed to identify the limitations of standard CNN-based localization regarding accuracy and model complexity. The primary stage of the study focused on YOLO-based object detection, in which YOLOv8n,

YOLOv11n, and YOLOv26n models were trained and evaluated under the specified experimental configurations. For the BUSI dataset, models were trained and evaluated using consistent fixed data partitions, whereas for the Figshare dataset, fixed patient-level partitions were applied as previously described.

Model performance was evaluated using Precision and Recall, together with IoUto measure the overlap between the predicted bounding boxes and the corresponding lesion or tumor regions. The same evaluation procedure was applied to all YOLO models to ensure a consistent comparison across the two datasets.

The following sections present the results obtained on the BUSI and Figshare test sets and discuss the main observations from the experiments.

Parameter	BUSI	Figshare
Maximum epochs	100	100
Image size	640 × 640	640 × 640
Batch size	16	16
Data partition	Fixed	Patient-level fixed
Early stopping patience	20	20

Table2 -Main training configurations used in the experiments.

3.6.1 Convolutional Neural Network Implementation

To address our objective of lesion localization, we first build a Convolutional Neural Network (CNN) model as a baseline approach. We implemented the model on Kaggle, which provided access to GPU resources for training.The CNN was designed to predict lesion bounding-box coordinates directly from BUSI images. The architecture consists of three convolutional layers, with Max-Pooling applied after each layer, followed by Global Average Pooling and fully connected layers for bounding-box regression. The final layer outputs four values representing the normalized bounding-box coordinates. The complete architecture and layer configurations are presented in Figure 24.

3.6.1.1Training CNN Model

Once the CNN architecture was defined, the model was trained for bounding-box regression using the BUSI dataset. The training process used the Adam optimizer with a learning rate of 0.001 and Mean Squared Error (MSE) as the loss function. A batch size of 16 was used, and the model was trained for up to 60 epochs. Checkpoints were saved at epochs 20, 40, and 60 to evaluate the evolution of the localization performance during training.Table 3 presents the localization results obtained for four test images from the BUSI dataset at different stages of training, specifically after 20, 40, and 60 epochs. The localization performance is evaluated using the Intersection over Union (IoU) metric, which measures the overlap between the predicted region and the ground-truth region. Higher IoU values indicate better localization performance.

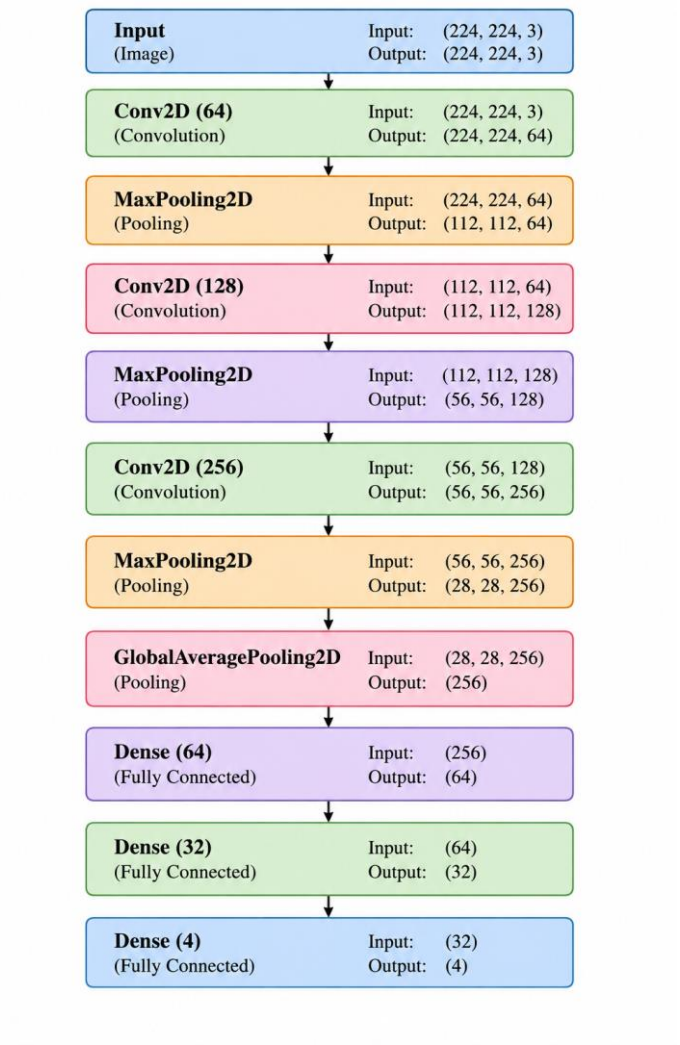


Figure 24- CNN Architecture.

Images	Number of epochs	IOU
1	20	**26**
	40	34
	60	41
2	20	**42**
	40	45
	60	52
3	20	38
	40	32
	60	**18**
4	20	**00**
	40	02
	60	**00**

Table 3 - IOU results for the Busi dataset, 4 samples.

As shown in Table 3, the localization performance varied across the four test images. For Images 1 and 2, the Intersection over Union (IoU) increased during training, rising from 26 to 41 and from 42 to 52, respectively. In contrast, Image 3 showed a decrease in IoU at later epochs, while Image 4 remained poorly localized, with IoU values close to zero, indicating that the convolutional neural network (CNN) identified effective localization patterns for certain images, but its performance was inconsistent across all test samples.

Samples of the CNN localization results at different training epochs are shown in Figure 25.

