



PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF IBN KHALDOUN - TIARET

THESIS

Presented to :

FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE

In order to obtain the degree of :

MASTER

Speciality : Software Engineering

Presented by :

ABDERRAHMANE Younes

On the theme

Preprocessing and Semantic Segmentation of Medical Images of Lung Diseases using AI Techniques

Defended publicly on 12 / 06 / 2024 in Tiaret before the jury composed of :

Mr BAGHDADI Mohamed

MCA Tiaret University

President

Mr MEZZOUG Karim

MCA Tiaret University

Supervisor

Mr BOUDAA Boudjema

MCA Tiaret University

Examiner

2023-2024

Acknowledgments

First and foremost, I would like to thank Allah for giving me the will, health, patience, and strength to complete this humble work.

I am deeply grateful to my supervisor, **Professor MEZZOUG Karim**, whose expertise was invaluable. His insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would like to extend my thanks to **Professor BAGHDADI Mohamed** and **Professor BOUDAA Boudjema** for agreeing to evaluate my final year thesis.

In addition, I would like to thank my parents, for their wise counsel and sympathetic ear. They provided me with the tools I needed to choose the right direction and successfully complete my master thesis. They have always been there for me.

Finally, I could not have completed this master thesis without the support of my family and friends, who provided happy distractions that allowed me to rest my mind outside of my research.

Dedication

This thesis is dedicated to the pillars of my strength and inspiration.

To my parents, who have always believed in me and provided unwavering support and encouragement. Your sacrifices and endless love have been the foundation of my journey.

Without your guidance and patience, this achievement would not have been possible.

To my friends and colleagues, whose camaraderie and insightful discussions have fueled my passion and curiosity. Your encouragement and shared experiences have made this academic pursuit an enjoyable and enriching experience.

To my advisors and mentors, who have guided me with their wisdom and expertise. Your invaluable feedback and continuous support have been crucial in shaping this work and pushing me to reach new heights.

And finally, to all those who dream and strive for excellence, may this work be a testament to the power of perseverance and dedication.

Abstract

Automated lung region segmentation in computed tomography (CT) images is vital for diagnosing and treating lung diseases. This research leverages artificial intelligence (AI) techniques to enhance the preprocessing and semantic segmentation of lung disease images. Two approaches are proposed: one integrating YOLOv8 and U-Net, and the other combining YOLOv8 with Attention U-Net for improved segmentation precision. These approaches efficiently identify and segment lung regions from CT images, enhancing accuracy and robustness. Using a dataset of 140 CT exams, the approaches were trained and evaluated using metrics such as Dice coefficient, IoU, precision, recall, and computational cost. The first approach showed significant improvements, with a training Dice Coefficient reaching 0.9900 and a validation Dice Coefficient of 0.9788. The second approach also demonstrated robust performance, with validation precision reaching about 0.996 and recall about 0.987. Additionally, a web application called Lung Vision, which facilitates medical image management, is introduced. The results underscore the high accuracy and efficiency of the proposed approaches, offering innovative solutions for lung segmentation and practical applications in clinical settings, thereby advancing patient care through advanced AI technologies.

Keywords: preprocessing, semantic segmentation, object detection, lung regions, computed tomography (CT) images, early diagnosis, lung diseases, AI technologies, artificial intelligence (AI), medical images, YOLOv8, U-Net, Attention-UNet.

Table of Contents

Acknowledgments	I
Dedication	I
Abstract	II
Table of Contents	III
List of Figures	VII
List of Tables	X
Glossary of Abbreviations and Acronyms	XI
General Introduction	1
Chapter I Artificial Vision	4
I. 1 Introduction	5
I. 2 Digital Image	5
I. 3 Digital Image Characteristics	6
I. 3. 1 Pixel	6
I. 3. 2 Dimension (size)	6
I. 3. 3 Resolution	7
I. 3. 4 Histogram	7
I. 3. 5 Luminance	8
I. 3. 6 Contrast	8
I. 3. 7 Depth	8
I. 3. 8 Contours and textures	9
I. 3. 9 Noise	9
I. 3. 10 Image weight	9
I. 4 Digital Image Types	9
I. 5 Digital image formats	11
I. 5. 1 Raster (Bitmap) image	11
I. 5. 2 Vector image	12
I. 6 Digital Image Processing	13

I. 6. 1	Purpose of Image processing.....	13
I. 6. 2	Fundamental steps in Digital Image Processing:	13
I. 7	Medical Imaging	16
I. 7. 1	Medical images modalities.....	16
I. 7. 2	Medical Image File Formats	18
I. 8	Computer Vision	20
I. 8. 1	What is computer vision?.....	20
I. 8. 2	How does computer vision work.....	20
I. 8. 3	Image Processing Vs. Computer Vision	21
I. 9	Conclusion.....	22
Chapter II	Artificial Intelligence	23
II. 1	Introduction	24
II. 2	Artificial intelligence	24
II. 2. 1	Definition	24
II. 2. 2	Types of artificial intelligence.....	24
II. 2. 3	Artificial intelligence applications in medicine	26
II. 3	Machine learning	27
II. 3. 1	Definition	27
II. 3. 2	Machine learning approaches.....	27
II. 4	Neural networks.....	30
II. 4. 1	Biological Inspiration.....	30
II. 4. 2	Artificial Neural Networks (ANN)	31
II. 4. 3	Global architecture:	33
II. 4. 4	Activation Functions	33
II. 4. 5	Optimizers	37
II. 5	Deep learning.....	40
II. 5. 1	Definition	40
II. 5. 2	The difference between ML and DL.....	41
II. 6	Convolutional neural networks.....	42
II. 6. 1	The different layers of a CNN.....	43

II. 6. 2	CNN known models	45
II. 6. 3	Transfer learning	46
II. 7	Conclusion	47
Chapter III	State of The Art	48
III. 1	Introduction	49
III. 2	Related work.....	49
III. 2. 1	The work of Bhattacharjee, A., et al.	49
III. 2. 2	The work of Qinhuo, Hu., et al.....	50
III. 2. 3	The work of Khanna, A., et al.	51
III. 2. 4	Results	52
III. 2. 5	Conclusion.....	53
Chapter IV	Experiments and Realization.....	55
IV. 1	Introduction	56
IV. 2	Tools and Libraries	56
IV. 2. 1	Python.....	56
IV. 2. 2	Google Colaboratory	56
IV. 2. 3	Anaconda Navigator.....	57
IV. 2. 4	Jupyter Notebook	58
IV. 2. 5	TensorFlow.....	58
IV. 2. 6	Keras.....	58
IV. 2. 7	OpenCV.....	59
IV. 2. 8	Visual Studio Code.....	59
IV. 2. 9	Django	59
IV. 2. 10	PostgreSQL	60
IV. 2. 11	React.js.....	60
IV. 2. 12	Draw.io.....	61
IV. 3	Experiments	62
IV. 3. 1	Datasets	62
IV. 3. 2	Preprocessing	63
IV. 3. 3	Environment	63

IV. 3. 4	Direct Approaches.....	64
IV. 4	Proposed Approaches	66
IV. 4. 1	YOLOv8 Object Detection.....	66
IV. 4. 2	YOLOv8 with U-Net (Approach 1)	68
IV. 4. 3	YOLOv8 with Attention U-Net (Approach 2)	69
IV. 5	Results and discussion	70
IV. 5. 1	Performance metrics.....	70
IV. 5. 2	Results obtained for approach 1	72
IV. 5. 3	Results obtained for approach 2	75
IV. 5. 4	Results comparison table.....	78
IV. 5. 5	State of the art comparison table	80
IV. 6	Web Application Development	81
IV. 6. 1	Architecture.....	81
IV. 6. 2	Features	82
IV. 6. 3	UML Diagrams	83
IV. 6. 4	Graphical User Interface (GUI).....	86
IV. 7	Conclusion.....	91
	General Conclusion.....	92
	Bibliography	95

List of Figures

Figure I.1: A rectangular digital image of resolution 16 x 8.	5
Figure I.2: A typical greyscale image of resolution 512x512.....	6
Figure I.3: Example showing each of the image's dimensions and pixels.....	6
Figure I.4: Example showing a grayscale image with its histogram	7
Figure I.5: Example showing an RGB image with its histogram	7
Figure I.6: An example represents the difference between a low-contrast (a) and high-contrast image (b).....	8
Figure I.7: Example of binary image	10
Figure I.8: Example of grayscale image	10
Figure I.9: Example of color image	11
Figure I.10: Example Multi-Spectral Image	11
Figure I.11: An example of a bitmap and vector image	13
Figure I.12: The fundamental Steps of Digital Image Processing	14
Figure I.13 X-Ray Lungs Radiography	17
Figure I.14 Ultrasound Echography.....	17
Figure I.15 Computed Tomography Scan.....	18
Figure II.1: Various types of machine learning techniques	27
Figure II.2: Neuron	30
Figure II.3: Artificial neural networks architecture	31
Figure II.4: The artificial neuron	32
Figure II.5: Binary step activation function.....	33
Figure II.6: Linear activation function.....	34
Figure II.7: Sigmoid activation function.....	35
Figure II.8: Tanh activation function	35
Figure II.9: ReLu activation function	36
Figure II.10: Softmax activation function.....	37
Figure II.11: Gradient Descent Optimization Process.	38
Figure II.12: The learning rate is too small	38
Figure II.13: The learning rate is too high	38
Figure II.14: Stochastic Gradient Descent.....	39
Figure II.15: Stochastic Gradient Descent with (left) and without (right) momentum	40
Figure II.16: The relation between AI, machine learning and deep learning	41
Figure II.17: CNN architecture	43
Figure II.18: Convolution operation	44
Figure II.19: Pooling operation performed by choosing a 2x2 window	45

Figure II.20: LeNet	46
Figure II.21: Training from scratch vs Transfer Learning.....	47
Figure III.1 The proposed U-Net architecture for lung segmentation	50
Figure III.2 Figure Mask R-CNN flowchart.....	51
Figure III.3 Overview of the proposed Residual U-Net based CNN architecture for lung CT segmentation.....	52
Figure IV.1 Python logo	56
Figure IV.2 Google Colaboratory	57
Figure IV.3 Anaconda navigator	57
Figure IV.4 Jupyter notebook	58
Figure IV.5 TensorFlow logo	58
Figure IV.6 Keras logo	58
Figure IV.7 OpenCV logo	59
Figure IV.8 Visual Studio Code	59
Figure IV.9 Django logo.....	60
Figure IV.10 PostgreSQL pgAdmin4	60
Figure IV.11 React logo.....	60
Figure IV.12 Draw.io.....	61
Figure IV.13 Example of CT lung detection and segmentation by image morphology. Lung mask overlaid in blue. Rendered by 3D Slicer.	62
Figure IV.14 Comparison of Original and Preprocessed CT Images	63
Figure IV.15 Structure of the U-Net architecture proposed by Ronnenberger et al.....	65
Figure IV.16 Structure of the attention U-Net architecture proposed by Ozan Oktay et al.	66
Figure IV.17 Schematic of the Attention Gate	66
Figure IV.18 YOLOv8 Architecture.....	67
Figure IV.19 Approach 1 Architecture	69
Figure IV.20 Approach 2 Architecture	70
Figure IV.21 Approach 1 Training and Validation Dice Coefficient and IOU Over Epochs ..	72
Figure IV.22 Approach 1 Training and Validation Loss Over Epochs	73
Figure IV.23 Approach 1 Training and Validation Recall and Precision Over Epochs	74
Figure IV.24 Approach 2 Training and Validation Dice Coefficient and IOU Over Epochs ..	76
Figure IV.25 Approach 2 Training and Validation Loss Over Epochs	76
Figure IV.26 Approach 2 Training and Validation Recall and Precision Over Epochs	77
Figure IV.27 Architectural Overview of the Lung Vision Web Application	82
Figure IV.28 Use case diagram of the Lung Vision Web Application.....	84
Figure IV.29 Entity Relationship Diagram of the Lung Vision Web Application	85
Figure IV.30 Sequence diagram of the segmentation.....	86
Figure IV.31 User Authentication - Login Screen.....	87
Figure IV.32 Dashboard Screen.....	87

Figure IV.33 Patient Management Screens	88
Figure IV.34 Scan Management Screens.....	89
Figure IV.35 Segmentation Screen.....	90
Figure IV.36 Workspace Screen.....	91

List of Tables

Table I.1: Difference between Image Processing and Computer Vision.....	21
Table II.1: The key differences between Machine Learning and Deep Learning	41
Table III.1 Comparative Analysis of Recent Studies on Lung Segmentation Using AI Techniques.....	52
Table IV.1 Performance Metrics Results of Approach 1.....	75
Table IV.2 Performance Metrics Results of Approach 2.....	78
Table IV.3 Results Comparison Table.....	79
Table IV.4 Our Proposed Approaches Vs State of the Art.....	80

Glossary of Abbreviations and Acronyms

AI Artificial Intelligence

ML Machine Learning

DL Deep Learning

MLP Multi-layer Perceptrons

GD Gradient Descent

SVM Support Vector Machine

SGD Stochastic Gradient Descent

CNN Convolutional Neural Network

BNN Biological Neural Networks

ANN Artificial Neural Networks

RNN Recurrent Neural Network

GPU Graphical Processing Unite

KNN K-nearest Neighbors

ReLU Rectified Linear Unit

Adam Adaptive Moment Estimation

IOU Intersection Over Union

DC Dice Coefficient

Resnet Residual Neural Network

CT Computer Tomography

MRI Magnetic Resonance Imaging

NIFTI Neuroimaging Informatics Technology Initiative

ROI Region Of Interest

General Introduction

Introduction

The field of medical imaging has seen remarkable advancements driven by technology and computational progress. Among these modalities, computed tomography (CT) plays a vital role in diagnosing and monitoring lung diseases, including cancer and hereditary conditions. Accurate analysis of CT images is crucial for early detection, treatment planning, and prognosis. However, the intricate and variable nature of lung morphology poses significant challenges in automating image segmentation and analysis. Artificial intelligence (AI), encompassing machine learning (ML) and deep learning (DL), offers robust tools to address these challenges. By leveraging AI techniques in preprocessing and semantic segmentation, there is potential to enhance the accuracy and efficiency of lung image analysis, ultimately improving clinical decision-making. This thesis explores the integration of AI techniques for preprocessing and semantic segmentation of medical images, with a specific focus on lung diseases, aiming to develop robust methodologies for enhanced precision in image segmentation.

Problematic

The intricate nature of lung diseases demands accurate image analysis for effective diagnosis and treatment planning. However, existing methods for preprocessing and segmentation of medical images often lack the precision required for optimal clinical decision-making. This thesis aims to investigate and implement AI techniques for preprocessing and semantic segmentation of lung disease images, aiming to enhance accuracy and efficiency in diagnosis and prognosis.

Objectives

The primary objectives of this thesis are:

- To investigate and implement effective preprocessing techniques that enhance the quality and consistency of CT images for lung disease analysis.
- To develop advanced AI approaches for the semantic segmentation of lung regions from CT images, focusing on integrating state-of-the-art object detection and segmentation algorithms.
- To create a practical application, named Lung Vision that incorporates the developed AI models into a user-friendly platform for managing medical images, patients, and scans.

Plan

Structured into four main chapters, this thesis aims to enhance lung disease image analysis through AI techniques.

In Chapter I, foundational concepts of artificial vision are introduced, covering digital image characteristics, types, processing, medical imaging modalities, and computer vision.

Chapter II builds upon these principles, exploring the broader domain of artificial intelligence, addressing AI types, machine learning approaches, neural networks, including CNNs and transfer learning, and concludes with a summary of insights.

Chapter III conducts a detailed review of the current state of the art in lung image segmentation, critically examining recent advancements in AI-driven methods such as Modified U-Net, Mask R-CNN, and Residual U-Net. This review identifies research gaps, laying the groundwork for the development of novel methodologies presented in this thesis.

Finally, Chapter IV outlines the experimental setup, implementation, and results of the proposed methodologies. Introducing two primary approaches one combining YOLOv8 with U-Net and the other integrating YOLOv8 with Attention-UNet this chapter also elaborates on the development and functionality of the Lung Vision application, demonstrating its practical utility in clinical settings.

Chapter I Artificial Vision

I. 1 Introduction

Artificial vision, a key area in computer science and engineering, enables machines to interpret and understand visual information. Building on image processing techniques, it enhances images or extracts valuable data, supporting applications like autonomous vehicles, medical imaging, and industrial automation. This technology is essential for advancing machine interaction with the environment.

I. 2 Digital Image

A *digital image* is a discrete two-dimensional function $f(x, y)$, which has been quantized over its domain and range. Without loss of generality, it will be assumed that the image is rectangular, consisting of Y rows and X columns. The *resolution* of such an image is written as $X \times Y$. By convention, $f(0, 0)$ is taken to be the top left corner of the image, and $f(X - 1, Y - 1)$ the bottom right corner [1]. As shown in (Figure I.1).

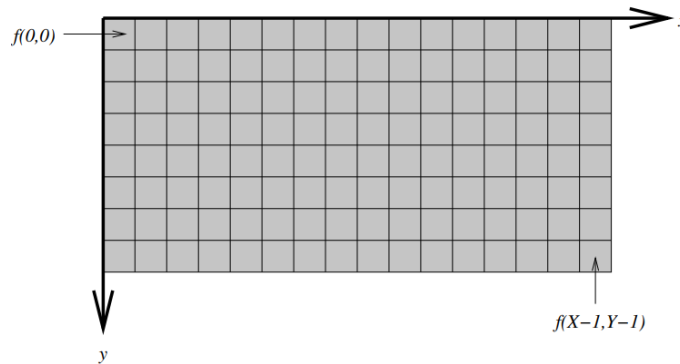


Figure I.1: A rectangular digital image of resolution 16 x 8.

Each distinct coordinate in an image is called a pixel, which is short for “picture element”.

The nature of the output of $f(x, y)$ for each pixel is dependent on the type of image. Most images are the result of measuring a specific physical phenomenon, such as light, heat, distance, or energy. The measurement could take any numerical form.

A *greyscale* image measures light intensity only. Each pixel is a scalar proportional to the brightness. The minimum brightness is called black, and the maximum brightness is called white. A typical example is given in (Figure I.2). A colour image measures the intensity and chrominance of light. Each colour pixel is a vector of colour components. Common colour spaces are RGB (red, green and blue), HSV (hue, saturation, value), and CMYK (cyan, magenta, yellow, black), which is used in the printing industry. Pixels in a range image measure the depth of distance to an object in the scene. Range data is commonly used in machine vision applications.



Figure I.2: A typical greyscale image of resolution 512x512.

For storage purposes, pixel values need to be quantized. The brightness in greyscale images is usually quantized to Z levels, so $(x, y) \in \{0, 1, \dots, Z - 1\}$. If Z has the form 2^L , the image is referred to as having L bits per pixel. Many common greyscale images use 8 bits per pixel, giving 256 distinct grey levels. This is a rough bound on the number of different intensities the human visual system is able to discern. For the same reasons, each component in a colour pixel is usually stored using 8 bits.

I. 3 Digital Image Characteristics

I. 3. 1 Pixel

The term pixel is an abbreviation of (Picture Elements). The pixel is the smallest constituent element of a digital image; these are numerical values representative of light intensities. These values are in a two-dimensional matrix through which the image is processed [2].

I. 3. 2 Dimension (size)

The dimension of a digital image represents the number of lines (denoted by N) of the matrix of pixels which represents the image multiplied by the number of columns (denoted by M) of this matrix. $N \times M$ (Figure I.3), This equation gives us the total number of pixels in an image [2].

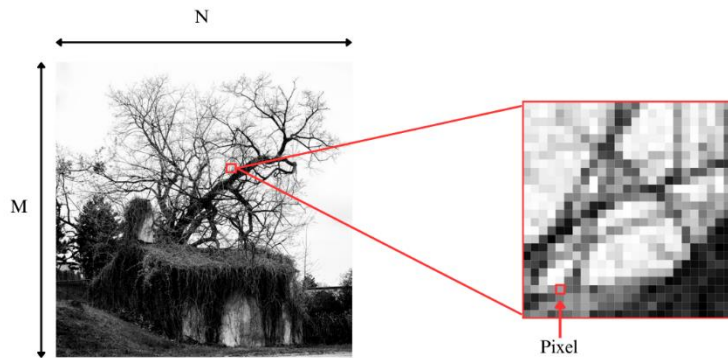


Figure I.3: Example showing each of the image's dimensions and pixels

I. 3. 3 Resolution

The resolution of a digital image is defined by the number of pixels per unit area in the image. The higher the number of pixels in the unit area, the higher the resolution of the image. represent the resolution by dpi (dots per inch) or PPI (Pixels Per Inch). one inch representing 2.54 cm [2].

I. 3. 4 Histogram

A histogram is the distribution of pixel intensities in a digital image. with another form, is the number of pixels that represents each light intensity in the image. The horizontal axis of the graph represents variations in light intensity, while the vertical axis represents the number of pixels in the image. (Figure I.4) represents an image with its histogram [2].

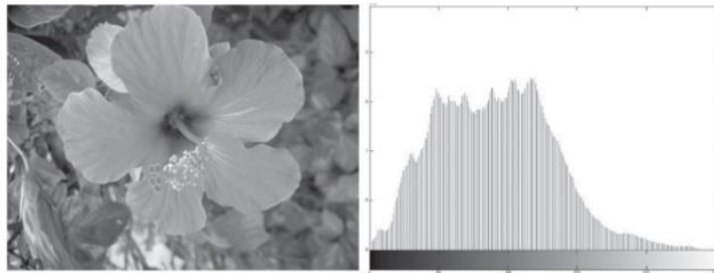


Figure I.4: Example showing a grayscale image with its histogram

RGB images are represented by four histograms (Figure I.5):

- A histogram representing the luminance distribution.
- A histogram representing the distribution of the values of the red components.
- A histogram representing the distribution of the values of the green components.
- A histogram representing the distribution of the values of the blue components.

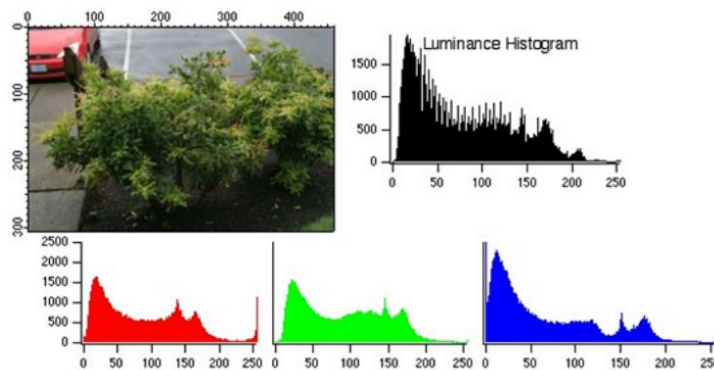


Figure I.5: Example showing an RGB image with its histogram

I. 3. 5 Luminance

Luminance is the degree of pixel brightness in the image. It is defined as being the quotient of the luminous intensity of a surface by the apparent area of this surface. A good luminance presents a bright image, a good contrast, and the absence of parasites [2].

I. 3. 6 Contrast

Contrast is the difference between dark regions and light regions in the image. The contrast C is defined by the ratio [3], and to calculate this last one this operation should be applied (see Eq I.1):

$$C = \frac{L_1 - L_2}{L_1 + L_2} \quad (\text{I.1})$$

With L_1 and L_2 are the degrees of the luminosity of two neighboring areas in the image.

A high-contrasted image presents a dynamic distribution of gray values over the entire range of possible values, with very clear whites and deep blacks. On the contrary, a low-contrast image has low dynamics (Figure I.6).



Figure I.6: An example represents the difference between a low-contrast (a) and high-contrast image (b)

I. 3. 7 Depth

Color Depth (bit depth) is measured by the number of bits per pixel (bpp). This value determines the number of distinct colors or grayscale levels in an image [2]. Common bpp values include:

- 2 bits/pixel = 2 colors (binary image)
- 8 bits/pixel = 256 colors (grayscale image)
- 16 bits/pixel = 65,536 colors (color image)
- 24 bits/pixel = 16,777,216 colors (color image)
- 32 bits/pixel = 4,294,967,296 colors (color image)

I. 3. 8 Contours and textures

The boundaries between objects in an image are known as contours. Contours represent the edges where there is a significant difference in pixel values (grayscale levels). Areas where the pixel values are similar are commonly referred to as textures. The process of identifying the points that separate two textures in an image is called edge detection [2].

I. 3. 9 Noise

In any digital image, the observed gray or color values have uncertainty. This uncertainty is due to the vagaries of counting the photons reaching each sensor.

The measured color values are disturbed because the sensors pick up parasitic photons and are subject to electrostatic fluctuations during charging and discharging.

When the sensor receives many photons from a well-lit scene, the noise is negligible compared to the actual photon flux. However, even in an image with sufficient exposure, dark pixels receive very few photons and are therefore "noisy". In short, noise is the sudden difference in the intensity of a pixel compared to neighboring pixels in the image [4]. Generally, two types of image noise that accumulate can be distinguished:

- Chrominance noise, which is the color component of noisy pixels: it is visible as random colored spots.
- Luminance noise, which is the light component of noisy pixels: it is visible as darker or lighter spots giving the image a grainy appearance

I. 3. 10 Image weight

The weight of the image determines according to the dimension and the depth, by counting the dimension ($N \times M$) of the image multiplied by its depth [4].

Example: For a 640x480 image in true colors (True colors):

Number of pixels (dimension): $640 \times 480 = 307200$ pixels

Weight of each pixel (depth): 24 bits = 3 bytes

The weight of the image is equal to $307200 \times 3 = 921600$ bytes.

I. 4 Digital Image Types

Digital images are typically classified into four primary categories:

a) Binary Images

One of the simplest digital image types are the binary images, they could be either white (1) or black (0), that is only take a single binary number for every element (Pixel) of image. Examples of this type are written. (Figure I.7) shows a binary image example [5].



Figure I.7: Example of binary image

b) Grayscale Images

Are monochrome images that contain only luminance levels and contain no gradients, and the value of every one of the elements in the image determines the grayscale (0-255), and each of its elements represents 8 bits/ pixel. In certain applications such as Medical imaging is often represented as 12 bit / pixel or 16 bit/pixel. This increase in intensity levels is useful only when a small area of the image has a high intensity. In this case, details that may be difficult to see without this increase levels of intensity [5]. (Figure I.8) shows a grayscale image example.



Figure I.8: Example of grayscale image

c) Color Images

These images are represented in a three bands monochromatic data format, each of which returns to the luminance in each of the red (R), green (G), and blue (B) colors using 8 bits per element, For each color pack bands, the color image is represented by (24bits) Total of the real colors in the picture is about 16 million equivalent $(2^8)^3$ [5]. (Figure I.9) shows color image.



Figure I.9: Example of color image

d) Multi-Spectral Images

Multi-spectral images contain information beyond the limits of visual perception, containing data collected in several packages from infrared, ultraviolet (UV), or x-ray (satellite) images. These images are collected in several packages ranging from spectral package, one to three of these packets fall within the Visible Spectrum range, while the other beams are located in invisible areas [5]. Figure (I.10) shows Multi-Spectral image.

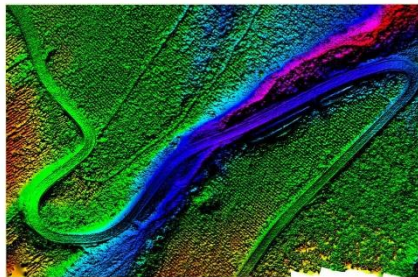


Figure I.10: Example Multi-Spectral Image

I. 5 Digital image formats

Generally, there are two main categories of digital images, bitmap images which are formed from a group of adjacent pixels, and vector images that represent by a set of geometric shapes. It will be explained in detail as follows [4]:

I. 5. 1 Raster (Bitmap) image

Bitmap or raster image is an image composed of little blocks of color called pixels organized in rows and columns (matrix). Each pixel is assigned a color code and a location, and this is how to form a picture. When zooming in on a bitmap image, the individual pixels become visible (Figure I.11), resulting in a loss of quality.

This is commonly referred to as resolution-dependent since the quality or sharpness of the image depends on the resolution. This type is found in the domain of image processing and analysis. Common bitmap image file types include

- **JPEG or JPG (Joint Photographic Experts Group):** this format, the most widely used on the Internet, allows images to be displayed in 16 million color modes. it can compress an image to 50 KB which takes 1MB, so it gives the smallest file size, with the least loss of quality.
- **Gif (Graphic Interchange Format):** this format uses RGB encoding, the GIF format takes all 256 colors out of 16 million colors.
- **BMP (Windows Bitmap):** this is the format is the native image format in the Microsoft Windows operating systems. It compressed the images and gives it with a good quality.
- **TIFF (Tag Image File Format):** a format of excellent quality, but it presents compatibility problems due to the multiplicity of versions. There is also a compressed version, which provides very compact files without noticeable loss of quality.

I. 5. 2 Vector image

The vector image is an image based on geometrical formulas (line segments, polygons, circles, rectangles, etc.), instead of pixels, to represent images. and This gives it the freedom to edit it resizing and changing colors without any losing resolution, making them ideal for icons, logos, and web-based imagery. When zooming in on this type of image, a pure image without distortion is seen (Figure I.11). This makes vector images resolution independent since the image quality is not affected by size or resolution settings.

To view this type of image on the computer screen, it must first be converted to bitmaps by changing the file type. This type of image is used in the domain of graphics and computer-aided design. Common vector image file types include:

- **SVG (Scalable Vector Graphics):** this format was developed to be used to compress or stretched images without losing the quality, and the element files are small.
- **ESP (Postscript / Encapsulated Postscript):** this is one of the best formats to export vector drawings, it encapsulates bitmaps and renders them in ESP format.
- **PDF (portable document format):** this format preserves the layout of a document and makes it small. Also, this format is easy to deal with it, It can be viewed, printed, or electronically transmitted by uploading, downloading, or attaching it to a message or email.
- **AI (Adobe Illustrator):** is the proprietary Adobe file type for vector images.

Other common vector image file types include PDF, EPS, and SVG. Vector files do not lose resolution when compressed because they are built on a graph-like formula that is infinitely expandable.

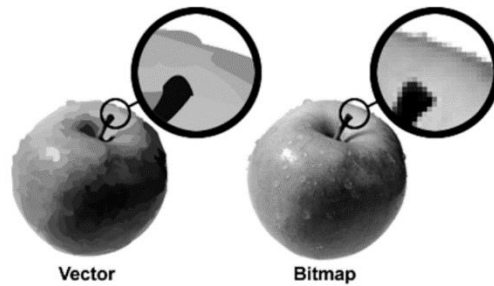


Figure I.11: An example of a bitmap and victor image

I. 6 Digital Image Processing

Image processing refers to computer-based imaging where human involvement in the visual loop is integral. In simpler terms, it entails the examination and manipulation of images by individuals. Key topics within this domain include image restoration, enhancement, and compression.

The continuum from image processing to computer vision lacks clear-cut boundaries. Within this spectrum, three types of computerized processes can be identified [6]:

- **Low-level processes:** These encompass primitive operations such as noise reduction, contrast enhancement, and sharpening.
- **Mid-level processes:** This category involves segmentation, which entails partitioning images into regions and objects.
- **High-level processes:** These processes focus on recognizing objects within images.

I. 6. 1 Purpose of Image processing

Image processing serves five main objectives, each aimed at distinct outcomes [7]:

- **Visualization:** Facilitates the observation of otherwise imperceptible objects.
- **Image sharpening and restoration:** Enhances image quality to produce clearer visuals.
- **Image retrieval:** Enables the search for specific images of interest.
- **Pattern measurement:** Quantifies various objects within an image.
- **Image recognition:** Identifies and distinguishes objects depicted in an image.

I. 6. 2 Fundamental steps in Digital Image Processing:

It is useful to categorize various image processing algorithms into broad subclasses to address different tasks and challenges. Oftentimes, it's beneficial to differentiate the nature of the task at hand. In essence, the steps involved in digital image processing can be illustrated as outlined below [6]. as shown in (Figure I.12).

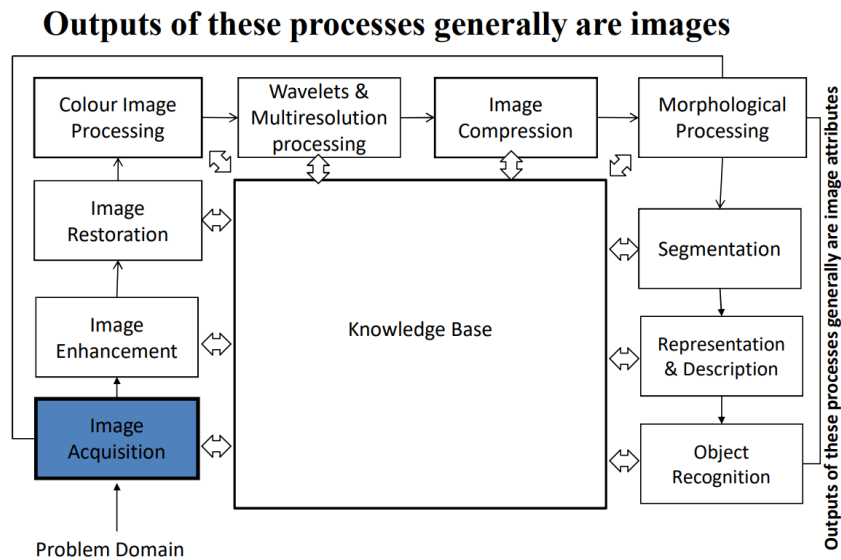


Figure I.12: The fundamental Steps of Digital Image Processing

a) Image Acquisition

Image acquisition refers to the initial step in the process of digital image processing, as depicted in (Figure I.12). This phase involves obtaining digital images, which may vary from receiving images already in digital format to undergoing preprocessing tasks like scaling.

b) Image Enhancement

Image enhancement is a fundamental aspect of digital image processing, aimed at refining and improving image quality. The primary goal of enhancement techniques is to reveal obscured details or emphasize specific features within an image. A common example of enhancement is adjusting the contrast of an image to enhance its visual appeal.

c) Image Restoration

Image restoration involves improving the appearance of an image through objective techniques based on mathematical or probabilistic models of image degradation. Unlike enhancement, which is subjective and relies on human preferences, restoration aims to objectively restore the image to its original state.

d) Color Image processing

Color image processing involves the manipulation and analysis of digital images with color content. This field has become increasingly important due to the widespread use of digital images, particularly on the Internet.

e) Wavelets and Multiresolution Processing

Wavelets form the basis for representing images at different levels of resolution. This concept is utilized in tasks such as image data compression and pyramidal representation, where images are progressively subdivided into smaller regions.

f) Image compression

Compression involves employing techniques to decrease the storage space needed to save an image or the bandwidth required to transmit it. Image compression is commonly encountered by computer users through file extensions like ".jpg", which utilize compression standards such as JPEG (Joint Photographic Experts Group).

g) Morphological Image processing

Morphological processing involves the utilization of tools to extract image components that aid in representing and describing shape characteristics.

h) Segmentation

Segmentation procedures involve dividing an image into its individual parts or objects. Autonomous segmentation, in particular, poses significant challenges in digital image processing. A robust segmentation procedure significantly enhances the likelihood of successfully solving imaging problems that necessitate the identification of individual objects. Conversely, unreliable or inconsistent segmentation algorithms typically lead to eventual failure. In general, the accuracy of segmentation directly influences the success of object recognition tasks.

i) Representation and Description

Representation and description entail the steps that typically follow the segmentation stage in image processing. This stage usually provides raw pixel data, representing either the boundary of a region (i.e., the pixels separating one image region from another) or all the points within the region itself. The decision of whether to represent the data as a boundary or a complete region depends on the focus of the analysis. Boundary representation is suitable for examining external shape characteristics like corners and inflections, while regional representation is preferred for analyzing internal properties such as texture or skeletal shape. In some cases, both representations are used together to provide complementary information. Selecting a representation is just one aspect of transforming raw data for subsequent computer processing. Additionally, a method must be defined for describing the data, highlighting features of interest.

Description, also known as feature selection, involves extracting attributes that provide quantitative information of interest or are fundamental for distinguishing between different classes of objects.

j) Object recognition

Object recognition refers to the process of assigning a label (e.g., "vehicle") to an object based on its descriptors. This process is integral to the field of digital image processing, as it enables the identification of individual objects within an image.

k) Knowledge Base

A knowledge base in the context of image processing serves as a repository of prior knowledge about the problem domain. This knowledge is encoded into the image processing system and can range from simple details about regions of interest within an image to more complex information such as lists of possible defects in a materials inspection problem or an image database containing high-resolution satellite images. The knowledge base guides the operation of processing modules and controls the interaction between these modules. It is represented as a separate component in image processing systems, distinct from the processing modules, and facilitates more efficient and accurate image analysis and interpretation.

I. 7 Medical Imaging

Digital image processing is used in medical imaging to enhance images, detect abnormalities, and aid in diagnosis. Medical imaging encompasses a wide array of imaging modalities, each offering unique insights into different aspects of the human body. From X-rays and computed tomography (CT) scans to magnetic resonance imaging (MRI) and ultrasound, enabling healthcare professionals to visualize anatomical structures and physiological processes with unprecedented detail and precision.

I. 7. 1 Medical images modalities

Medical imaging encompasses various modalities, each offering unique insights into the human body. From ionizing radiation to sound waves and magnetic fields, these modalities capture intricate details of organs and tissues, aiding in diagnosis and treatment planning.

a) X-Ray

X-rays employ ionizing radiation to capture images of the internal structure of the body by directing beams through tissues. The extent of absorption varies based on tissue density. Medical

imaging using X-ray radiation encompasses three primary types: conventional X-ray imaging, angiography, and fluoroscopy (see Figure I.13) [8].



Figure I.13 X-Ray Lungs Radiography

b) Ultrasound

Commonly referred to as medical sonography or ultrasonography, this imaging technique utilizes high-frequency sound waves to generate internal body images. By emitting sound waves into the body and capturing the returning echoes, the ultrasound machine translates them into visual images (see Figure I.14) [8].



Figure I.14 Ultrasound Echography

c) Computed tomography (CT)

Often termed as a CT scan, Computed Tomography (CT) is an imaging method that merges numerous X-ray images captured from diverse angles to generate highly detailed cross-sectional internal images (see Figure I.15) [8].

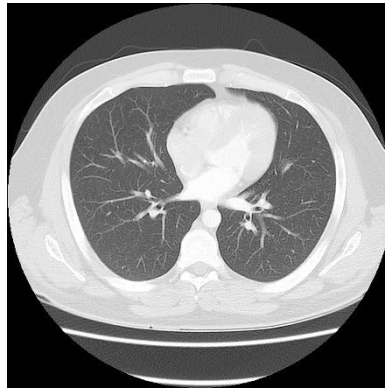


Figure I.15 Computed Tomography Scan

d) Magnetic resonance imaging (MRI)

Magnetic Resonance Imaging (MRI) employs radio waves and a magnetic field to produce intricate images of organs and tissues. This radiation type captures images of soft tissues while excluding bones, offering detailed insights into anatomical structures [8].

I. 7. 2 Medical Image File Formats

Medical image file formats play a crucial role in the storage and management of diagnostic imaging data. These formats are tailored to meet specific needs, whether for multidimensional data representation, neuroimaging informatics, or versatile data manipulation. Here, we delve into four prominent file formats: Analyze, NIFTI, Minc, and DICOM. Each format offers unique features and advantages, addressing distinct requirements in medical imaging.

a) Analyze

Analyze 7.5 is a file format developed in the late 1980s for medical imaging post-processing, primarily used by the Analyze software from the Mayo Clinic in Rochester, MN, USA. It served as the standard for over a decade, known for its ability to store multidimensional data, including 3D or 4D volumes (with temporal information). Each Analyze 7.5 volume comprises two binary files: an ".img" file containing voxel raw data and a ".hdr" file containing metadata like pixel dimensions, voxel size, and data type. The header, with a fixed size of 348 bytes, is structured in the C programming language and requires specialized software for reading and editing. Despite being considered "old," Analyze 7.5 remains widely used and supported by various processing software packages, viewers, and conversion utilities [9].

b) NIFTI

NIFTI (Neuroimaging Informatics Technology Initiative) is a file format introduced in the early 2000s by a committee at the National Institutes of Health. It was designed to address the limitations of the Analyze format while retaining its benefits, primarily for neuroimaging data. NIFTI enhances Analyze by utilizing previously unused fields in the header to store additional information, such as image orientation, thus mitigating ambiguities in brain studies. It also introduces support for data types not present in Analyze, like unsigned 16-bit. Typically saved as a single ".nii" file, NIFTI merges both the header and pixel data, with header sizes varying depending on storage configurations. It provides two methods for image volume orientation, enabling mapping to the scanner frame of reference and alignment to a standard coordinate system. Widely adopted in neuroimaging research, NIFTI has become the default format for many software packages and utilities, offering compatibility with various viewers and analysis tools [9].

c) Minc

The Minc file format, originating from the Montreal Neurological Institute (MNI) in 1992, was designed to offer versatility in medical imaging data storage. Initially based on NetCDF (Network Common Data Format), the first version, Minc1, provided a foundation for subsequent developments. To address limitations in handling large data files and incorporate new features, Minc transitioned to Hierarchical Data Format version 5 (HDF5), resulting in Minc2. This transition rendered Minc2 incompatible with Minc1. Primarily utilized by software tools from the MNI Brain Imaging Center, including viewers and processing software libraries, Minc format facilitates flexible data manipulation. Additionally, utilities facilitating conversion between Dicom, Nifti formats, and interconversion between Minc1 and Minc2 have been developed by the MNI Brain Imaging Center [9].

d) DICOM

DICOM, established by the American College of Radiology and the National Electric Manufacturers Association, emerged as the cornerstone of medical imaging departments, despite its 1993 inception. Renowned for its pivotal role in enhancing access, exchange, and usability of diagnostic medical images, DICOM transcends mere file format status to encompass a comprehensive network communication protocol. By merging pixel data seamlessly with procedural metadata, DICOM ensures that images retain intrinsic clinical meaning. The DICOM header encapsulates exhaustive details of the image acquisition protocol and patient information, rendering each image self-descriptive and facilitating software-driven replication of acquisition protocols. While DICOM exclusively stores pixel values as integers, it accommodates diverse metadata types, utilizing linear transformations to scale voxel values to real-world units. Additionally, DICOM supports compressed image data through encapsulation, offering

compatibility with a spectrum of compression schemes, including JPEG, RLE, JPEG-LS, and JPEG-2000, thereby ensuring interoperability and versatility across medical imaging systems [9].

I. 8 Computer Vision

I. 8. 1 What is computer vision?

Computer vision, a subset of artificial intelligence (AI), utilizes machine learning and neural networks to enable computers and systems to extract meaningful insights from digital images, videos, and visual inputs. It empowers them to provide recommendations or take actions upon detecting defects or anomalies. If AI grants computers the capacity to think, computer vision equips them with the ability to observe and comprehend visual data.

Comparable to human vision, computer vision operates similarly, although humans possess a significant advantage. Human sight benefits from a lifetime of context to distinguish objects, assess distances, detect motion, and identify abnormalities in images. In contrast, computer vision trains machines to perform these tasks, relying on cameras, data, and algorithms instead of human anatomical components like retinas, optic nerves, and a visual cortex. Despite this difference, computer vision can rapidly analyze thousands of products or processes per minute, detecting imperceptible defects or anomalies and thereby exceeding human capabilities [10].

I. 8. 2 How does computer vision work

Computer vision relies heavily on data to train its algorithms, which iteratively analyze the data until they can distinguish patterns and ultimately recognize images. For instance, training a computer to identify automobile tires requires feeding it extensive datasets of tire images and related items to learn the nuances and recognize tires, particularly those without defects [10].

Two primary technologies are instrumental in achieving this: deep learning, a subset of machine learning, and convolutional neural networks (CNNs). Machine learning employs algorithmic models that enable computers to autonomously understand the context of visual data. With sufficient data input, the computer learns to differentiate between various images without explicit programming.

A CNN aids machine learning or deep learning models in visual perception by segmenting images into pixels, which are then labeled or tagged. It utilizes these labels to perform convolutions, a mathematical operation combining two functions to produce a third, and subsequently makes predictions about the visual content. The neural network iteratively refines its predictions, assessing their accuracy until they align with reality, akin to human perception.

Similar to how humans perceive distant images, a CNN initially identifies sharp edges and basic shapes before refining its understanding through iterative prediction cycles. While CNNs are adept at understanding individual images, recurrent neural networks (RNNs) serve a similar function in video applications, helping computers comprehend the relationships between frames in a sequence.

I. 8. 3 Image Processing Vs. Computer Vision

a) Image processing

Image Processing involves enhancing images by adjusting various parameters and features. It is a subset of Computer Vision. In this field, transformations are applied to an input image, resulting in an output image. Common transformations include sharpening, smoothing, and stretching [11].

b) Computer vision:

Computer vision involves creating explicit, meaningful descriptions of physical objects based on their images. The output of computer vision includes descriptions, interpretations, or quantitative measurements of structures in the 3D scene. Techniques such as image processing and pattern recognition contribute to achieving these goals [11].

c) Difference between Image Processing and Computer Vision:

The difference between image processing and computer vision is summarized in the table (Table I.1) [11].

Table I.1: Difference between Image Processing and Computer Vision

Image Processing	Computer Vision
Image processing is mainly focused on processing the raw input images to enhance them or preparing them to do other tasks	Computer vision is focused on extracting information from the input images or videos to have a proper understanding of them to predict the visual input like human brain.
Image processing uses methods like Anisotropic diffusion, Hidden Markov models, Independent component analysis, Different Filtering etc.	Image processing is one of the methods that is used for computer vision along with other Machine learning techniques, CNN etc.
Image Processing is a subset of Computer Vision.	Computer Vision is a superset of Image Processing.
Examples of some Image Processing applications are- Rescaling image (Digital Zoom), Correcting illumination, Changing tones etc.	Examples of some Computer Vision applications are- Object detection, Face detection, Hand writing recognition etc.

I. 9 Conclusion

In this chapter, image processing within the context of computer vision has been explored, emphasizing its role in mimicking human vision and enabling automated recognition. Looking forward, the prospects of digital image processing appear promising, with potential applications spanning healthcare, education, defense, transportation, and urban development.

Chapter II Artificial Intelligence

II. 1 Introduction

Artificial Intelligence (AI) revolutionizes modern technology by enabling machines to perform tasks requiring human intelligence. This chapter explores foundational AI concepts, including Machine Learning (ML) and Artificial Neural Networks (ANNs), with a focus on Deep Learning (DL). It aims to offer insights into AI's theoretical foundations, practical applications, and evolving technologies.

II. 2 Artificial intelligence

II. 2. 1 Definition

Artificial Intelligence (AI) can be defined as the interdisciplinary science and engineering of developing systems and machines capable of performing tasks that would typically require human intelligence. These tasks include but are not limited to visual perception, speech recognition, decision-making, and language translation. Originating from the foundational perspectives of pioneers like Marvin Minsky, who described AI as "the science of machines that perform tasks requiring intelligence when executed by humans," and John McCarthy, who emphasized AI as "the science and engineering of making intelligent machines and computer programs," the field has evolved to incorporate a broad spectrum of technologies and methodologies. These include machine learning, natural language processing, deep learning, and cognitive computing, which enable machines to mimic and potentially exceed human cognitive functions [12].

II. 2. 2 Types of artificial intelligence

There are four types of artificial intelligence: reactive machines, limited memory, theory of mind and self-awareness.

a) **Reactive Machine AI :**

Reactive machines are AI systems with no memory and are designed to perform a very specific task. Since they can't recollect previous outcomes or decisions, they only work with presently available data. Reactive AI stems from statistical math and can analyze vast amounts of data to produce a seemingly intelligence output [12].

Example of Reactive Machine AI :

- **IBM Deep Blue:** IBM's chess-playing supercomputer AI beat chess grandmaster Garry Kasparov in the late 1990s by analyzing the pieces on the board and predicting the probable outcomes of each move.

b) **Limited Memory AI :**

Unlike Reactive Machine AI, this form of AI can recall past events and outcomes and monitor specific objects or situations over time. Limited Memory AI can use past- and present-moment data to decide on a course of action most likely to help achieve a desired outcome. However, while Limited Memory AI can use past data for a specific amount of time, it can't retain that data in a library of past experiences to use over a long-term period. As it's trained on more data over time, Limited Memory AI can improve in performance [12].

Examples of Limited Memory AI :

- **Generative AI:** Generative AI tools such as ChatGPT, Bard and DeepAI rely on limited memory AI capabilities to predict the next word, phrase or visual element within the content it's generating
- **Virtual assistants and chatbots:** Siri, Alexa, Google Assistant, Cortana and IBM Watson Assistant combine natural language processing (NLP) and Limited Memory AI to understand questions and requests, take appropriate actions and compose responses
- **Self-driving cars:** Autonomous vehicles use Limited Memory AI to understand the world around them in real-time and make informed decisions on when to apply speed, brake, make a turn, etc.

c) **Theory of Mind AI :**

Theory of Mind AI is a functional class of AI that falls underneath the General AI. Though an unrealized form of AI today, AI with Theory of Mind functionality would understand the thoughts and emotions of other entities. This understanding can affect how the AI interacts with those around them. In theory, this would allow the AI to simulate human-like relationships. Because Theory of Mind AI could infer human motives and reasoning, it would personalize its interactions with individuals based on their unique emotional needs and intentions. Theory of Mind AI would also be able to understand and contextualize artwork and essays, which today's generative AI tools are unable to do [12].

d) **Self-Aware AI**

Self-Aware AI is a kind of functional AI class for applications that would possess super AI capabilities. Like theory of mind AI, Self-Aware AI is strictly theoretical. If ever achieved, it would have the ability to understand its own internal conditions and traits along with human emotions and thoughts. It would also have its own set of emotions, needs and beliefs [12].

II. 2. 3 Artificial intelligence applications in medicine

AI has the potential to improve medical practice in a variety of ways, including accelerating the rate of research and assisting clinicians in making more informed decisions. Here are a few applications for artificial intelligence:

a) AI in disease detection and diagnosis :

Unlike humans, AI never needs to sleep. Machine learning models could be used to observe the vital signs of patients receiving critical care and alert clinicians if certain risk factors increase. While medical devices like heart monitors can track vital signs, AI can collect the data from those devices and look for more complex conditions, such as sepsis. One IBM client has developed a predictive AI model for premature babies that is 75% accurate in detecting severe sepsis [13].

b) Personalized disease treatment :

Precision medicine could become easier to support with virtual AI assistance. Because AI models can learn and retain preferences, AI has the potential to provide customized real-time recommendations to patients around the clock. Rather than having to repeat information with a new person each time, a healthcare system could offer patients around-the-clock access to an AI-powered virtual assistant that could answer questions based on the patient's medical history, preferences and personal needs [13].

c) AI in medical imaging :

AI is already playing a prominent role in medical imaging. Research has indicated that AI powered by artificial neural networks can be just as effective as human radiologists at detecting signs of breast cancer as well as other conditions. In addition to helping clinicians spot early signs of disease, AI can also help make the staggering number of medical images that clinicians have to keep track of more manageable by detecting vital pieces of a patient's history and presenting the relevant images to them [13].

d) Clinical trial efficiency :

A lot of time is spent during clinical trials assigning medical codes to patient outcomes and updating the relevant datasets. AI can help speed this process up by providing a quicker and more intelligent search for medical codes. Two IBM Watson Health clients recently found that with AI, they could reduce their number of medical code searches by more than 70% [13].

e) **Accelerated drug development :**

Drug discovery is often one of the longest and most costly parts of drug development. AI could help reduce the costs of developing new medicines in primarily two ways: creating better drug designs and finding promising new drug combinations. With AI, many of the big data challenges facing the life sciences industry could be overcome [13].