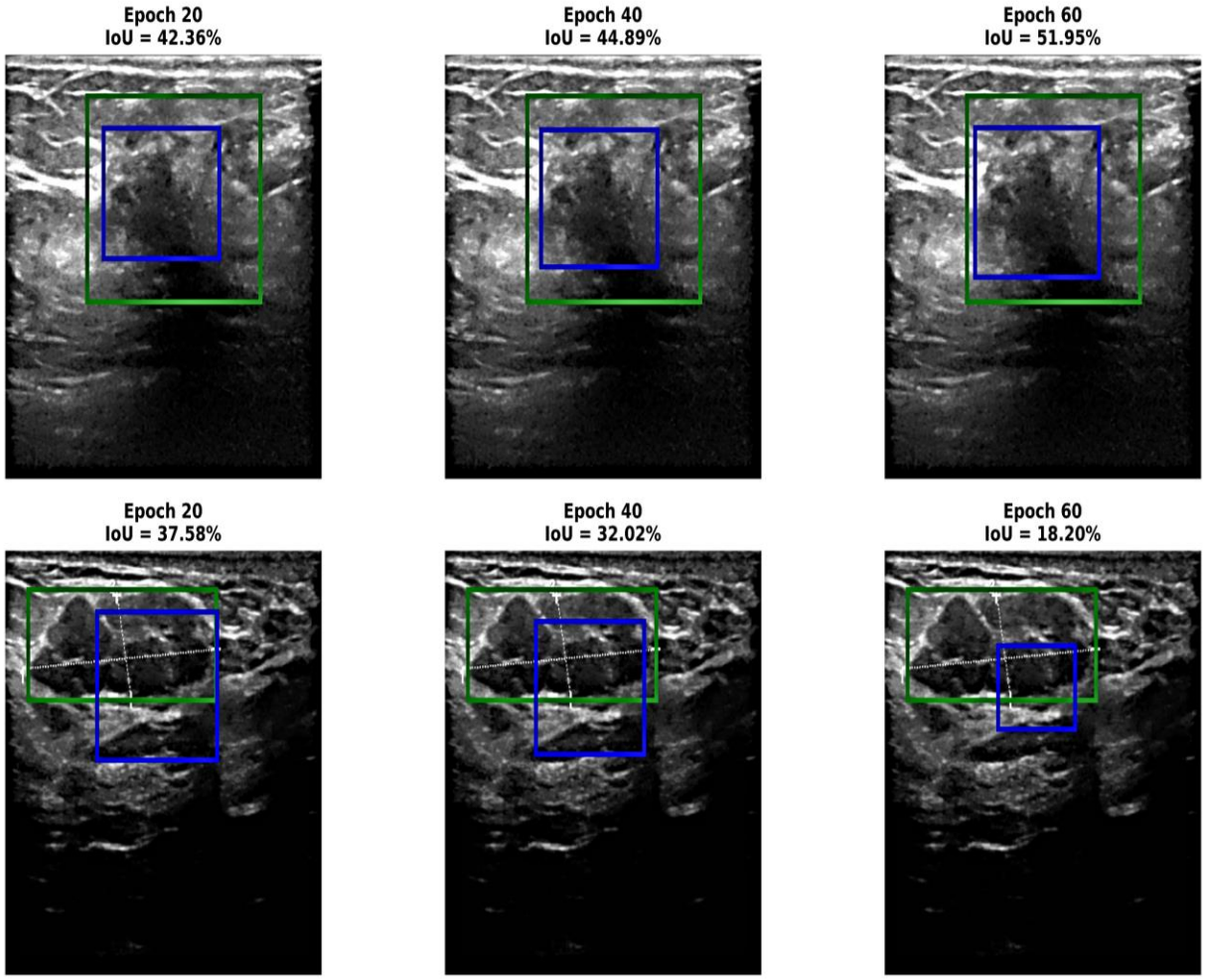


Figure 25 – Localization results for two BUSI test images at 20, 40, and 60 training epochs.

In the first image, the Intersection over Union (IoU) increases from 42.36% at epoch 20 to 44.89% at epoch 40 and 51.95% at epoch 60, indicating improved lesion localization as training advances. In contrast, the second image shows a decrease in IoU from 37.58% to 32.02% and ultimately to 18.20%, demonstrating that further training did not enhance localization for this sample. These examples highlight the variability of the model's localization performance.

Overall,the comparison between the different training stages shows that increasing the number of epochs did not consistently improve the predictions. Although some samples benefited from additional training, others showed little improvement or decreased overlap at later epochs. Therefore, increasing the number of epochs alone did not guarantee better localization performance.Although the proposed CNN provided a useful baseline for the localization task, its overall performance remained limited. Therefore, a pretrained VGG16 architecture was adopted in the next stage to obtain better feature representations for lesion localization.

3.6.2 Transfer Learning Implementation

In this part, transfer learning was applied using the VGG16 architecture with pre-trained ImageNet weights, without its original classification layers. The earlier convolutional layers were frozen, while the final block was fine-tuned during training.A Conv2D layer with 256 filters was added, followed by Batch Normalization and Global Average Pooling2D. Two fully connected layers with 128 and 64 units using ReLU activation were then added, with a Dropout layer between them. Finally, a dense layer with four units was used to predict the normalized bounding-box coordinates. The model was trained using the Adam optimizer with a learning rate of 0.00001.

3.6.2.1Result Interpretation of the VGG16 Model

Table 4 presents the IoU results obtained using the VGG16 model on four selected test images. We trained the model for 20, 40, and 60 epochs and recorded the IoU results for each selected test image.After training the VGG16 model, an improvement in localization performance was observed compared with the CNN baseline.

Images	Number of epochs	IOU
1	20	**90**
	40	87
	60	**90**
2	20	87
	40	85
	60	**89**
3	20	87
	40	**88**
	60	**88**
4	20	85
	40	85
	60	**86**

Table 4- IOU results for Busi dataset using VGG16.

In Table 4, the IoU values remained consistently high across all samples, ranging from 85% to 90%. For Image 1, the highest IoU of 90% was achieved at both 20 and 60 epochs. Image 2

Image 2 showed a slight decrease from 87% to 85% at 40 epochs, followed by an increase to 89% at 60 epochs. For Image 3, the IoU increased marginally from 87% to 88% and then stabilized. Image 4 showed a small improvement from 85% to 86% at 60 epochs.These results demonstrate that VGG16 provided stable and consistent localization performance across the tested images. Increasing the number of epochs resulted in only minor changes, indicating that the pretrained model was able to extract effective features even during the early stages of training.

Samples of the VGG16 localization results at different training epochs are shown in Figure 26.

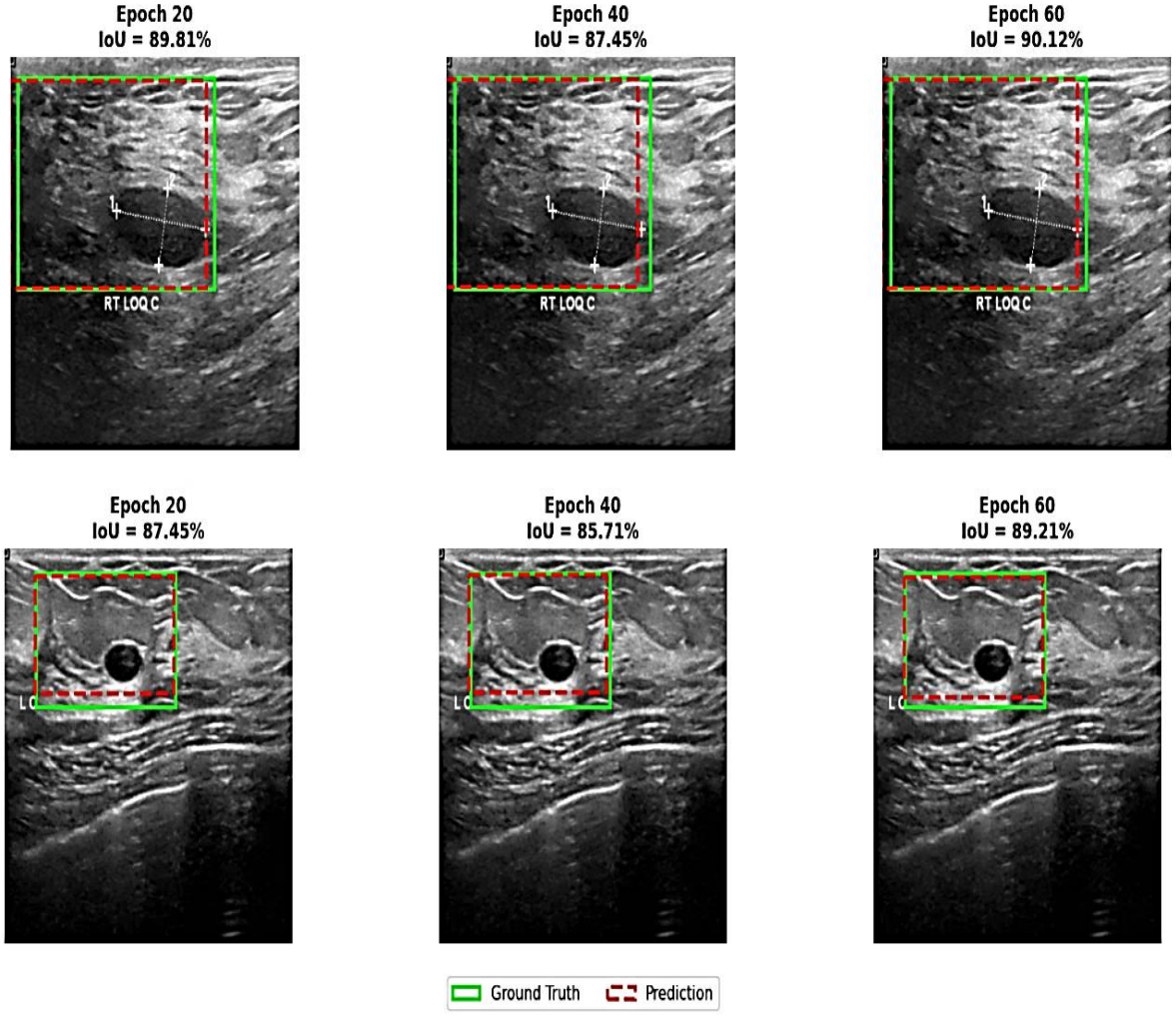


Figure 26 – VGG16 Localization Results on Two BUSI Test Images Across Different Training Epochs.