II. 3 Machine learning

II. 3. 1 Definition

Machine learning, as elucidated by various scholars including Arthur Samuel, Tom Mitchell, and Ethem Alpaydin, is a discipline within computer science aimed at empowering computers to acquire knowledge and improve task performance through experience rather than explicit programming. It involves training computer programs to optimize performance metrics by leveraging example data or past encounters, thereby enabling them to intelligently execute tasks beyond conventional numerical computations [14].

II. 3. 2 Machine learning approaches

Machine Learning algorithms are mainly divided into four categories: Supervised learning, Unsupervised learning, Semi-supervised learning, and Reinforcement learning [15], as shown in (Figure II.1).

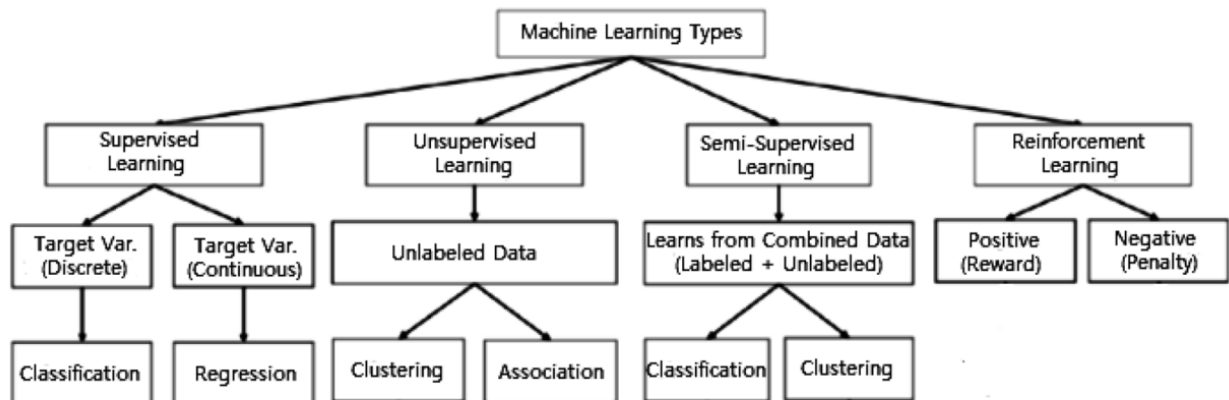


Figure II.1: Various types of machine learning techniques

a) Supervised machine learning

The primary approach in practical machine learning is supervised learning. In supervised learning, there are input variables (X) and an output variable (Y), and an algorithm is used to learn the mapping function from the input to the output (see Eq II.1).

$$Y = f(X) \tag{II.1}$$

The objective is to approximate the mapping function so well that when new input data (X) is available, the output variables (Y) can be predicted for that data. It's termed supervised learning because the algorithm learns from a training dataset under the guidance of a teacher. The correct answers are known, and the algorithm iteratively makes predictions on the training data, receiving corrections from the teacher. Learning ceases when the algorithm reaches an acceptable level of performance. Supervised learning problems can be categorized into regression and classification problems [16].

- **Classification:** In classification problems, the output variable represents categories, such as red or blue, or disease and no disease.
- **Regression:** Regression problems involve output variables that are real values, such as dollars or weight.

Common problems derived from classification and regression include recommendation and time series prediction, respectively. Some popular supervised machine learning algorithms include:

- Linear regression for regression problems.
- Random forest for classification and regression problems.
- Support vector machines for classification problems.

b) Unsupervised machine learning

Unsupervised learning involves having only input data (X) without corresponding output variables. The objective of unsupervised learning is to model the underlying structure or distribution in the data to gain insights into the data [16].

These methods are termed unsupervised learning because, unlike supervised learning, there are no correct answers or teachers to guide the algorithms. Algorithms are left to their own devices to uncover and present the meaningful structure in the data. Unsupervised learning problems can be categorized into clustering and association problems.

- **Clustering:** Clustering involves identifying inherent groupings in the data, such as grouping customers by their purchasing behavior.
- **Association:** Association rule learning entails discovering rules that describe significant portions of the data, such as identifying patterns like people who buy A also tend to buy B.

Some popular unsupervised learning algorithms include:

- k-means for clustering problems.
- Apriori algorithm for association rule learning problems.

c) **Semi-supervised learning**

Semi-supervised learning addresses scenarios where there's a large volume of input data (X) with only some data labeled (Y), falling between the realms of supervised and unsupervised learning. A classic example is a photo archive where only a fraction of images are labeled (e.g., dog, cat, person), while the majority remain unlabeled. Many real-world machine learning problems fall into this category due to the expense or time required for labeling, often necessitating access to domain experts. Conversely, collecting and storing unlabeled data is relatively inexpensive and straightforward [16].

In semi-supervised learning, unsupervised techniques can be employed to unveil and understand the structure within the input variables. Additionally, supervised learning techniques can be utilized to generate predictions for the unlabeled data, leveraging these predictions as training data for the supervised learning algorithm. This trained model can then be applied to make predictions on new, unseen data.

d) **Reinforcement learning**

Reinforcement learning (RL) constitutes a subset of machine learning, focusing on how intelligent agents can acquire decision-making abilities and take actions within an environment to optimize cumulative rewards. It draws inspiration from the trial-and-error learning process observed in humans and animals [17].

In RL, an agent interacts with its environment, learning to make decisions based on feedback received in the form of rewards or penalties. By taking actions within the environment, the agent receives a reward signal indicating its performance. The objective is for the agent to learn a policy -a mapping from states to actions- that maximizes the anticipated cumulative reward over time.

RL entails a delicate balance between exploration and exploitation. Initially, the agent explores the environment through random or exploratory actions to understand various outcomes. As it

accumulates knowledge, the agent transitions to exploiting its learned policy, prioritizing actions expected to yield higher rewards.

II. 4 Neural networks

By inspiration, a Neural Network (NN) is simulated from the human brain. it has somewhat an ability of distinguishing between things.

II. 4. 1 Biological Inspiration

The brain is mainly constituted by neurons to realize signal exchange and information processing. It is estimated that there are nearly 100 billion neurons in the human central nervous system. The neuron can receive, integrate, conduct and output the excitement. Structurally, it can be divided into three parts: dendrite, cell body and axon [18]. As shown in the image (Figure II.2) [19].

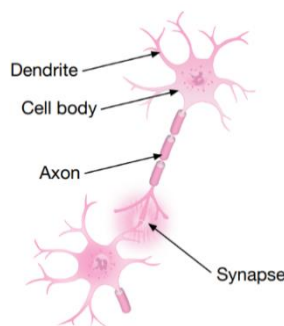


Figure II.2: Neuron

- **Dendrites** are composed of several thin extensions that form the dendritic tree (Figure II.2). The fundamental purpose of dendrites is to acquire, continuously, stimuli from several other neurons (connectors) or from the external environment, which is the case of some neurons in contact with the environment (also called sensory neurons) [20].
- **The cell body** is responsible for processing all the information that comes from the dendrites, to produce an activation potential that indicates if the neuron can trigger an electric impulse along its axon [20].
- **The axon** is composed of a single extension whose mission is to guide the electrical impulses to other connecting neurons, or to neurons directly connected to the muscular tissue (efferent neurons). The axon termination is also composed of branches called synaptic terminals [20].
- **The synapses** are the connections which enable the transfer of electric axon impulses from a particular neuron to dendrites of other neurons, as illustrated in (Figure II.2) It is important to note that there is no physical contact between the neurons forming the synaptic junction, so the

neurotransmitter elements released on the junction are in charge of weighting the transmission from one neuron to another [20].

II. 4. 2 Artificial Neural Networks (ANN)

The basic concept of Artificial Neural Networks (ANNs) is partially inspired by how the human brain functions. (Figure II.3) shows artificial neural networks architecture. Neural networks are multi layers networks that consist of a single input layer, one or multi hidden layers and one output layers. The input to neural networks is a set of input values. The goal of neural networks is to predict and classify those values into predefined categories [18].

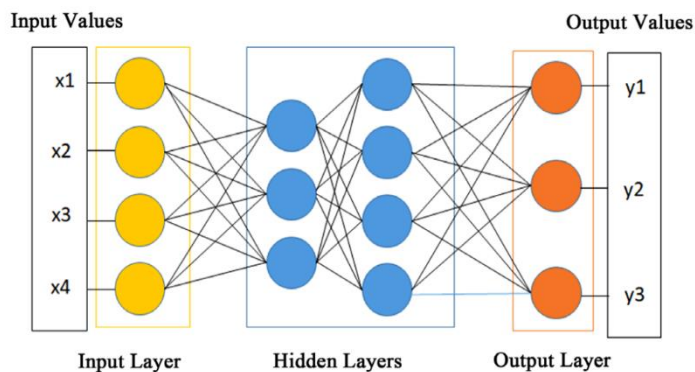


Figure II.3: Artificial neural networks architecture

The computational components or processing units, called artificial neurons, are simplified models of biological neurons. These models were inspired by the analysis of how a cell membrane of a neuron generates and propagates electrical impulses (Hodgkin and Huxley 1952).

The artificial neurons used in artificial neural networks are nonlinear, usually providing continuous outputs, and performing simple functions, such as gathering signals available on their inputs, assembling them according to their operational functions, and producing a response considering their innate activation functions [20].

Each neuron from a network can be implemented as shown in (Figure II.4) [19]. The multiple input signals coming from the external environment (application) are represented by the set $\{x_1, x_2, x_3, \dots, x_n\}$, analogous to the external electrical impulses gathered by the dendrites in the biological neuron.

The weighing carried out by the synaptic junctions of the network are implemented on the artificial neuron as a set of synaptic weights $\{w_1, w_2, w_3, \dots, w_n\}$.

Analogously, the relevance of each of the $\{x_i\}$ neuron inputs is calculated by multiplying them by their corresponding synaptic weight $\{w_i\}$, thus weighting all the external information arriving

to the neuron. Therefore, it is possible to verify that the output of the artificial cellular body, denoted by Z , is the weighted sum of its inputs.

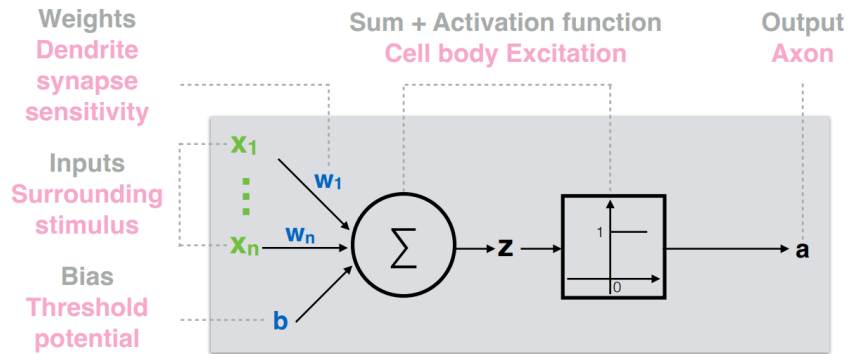


Figure II.4: The artificial neuron

Considering (Figure II.4), it is possible to see that the artificial neuron is composed of seven basic elements, namely:

- **Input signals** ($x_1, x_2, x_3, \dots, x_n$) are the signals or samples coming from the external environment and representing the values assumed by the variables of a particular application. The input signals are usually normalized in order to enhance the computational efficiency of learning algorithms.
- **Synaptic weights** ($w_1, w_2, w_3, \dots, w_n$) are the values used to weight each one of the input variables, which enables the quantification of their relevance with respect to the functionality of the neuron.
- **Linear aggregator** (Σ) gathers all input signals weighted by the synaptic weights to produce an activation voltage.
- **Activation threshold or bias** (b) is a variable used to specify the proper threshold that the result produced by the linear aggregator should have to generate a trigger value toward the neuron output.
- **Activation potential** (Z) is the result produced by the difference between the linear aggregator and the activation threshold. If this value is positive, i.e. if $u \geq b$, then the neuron produces an excitatory potential; otherwise, it will be inhibitory.
- **Activation function** whose goal is limiting the neuron output within a reasonable range of values, assumed by its own functional image.
- **Output signal** (a) consists on the final value produced by the neuron given a particular set of input signals, and can also be used as input for other sequentially interconnected neurons.

The two following expressions synthesize the result produced by the artificial neuron (Eq II.2, II.3):

$$u = \sum_{i=1}^n w_i \cdot x_i - \theta \quad (\text{II.2})$$

$$a = g(u) \quad (\text{II.3})$$

II. 4. 3 Global architecture:

Neural networks are organized in various layers [21]:

- **Input layer:** the input layer neurons receive the information supposed to explain the problem to be analyzed;
- **Hidden layer:** the hidden layer is an intermediate layer allowing neural networks to model nonlinear phenomena. This said to be “hidden” because there is no direct contact with the outside world. The outputs of each hidden layer are the inputs of the units of the following layer;
- **Output layer:** the output layer is the last layer of the network; it produces the result, the prediction.

II. 4. 4 Activation Functions

Activation functions decide whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. They are differentiable operators for transforming input signals to outputs, while most of them add nonlinearity [22]. Some of the popular activation functions are described below along with their other characteristics.

a) Binary Step Function:

A binary step function is a threshold-based activation function. If the input value is above or below a certain threshold [23], the neuron is activated and sends exactly the same signal to the next layer (see Figure II.5) [24].

Function (see Eq II.4):

$$f(z) = 1 \text{ if } z > 0 \text{ else } 0 \text{ if } z < 0 \quad (\text{II.4})$$

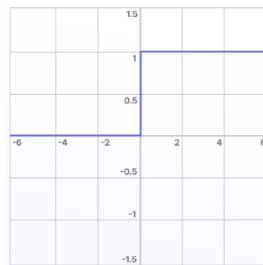


Figure II.5: Binary step activation function

Derivative:

The gradient of the step function is zero, which causes a hindrance in the back propagation process. That is, the derivative of $f(z)$ with respect to x , it comes out to be 0.

b) Linear Activation Functions:

It takes the inputs, multiplied by the weights for each neuron, and creates an output signal proportional to the input. In one sense, a linear function is better than a step function because it allows multiple outputs, not just yes and no (see Figure II.6) [24].

Function (see Eq II.5):

$$f(z) = a * z \tag{II.5}$$

‘a’ in this case can be any constant value.

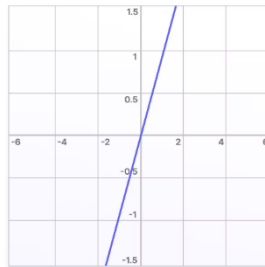


Figure II.6: Linear activation function

Derivative (see Eq II.6):

$$f'(z) = a \tag{II.6}$$

c) Non-Linear Activation Functions:

- **Sigmoid**

Sigmoid takes a real value as input and outputs another value between 0 and 1. The larger the input (more positive), the closer the output value will be to 1, whereas the smaller the input (more negative), the closer the output will be to 0 (see Figure II.7) [24].

Function (see Eq II.7):

$$f(z) = \frac{1}{1+e^{-z}} \tag{II.7}$$

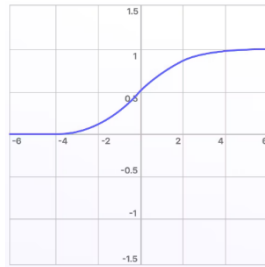


Figure II.7: Sigmoid activation function

Derivative (see Eq II.8):

$$f'(z) = \text{sigmoid}(z) * (1 - \text{sigmoid}(z)) \quad (\text{II.8})$$

- **Tanh (Hyperbolic Tangent)**

The tanh function is very similar to the sigmoid function. The only difference is that it is symmetric around the origin. The range of values in this case is from -1 to 1. Thus the inputs to the next layers will not always be of the same sign (see Figure II.8) [24]. The *tanh* function is defined as (see Eq II.9)

Function:

$$f(z) = \text{tanh}(z) = \frac{2}{1+e^{-2z}} - 1 \quad (\text{II.9})$$

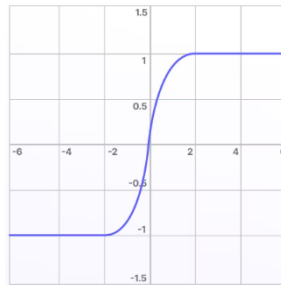


Figure II.8: Tanh activation function

Derivative (see Eq II.10):

$$\text{tanh}'(z) = 1 - \text{tanh}(z)^2 \quad (\text{II.10})$$

- **ReLU (Rectified Linear unit)**

A recent invention which stands for Rectified Linear Units. The formula is deceptively simple: $\max(0, z)$. Despite its name and appearance, it is not linear and provides the same benefits as Sigmoid but with better performance (see Figure II.9) [24].

Function (see Eq II.11):

$$f(z) = \max(0, z) \quad (\text{II.11})$$

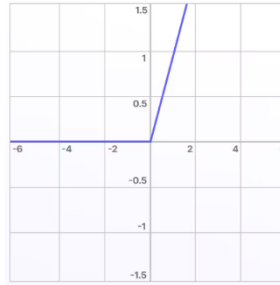


Figure II.9: ReLu activation function

For the negative input values, the result is zero, that means the neuron does not get activated. Since only a certain number of neurons are activated, the ReLU function is far more computationally efficient when compared to the sigmoid and tanh function.

Derivative (see Eq II.12):

$$\begin{aligned} f'(z) &= 1, \quad z \geq 0 \\ &= 0, \quad z < 0 \end{aligned} \quad (\text{II.12})$$

- **Softmax**

The softmax function is a function that turns a vector of K real values into a vector of K real values that sum to 1. The input values can be positive, negative, zero, or greater than one, but the softmax transforms them into values between 0 and 1, so that they can be interpreted as probabilities. If one of the inputs is small or negative, the softmax turns it into a small probability, and if an input is large, then it turns it into a large probability, but it will always remain between 0 and 1 (see Figure II.10) [24].

Function (see Eq II.13):

$$\frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}} \quad (\text{II.13})$$

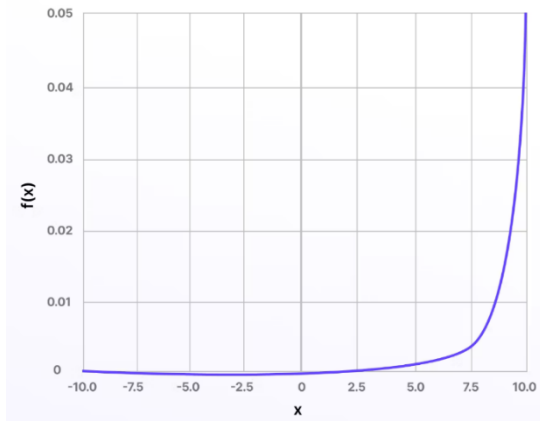


Figure II.10: Softmax activation function

II. 4. 5 Optimizers

Optimizers are algorithms or methods used to minimize an error function (loss function) or to maximize the efficiency of production. Optimizers are mathematical functions, which are dependent on model's learnable parameters (i.e. Weights & Biases). Optimizers help to know how to change weights and learning rate of neural network to reduce the losses [25].

a) Gradient Descent

Gradient descent is an optimization algorithm based on a convex function and tweaks its parameters iteratively to minimize a given function to its local minimum. Gradient Descent iteratively reduces a loss function by moving in the direction opposite to that of steepest ascent. It is dependent on the derivatives of the loss function for finding minima. Uses the data of the entire training set to calculate the gradient of the cost function to the parameters, which requires large amount of memory and slows down the process (see Figure II.11) [25].

The gradient descent calculates the slope of the landscape, which is the derivative of the function at this point with respect to the weights, as shown in Eq. (II.14).

$$w = w - lr \cdot \nabla_w L(w) \quad (\text{II.14})$$

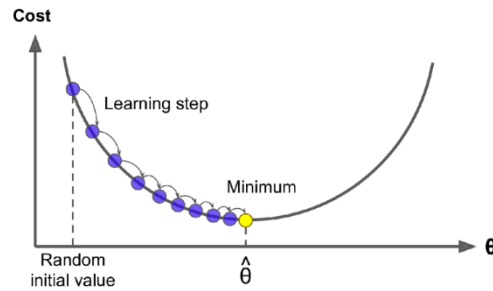


Figure II.11: Gradient Descent Optimization Process.

An important parameter in Gradient Descent is the size of the steps, determined by the learning rate hyperparameter. If the learning rate is too small, then the algorithm will have to go through many iterations to converge, which will take a long time (Figure II.12) [26].

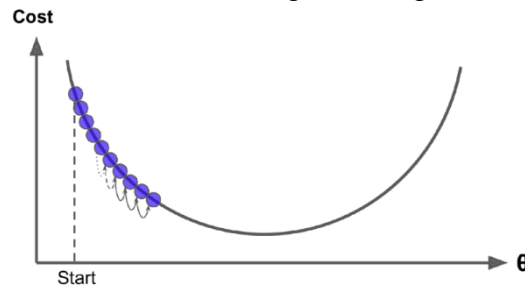


Figure II.12: The learning rate is too small

Alternatively, if the learning rate is too high, then the algorithm might leap across the minimum and land on the other side, potentially at a higher point than before. This could cause the algorithm to diverge, with values increasing progressively, and it might fail to find an optimal solution (see Figure II.13) [26].

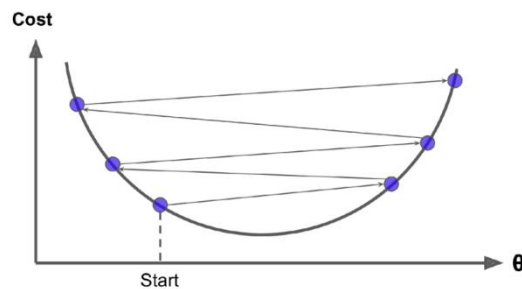


Figure II.13: The learning rate is too high

b) Stochastic Gradient Descent (SGD)

SGD is a basic algorithm and widely used in ML algorithms. Instead of calculating the gradient over all training examples and updating the weights (Figure II.14) [26], the SGD updates the weights of each training example x_i, y_i , as shown in Eq. (II.14) [27].

$$w = w - lr \cdot \nabla_w L(x_i, y_i, W) \quad (\text{II.15})$$

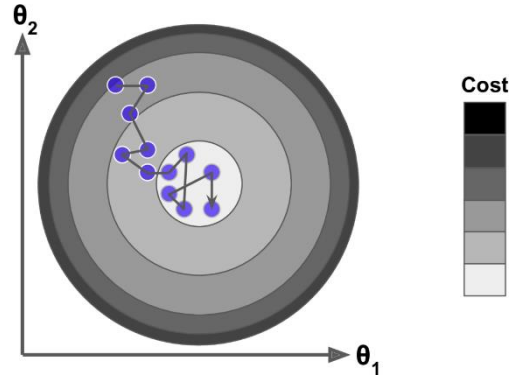


Figure II.14: Stochastic Gradient Descent

c) Mini-Batch Gradient Descent

It is a combination of the concepts of SGD and batch gradient descent. It simply splits the training dataset into small batches and performs an update for each of those batches. This creates a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It can reduce the variance when the parameters are updated, and the convergence is more stable. The Mini-Batch Gradient Descent updates the weights, as shown in Eq. (II.16) [27].

$$w = w - lr \cdot \nabla_w L(x^{(i:i+n)}; y^{(i:i+n)}, W) \quad (\text{II.16})$$

d) Stochastic Gradient Descent with Momentum:

In this approach, a momentum term is added to the regular SGD to overcome the limitations of the gradient descent algorithm, i.e., a gradient descent with a momentum. By using the principle of momentum from physics, the SGD is forced to continue moving in the same direction as those in the previous time steps (see Figure II.15) [27]. This momentum is accomplished by introducing two new variables, namely, velocity and friction, as given by Eqs. (II.17) and (II.18), respectively.

$$vt + 1 = pvt + \nabla_w L(x, w) \quad (\text{II.17})$$

$$w = w - lr \cdot vt + 1 \quad (\text{II.18})$$

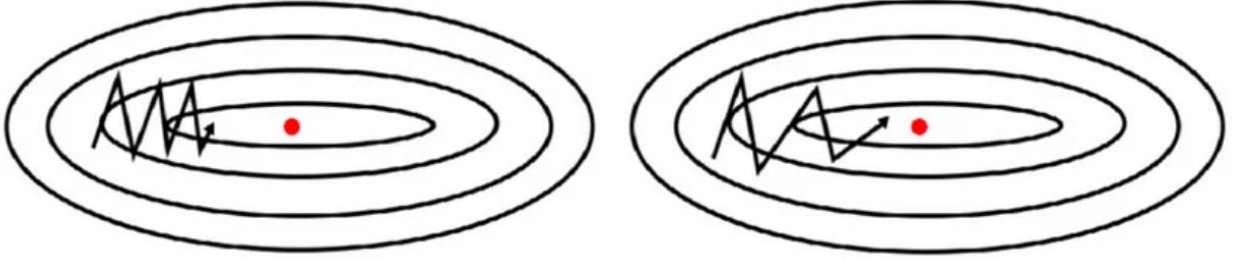


Figure II.15: Stochastic Gradient Descent with (left) and without (right) momentum

e) RMS-Prop (Root Mean Square Propagation)

The sizes of gradients vary by weight and change over time, hence the difficulty in selecting a single global LR. This aspect is addressed by RMSProp by retaining a moving average of the squared gradient and altering the weight updates by this magnitude [27]. The gradient updates are elaborated as shown in Eqs. (II.19) and (II.20).

$$v_t = \delta_{vt-1} + (1 - \delta)(\nabla_w L(x, y, w_t))^2 \quad (\text{II.19})$$

$$w = w - lr \cdot \frac{1}{\sqrt{v_t + \epsilon}} \odot (\nabla_w L(x, y, w_t))^2 \quad (\text{II.20})$$

f) Adam (Adaptive Moment Estimation)

Adam is a first-order gradient-based optimization technique for stochastic objective functions based on adaptive lower-order moment estimates. Instead of using the usual SGD approach, Adam is used to iteratively update the network weights depending on the training data [27]. Adam stems from evolutionary moment calculation and algorithm features as shown in Eqs. (II.21), (II.22), and (II.23).

$$m_t = \delta_{m_t} + (1 - \delta_1)(\nabla_w L(x, y, w_t)) \quad (\text{II.21})$$

$$v_t = \delta_{2vt-1} + (1 - \delta_2)(\nabla_w L(x, y, w_t))^2 \quad (\text{II.22})$$

$$w = w - lr \cdot \frac{m_t}{\sqrt{v_t + \epsilon}} \quad (\text{II.23})$$

II. 5 Deep learning

II. 5. 1 Definition

Deep learning is a sub-field of machine learning dealing with algorithms inspired by the structure and function of the brain called artificial neural networks (see Figure II.16) [28]. In other words, it mirrors the functioning of our brains. Deep learning algorithms are similar to how nervous system structured where each neuron connected each other and passing information [29].

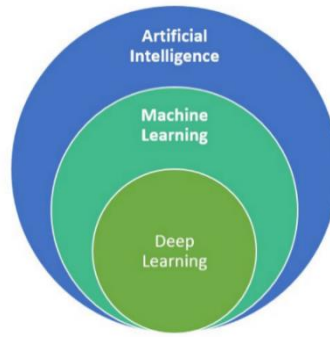


Figure II.16: The relation between AI, machine learning and deep learning

II. 5. 2 The difference between ML and DL

Machine learning and deep learning both fall under the category of artificial intelligence, while deep learning is a subset of machine learning. Therefore, deep learning is a part of machine learning, but it is different from traditional machine learning methods [30].

The table below (Table II.1) shows the key differences between deep learning and machine learning [31].

Table II.1: The key differences between Machine Learning and Deep Learning

Machine Learning	Deep Learning
Machine Learning is a superset of Deep Learning	Deep Learning is a subset of Machine Learning
The data represented in Machine Learning is quite different compared to Deep Learning as it uses structured data	The data representation used in Deep Learning is quite different as it uses neural networks(ANN).
Machine learning consists of thousands of data points.	Big Data: Millions of data points.
Outputs: Numerical Value, like classification of the score.	Anything from numerical values to free-form elements, such as free text and sound.
Uses various types of automated algorithms that turn to model functions and predict future action from data.	Uses a neural network that passes data through processing layers to, interpret data features and relations.
Algorithms are detected by data analysts to examine specific variables in data sets.	Algorithms are largely self-depicted on data analysis once they're put into production.
Machine Learning is highly used to stay in the competition and learn new things.	Deep Learning solves complex machine-learning issues.
Training can be performed using the CPU (Central Processing Unit).	A dedicated GPU (Graphics Processing Unit) is required for training.
More human intervention is involved in getting results.	Although more difficult to set up, deep learning requires less intervention once it is running.
Machine learning systems can be swiftly set up and run, but their effectiveness may be constrained.	Although they require additional setup time, deep learning algorithms can produce results immediately (although the quality is likely to

	improve over time as more data becomes available).
Its model takes less time in training due to its small size.	A huge amount of time is taken because of very big data points.
Humans explicitly do feature engineering.	Feature engineering is not needed because important features are automatically detected by neural networks.
Machine learning applications are simpler compared to deep learning and can be executed on standard computers.	Deep learning systems utilize much more powerful hardware and resources.
The results of an ML model are easy to explain.	The results of deep learning are difficult to explain.
Machine learning models can be used to solve straightforward or a little bit challenging issues.	Deep learning models are appropriate for resolving challenging issues.
Banks, doctor's offices, and mailboxes all employ machine learning already.	Deep learning technology enables increasingly sophisticated and autonomous algorithms, such as self-driving automobiles or surgical robots.
Machine learning involves training algorithms to identify patterns and relationships in data.	Deep learning, on the other hand, uses complex neural networks with multiple layers to analyze more intricate patterns and relationships.
Machine learning algorithms can range from simple linear models to more complex models such as decision trees and random forests.	Deep learning algorithms, on the other hand, are based on artificial neural networks that consist of multiple layers and nodes.
Machine learning algorithms typically require less data than deep learning algorithms, but the quality of the data is more important.	Deep learning algorithms, on the other hand, require large amounts of data to train the neural networks but can learn and improve on their own as they process more data.
Machine learning is used for a wide range of applications, such as regression, classification, and clustering.	Deep learning, on the other hand, is mostly used for complex tasks such as image and speech recognition, natural language processing, and autonomous systems.
Machine learning algorithms for complex tasks, but they can also be more difficult to train and may require more computational resources.	Deep learning algorithms are more accurate than machine learning algorithms.

II. 6 Convolutional neural networks

Convolutional neural networks or CNNs, are a specialized kind of neural network for processing data that has a known, grid-like topology. Examples include time-series data, which can be thought of as a 1D grid taking samples at regular time intervals, and image data, which can be thought of as a 2D grid of pixels. Convolutional networks have been tremendously successful in practical applications. The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution. Convolution is a specialized kind of linear operation. Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers [32].

The design of the Convolutional Neural Network is motivated by the discovery of the visual mechanism; the visual cortex presents in the brain. The visual cortex contains many cells that are responsible for detecting light in the small, layered sub-region of the visual field, called the receptive plane. These cells act as local filters above the input space, and the more complex cells have larger receptive fields [33].

In the Convolutional Neural Network (CNN) algorithm applied to the image, the image input will pass through several layers until it is finally classified into one class group. This layer includes the Convolutional layer, Pooling Layer, and Fully Connected Layer (Heaton, 2015). While the learning process is divided into two parts: forward propagation and back propagation.

The convolution layer in CNN performs the functions performed by cells in the visual cortex. In general, the convolutional layer of the convolution layer will detect the edge feature of the image, then the subsampling layer will reduce the dimensions of the feature obtained from the convolution layer, and finally forwarded to the output node through forward propagation process, and the predicted data class is finally determined by Softmax method on dense layer or fully connected layer (see Figure II.17) [34].

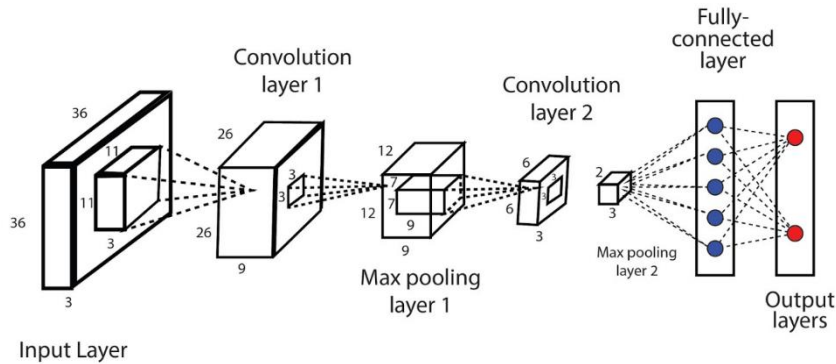


Figure II.17: CNN architecture

II. 6. 1 The different layers of a CNN

a) Convolutional Layer

The first layer to be passed by the input data is the convolution layer. As the name suggests, this layer plays an important role in how CNN works. This layer has several parameters that must be determined by the researcher [33].

Convolution Layer can significantly reduce model complexity through optimizing output. Accordingly, some parameters for this optimization are called hyper parameters consisting of the number of Filters, the size of Filter, Stride, Padding and activation functions (see Figure II.18) [32].

Filter size parameters (f), image size (w), magnitude of shift (s) and padding (p) must be combined in order to make the value of the step count or the value of the step variable being an integer (see Eq II.24).

$$steps = \frac{w-f+2p}{s} + 1 \quad (\text{II.24})$$

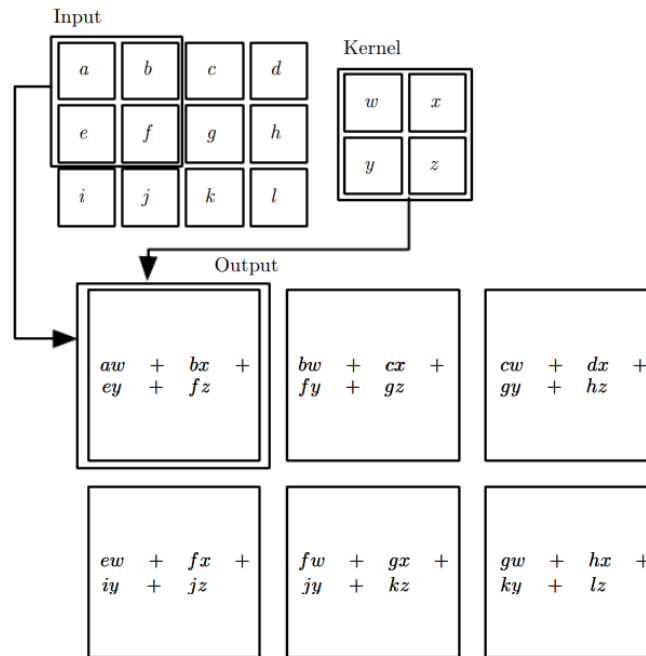


Figure II.18: Convolution operation

b) Pooling layer

In simple terms this layer is responsible for down sampling so that the size of the input data will be smaller. The literature mentions that this layer is not affected by the training phase. This is because this layer has no weight as any other layer [33].

This layer has several parameters that are static, the parameters are spatial extend and stride parameters. Illustration of down sampling on this layer can be seen in (Figure II.19) [35].

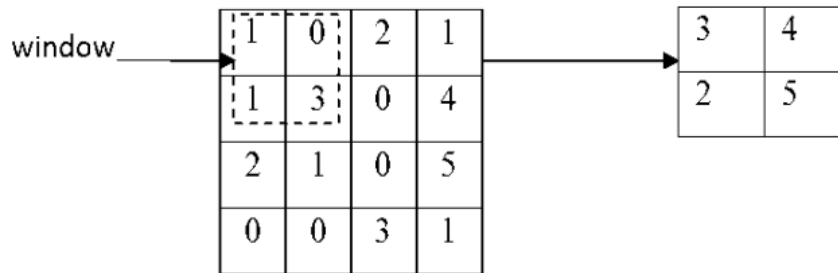


Figure II.19: Pooling operation performed by choosing a 2x2 window

The purpose of using pooling or subsampling layer is to reduce the data volume and complexity of the data that will enter in the next layer. The window size used in the subsampling process determines how much information is omitted. There are two main types of pooling [36]:

Max Pooling selects the maximum element from each of the windows of the feature map. Thus, after the max-pooling layer, the output would be a feature map containing the most dominant features of the previous feature map.

Average Pooling computes the average of the elements present in the region of the feature map covered by the filter. It simply averages the features from the feature map.

c) Fully-Connected Layer

This layer is a layer that has interconnected neurons. In this layer the feature maps of the previous layer (Pooling layer) will be decomposed into a node. The way this layer works is the same as the forward propagation process in ANN, so in this layer there is also a weight that connects between neurons. These weights will change with the training done on the network. Like neurons on artificial neural networks, neurons on this Dense layer must also have an activation function, in the Deep Learning and Neural Network mentioned that the activation functions that can be used in neurons in this layer are very diverse as ReLU, Sigmoid or hyperbolic activation function [33].

II. 6. 2 CNN known models

a) LeNet

LeNet-5, introduced by Yann LeCun et al. in 1998 [36], was one of the pioneering CNN architectures. It was designed for handwritten digit recognition and consisted of two convolutional layers, followed by average pooling, fully connected layers, and a softmax output layer. LeNet-5's lightweight architecture and efficient use of shared weights laid the foundation for modern CNNs [37], influencing subsequent developments in the field (see Figure II.20) [38].

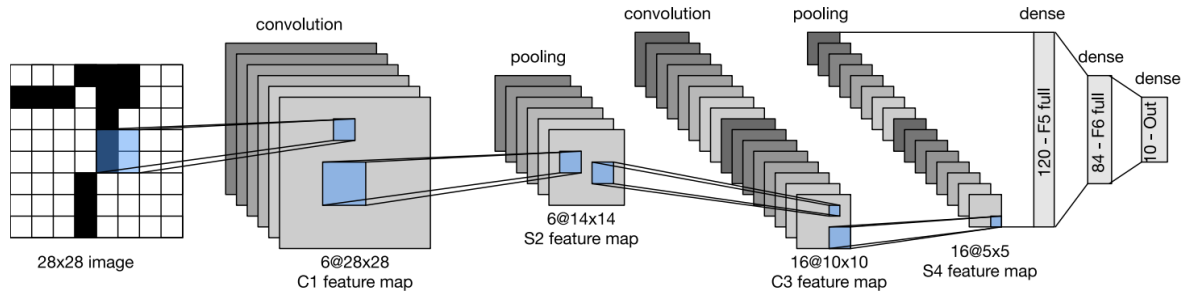


Figure II.20: LeNet

b) AlexNet

AlexNet developed by Alex Krizhevsky et al. in 2012 [39], made significant strides in image classification. It won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 and achieved a significant improvement over previous methods. AlexNet featured a deeper architecture with multiple convolutional layers, introduced the concept of ReLU activations, and employed techniques like dropout for regularization. Its success paved the way for the resurgence of deep learning in computer vision [37].

c) VGG

VGGNet, proposed by Karen Simonyan and Andrew Zisserman in 2014 [40], introduced deeper architectures with up to 19 layers. VGGNet's key characteristic was the consistent use of 3x3 convolutional filters throughout the network, which enabled better modeling of complex visual patterns. The VGGNet architecture is known for its simplicity and effectiveness, and it has served as a baseline for subsequent CNN models [37].

d) ResNet

ResNet developed by Kaiming He et al. in 2015 [41], brought significant advancements in training very deep neural networks. ResNet introduced the concept of residual connections, allowing information to bypass certain layers and alleviating the vanishing gradient problem. This breakthrough enabled the successful training of CNNs with 50, 100, or even more layers. ResNet's skip connections revolutionized deep learning architectures and enabled the training of extremely deep and accurate networks [37].

II. 6. 3 Transfer learning

Transfer learning for machine learning is when existing models are reused to solve a new challenge or problem. Transfer learning is not a distinct type of machine learning algorithm, instead

it's a technique or method used whilst training models. The knowledge developed from previous training is recycled to help perform a new task. The new task will be related in some way to the previously trained task, which could be to categorise objects in a specific file type. The original trained model usually requires a high level of generalisation to adapt to the new unseen data [42].

Transfer learning means that training won't need to be restarted from scratch for every new task. Training new machine learning models can be resource-intensive, so transfer learning saves both resources and time. The accurate labelling of large datasets also takes a huge amount of time. The majority of data encountered by organisations can often be unlabelled, especially with the extensive datasets required to train a machine learning algorithm. With transfer learning, a model can be trained on an available labelled dataset, then be applied to a similar task that may involve unlabelled data (see Figure II.21) [43].

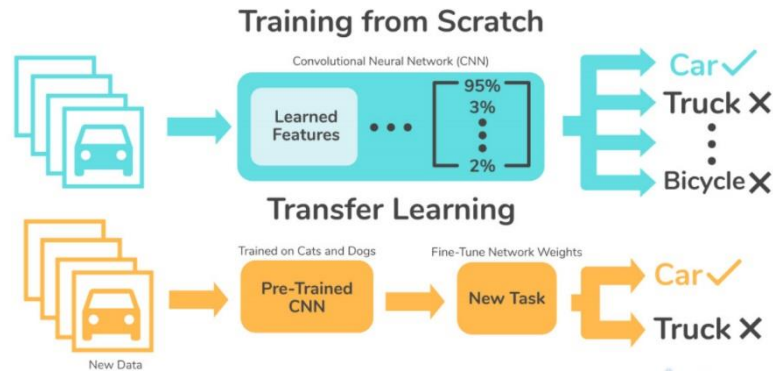


Figure II.21: Training from scratch vs Transfer Learning

II. 7 Conclusion

In conclusion, Artificial Intelligence emerges as a transformative force, driving innovation across industries. This chapter provides a comprehensive overview, emphasizing Machine Learning's role in enabling systems to learn from data and Deep Learning's mimicry of human neural processes. The distinctions between these technologies highlight AI's versatility and potential in solving real-world challenges. As AI evolves, it promises to revolutionize efficiency, productivity, and innovation. The next chapter will explore the state of the art in AI.

Chapter III State of The Art

III. 1 Introduction

Automated lung segmentation in computed tomography (CT) images is a crucial step in the early diagnosis and treatment of lung diseases. Due to the complex and variable nature of lung morphology, developing robust and accurate segmentation algorithms remains a significant challenge. Traditional methods, such as morphological techniques and thresholding, have shown limitations in generalization and computational efficiency. In contrast, deep learning approaches, particularly Convolutional Neural Networks (CNNs), have demonstrated substantial promise in addressing these challenges due to their ability to learn intricate patterns from data. This chapter reviews state-of-the-art methods in lung segmentation, with a focus on recent advancements in deep learning architectures such as U-Net, Mask R-CNN, and Residual U-Net, which have significantly enhanced segmentation accuracy and efficiency.

III. 2 Related work

III. 2. 1 The work of Bhattacharjee, A., et al.

This research introduces a refined U-Net segmentation model tailored for lung segmentation, addressing the complexities posed by anatomical variations. The model was trained on a modest dataset of 240 images, highlighting its ability to achieve robust segmentation with limited training data. Key innovations include structural modifications to the original U-Net architecture, incorporating batch normalization and dropout layers to prevent overfitting, along with additional filters for dimensionality reduction. These adjustments result in superior segmentation performance, outperforming both the baseline U-Net model and contemporary techniques. Noteworthy achievements include an accuracy of 98.3%, a binary cross-entropy loss of 0.04, and high scores for Intersection over Union (IoU) and Dice Similarity Coefficient (DSC). This research contributes a potent semantic segmentation framework tailored specifically for lung segmentation, offering excellent performance even with limited CT image data (see Figure III.1) [44].

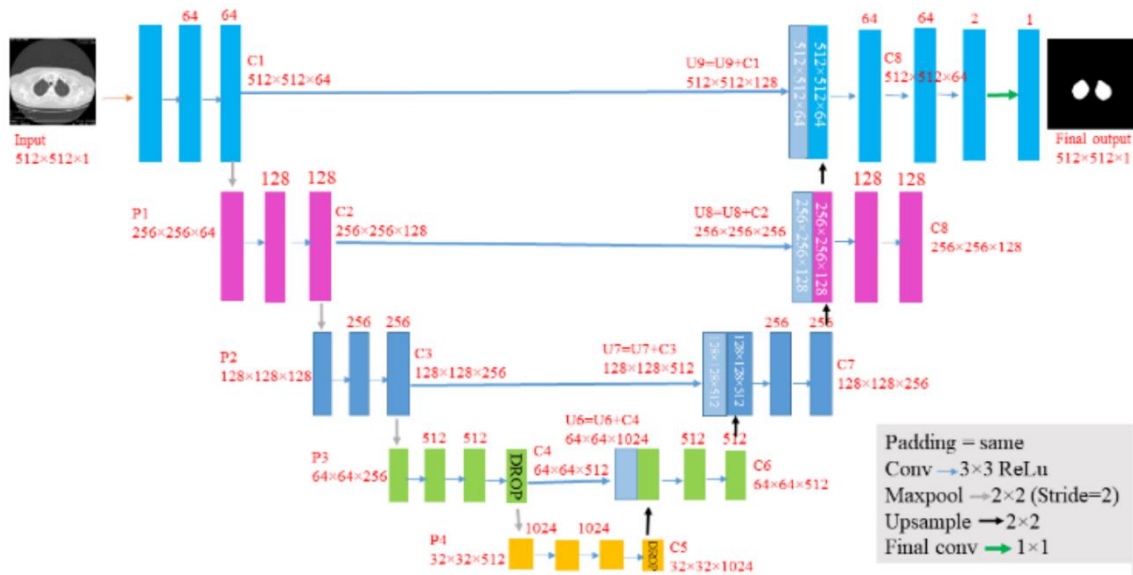


Figure III.1 The proposed U-Net architecture for lung segmentation

III. 2. 2 The work of Qinhua, Hu., et al.

This study presents an advanced approach for automating lung segmentation in CT images, utilizing the power of Convolutional Neural Network (CNN) Mask R-CNN, enhanced by supervised and unsupervised machine learning techniques, including Bayes, Support Vector Machine (SVM), K-means, and Gaussian Mixture Models (GMMs). The method achieves exceptional accuracy ($97.68 \pm 3.42\%$) with an average runtime of 11.2 seconds when employing Mask R-CNN with the K-means kernel. Comparative analysis against existing methodologies confirms the superiority of the proposed approach in both accuracy and computational efficiency. The innovation lies in the comprehensive integration of transfer learning, erosion and dilation strategies, clustering for precise region mapping, and a guiding kernel factor within the Mask R-CNN architecture. This holistic approach minimizes the dependence on extensive training data while maximizing segmentation performance, distinguishing it from traditional methods reliant on preprocessing or semi-automatic procedures (see Figure III.2) [45].

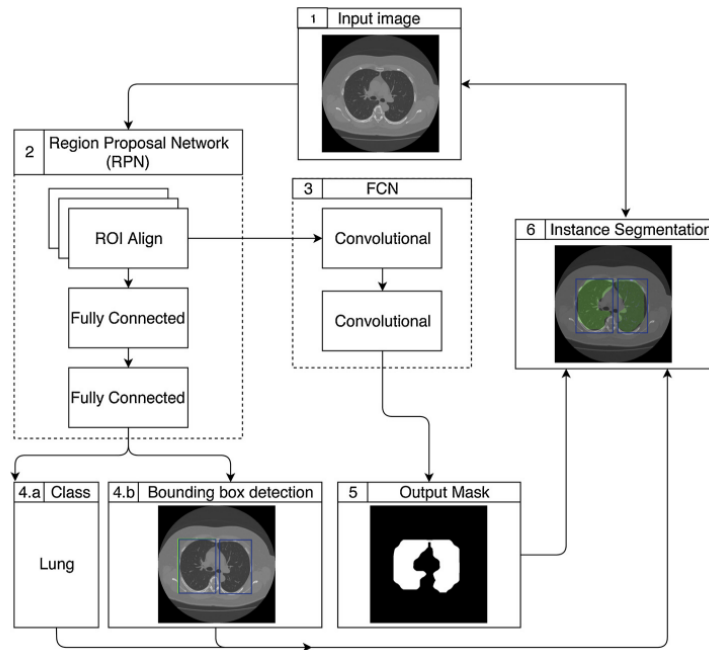


Figure III.2 Figure Mask R-CNN flowchart

III. 2. 3 The work of Khanna, A., et al.

This research introduces a deep Residual U-Net convolutional neural network for automated lung segmentation in CT images, addressing challenges such as irregular lung shapes, low contrast, and fuzzy boundaries. By integrating residual learning with the U-Net architecture, the model enhances feature extraction and segmentation accuracy. The approach employs data augmentation techniques to improve generalization and a connected component-based method to reduce false positives. Experimental results show that the proposed method achieves Dice Similarity Coefficients of 98.63%, 99.62%, and 98.68% on the LUNA16, VESSEL12, and HUG-ILD datasets, respectively, outperforming existing methods. This robust and precise segmentation framework holds promise for improving early diagnosis and treatment of lung diseases, with potential applications in clinical decision-making and computer-aided diagnosis systems. Future research aims to further enhance generalization through generative adversarial networks and extend the model to other medical imaging domains (see Figure III.3) [46].

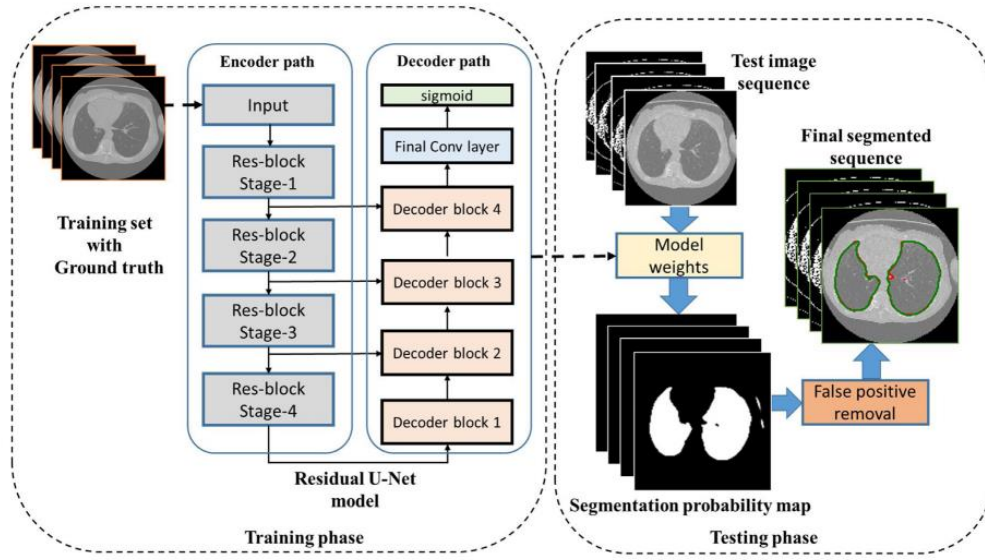


Figure III.3 Overview of the proposed Residual U-Net based CNN architecture for lung CT segmentation.

III. 2. 4 Results

The table below (Table III.1) summarizes the results of various state-of-the-art methods for automated lung segmentation in CT images. These methods, which include modifications to U-Net architectures, Mask R-CNN with machine learning techniques, and deep Residual U-Nets, demonstrate significant advancements in segmentation accuracy and efficiency across different datasets. The performance metrics highlight the effectiveness and precision of these approaches in handling the complexities of lung segmentation (Table III.1).