The predicted bounding boxes exhibit substantial overlap with the ground-truth regions in both test images, as indicated by Intersection over Union (IoU) values consistently exceeding 85%.The experiments with the CNN and VGG16 architectures revealed a clear trade-off between model complexity and localization performance. The CNN is lightweight, with only 392,548 parameters, but provides limited localization accuracy, whereas VGG16 achieves better results at the cost of nearly 16 million parameters.

This motivated the use of YOLO as a more balanced alternative. As shown in Table 5, YOLOv8n, YOLOv11n, and YOLO26n contain approximately 3.15, 2.62, and 2.41 million parameters, respectively. These models are considerably smaller than VGG16 while being specifically designed for object detection and bounding-box localization. Therefore, YOLOv8n, YOLOv11n, and YOLO26n were evaluated to investigate a more efficient balance between localization performance and model complexity.

Model	Total Parameters
CNN	**392,548**
VGG16	**15,969,796**
YOLOv8n	**3,151,904**
YOLOv11n	**2,616,248**
YOLO26n	**2,408,932**

Table 5- Comparison of Model Complexity in Terms of Number of Parameters.

3.6.3 YOLO Object Detection Results and Comparison

In this section, we present the main object detection results that we obtained using different YOLO models on the BUSI and Figshare datasets. We first analyze the results separately for each dataset to compare the performance of the different YOLO versions.

3.6.3.1BUSI Object Detection Results

YOLOv8n

The YOLOv8n model was first evaluated on the BUSI dataset using the experimental configurations described above. The obtained detection results are presented below, followed by representative detection outputs for the best-performing configuration.Figure 27

presents the evolution of the main YOLOv8n training metrics on the BUSI dataset, including precision, recall, and box lossover the training epochs.

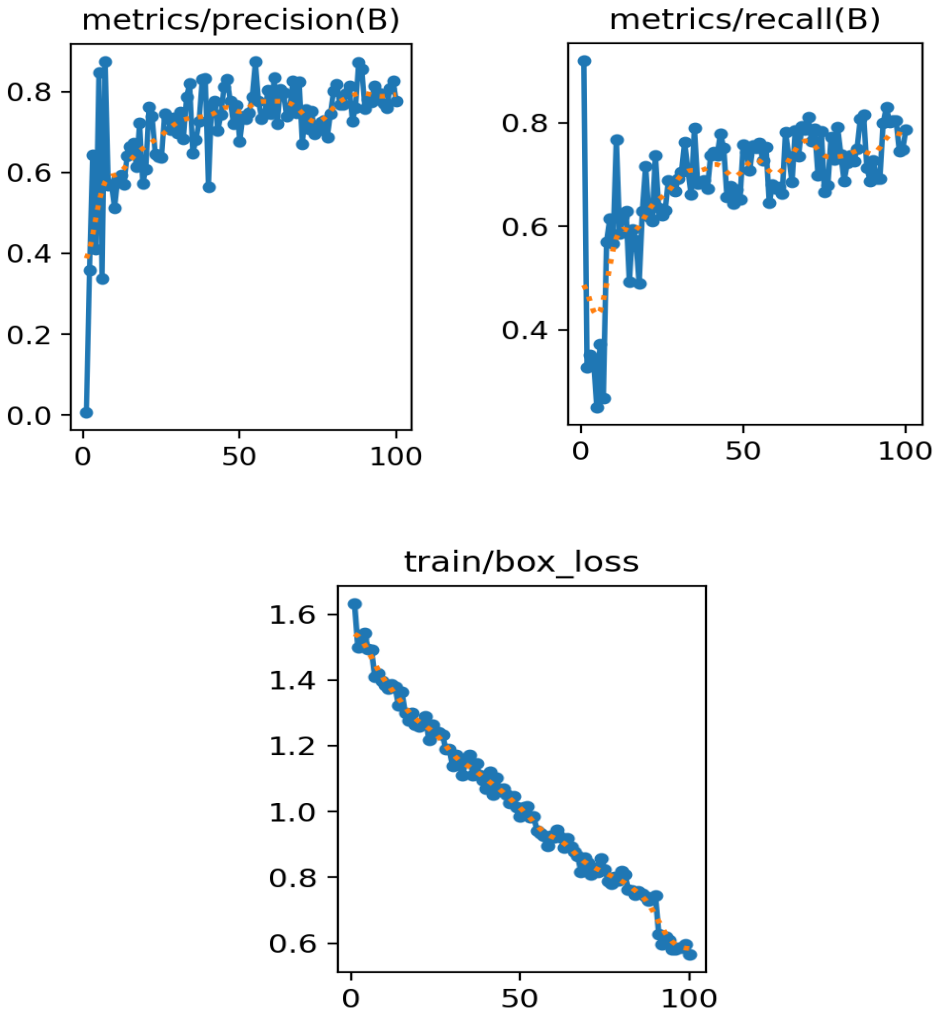


Figure 27– Evolution of YOLOv8n Training Metrics on the BUSI Dataset.

As shown in Figure 27, the YOLOv8n model showed an initial improvement during the early stages of training, followed by some fluctuations in precision and recall. After these variations, both metrics remained relatively stable during the later epochs. At the same time, the box loss generally decreased throughout training, indicating an improvement in the model's ability to localize lesions. Overall, the curves indicate that the model learned progressively and reached a relatively stable training behavior in the later epochs.

Figure 28 presents representative benign and malignantimages from the BUSI test set, comparing the ground-truth annotations with the corresponding predictions produced by YOLOv8n.

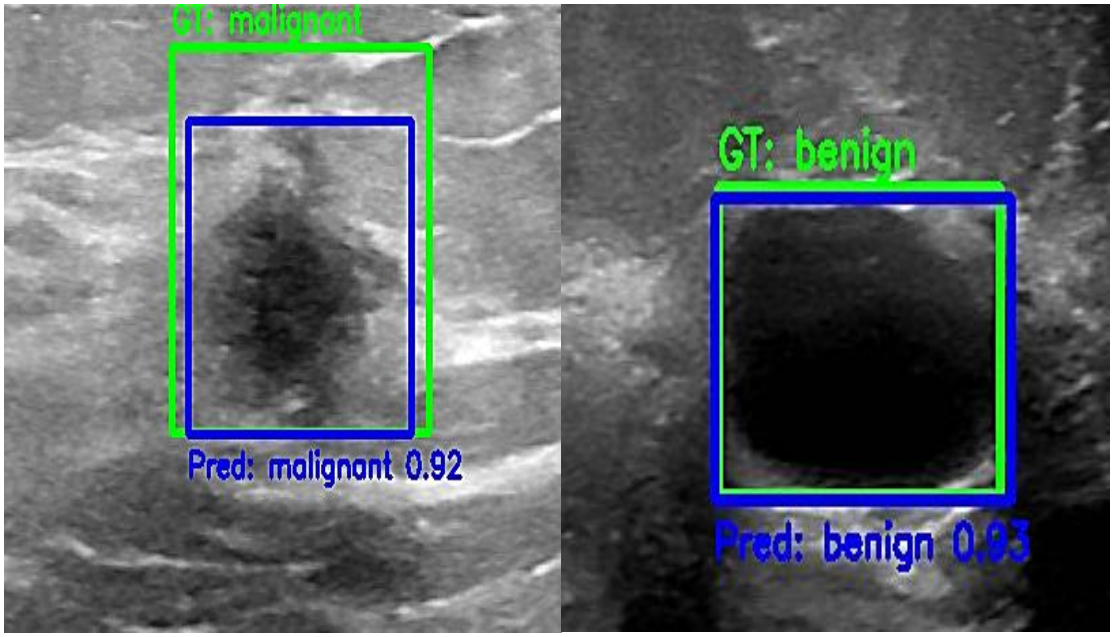


Figure 28– Comparison of Ground-Truth Annotations and YOLOv8n Predictions on BUSI Test Images.

The predicted bounding boxes, class labels, and confidence scores illustrate the model's ability to classify and localize breast lesions.