Table III.1 Comparative Analysis of Recent Studies on Lung Segmentation Using AI Techniques

DOI	Authors	Title	Dataset	Results
10.1109/ACTS53447.2021.9708190	Ananya Bhattacharjee, Tripti Goel, R Murugan, Badal Soni	Semantic segmentation of lungs using a modified U-Net architecture through limited Computed	240 training images and tested on 240 testing images of Lung Image Database Consortium and Image Database Resource	DSC : 96.29% Accuracy : 98.3% Jaccard coefficient : 93.63% Loss : 0.03%

		Tomography images	Initiative (LIDCIDRI)	
10.1016/j.artmed.2020.101792	Qinhua Hua, Luís Fabrício de F. Souza, Gabriel Bandeira Holandab, Shara S.A. Alvares, Francisco Hércules dos S. Silvab, Tao Hand, Pedro P. Rebouças Filhob	An effective approach for CT lung segmentation using mask region-based convolutional neural networks	998 images of 39 lung computed tomography (CT) exams in Digital Imaging and Communications in Medicine (DICOM) format.	DC : 97.33 ± 3.24 Segmentation time (s): 11.24 ± 2.57
10.1016/j.bbe.2020.07.007	Anita Khanna, Narendra D. Londhe, S. Gupta, Ashish Semwal.	A deep Residual U-Net convolutional neural network for automated lung segmentation in computed tomography images	Publicly available benchmark datasets, namely LUNA, VESSEL12, and HUG-ILD	DSC: 98.63 ± 0.1 JI : 97.32 ± 0.2 Recall : 98.61 ± 0.3

III. 2. 5 Conclusion

The reviewed works highlight the evolution and effectiveness of deep learning techniques in the domain of lung segmentation. Bhattacharjee et al. presented a modified U-Net model that achieves high accuracy with minimal training data, incorporating batch normalization and dropout layers to

prevent overfitting. Hu et al. introduced a Mask R-CNN approach augmented by various machine learning techniques, excelling in both accuracy and computational efficiency. Khanna et al. developed a deep Residual U-Net that integrates residual learning with the U-Net architecture, demonstrating superior performance across multiple datasets. These advancements underscore the potential of deep learning models to overcome traditional limitations, providing robust and precise segmentation solutions essential for clinical decision-making and further medical imaging applications.

Chapter IV Experiments and Realization

IV. 1 Introduction

This chapter describes the experimental setup and realization of our proposed models for lung segmentation in CT images. It details the tools and libraries used, the dataset, preprocessing steps, network architectures, implementation, and results. Additionally, it covers the development of a web application, Lung Vision, for managing medical images, patients, and segmenting scans using the developed machine learning models.

IV. 2 Tools and Libraries

IV. 2. 1 Python

Python is a powerful and easy-to-learn programming language with efficient high-level data structures and a straightforward approach to object-oriented programming. Its elegant syntax and dynamic typing, combined with its interpreted nature, make it ideal for scripting and rapid application development across many platforms [47].



Figure IV.1 Python logo

IV. 2. 2 Google Colaboratory

Google Colaboratory (Colab) is a hosted Jupyter Notebook service that requires no setup to use and provides free access to computing resources, including GPUs and TPUs. Colab is especially well-suited for machine learning, data science, and education [48].

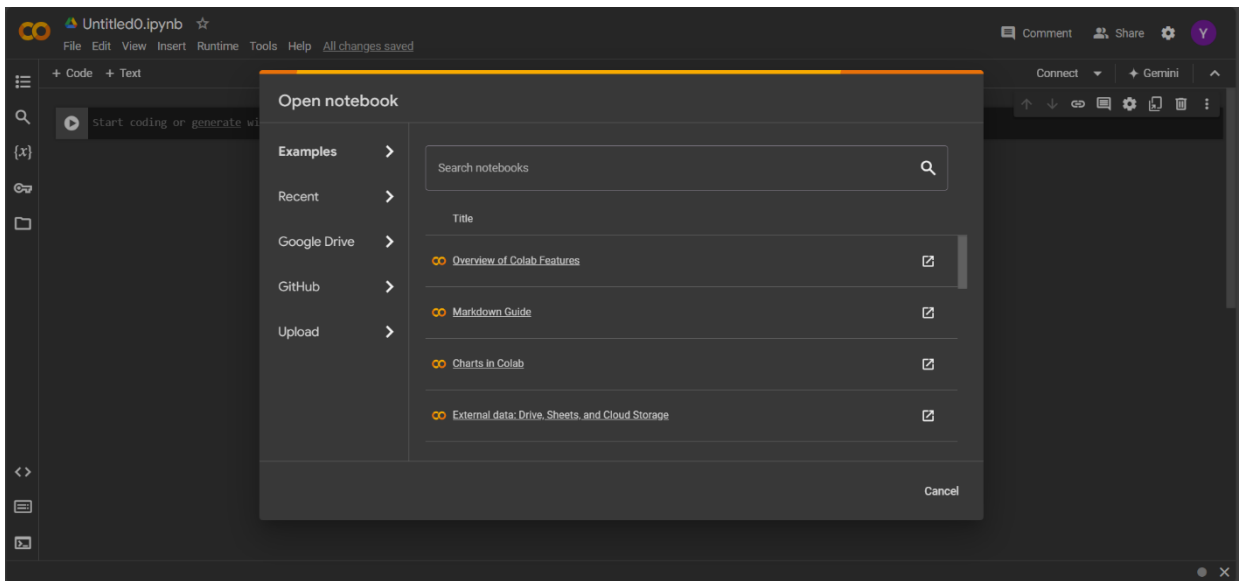


Figure IV.2 Google Colaboratory

IV. 2. 3 Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in the Anaconda Distribution that allows users to launch applications and manage conda packages, environments, and channels without using the command line interface (CLI). It is available for Windows, macOS, and Linux [49].

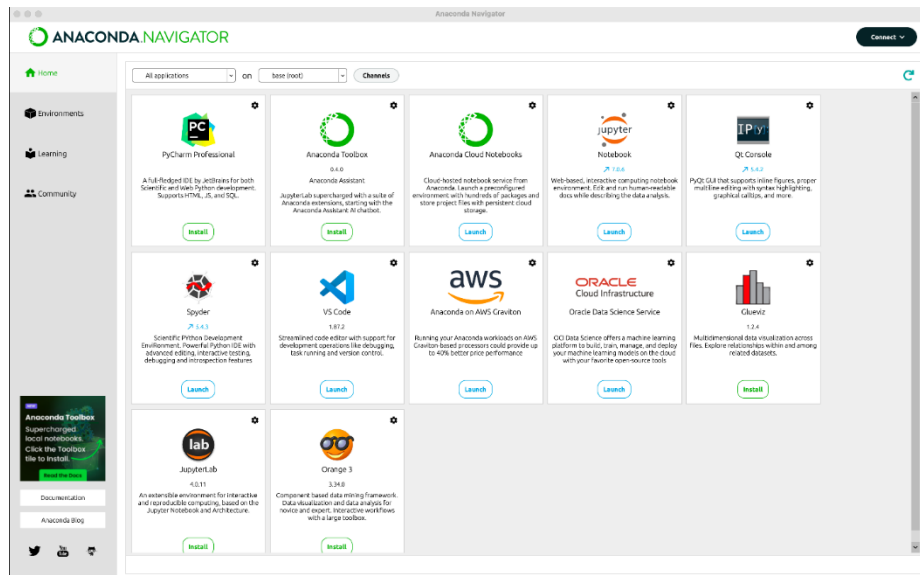


Figure IV.3 Anaconda navigator

IV. 2. 4 Jupyter Notebook

Jupyter Notebook is a user-friendly notebook authoring tool that is part of Project Jupyter. It enables the creation and sharing of computational notebooks, which are interactive documents combining computer code, text descriptions, data, rich visualizations, and interactive controls [50].

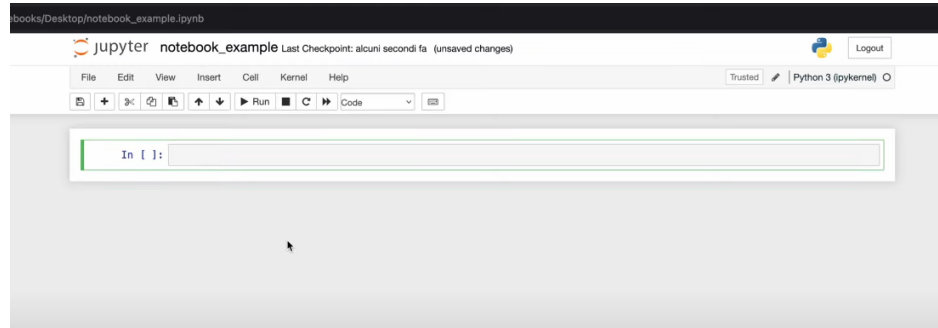


Figure IV.4 Jupyter notebook

IV. 2. 5 TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that allows researchers to push the state-of-the-art in ML and developers easily build and deploy ML-powered applications [51].



Figure IV.5 TensorFlow logo

IV. 2. 6 Keras

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. allowing researchers to go from idea to result as quickly as possible [52].



Figure IV.6 Keras logo

IV. 2. 7 OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. It provides a common infrastructure for computer vision applications and accelerates the use of machine perception in the commercial products [53].



Figure IV.7 OpenCV logo

IV. 2. 8 Visual Studio Code

Visual Studio Code is a lightweight but powerful source code editor available for Windows, macOS, and Linux. It comes with built-in support for JavaScript, TypeScript, and Node.js and has a rich ecosystem of extensions for other languages and runtimes [54].

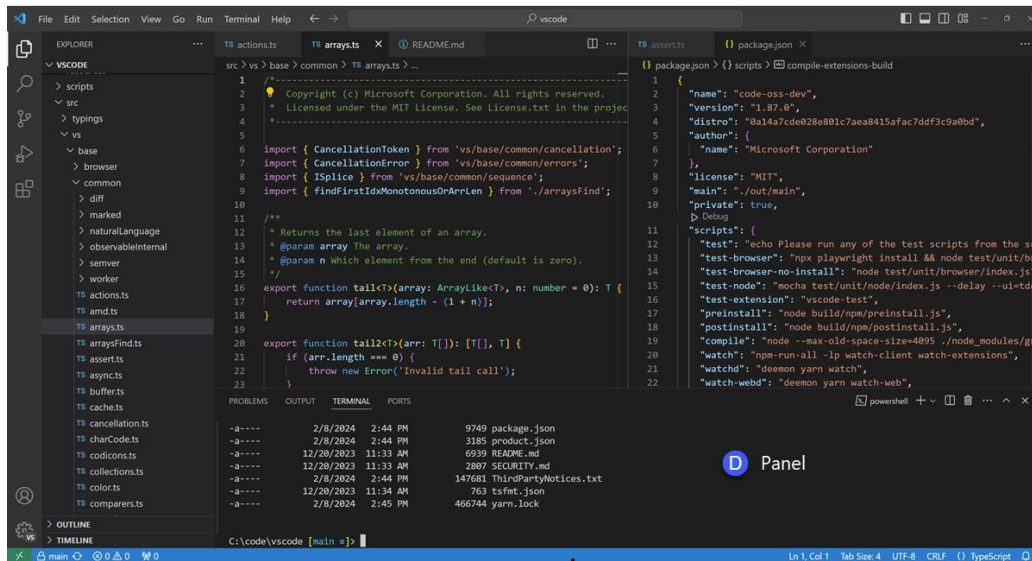


Figure IV.8 Visual Studio Code

IV. 2. 9 Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, allowing developers to focus on writing their applications [55].



Figure IV.9 Django logo

IV. 2. 10 PostgreSQL

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. It has a history of more than 35 years of active development [56].

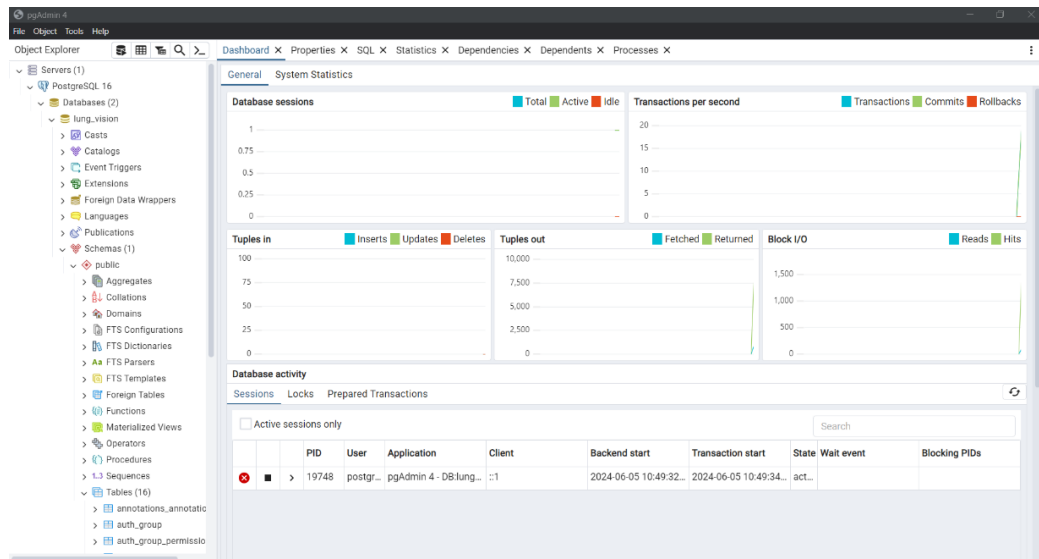


Figure IV.10 PostgreSQL pgAdmin4

IV. 2. 11 React.js

React.js is a JavaScript library developed by Facebook for building user interfaces, particularly single-page applications. It enables developers to create reusable UI components that manage their own state and efficiently update and render the right components in response to data changes [57].

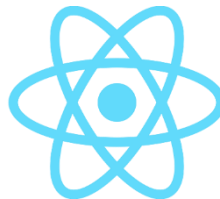


Figure IV.11 React logo

IV. 2. 12 Draw.io

draw.io is a free online diagram software that allows users to create various types of diagrams and charts, such as flowcharts, network diagrams, UML diagrams, ER diagrams, BPMN diagrams, and circuit diagrams. It can be used in the browser or as a desktop app [58].

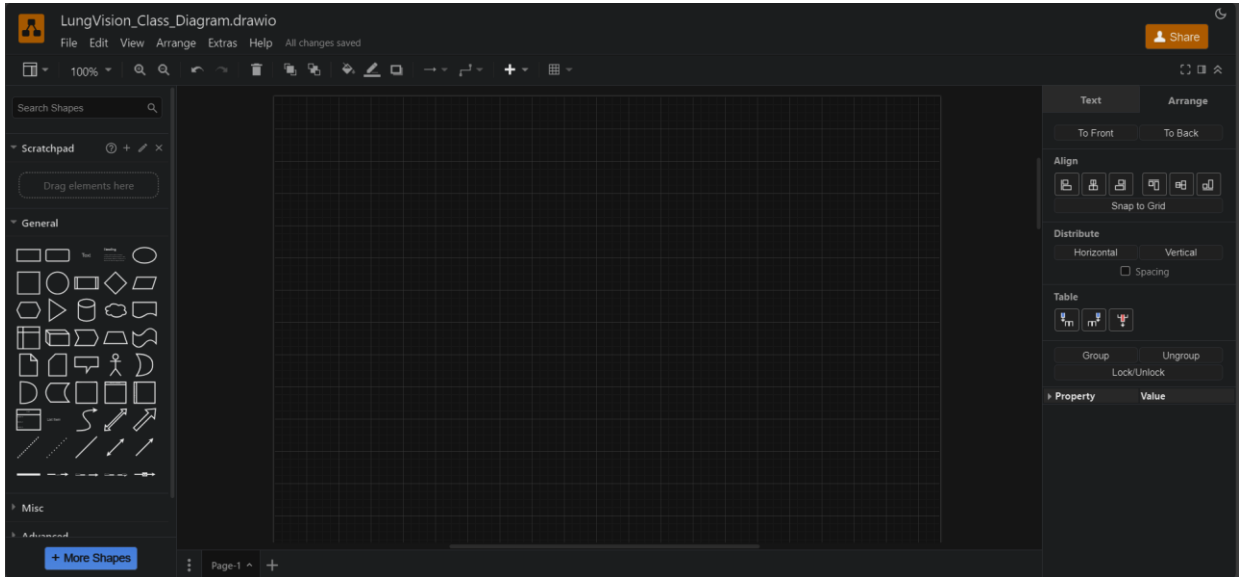


Figure IV.12 Draw.io

IV. 3 Experiments

The goal of this thesis is to develop robust approaches for the preprocessing and semantic segmentation of medical images, specifically focusing on lung diseases. In this research, we propose a two-step approach combining object detection and segmentation to enhance the accuracy and efficiency of medical image analysis.

IV. 3. 1 Datasets

The dataset used in this study comprises 140 computed tomography (CT) exams, each representing a distinct patient and covering six organ classes: liver, lungs, bladder, kidney, bones, and brain. The dataset, formatted in Neuroimaging Informatics Technology Initiative (NIFTI), was obtained from the CT-ORG dataset for multiple organ segmentation in computed tomography. The patient demographic includes 62.5% male and 37.5% female [59], with some exhibiting metastatic diseases originating from cancers in the breast, colon, bones, and lungs. The data are divided into 119 training volumes and 21 testing volumes. For this study, we only used volumes that contain the full lungs because some scans only cover half of the chest, focusing on informative slices based on the provided lung mask (lungs = 3). Each exam contains approximately 200 images, resulting in a dataset of 5,930 images, each 512×512 pixels in size. We conducted experiments on 500 randomly selected images, using 400 for training and 100 for testing, corresponding to an 80% training and 20% testing split.

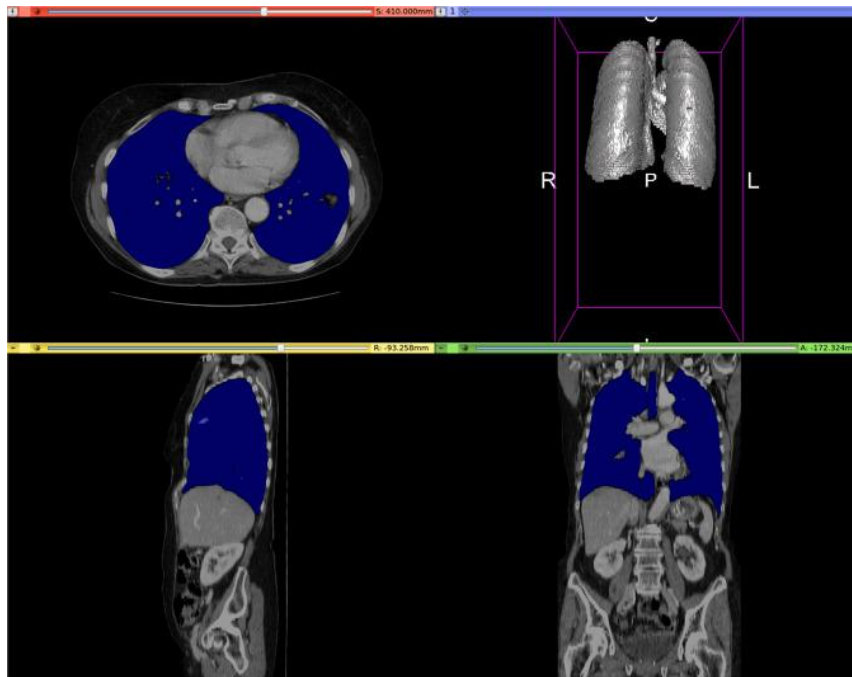


Figure IV.13 Example of CT lung detection and segmentation by image morphology. Lung mask overlaid in blue. Rendered by 3D Slicer.

IV. 3. 2 Preprocessing

Preprocessing is a crucial part of our fully automated segmentation pipeline, designed to streamline the process without requiring any user intervention (see Figure IV.14). The steps involved are:

a) Thresholding:

To isolate lung tissues and exclude irrelevant structures, the Hounsfield unit (HU) values are selected within the range of -1000 to -200. This range effectively captures the lung parenchyma and air-filled spaces, ensuring that only relevant anatomical features are included for further analysis. This step helps in emphasizing the lung region and improving the segmentation process.

b) Normalization:

Subsequently, normalization is applied to standardize the pixel values across the dataset. By scaling the pixel values to the range [0-1], normalization ensures uniformity and facilitates consistent processing across different images.

c) Conversion to Grayscale:

To simplify subsequent analysis and enhance visualization, the images are converted to grayscale. This conversion reduces complexity by representing each pixel with a single intensity value, streamlining further processing steps.

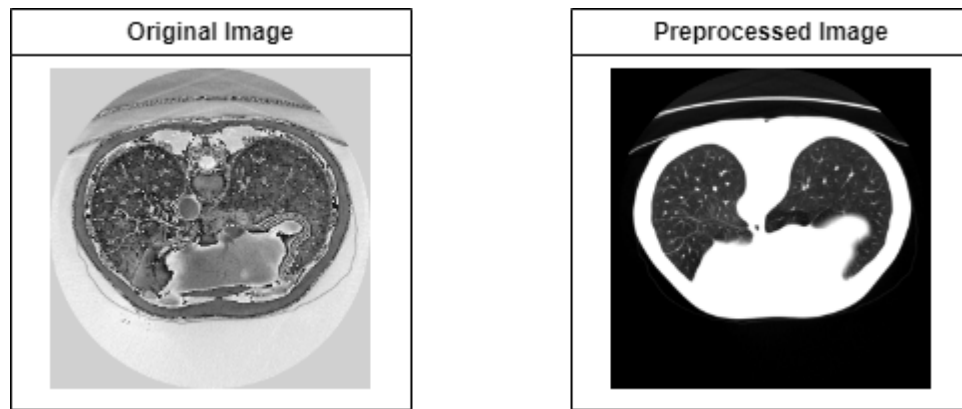


Figure IV.14 Comparison of Original and Preprocessed CT Images

IV. 3. 3 Environment

In our academic research, because of the unavailability of resources and the high cost associated with acquiring advanced graphic cards, we initially attempted to utilize the GPU resources provided

by Google Colab. However, due to limitations in the available resources (a 12GB NVIDIA Tesla K80 GPU with a usage cap of 12 hours), coupled with the significant volume of data, we encountered frequent session crashes, restarts, and subsequent instability. To address this challenge, we shifted our experimentation to a desktop computer powered by a Ryzen 5600G processor and equipped with 16GB of RAM. Our software environment remained consistent, utilizing Python version 3.8 and employing the Keras and TensorFlow libraries for model development and experimentation.

IV. 3. 4 Direct Approaches

a) U-Net

The U-Net architecture was first introduced by Olaf Ronnenberger et al. in 2015, its application focuses on biomedical image segmentation. They improve the architecture of the so-called "fully convolutional network" by changing and extending it so that it can work with fewer training images and produce more precise segmentations. As its name implies, it is an architecture that has a symmetrical "U" shape, as shown in (Figure IV.15), and is constituted by two fundamental parts. On the left side of the "U", there is the contraction path, also known as the encoder. This path is formed by several blocks which are based on convolutional processes that capture the context of the input image. The left path, the expansion path, or also called decoder is used to allow precise localization by means of transposed convolutions. Both paths are concatenated by skip connections, the skip connection combines the spatial information of the encoder with the decoder to retain good spatial information. The network is essentially a stack of convolutional and maximum density layers with activation and loss functions. As a result, it is an end-to-end fully convolutional network, meaning that it has only convolutional layers and no dense layers, allowing it to accept images of any size. The u-net architecture performs effectively in a variety of biomedical segmentation applications [60].

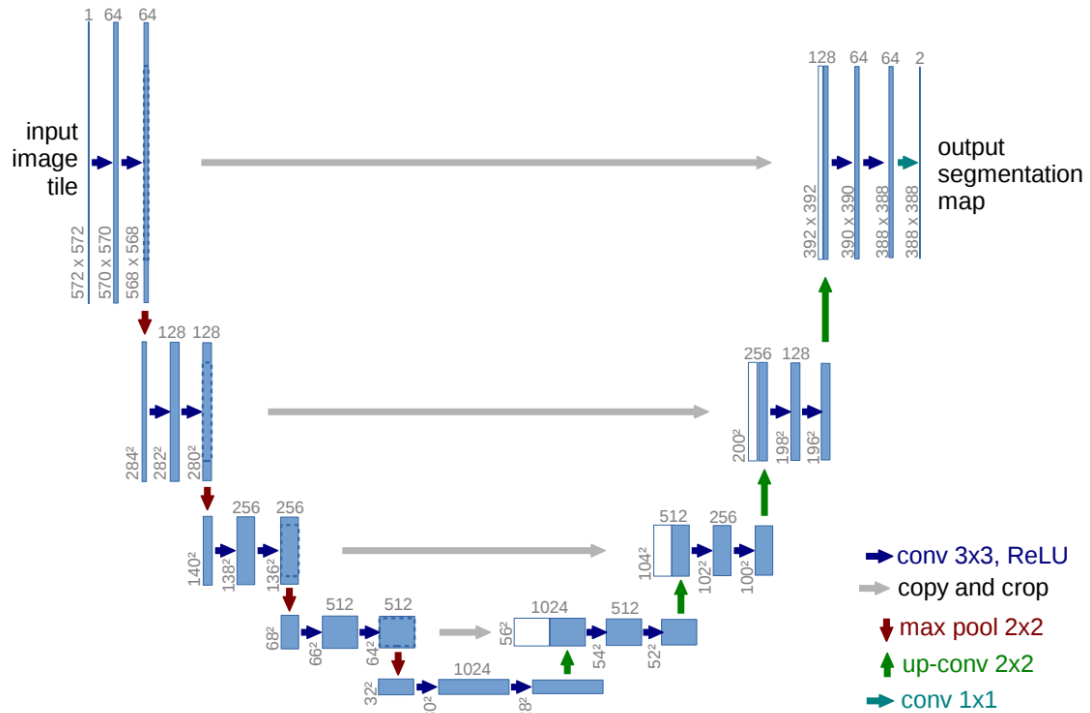


Figure IV.15 Structure of the U-Net architecture proposed by Ronnenberger et al.

b) Attention U-Net

Ozan Oktay et al. proposed for the first time attention gates (AGs) in the context of U-Net. (Figure IV.16) shows the diagram of the architecture. The objective of applied AGs to a neural network model is to highlight only the relevant features that are need to train the proposed model (Figure IV.17). Incorporating AGs into a common CNN architecture, such as the U-Net model, does not require heavy computational resources, on the contrary, it reduces those wasted on irrelevant activations and provides better network generalization. Traditional approaches like cascaded framework lead to overuse and redundancy of computing resources and model parameters; for example, all models in the cascade repeatedly extract similar low-level features. Attention U-Net learn to remove unimportant areas in the input image and to focus important structures that are helpful for a particular task without additional supervision. By reducing feature activations in unnecessary regions, AGs increase model sensitivity, and accuracy for dense label predictions. In this method, the use of an external organ localization model can be avoided while yet achieving good prediction accuracy. AGs have been shown to be quite useful for tissue/organ identification and localization [61].