YOLOv11n

Next, we present the results of the YOLOv11n implementation on the BUSI dataset. The obtained results are presented below.Figure 29shows the precision and recall metrics during the training of YOLOv11n on BUSI Dataset.

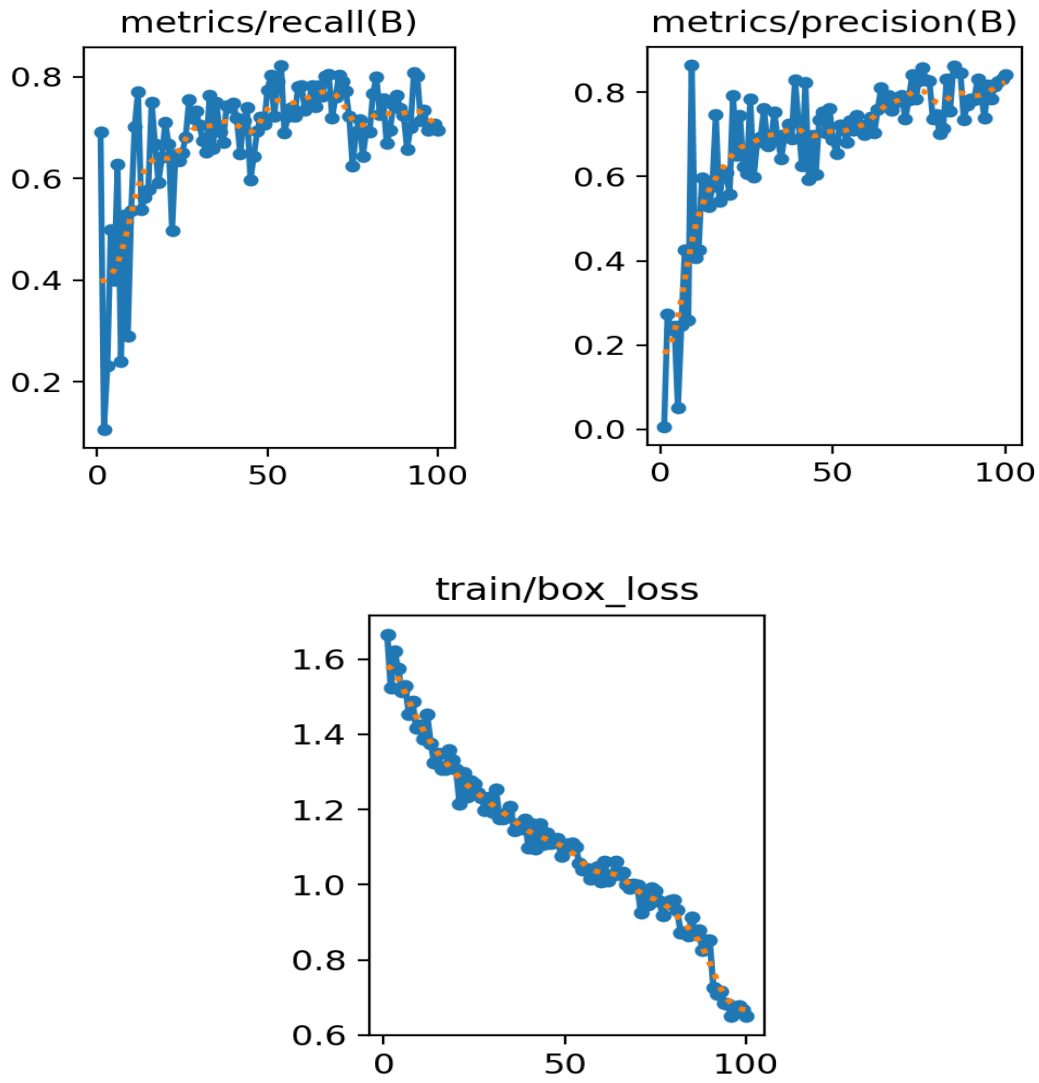


Figure 29 – Evolution of YOLOv11n Training Metrics on the BUSI Dataset.

As shown in Figure 29, the YOLOv11n model showed a clear improvement throughout the training process. Precision and recall generally increased across the epochs, with some variations during training, before reaching more stable values toward the later epochs. The box loss gradually decreased, indicating better localization of the detected lesions. Overall, the curves show that the model continued to improve during training and reached a more stable behavior in the final epochs.

Figure 30 presents representative benign and malignant images from the BUSI test set, showing the ground-truth annotations alongside the corresponding predictions generated by YOLO11n. The examples illustrate the model's ability to detect and localize breast lesions in both classes.

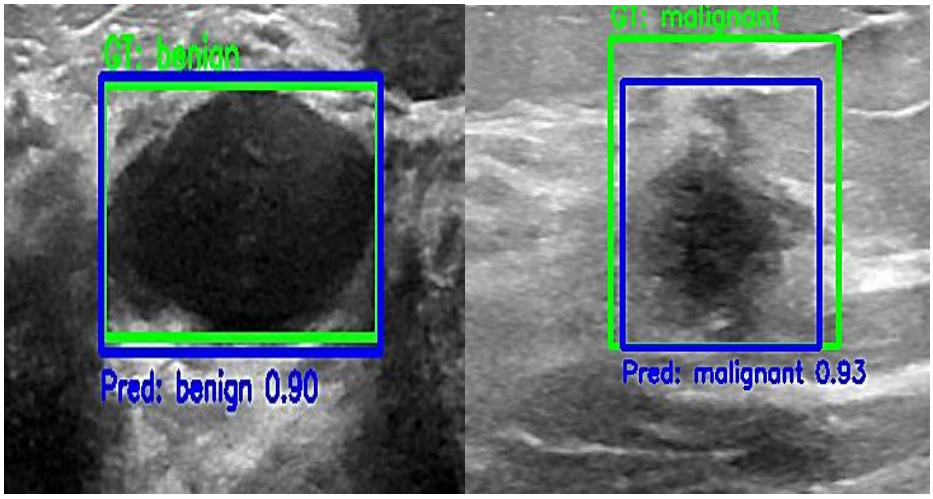


Figure 30– Comparison of Ground-Truth Annotations and YOLOv11n Predictions on BUSI Test Images.

YOLOv26n

Now we move to the results of the YOLO26n implementation on the BUSI dataset. The obtained results are presented below (Figure 31). Figure 32shows sample YOLO26n predictions and their corresponding ground-truth annotations on the BUSI test set.

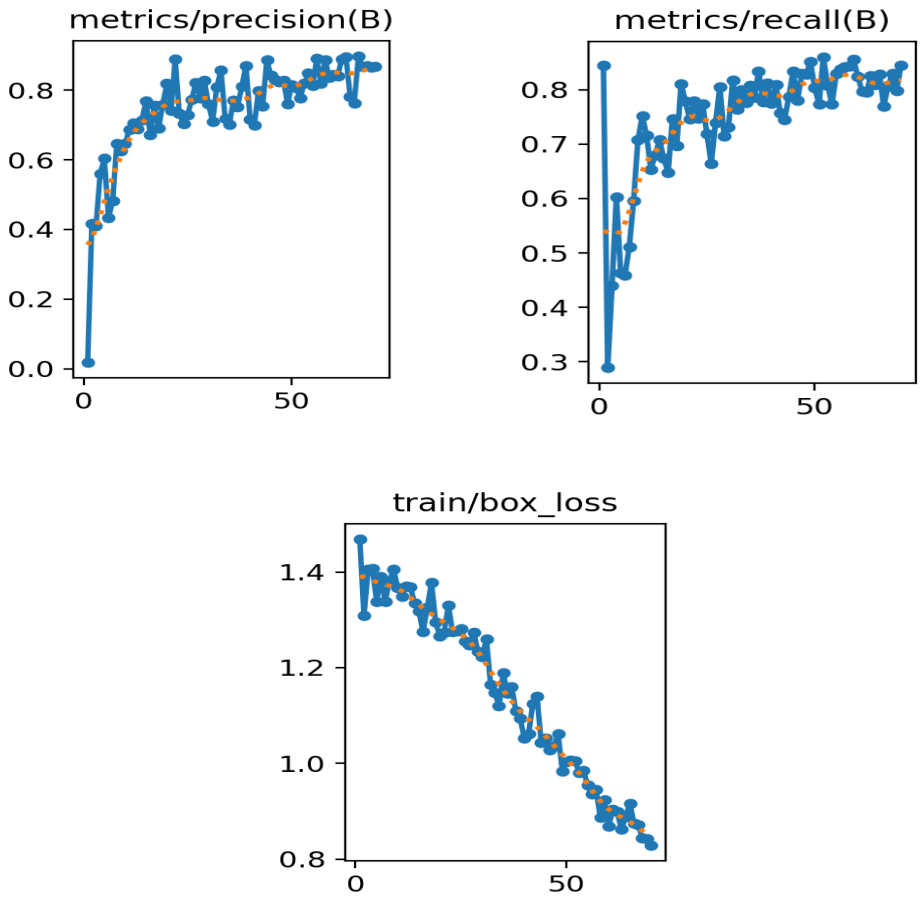


Figure 31– Evolution of YOLOv26n Training Metrics on the BUSI Dataset.