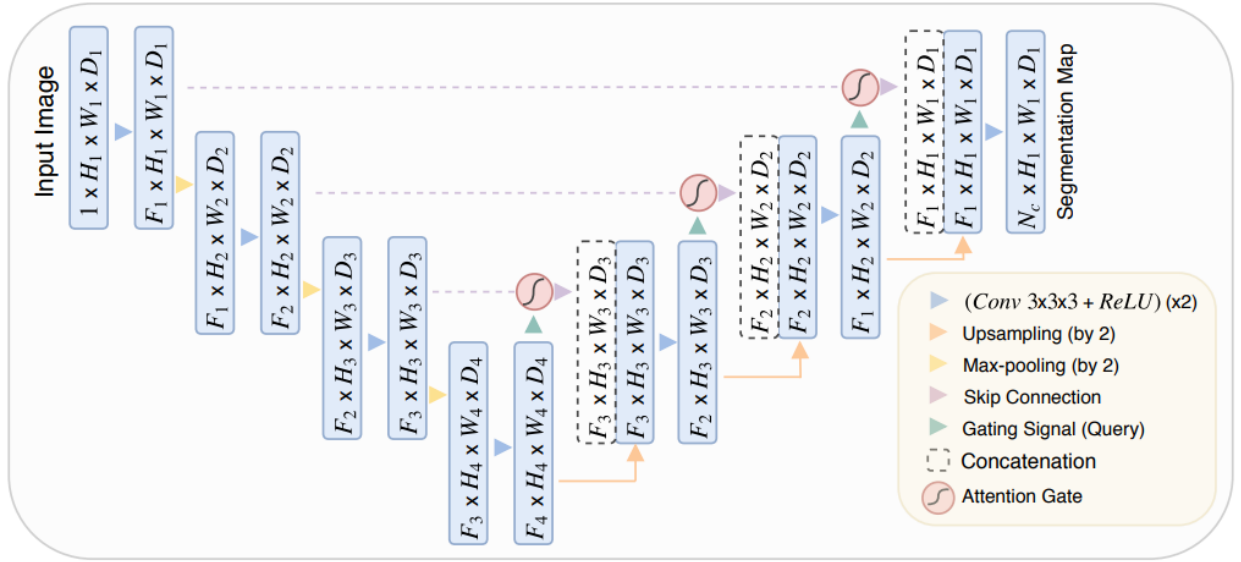


Figure IV.16 Structure of the attention U-Net architecture proposed by Ozan Oktay et al.

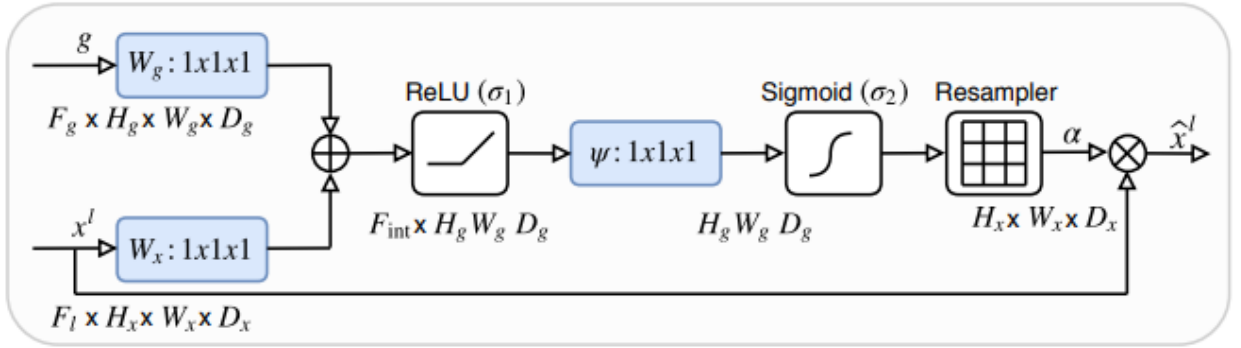


Figure IV.17 Schematic of the Attention Gate

IV. 4 Proposed Approaches

In this section, we present two advanced approaches designed for the segmentation of lung regions in CT images. Both approaches begin with lung detection using the YOLOv8 algorithm and proceed to segmentation, with Approach 1 utilizing the U-Net framework and Approach 2 employing the Attention U-Net. These integrated approaches leverage state-of-the-art deep learning techniques to enhance segmentation accuracy and efficiency, making them suitable for clinical applications in diagnosing and treating lung diseases.

IV. 4. 1 YOLOv8 Object Detection

As a pivotal component of our proposed approaches, YOLOv8 is an advanced object detection model that builds on the architecture of YOLOv5, incorporating significant improvements.

Introduced as the latest in the YOLO (You Only Look Once) series, YOLOv8 focuses on enhancing detection accuracy and efficiency. The architecture features a new C2f module (cross-stage partial bottleneck with two convolutions), replacing the previous CSPLayer to better combine high-level features with contextual information. YOLOv8 utilizes an anchor-free model with a decoupled head, which processes objectness, classification, and regression tasks independently, allowing for more specialized and accurate results. The output layer employs the sigmoid function for objectness scores and the softmax function for class probabilities, enhancing the model's prediction capabilities. Additionally, YOLOv8 incorporates CIoU and DFL loss functions for bounding box regression and binary cross-entropy for classification loss, which improve performance, especially with smaller objects. YOLOv8's versatility is reflected in its command line interface (CLI) usage, PIP package installation, and multiple integrations for labeling, training, and deployment. YOLOv8 demonstrates significant improvements over its predecessors. (Figure IV.18) shows the detailed architecture of YOLOv8 [62].

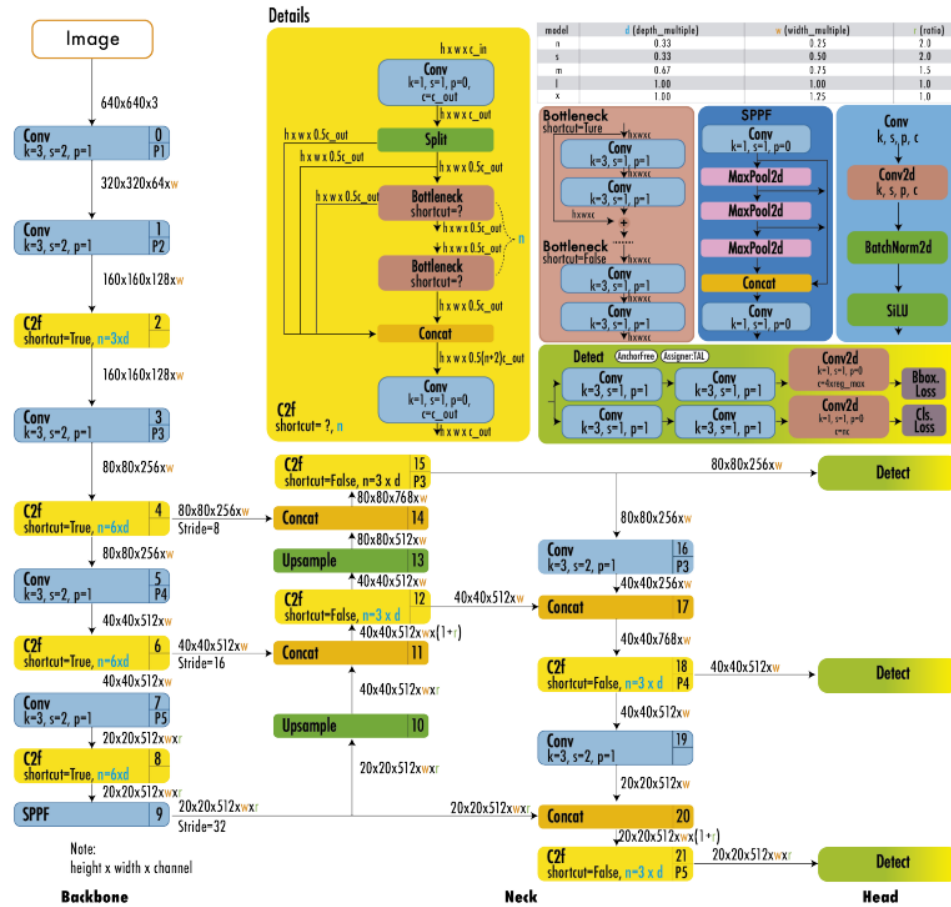


Figure IV.18 YOLOv8 Architecture

IV. 4. 2 YOLOv8 with U-Net (Approach 1)

Our initial approach to lung segmentation aims to efficiently and accurately identify and segment lung regions from CT images. The method encompasses a multi-step process integrating cutting-edge techniques:

a) Lung Detection with YOLOv8:

Preprocessed images are fed into the pre-trained YOLOv8 model, which is renowned for its exceptional object detection capabilities. YOLOv8 efficiently identifies lung regions within the images, generating bounding boxes to delineate these areas. This bounding box information serves as a region of interest (ROI), highlighting the specific areas containing the lungs. Subsequently, pixels within these bounding boxes are selectively retained for further processing, while background pixels are eliminated. This meticulous selection ensures that only relevant lung region pixels are retained, optimizing subsequent analysis and segmentation tasks.

b) Lung Segmentation using U-Net:

This approach employs the U-Net architecture for segmentation (Figure IV.19), which is particularly effective for biomedical image segmentation tasks. U-Net has a symmetrical "U" shape, comprising a contraction path (encoder) on the left and an expansion path (decoder) on the right. The encoder captures the context of the processed images through convolutional processes, while the decoder enables precise localization via transposed convolutions. Skip connections between the encoder and decoder combine spatial information, retaining high-resolution features. For training, the U-Net model uses an input shape of (512, 512, 1), with a learning rate of 1e-4, a batch size of 16, and 50 epochs. The model is compiled with the Dice loss function and the Adam optimizer, and the metrics include dice_coef, IoU, Recall, and Precision. Training incorporates callbacks such as ModelCheckpoint, ReduceLROnPlateau, TensorBoard, and EarlyStopping to enhance performance and prevent overfitting.

In summary, Approach 1 integrates preprocessing steps, lung detection using YOLOv8, and final segmentation with U-Net. This combination leverages object detection and deep learning segmentation techniques, providing an effective and robust approach for lung segmentation in CT images. The automation of these steps ensures consistency and efficiency, making the approach suitable for clinical applications and aiding in the early diagnosis and treatment of lung diseases.

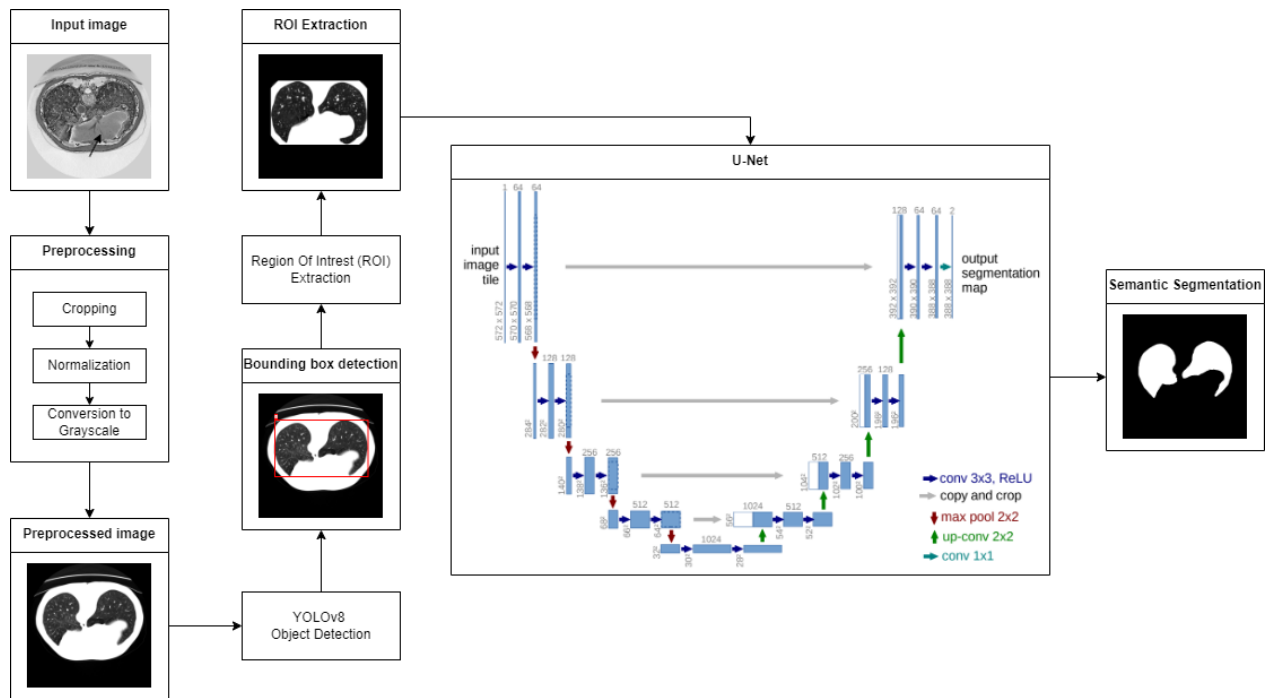


Figure IV.19 Approach 1 Architecture

IV. 4. 3 YOLOv8 with Attention U-Net (Approach 2)

Our second proposed approach for lung segmentation also uses advanced deep learning techniques, with the following steps:

a) Lung Detection using YOLOv8:

Leveraging the pre-trained YOLOv8 model, the system conducts lung detection on uploaded CT scans. Preprocessed images are fed into the YOLOv8 model, which promptly identifies lung regions and produces bounding boxes to demarcate these areas, serving as regions of interest (ROIs). The subsequent processing selectively retains pixels within these bounding boxes while eliminating background pixels, ensuring focused consideration of relevant lung region pixels. This meticulous selection process enhances the accuracy and efficiency of subsequent segmentation tasks by prioritizing the areas of interest identified by YOLOv8.

b) Lung Segmentation using Attention U-Net:

Approach 2 utilizes the Attention U-Net architecture (see Figure IV.20), an enhancement of the traditional U-Net with attention mechanisms. These mechanisms focus on relevant regions in the input image (processed images), improving segmentation accuracy, particularly in complex scenes. The architecture maintains the same U-shaped structure with a contraction path, expansion path, and skip connections, but integrates attention gates to highlight important features. This approach

follows the same training setup as Approach 1, using an input shape of (512, 512, 1), a learning rate of $1e-4$, a batch size of 16, and 50 epochs. The model is compiled with the Dice loss function and the Adam optimizer, using `dice_coef`, IoU, Recall, and Precision as metrics. The same set of callbacks is used to optimize training.

In summary, Approach 2 integrates preprocessing steps, lung detection using YOLOv8, and final segmentation with Attention-UNet. By incorporating attention mechanisms into the U-Net architecture, this model enhances segmentation accuracy and robustness, particularly in challenging cases. The automation of these steps ensures consistency and efficiency, making the approach suitable for clinical applications and aiding in the early diagnosis and treatment of lung diseases.

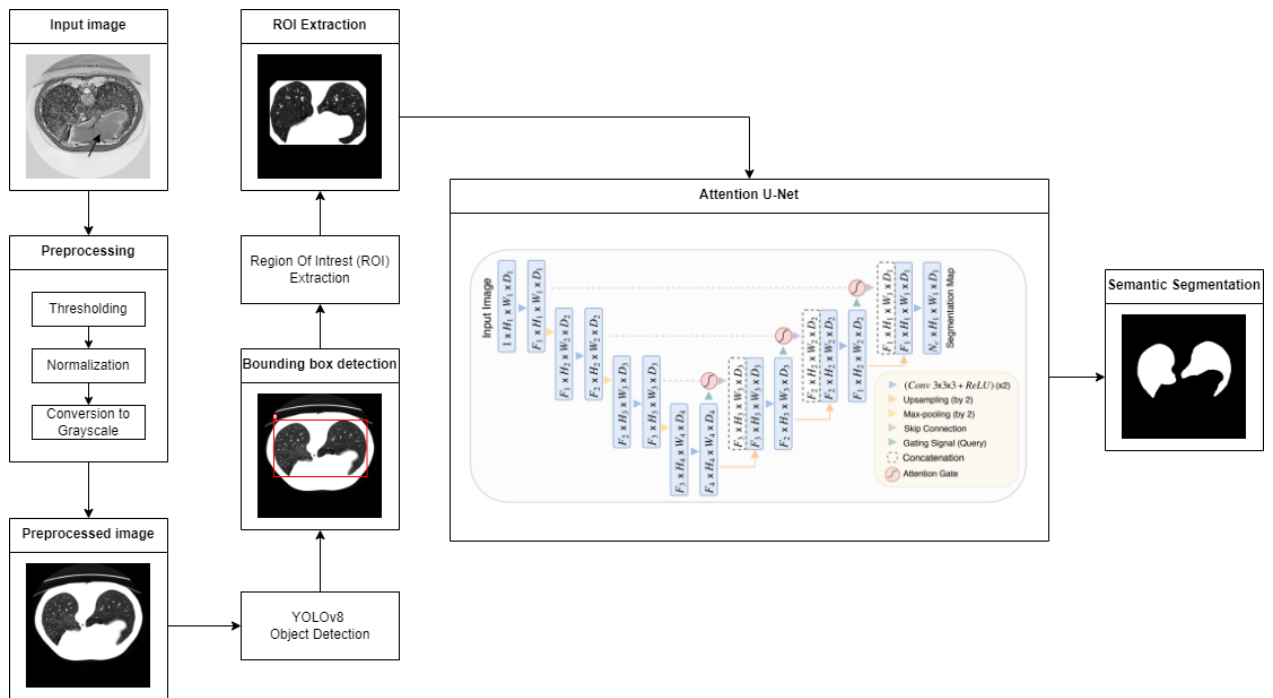


Figure IV.20 Approach 2 Architecture

IV. 5 Results and discussion

This section outlines the experimental setup and presents the results. The results are divided into two parts. The first part discusses the outcomes and analysis of the approaches proposed by our methodology. In the second part, we compare our approaches with three related works.

IV. 5. 1 Performance metrics

The performance of the models was evaluated using the following metrics:

a) Dice Coefficient (Dice):

Measures the overlap between the predicted and ground truth masks. It is particularly useful for evaluating segmentation models, providing a measure of how well the predicted mask matches the ground truth mask. The Dice Coefficient is calculated using the formula:

$$DSC = \frac{2 \times |A \cap B|}{|A| + |B|} \quad (IV.1)$$

Where $|A \cap B|$ is the intersection of the predicted (A) and ground truth (B) masks, and $|A|$ and $|B|$ are the sizes of the predicted and ground truth masks.

b) Intersection over Union (IoU):

Evaluates the overlap between the predicted and ground truth regions by calculating the ratio of the intersection to the union of the regions. Intersection is defined as the area of overlap between the predicted and ground truth regions, while union is defined as the total area covered by both the predicted and ground truth regions. The IoU is calculated using the formula:

$$IoU = \frac{Intersection}{Union} \quad (IV.2)$$

c) Precision:

Quantifies the ability of a model to correctly predict positive samples out of all the samples predicted as positive. It focuses on minimizing false positives and is calculated using the formula:

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

Where TP is the number of true positive samples and FP is the number of false positive samples.

d) Recall:

Measures the proportion of actual positive samples that are correctly identified by the model. It focuses on minimizing false negatives and is calculated using the formula:

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

Where TP is the number of true positive samples and FN is the number of false negative samples.

e) Dice Loss:

Is a loss function used during model training that is based on the Dice Coefficient. It is defined as:

$$L_{Dice} = 1 - DSC \quad (IV.5)$$

Where DSC is the Dice Coefficient. Dice Loss helps optimize the model to maximize the Dice Coefficient, thus improving the segmentation performance.

These metrics provide a comprehensive evaluation of the model's performance in image segmentation tasks, ensuring a robust assessment of its ability to accurately predict the segmented regions compared to the ground truth.

IV. 5. 2 Results obtained for approach 1

The performance of the approach was tracked over 50 epochs, with key metrics such as Dice Coefficient, Intersection over Union (IoU), Loss, Learning Rate (lr), Precision, Recall, and their corresponding validation metrics recorded. The results provide insights into the approach training progress, convergence, and overall performance. Here we discuss these metrics in detail, focusing on trends, convergence behaviors, and the final model performance.

a) Dice Coefficient and IoU

The plots shown in (Figure IV.21) of the Dice Coefficient and IoU reveals a clear trajectory of model performance improvement over the training epochs. The Dice Coefficient, a measure of overlap between predicted and ground truth masks, shows a consistent upward trend, starting from 0.2075 at epoch 0 and reaching 0.9900 by epoch 49. This steady increase indicates that the model

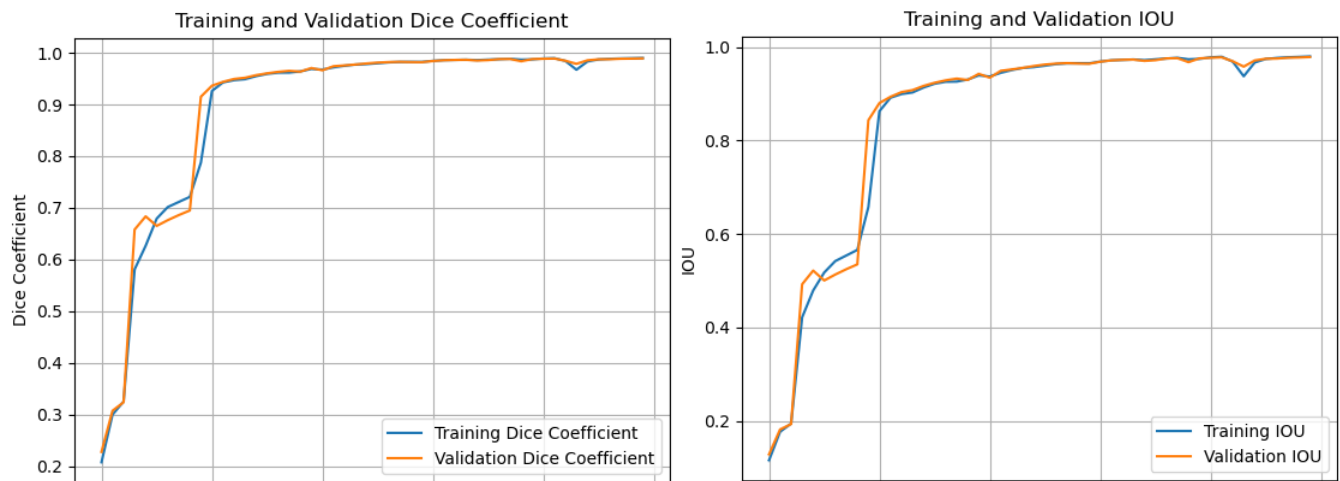


Figure IV.21 Approach 1 Training and Validation Dice Coefficient and IOU Over Epochs

progressively refined its predictions to closely match the ground truth. Similarly, the validation Dice Coefficient mirrors this trend, rising to 0.9788 at epoch 49, suggesting strong generalization to unseen data. However, the rate of improvement in validation Dice Coefficient tapers off after epoch 30, indicating a convergence point. The IoU metric follows a parallel pattern, confirming the model's enhanced ability to accurately predict the overlap between predicted and actual segments over time. These trends collectively demonstrate the model's robust learning and generalization capabilities.

b) Loss

The training loss consistently decreased from 0.7925 at epoch 0 to 0.00998 at epoch 49, indicating the model's progressively improved accuracy in predictions. Similarly, the validation loss followed a decreasing trend, although it exhibited some fluctuations, such as a higher loss at epoch 43. These fluctuations can be attributed to model adjustments and changes in the learning rate. By epoch 49, the validation loss had decreased to 0.01012, closely aligning with the training loss and suggesting minimal overfitting. See (Figure IV.22)

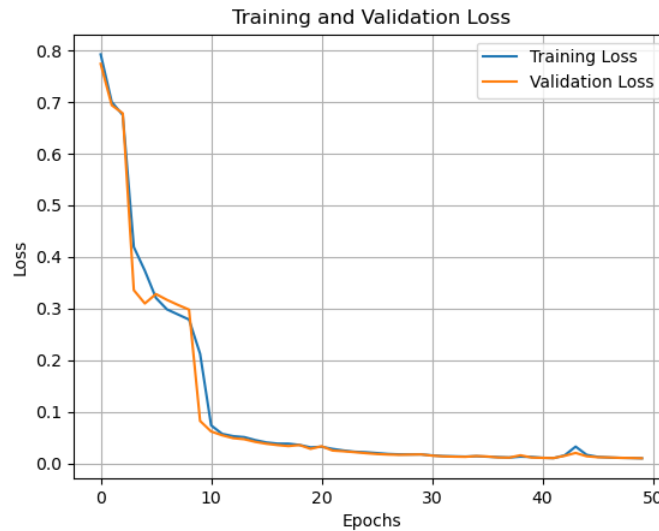


Figure IV.22 Approach 1 Training and Validation Loss Over Epochs

c) Precision and Recall

The precision and recall plots demonstrate the model's performance and balance between true positive identification and sensitivity. Training precision increased significantly from 0.1254 to 0.9916 over the epochs, indicating the model's growing ability to correctly identify true positives.