As shown in Figure 31, the YOLO26n model showed a continuous improvement throughout the training process. Precision and recall generally increased across the epochs, despite some fluctuations during training, while the box loss gradually decreased. This indicates that the model progressively improved its ability to detect and localize the lesions. The curves also show a more stable behavior toward the later epochs, indicating that the model continued to learn effectively throughout training. Figure 32 shows representative benign and malignant cases with their ground-truth and YOLOv26n predictions. The predicted boxes closely match the ground-truth annotations, showing accurate lesion localization in both classes.

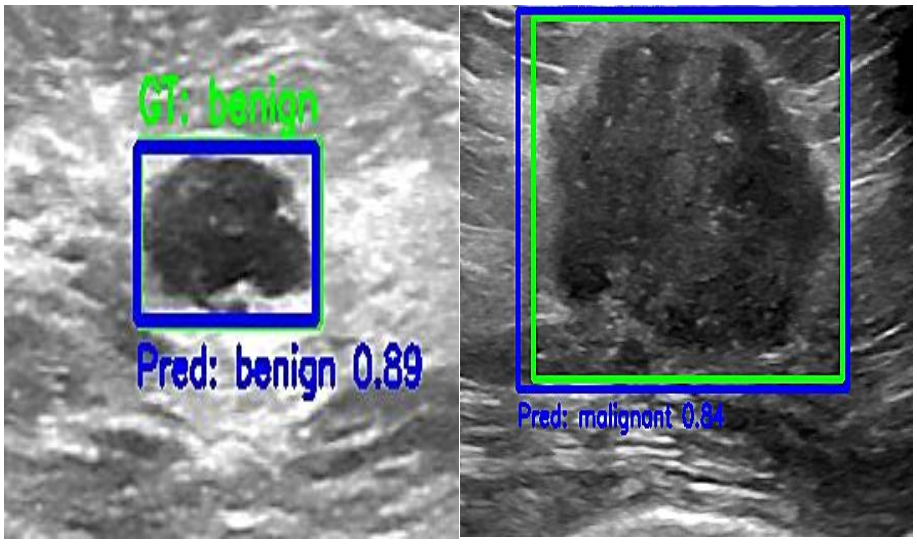


Figure 32 – Comparison of Ground-Truth Annotations and YOLOv26n Predictions on BUSI Test Images.

The results obtained on the BUSI test set show that the three YOLO models were able to detect and localize breast lesions with reasonable performance.

Table 6 summarizes the performance of the three YOLO models on the BUSI test set in terms of Precision, Recall, and IoU.

Model	Precision	Recall	IoU
YOLOv8n	79.14%	63.24%	80.64%
YOLOv11n	80.39%	70.16%	77.94%
YOLOv26n	**92.23%**	**76.30%**	**81%**

Table 6 – Performance comparison of YOLO models on the BUSI test set.

YOLO26n achieved the strongest overall results, with precision of 0.9223, recall of 0.7630, and IoU of 0.8100. YOLOv8n achieved a similar IoU of 0.8064 but with lower precision and recall, while YOLO11n achieved an IoU of 0.7794. These results show that YOLO26n provides the best balance between correct detections, lesion detection rate, and localization accuracy among the three YOLO variants.

In comparison to the initial CNN,the YOLO models demonstrate significantly stronger and more consistent localization results. While VGG16 achieves high IoU values on the selected test samples, it requires a much larger number of parameters. In contrast, the YOLO Nano

models maintain strong localization performance with considerably lower model complexity, supporting their use as a more efficient solution for lesion detection and localization.

3.6.3.2Figshare Dataset Results

To further evaluate the YOLO models on a larger and more diverse dataset, the experiments were extended to the Figshare dataset.As shown in Table 1, Figshare contains 3,064 images, whereas BUSI includes 778 usable images. The larger dataset offers more training examples and enables evaluation of whether the models maintain localization performance across varying imaging conditions and tumor classes.

YOLOv8n

The initial model evaluated on the Figshare dataset was YOLOv8n. Figure33 display the precision, recall, and loss curves recorded during training. These curves facilitate analysis of the model's learning dynamics and localization performance throughout the training epochs.

Precision and recall increased rapidly during the initial epochs, demonstrating that the model quickly improved its ability to detect tumor regions. Following this initial improvement, both metrics remained relatively high, despite some fluctuations during training. Concurrently, the box loss decreased steadily from approximately 1.4 to 0.2, indicating continuous improvement in bounding-box prediction and localization accuracy. These results indicate that YOLOv8n learned effectively from the Figshare dataset and achieved a stable level of detection performance by the end of training.

A representative detection result is shown in Figure 34.

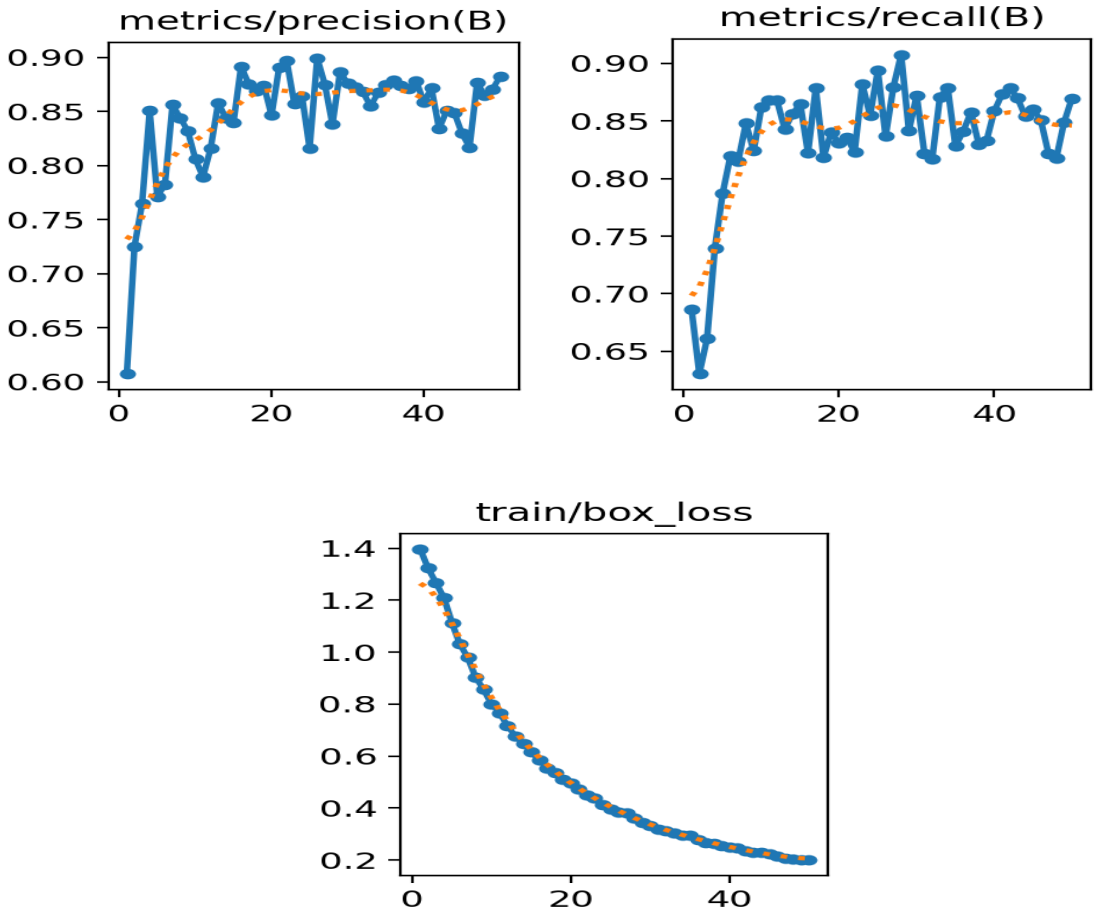


Figure 33 – Evolution of YOLOv8n Training Metrics on the Figshare Dataset.