Similarly, validation precision followed this upward trend, peaking at 0.9945, showcasing the model's robust performance on new data. Training recall remained consistently high, starting near 1 and ending slightly lower at 0.9895, reflecting a balanced performance with high sensitivity. Validation recall also exhibited high consistency, concluding at 0.9871, further confirming the model's robustness and its ability to maintain a balance between precision and recall. See (Figure IV.23)

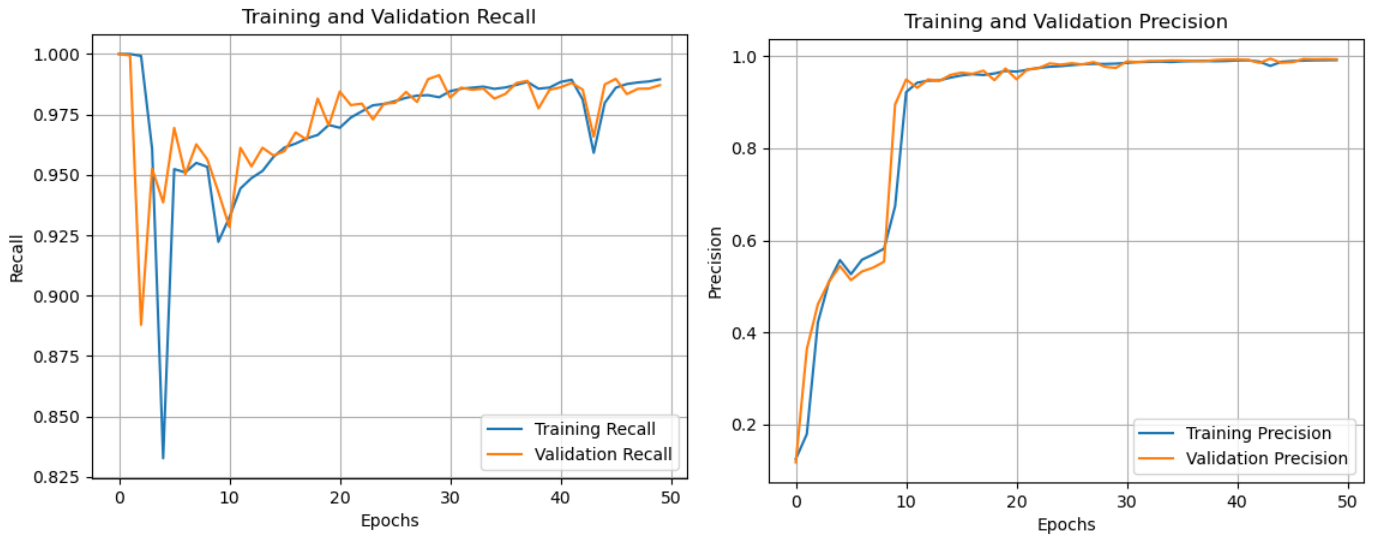


Figure IV.23 Approach 1 Training and Validation Recall and Precision Over Epochs

d) Detailed Epoch Analysis

Early Epochs (0-9): Initial epochs show rapid improvements in Dice Coefficient and IoU, indicating that the model is learning fundamental features and structure. Loss decreases significantly, showing effective learning.

Middle Epochs (10-30): During these epochs, there are consistent improvements in both training and validation metrics, though the rate of improvement starts to plateau, indicating the model is nearing its optimal performance.

Late Epochs (31-49): The metrics stabilize, with slight fluctuations but overall showing minor improvements. This phase suggests the model has largely converged and is fine-tuning its parameters.

e) **Convergence and Overfitting**

The close alignment between training and validation metrics throughout the epochs, particularly in the later stages, suggests good model generalization and minimal overfitting. The slight fluctuations in validation loss and precision are typical in training processes and do not indicate significant overfitting.

f) **Final Approach Performance**

By the end of the training at epoch 49, this approach demonstrates high performance across all metrics shown in the table (Table IV.1):

Table IV.1 Performance Metrics Results of Approach 1

Approach 1									
Dice Coefficient		IoU		Loss		Precision		Recall	
train	val	train	val	train	val	train	val	train	val
0.9900	0.9893	0.9802	0.9788	0.00998	0.01012	0.9916	0.9934	0.9895	0.9871

These results indicate that the model is highly effective at segmenting the target features with excellent accuracy and robustness.

g) **Conclusion**

The detailed analysis of the approach's performance metrics over 50 epochs reveals a well-trained model with strong generalization capabilities. The consistent improvements and eventual stabilization of the metrics suggest that the model has effectively learned the task and is unlikely to benefit from further training under the current settings. Future work could explore fine-tuning the learning rate or introducing advanced regularization techniques to push the performance boundaries even further.

IV. 5. 3 Results obtained for approach 2

Analyzing the performance metrics of the second approach over the 50 epochs, we can observe several trends and points of interest:

a) **Dice Coefficient and IoU:**

The Dice coefficient and IoU metrics exhibit a consistent upward trajectory across epochs (Figure IV.24), with the Dice coefficient initially hovering around 0.198 and ultimately peaking near 0.989, and the IoU starting at 0.111 and reaching approximately 0.979. This consistent improvement indicates a steady enhancement in the approach's segmentation quality throughout

training. Similarly, the validation metrics mirror this trend, with the validation Dice increasing from 0.195 to 0.989 and the validation IoU improving from 0.108 to 0.979. The close correspondence between the training and validation metrics implies robust generalization capabilities of the approach, reinforcing its reliability beyond the training dataset.

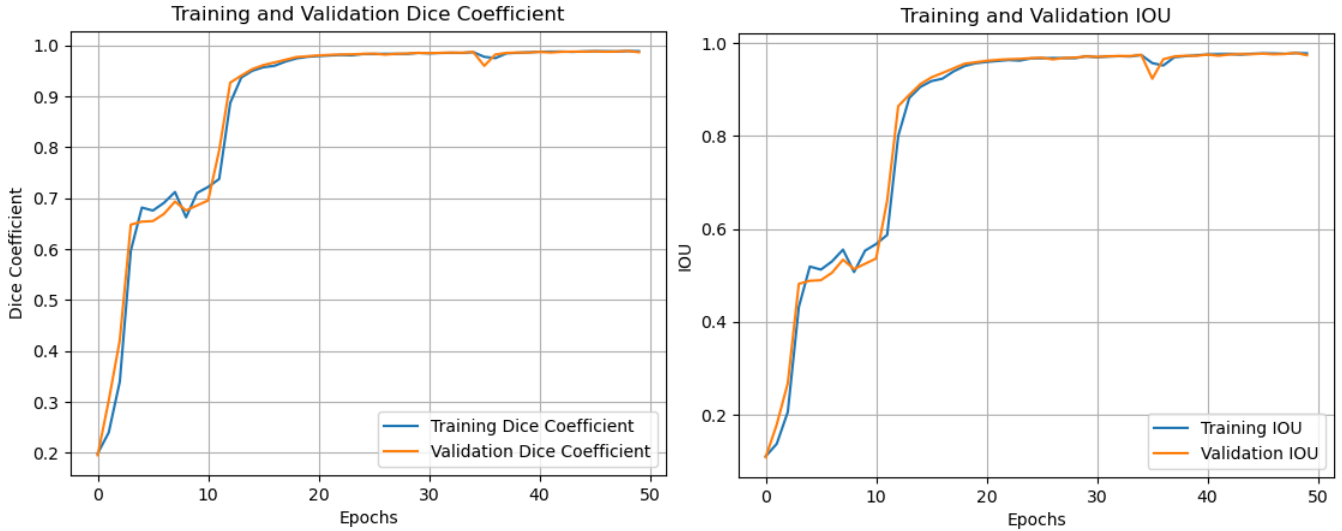


Figure IV.24 Approach 2 Training and Validation Dice Coefficient and IOU Over Epochs

b) Loss:

The loss steadily decreases from an initial value of 0.801 to a remarkably low 0.011, underscoring the approach's improved performance and increasingly accurate predictions as training progresses. Similarly, the validation loss diminishes from 0.806 to 0.010, affirming the approach's ability to generalize effectively and perform well on unseen data, without succumbing

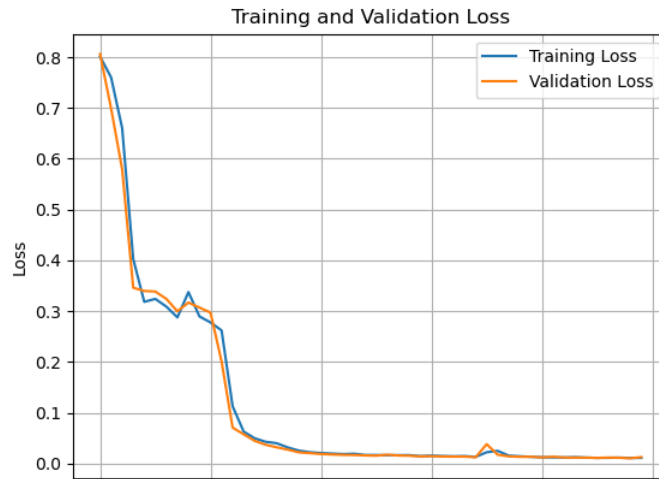


Figure IV.25 Approach 2 Training and Validation Loss Over Epochs

to overfitting. This consistent reduction in both training and validation loss reflects the approach's capacity to learn and adapt to the dataset while maintaining its predictive accuracy across various scenarios (see Figure IV.25).

c) Precision and Recall:

Precision initially starts at a relatively low level but undergoes a substantial increase over the course of training, ultimately maintaining high values towards the conclusion of the epochs. Meanwhile, recall remains consistently high throughout, signifying the model's adeptness at correctly identifying the majority of relevant instances within the dataset. Across the epochs, both precision and recall show significant improvement, with precision reaching approximately 0.996 and recall around 0.987 by the end of the training process. This progression underscores the approach's increasing ability to make precise predictions while effectively capturing relevant instances, culminating in a high-performing and reliable system (Figure IV.26).



Figure IV.26 Approach 2 Training and Validation Recall and Precision Over Epochs

d) Detailed Epochs Analysis

Early Epochs (0-10): In the early epochs, there are rapid improvements observed. For instance, the Dice coefficient rises from 0.198 to 0.722, and IoU from 0.111 to 0.567. The sharp decrease in loss indicates quick learning by the model during this phase. Validation metrics show a similar trend, albeit with a slight lag compared to training metrics.

Mid Epochs (11-30): As the training progresses, the pace of improvement slows down but remains consistent. The Dice coefficient rises from 0.737 to 0.985, and IoU from 0.587 to 0.971,

showing continued enhancement. While the loss continues to decrease, the rate of decrease becomes more gradual over time. However, validation metrics closely track the training metrics, suggesting effective generalization of the approach's performance beyond the training data.

Late Epochs (31-50): The metrics reach a plateau, exhibiting only marginal improvements. The Dice coefficient stabilizes around 0.988, while IoU stabilizes around 0.978, suggesting a convergence in performance. Concurrently, loss values also stabilize at exceptionally low levels. The close alignment between training and validation metrics signifies that the approach has achieved a state of near-optimal performance without notable overfitting, highlighting its robustness and effectiveness in generalizing to unseen data.

e) Final Approach Performance

By the end of the training at epoch 49, the approach 2 demonstrates high performance across all metrics shown in the table (Table IV.2):

Table IV.2 Performance Metrics Results of Approach 2

Approach 2									
Dice Coefficient		IoU		Loss		Precision		Recall	
train	val	train	val	train	val	train	val	train	val
0.9889	0.9868	0.9780	0.9740	0.01113	0.01261	0.9912	0.9960	0.9878	0.9795

f) Conclusion:

Approach 2 demonstrates strong performance and generalization ability. The steady improvement across all key metrics suggests that the approach has been effectively trained. High precision and recall further indicate the model's robustness and reliability in practical segmentation tasks.

IV. 5. 4 Results comparison table

In this section, we conducted a comparative analysis of the performance of our two proposed approaches Approach 1 (YOLOv8 with U-Net) and Approach 2 (YOLOv8 with Attention U-Net) against the direct approaches : U-Net and Attention U-Net. The evaluation was based on a comprehensive set of metrics including Dice coefficient, Intersection over Union (IoU), loss, precision, and recall. The results presented correspond to the 49th epoch for each approach (Table IV.3).

Table IV.3 Results Comparison Table

Approach	Dice Coefficient		IoU		Loss		Precision		Recall	
	train	val	train	val	train	val	train	val	train	val
U-Net	0.9830	0.9882	0.9668	0.9767	0.01699	0.01142	0.9840	0.9937	0.9838	0.9848
Attention U-Net	0.9812	0.9874	0.9635	0.9752	0.0187	0.01220	0.9825	0.9918	0.9820	0.9854
YOLOv8 U-Net	0.9900	0.9893	0.9802	0.9788	0.00998	0.01012	0.9916	0.9934	0.9895	0.9871
YOLOv8 Attention U-Net	0.9889	0.9868	0.9780	0.9740	0.01113	0.01261	0.9912	0.9960	0.9878	0.9795

a) Dice Coefficient and IoU:

Among the segmentation approaches evaluated, U-Net attained a Dice coefficient of 0.983 and an Intersection over Union (IoU) of 0.967. Attention U-Net achieved slightly lower scores, with a Dice coefficient of 0.981 and an IoU of 0.964. YOLOv8 U-Net exhibited the highest performance, boasting a Dice coefficient of 0.990 and an IoU of 0.980, surpassing all other approaches. YOLOv8 Attention U-Net also delivered outstanding results, with a Dice coefficient of 0.989 and an IoU of 0.978, demonstrating its effectiveness in segmentation tasks.

b) Loss:

In terms of loss, U-Net incurred a value of 0.017, while Attention U-Net registered a marginally higher loss at 0.019. YOLOv8 U-Net demonstrated the most efficient performance in this aspect, achieving the lowest loss of 0.010. Conversely, YOLOv8 Attention U-Net experienced a slightly elevated loss of 0.011. These findings illustrate the varying degrees of efficiency among the approaches in minimizing loss during the segmentation process.

c) Precision and Recall:

In terms of precision and recall, U-Net achieved values of 0.984 and 0.984, respectively. Attention U-Net exhibited slightly lower precision and recall metrics, recording values of 0.983 and 0.982, respectively. YOLOv8 U-Net emerged as the top performer in this regard, showcasing the highest precision and recall values of 0.992 and 0.990. Similarly, YOLOv8 Attention U-Net delivered exceptional results, with precision and recall values of 0.991 and 0.988, demonstrating its effectiveness in accurately identifying relevant instances within the dataset.

d) **Validation Metrics:**

The validation metrics generally followed similar trends to the training metrics for all approaches, indicating good generalization ability.

e) **Discussion:**

Model Performance: Among the approaches evaluated, YOLOv8 U-Net and YOLOv8 Attention U-Net outperformed traditional U-Net and Attention U-Net in terms of Dice coefficient, IoU, loss, precision, and recall. This suggests that integrating YOLOv8 architecture with U-Net significantly improves segmentation performance.

Trade-offs: While YOLOv8 U-Net achieved the lowest loss and highest Dice coefficient, it's essential to consider the trade-offs between precision and recall. YOLOv8 Attention U-Net, despite slightly lower performance metrics, still maintains a balance between precision and recall, making it a robust choice for segmentation tasks.

Generalization: The validation metrics closely mirrored the training metrics for all approaches, indicating that they generalize well to unseen data. This suggests that the approaches have learned meaningful representations from the training data.

f) **Conclusion:**

In conclusion, the integration of YOLOv8 architecture with U-Net, particularly in the Attention U-Net variant, shows promising results for medical image segmentation tasks. While traditional U-Net and Attention U-Net remain competitive, the YOLOv8-based variants demonstrate superior performance across multiple evaluation metrics, making them favorable choices for practical segmentation applications.

IV. 5. 5 State of the art comparison table

Our proposed approaches in comparison to existing state of the art methods (Table IV.4):

Table IV.4 Our Proposed Approaches Vs State of the Art

Approach	Dice Coefficient		IoU		Loss		Precision		Recall	
	train	val	train	val	train	val	train	val	train	val
YOLOv8 U-Net	0.9900	0.9893	0.9802	0.9788	0.00998	0.01012	0.9916	0.9934	0.9895	0.9871
YOLOv8 Attention U-Net	0.9889	0.9868	0.9780	0.9740	0.01113	0.01261	0.9912	0.9960	0.9878	0.9795

State of Art: Bhattacharjee, A., et al.	0.9630	0.9882	0.9668	0.9767	0.01699	0.01142	0.9840	0.9937	0.9838	0.9848
State of Art: Qinhua, Hu., et al.	0.9812	0.9874	0.9635	0.9752	0.0187	0.01220	0.9825	0.9918	0.9820	0.9854
State of Art: Khanna, A., et al.	0.9863	0.9863	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

The results indicate that both our approaches YOLOv8 U-Net and YOLOv8 Attention U-Net outperform the state-of-the-art approaches in terms of Dice coefficient, IoU, loss, precision, and recall. YOLOv8 U-Net achieves the highest Dice coefficient and IoU, indicating superior segmentation accuracy. Additionally, it demonstrates lower loss values compared to the state-of-the-art methods, showcasing better optimization during training. Both YOLOv8 variants also exhibit higher precision and recall, suggesting more accurate identification of relevant instances within the dataset. However, further analysis is required to compare the generalization ability and computational efficiency of these approaches against the state-of-the-art methods.

IV. 6 Web Application Development

Lung Vision is a web application designed for managing medical images, patients, and scans, with integrated functionalities for segmenting scans using the developed machine learning models. The application is built using Django for the backend, PostgreSQL for the database, and React.js for the frontend.

IV. 6. 1 Architecture

In our system architecture (Figure IV.27), Django serves as the backend framework, responsible for managing user authentication, data handling, and facilitating the deployment of machine learning models utilized for image segmentation tasks. The backend is further equipped with a REST API, enabling seamless communication with the frontend. PostgreSQL serves as the database management system, offering robust storage capabilities for patient data, scan metadata, and segmentation results. Its implementation ensures data integrity and supports the execution of complex queries essential for efficiently managing medical records. On the frontend, React.js is employed to develop a responsive and interactive user interface. Leveraging React.js, users can effortlessly upload medical images, access patient records, and visualize segmentation outcomes, thereby enhancing the overall user experience.

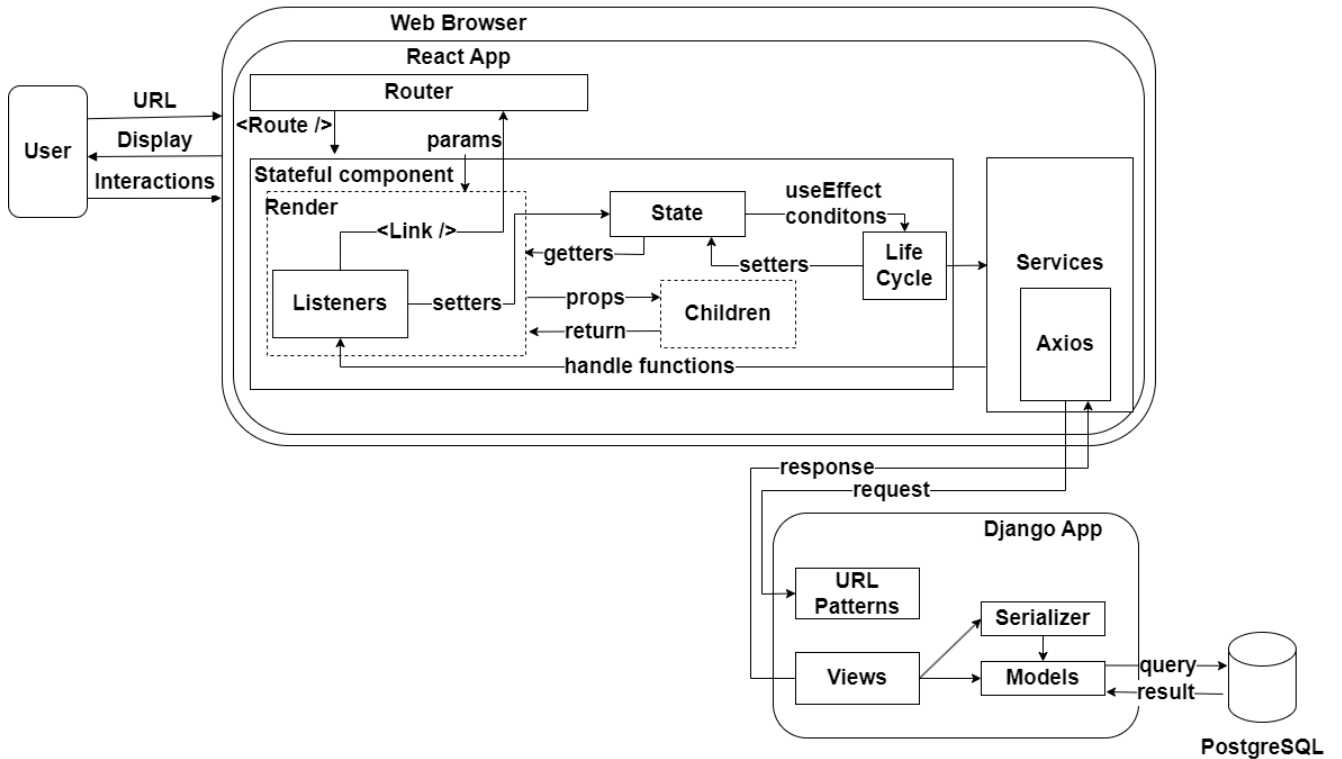


Figure IV.27 Architectural Overview of the Lung Vision Web Application

IV. 6. 2 Features

The proposed system encompasses a range of features designed to facilitate efficient and secure management of medical data while enabling advanced analysis and visualization capabilities. The key features of the system are outlined below:

User Authentication: A robust user authentication system is implemented to ensure secure login and registration processes. Authentication mechanisms are designed to safeguard sensitive medical information and comply with relevant privacy regulations.

Patient Management: The system provides comprehensive CRUD (Create, Read, Update, Delete) operations for managing patient records. This includes functionalities for adding new patients, updating existing records, retrieving patient information, and removing outdated data.

Scan Management: Users can upload, view, and manage CT scans within the system. The platform supports the seamless uploading of medical images, facilitating easy access and retrieval of scan data for analysis and review.

Segmentation: Leveraging state-of-the-art machine learning models developed as part of this research, the system offers advanced segmentation capabilities. Lung regions within CT scans can be accurately segmented using the developed ML models, enabling precise delineation of anatomical structures and pathological regions.

Data Visualization: The system incorporates interactive visualization tools for exploring segmentation results and patient data. Visualizations allow users to gain insights into medical imaging data, facilitating interpretation and analysis. Interactive features enable users to manipulate and explore data dynamically, enhancing understanding and decision-making.

These features collectively empower healthcare professionals with tools for efficient management, analysis, and visualization of medical data. By integrating advanced machine learning algorithms for segmentation and providing intuitive visualization capabilities, the system facilitates enhanced clinical decision-making and patient care.