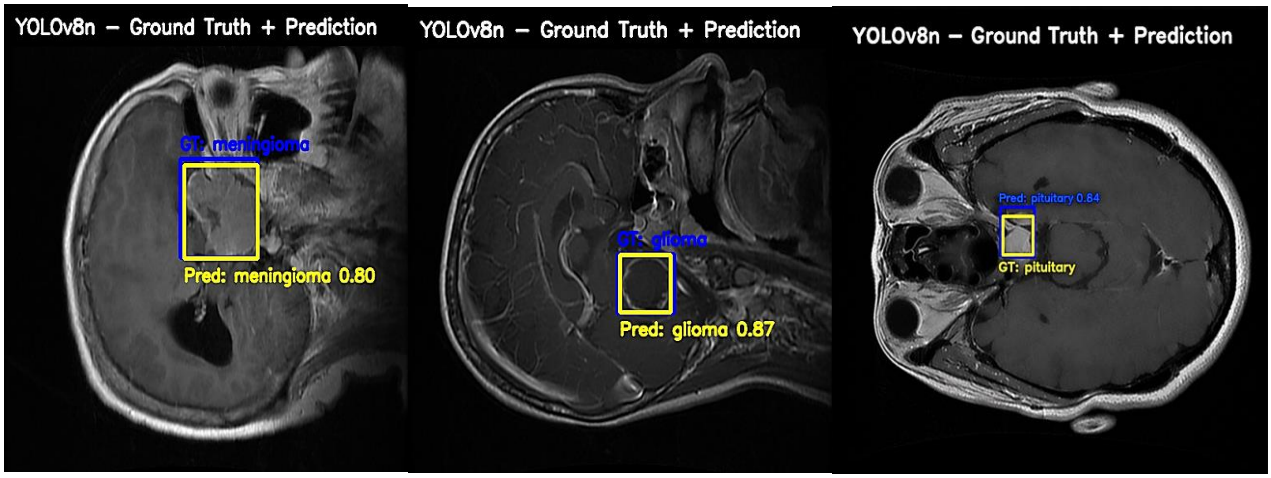


Figure 34 – Ground-Truth and YOLOv8n Predictions for Representative Meningioma, Glioma, and Pituitary Tumor Cases from the Figshare Test Set.

Figure 34 presents representative examples of meningioma, glioma, and pituitary tumor cases from the Figshare test set. In all three cases, the predicted bounding boxes closely overlap

with the ground-truth annotations, showing that YOLOv8n localized the tumor regions effectively. The model also correctly classified the three tumors, with confidence scores of 0.80 for meningioma, 0.87 for glioma, and 0.84 for pituitary tumor. These visual examples complement the quantitative results by showing that YOLOv8n can correctly detect, classify, and localize different types of brain tumors.

YOLOv11n

Now we move to the results of the YOLOv11n implementation on the Figshare dataset. The Figure 35 below presents the Precision, Recall and loss curves recorded during the training of YOLO11n on the Figshare dataset.

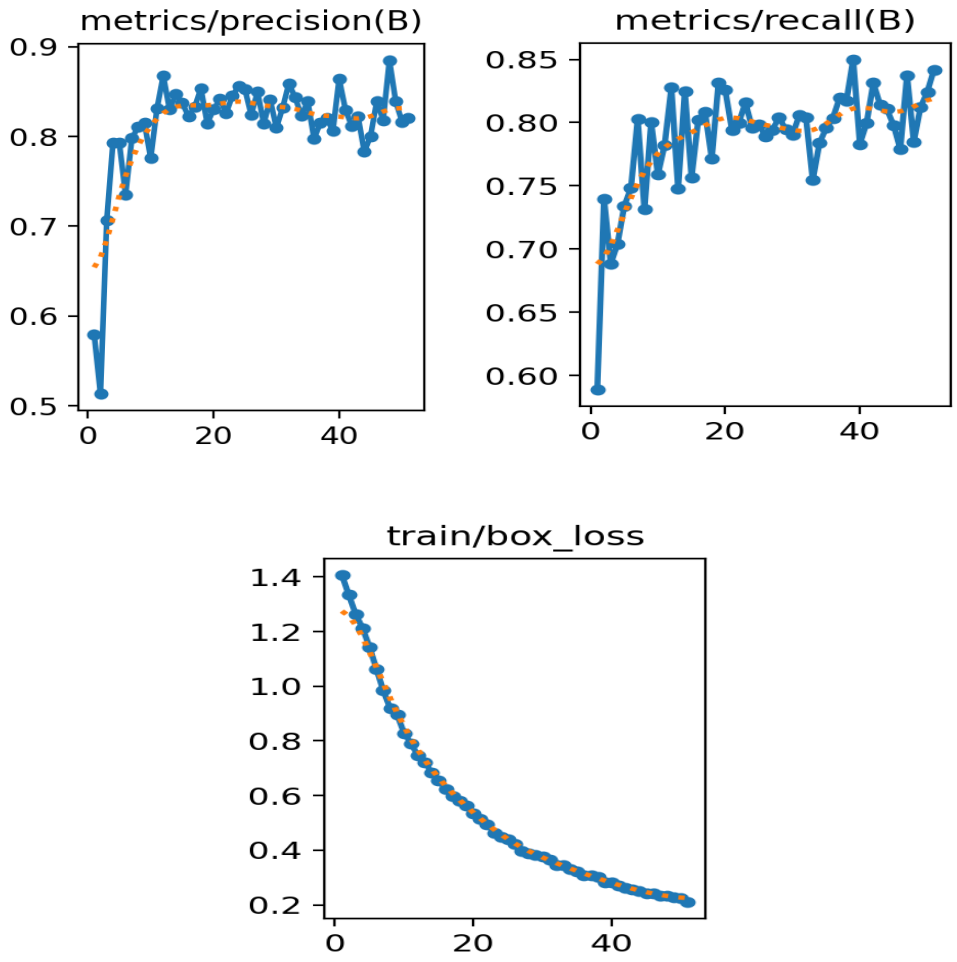


Figure 35 – Evolution of YOLOv11n Training Metrics on the Figshare Dataset.

Figure 35 shows the training progress of YOLO11n on the Figshare dataset. The curves show a good and consistent improvement across the epochs, indicating that the model progressively

learned and improved its detection performance during training. Figure 36 presents representative examples of glioma, pituitary tumor, and meningioma cases from the Figshare test set.

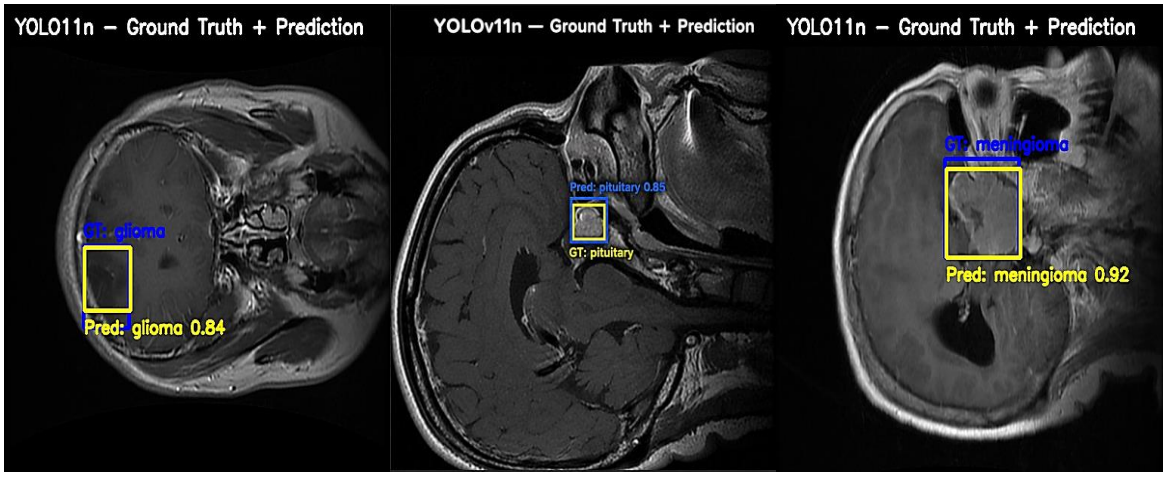


Figure 36 – Ground-Truth and YOLO11n Predictions for Representative Glioma, Pituitary Tumor, and Meningioma Cases from the Figshare Test Set.

In all three cases, the predicted bounding boxes closely overlap with the ground-truth annotations, indicating that YOLO11n localized the tumor regions accurately. The model also correctly classified the three tumors with confidence scores of 0.84 for glioma, 0.85 for pituitary tumor, and 0.92 for meningioma. These visual examples support the quantitative results and show that YOLO11n can effectively detect, classify, and localize different brain tumor types.

YOLOv26n

Now we move to the results of the YOLOv26n implementation on the Figshare dataset. Figure 37 presents the evolution of the main YOLOv26n training metrics on the Figshare dataset.

Precision and recall increased rapidly during the early epochs and then remained relatively stable, with some fluctuations throughout training. Precision stayed mostly above 0.80, while recall remained around 0.80 in the later epochs. At the same time, the box loss decreased steadily from about 1.3 to nearly 0.15, indicating a continuous improvement in bounding-box localization. Overall, the curves show that YOLO26n learned effectively from the Figshare dataset and reached stable detection performance toward the end of training.

Figure 38 presents representative examples of glioma, meningioma, and pituitary tumor cases from the Figshare test set.

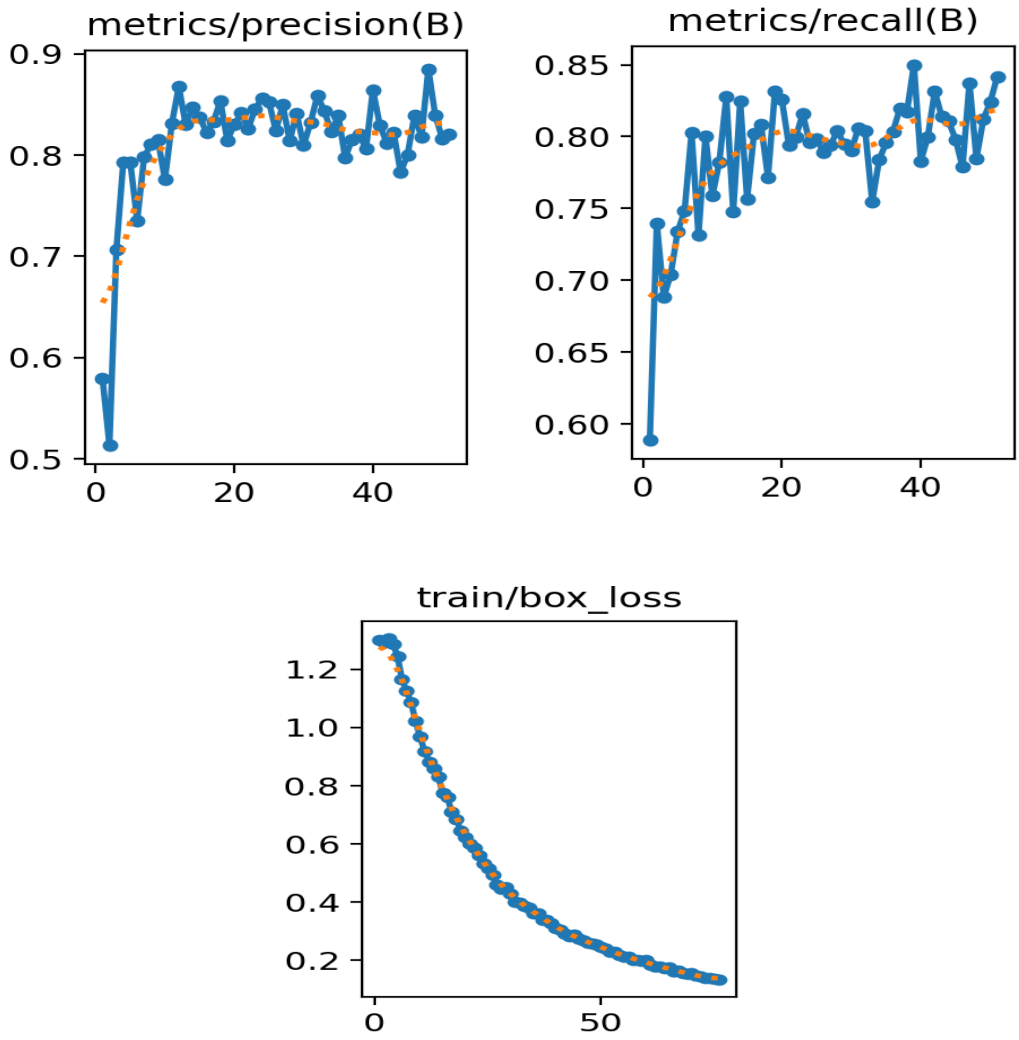


Figure 37– Evolution of YOLOv26n Training Metrics on the Figshare Dataset.

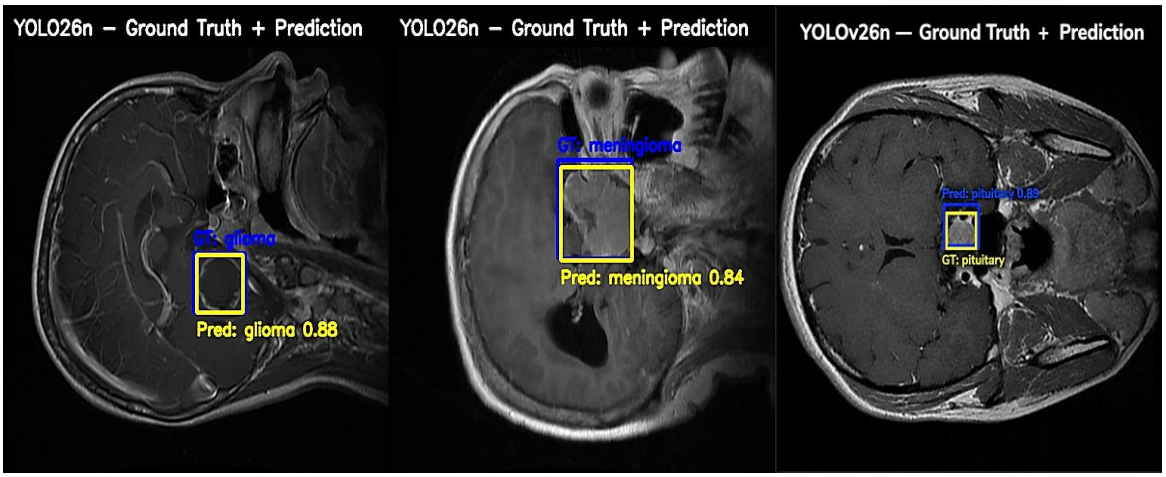


Figure 38 – Ground-Truth and YOLO26n Predictions for Representative Glioma, Pituitary Tumor, and Meningioma Cases from the Figshare Test Set.

In the three examples, the predicted bounding boxes show a close overlap with the ground-truth annotations, indicating that YOLO26n was able to localize the tumor regions effectively. The model also correctly classified the three tumor types, with high confidence scores for the presented cases. These visual results support the quantitative evaluation and show that YOLO26n can successfully detect, classify, and localize different brain tumor types.

Before concluding this section, Table 7 summarizes the overall performance of the three YOLO models on the Figshare test set.

Model	Precision	Recall	IoU
YOLOv8n	86.55%	87.06%	80.78%
YOLO11n	84.26%	82.37%	80.78%
YOLO26n	**89.44%**	**93.84%**	**81.42%**

Table 7– Overall performance comparison of YOLO models on the Figshare test set.

As shown in Table 7, the three YOLO models demonstrated robust detection and localization performance on the Figshare test set. YOLO26n achieved the highest scores across all evaluation metrics, with a precision of 89.44%, recall of 93.84%, and IoU of 81.42%. YOLOv8n also exhibited strong performance, whereas YOLO11n yielded slightly lower precision, re-

call. The comparable IoU values among the three models indicate that all versions effectively localized tumor regions, with YOLO26n delivering the most comprehensive performance.

These results also support the progression followed in this study. The initial CNN offered a lightweight solution but demonstrated limited and inconsistent localization performance, while VGG16 improved localization accuracy but required a substantially larger number of parameters. The YOLO Nano models achieved a better balance between model size and localization performance. Their robust results on both BUSI and the larger Figshare dataset show the advantage of using architectures specifically designed for object detection and support the choice of YOLO as the main localization approach in this study.

3.7 Conclusion

This chapter presented the preparation, implementation, and evaluation of CNN, VGG16, and lightweight YOLO models for medical image localization. The CNN provided a simple baseline but showed limited and inconsistent localization results, while VGG16 improved performance at the cost of a much larger number of parameters. This motivated the use of YOLOv8n, YOLO11n, and YOLO26n as more efficient object-detection approaches.

On both the BUSI and Figshare datasets, the YOLO models achieved strong detection and localization performance. YOLO26n obtained the strongest overall results, particularly in terms of Precision, Recall, mAP50, and IoU.

Overall, the experiments show that lightweight YOLO models provide a good balance between localization performance and model complexity, supporting their use as the main detection approach in this study.

Conclusion

In this study, we addressed the problem of object detection and localization in medical images using breast ultrasound and brain MRI datasets. Before conducting the experiments, we first established the theoretical foundations needed to understand the problem and the selected methods. In Chapter 1, we reviewed the main concepts of Artificial Intelligence, Machine Learning, Deep Learning, and convolutional neural networks. This provided the background required to understand how image features are learned and how CNN-based models can be applied to medical image analysis. In Chapter 2, we focused on object detection, including the principles of localization, bounding-box prediction, evaluation metrics, and the YOLO architecture. This theoretical study helped us understand the evolution of YOLO and the reasons behind its efficiency for real-time object detection.

Building on this background, the experimental work was carried out in Chapter 3. We first explored conventional CNN-based approaches, including a custom CNN and VGG16, to better understand the localization task and establish initial results. The CNN was lightweight but showed limited and inconsistent localization performance, while VGG16 provided better results at the cost of a much larger model. These first experiments helped us identify the need for an approach that could provide a better balance between localization performance and model complexity.

Based on these observations, we then evaluated and compared three lightweight YOLO architectures, YOLOv8n, YOLO11n, and YOLO26n. The results showed that the YOLO Nano models provided a more suitable balance between detection performance and model complexity, with YOLO26n achieving the strongest overall results in our experiments.

Through this work, we concluded that lightweight object detection models have strong potential for medical image analysis, particularly in the context of Embedded Systems. As future work, these models could be evaluated on larger and more diverse datasets and deployed on embedded or edge devices. Such implementations could contribute to real-time computer-assisted applications that support doctors in detecting and localizing suspicious regions. However, further clinical evaluation and validation would be necessary before their use in real medical practice.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in Proc. 27th Int. Conf. on Machine Learning (ICML), Haifa, Israel, 2010, pp. 807–814.

[4] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[5] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. 3rd Int. Conf. on Learning Representations (ICLR), San Diego, CA, USA, 2015.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779–788.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137–1149, 2017.

[8] R. Girshick, "Fast R-CNN," in Proc. IEEE Int. Conf. on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1440–1448.

[9] Ultralytics, "YOLOv8 documentation," Ultralytics GitHub repository, 2023. [Online]. Available: https://docs.ultralytics.com

[10] Ultralytics, "YOLO11 documentation," Ultralytics GitHub repository, 2024. [Online]. Available: https://docs.ultralytics.com

[11] Ultralytics, "YOLO26 documentation," Ultralytics GitHub repository, 2026. [Online]. Available: https://docs.ultralytics.com

[12] Kaggle Inc., "Kaggle Notebooks documentation," 2024. [Online]. Available: https://www.kaggle.com/docs/notebooks

[13] W. Al-Dhabyani, M. Gomaa, H. Khaled, and A. Fahmy, "Dataset of breast ultrasound images," Data in Brief, vol. 28, art. no. 104863, 2020.

[14] J. Cheng, J. Huang, C. Zhang, Y. Jiang, R. Wang, S. Wang, and Y. Cheng, "Enhanced Performance of Brain Tumor Classification via Tumor Region Augmentation and Partition," *PLoS ONE*, vol. 10, no. 10, 2015.

Abstract

This thesis focuses on the use and evaluation of lightweight object detection models for medical images. Two datasets, BUSI and Figshare, were prepared to evaluate three YOLO models: YOLOv8n, YOLO11n, and YOLO26n. CNN and VGG16 were also used for lesion localization on the BUSI dataset and compared with the YOLO models. The results show that the models can detect and localize lesions and tumors in both datasets. Among the YOLO models, YOLO26n obtained the best overall results. The lightweight models are also suitable for embedded systems applications.

Keywords: YOLO, object detection, medical images, deep learning, lesion localization, embedded systems.

Résumé

Ce mémoire porte sur la mise en œuvre et l'évaluation de modèles légers de détection d'objets pour l'analyse d'images médicales. Deux jeux de données, BUSI et Figshare, ont été préparés pour évaluer trois modèles YOLO : YOLOv8n, YOLO11n et YOLO26n. Les modèles CNN et VGG16 ont également été utilisés pour la localisation des lésions sur BUSI et comparés aux modèles YOLO. Les résultats montrent que les modèles étudiés peuvent détecter et localiser les lésions et les tumeurs dans les deux jeux de données. Parmi les modèles YOLO, YOLO26n a obtenu les meilleurs résultats globaux. Les modèles légers sont également adaptés aux applications liées aux systèmes embarqués.

Mots-clés : YOLO, détection d'objets, images médicales, apprentissage profond, localisation des lésions, systèmes embarqués.

الملخص

يركز هذا العمل على تنفيذ وتقييم نماذج خفيفة لاكتشاف الأجسام في الصور الطبية. تم إعداد مجموعتي بيانات، BUSI وFigshare، لتقييم ثلاثة نماذج YOLO وهي YOLOv8n و YOLO11nو YOLO26n. كما تم استخدام نموذجي CNN و VGG16لتحديد مواقع الآفات في مجموعة BUSI ومقارنتهما بنماذج YOLO. أظهرت النتائج أن النماذج المدروسة قادرة على اكتشاف وتحديد مواقع الآفات والأورام في مجموعتي البيانات. ومن بين نماذجYOLO ، حقق YOLO26n أفضل النتائج الإجمالية. كما أن النماذج الخفيفة مناسبة للتطبيقات المرتبطة بالأنظمة المدمجة.

الكلمات المفتاحية : اكتشاف الأجسام، الصور الطبية، التعلم العميق، تحديد مواقع الآفات، الأنظمة المدمجة