IV. 6. 3 UML Diagrams

In this section, we provide a comprehensive overview of the Lung Vision Web Application's architecture and functionality through Unified Modeling Language (UML) diagrams. These diagrams serve as visual aids to understand the system's structure, interactions, and processes. We commence with the Use Case Diagram (Figure IV.28), offering a high-level depiction of the system's functionalities from a user's perspective. Following this, the Class Diagram (Figure IV.29) provides insight into the system's object-oriented design, showcasing the classes, attributes, and relationships within the application. Finally, the Sequence Diagram (Figure IV.30) illustrates the step-by-step interactions between various components during the segmentation process, elucidating the system's dynamic behavior. Together, these UML diagrams offer a comprehensive understanding of the Lung Vision Web Application's architecture and functionality, facilitating effective system comprehension and development.

a) Use Case Diagram

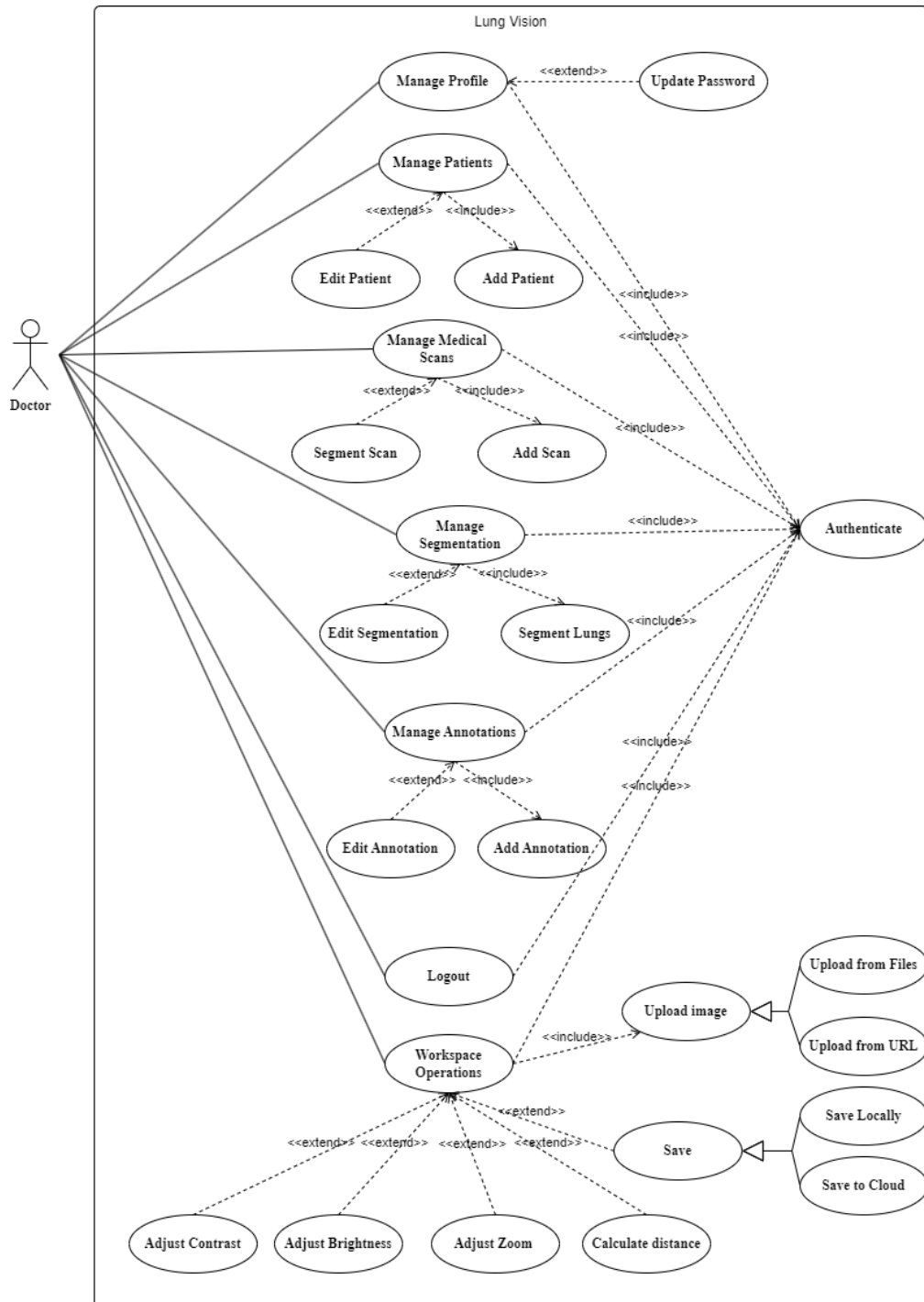


Figure IV.28 Use case diagram of the Lung Vision Web Application

b) Entity Relationship Diagram

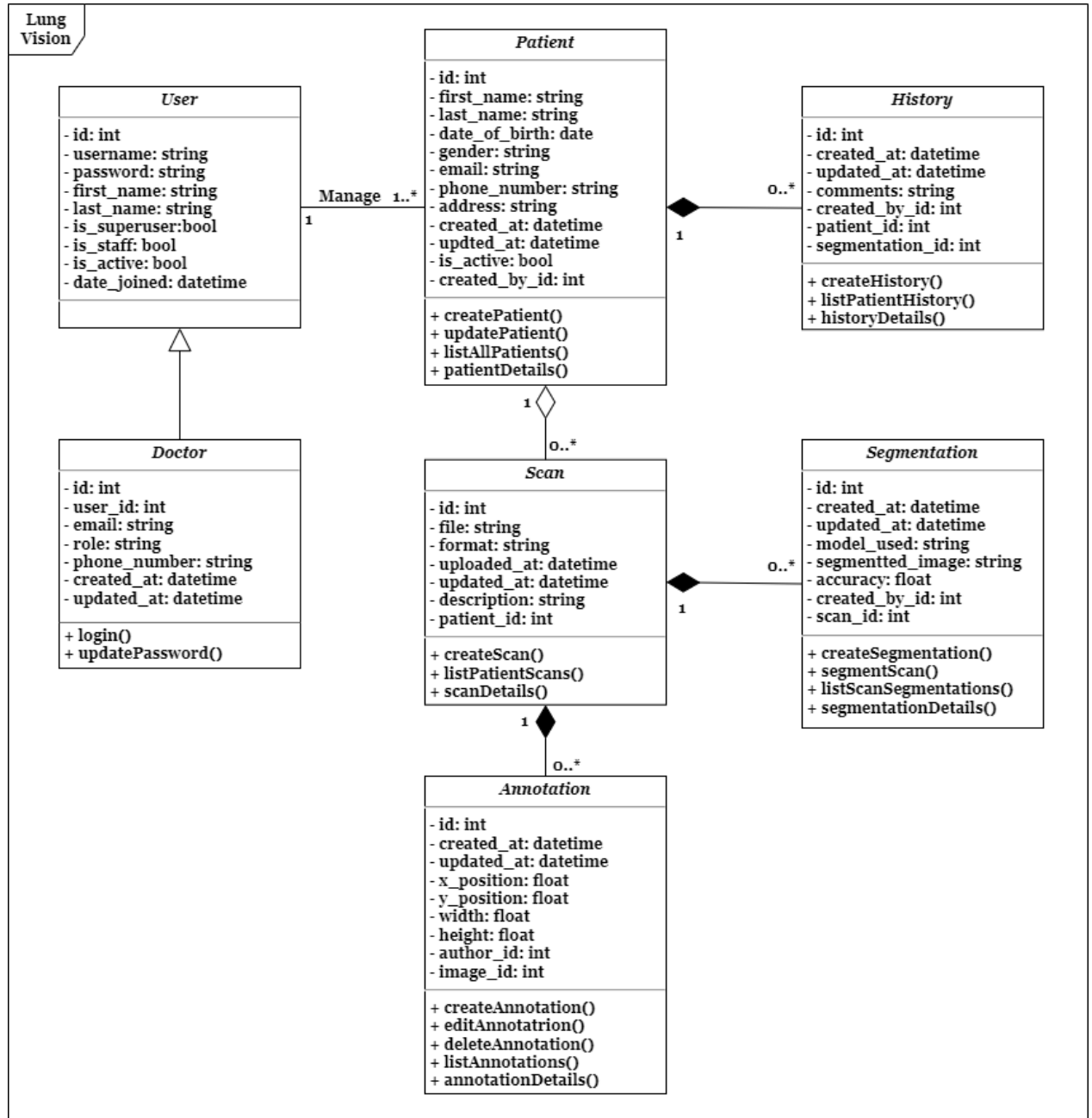


Figure IV.29 Entity Relationship Diagram of the Lung Vision Web Application

c) Sequence Diagram

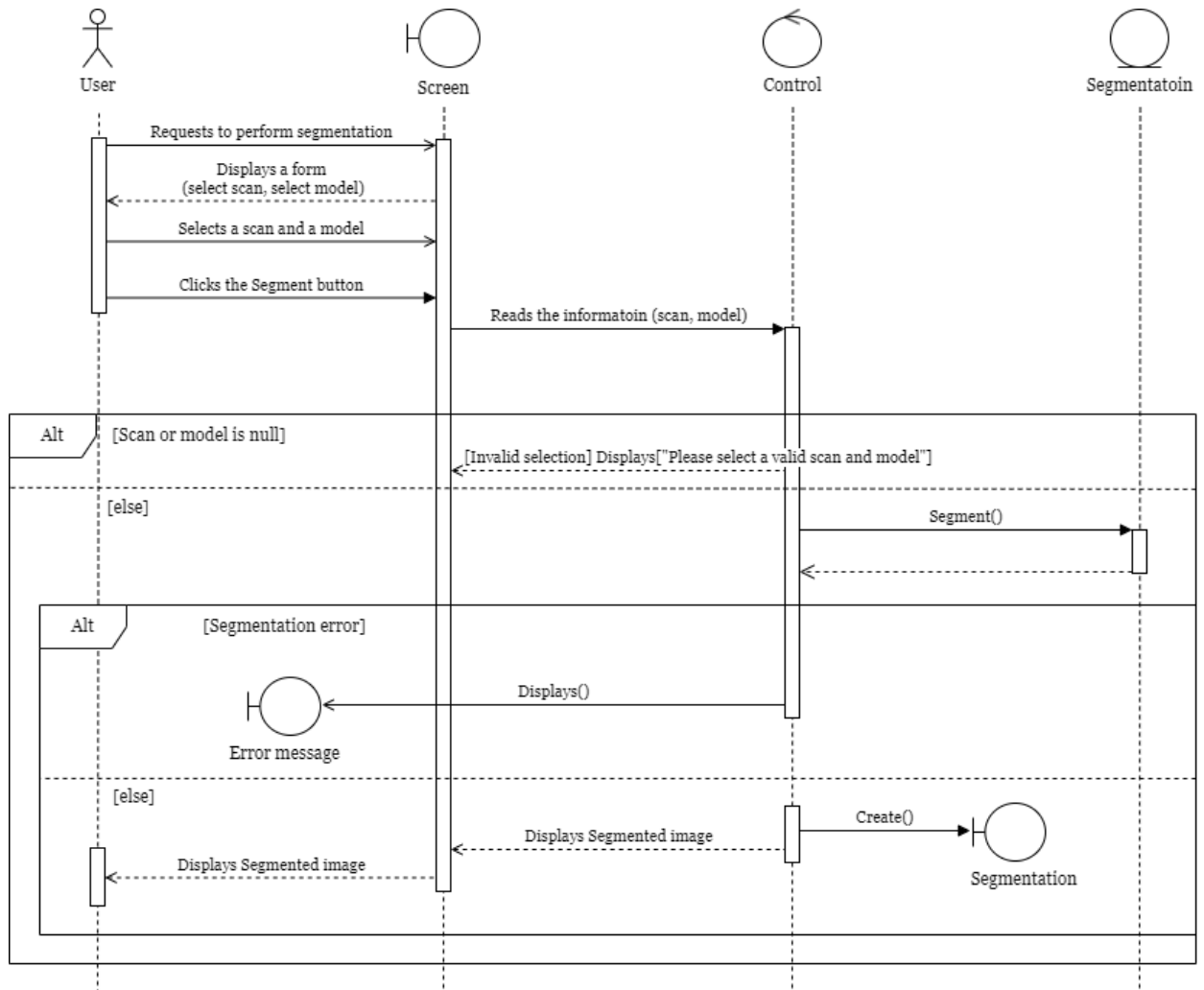


Figure IV.30 Sequence diagram of the segmentation

IV. 6. 4 Graphical User Interface (GUI)

In this section, we present the user interface of the Lung Vision web application, highlighting its key features and functionalities. The following UI captures provide an overview of the application’s interface and user experience.

a) User Authentication

The login screen allows users to securely log into the application. Users must provide their credentials to access the system (see Figure IV.31).

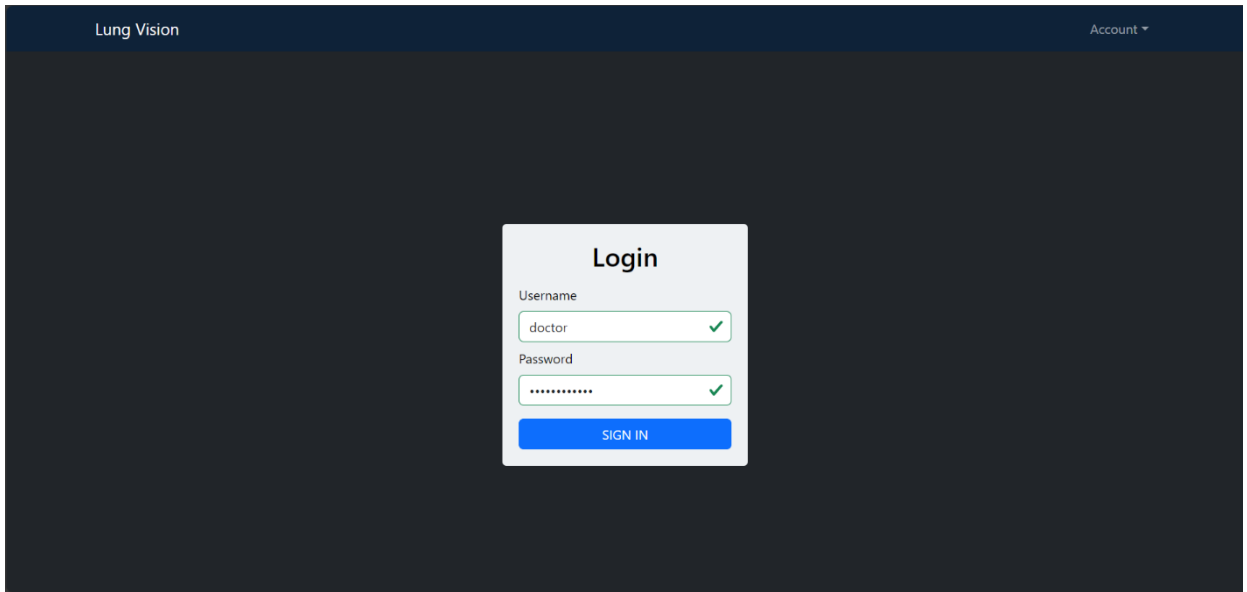


Figure IV.31 User Authentication - Login Screen

b) Dashboard

The dashboard provides an overview of the application's status and key metrics. Doctors can see the number of active patients, total patients, and a list of all patients (see Figure IV.32).

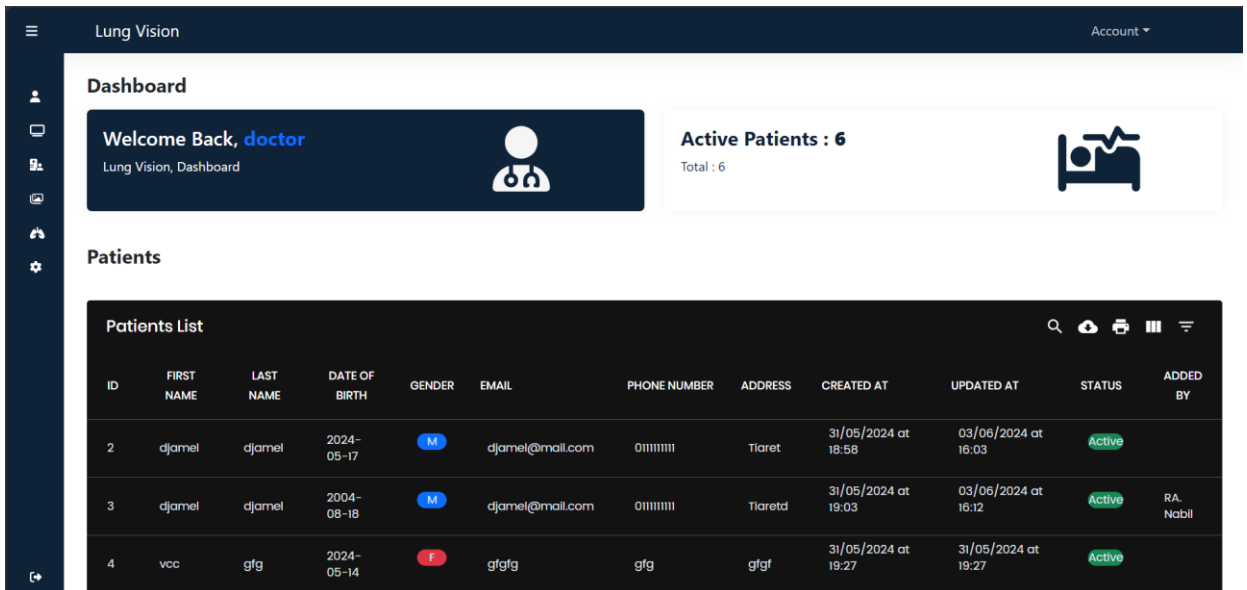


Figure IV.32 Dashboard Screen

c) Patient Management

The patient management interface enables users to perform CRUD operations on patient records. This includes adding new patients, updating existing records, and deleting records if necessary (see Figure IV.33).

ID	FIRST NAME	LAST NAME	DATE OF BIRTH	GENDER	EMAIL	PHONE NUMBER	ADDRESS	CREATED AT	UPDATED AT	STATUS	ADDED BY
2	djamel	djamel	2024-05-17	M	djamel@mail.com	011111111	Tiaret	31/05/2024 at 18:58	03/06/2024 at 16:03	Active	
3	djamel	djamel	2004-08-18	M	djamel@mail.com	011111111	Tiaret	31/05/2024 at 19:03	03/06/2024 at 16:12	Active	RA, Nabil
4	vcc	gfg	2024-05-14	F	gfgfg	gfg	gfgf	31/05/2024 at 19:27	31/05/2024 at 19:27	Active	
5	react	react	2001-01-18	M	mail@mail.com	0555555555	Tiaret	02/06/2024 at 15:16	02/06/2024 at 15:16	Active	
6	user	user	2024-06-28	M	user@mail.com	0555555555	user	02/06/2024 at 15:30	02/06/2024 at 15:30	Active	

Add Patient

Note: All fields are required

First Name:

Last Name:

Date of Birth:

Gender:

Email:

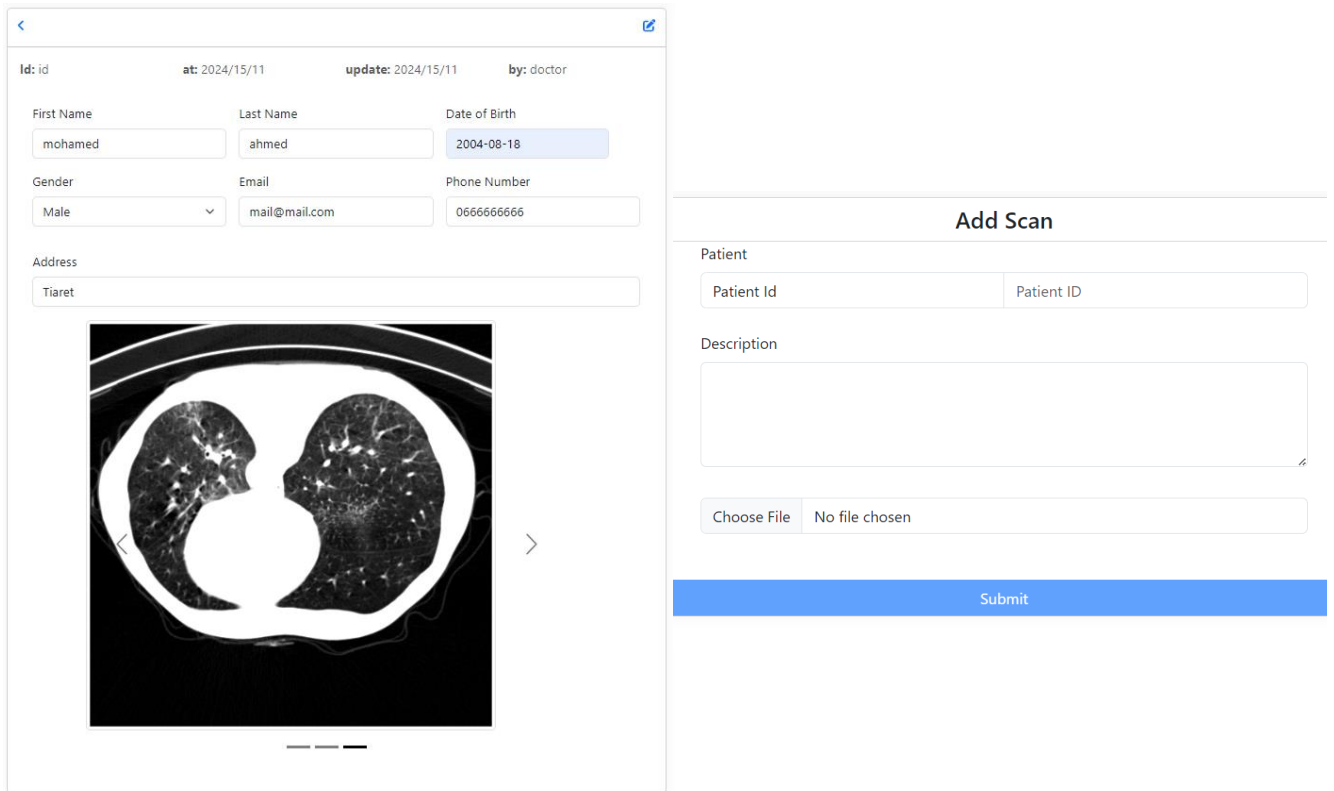
Phone Number:

Address:

Figure IV.33 Patient Management Screens

d) Scan Management

Users can upload and manage CT scans through this interface. The uploaded scans are displayed in a list format, allowing users to easily access and view them (see Figure IV.34).



The image displays two side-by-side screenshots of a web application interface for scan management.

The left screenshot shows a patient profile form with the following fields and values:

- id: id
- at: 2024/15/11
- update: 2024/15/11
- by: doctor
- First Name: mohamed
- Last Name: ahmed
- Date of Birth: 2004-08-18
- Gender: Male
- Email: mail@mail.com
- Phone Number: 0666666666
- Address: Tiaret

Below the form is a CT scan image of a chest cross-section, with left and right navigation arrows.

The right screenshot shows the 'Add Scan' form with the following fields:

- Patient: Patient Id (dropdown menu)
- Description: Text area
- Choose File: No file chosen
- Submit: Blue button

Figure IV.34 Scan Management Screens

e) Segmentation

The segmentation feature allows users to select a scan and apply machine learning approaches (U-Net, Attention U-Net, YOLOv8 with UNet, YOLOv8 with Attention UNet) to segment lung regions. The results are displayed interactively (see Figure IV.35).

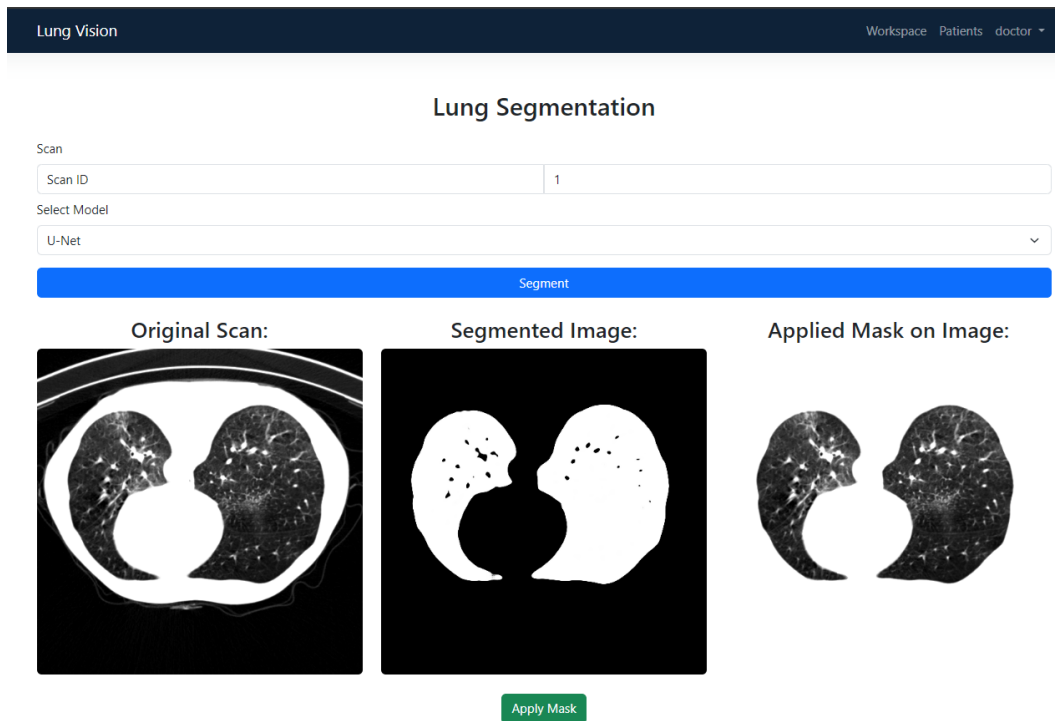


Figure IV.35 Segmentation Screen

f) Workspace

The workspace interface provides tools for users to interact with the uploaded scans. Users can adjust contrast, brightness, calculate distances between points, zoom, rotate, and save their adjustments (see Figure IV.36).

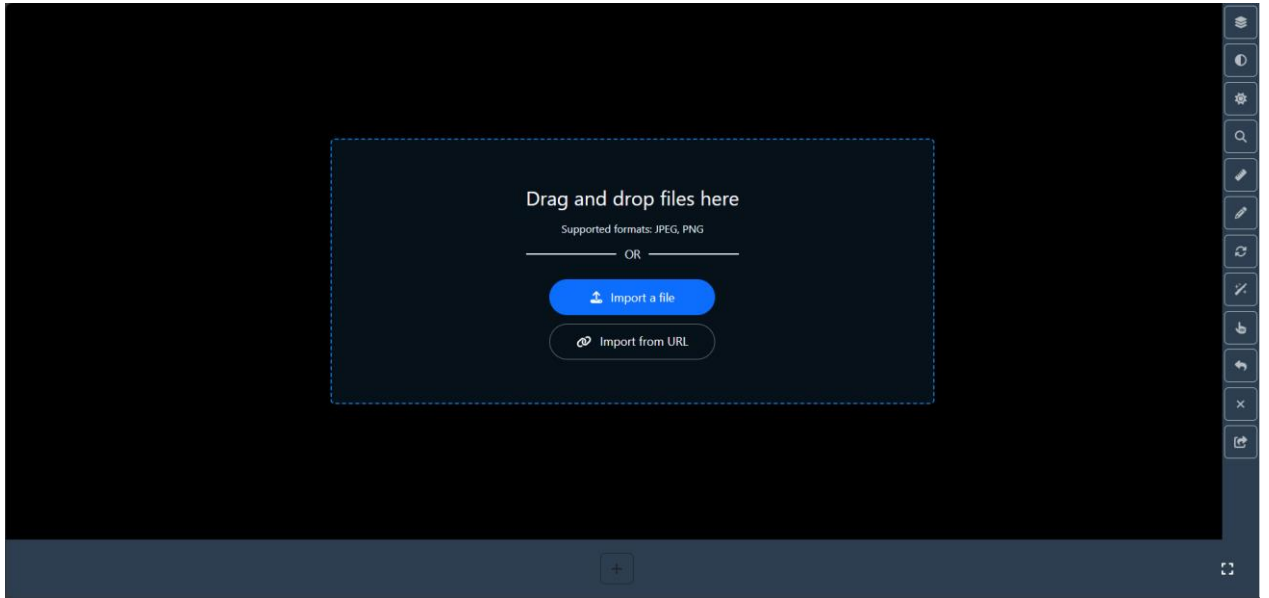


Figure IV.36 Workspace Screen

IV. 7 Conclusion

In this chapter, we presented the experimental setup, realization, and results of our proposed models for lung segmentation in CT images. We utilized state-of-the-art deep learning techniques, including YOLOv8 for lung detection and U-Net/Attention U-Net for segmentation. Additionally, we introduced Lung Vision, a web application designed to manage medical images and segment scans using the developed models. Our models demonstrated high accuracy and robustness, making them suitable for clinical applications and aiding in the early diagnosis and treatment of lung diseases. Future work will focus on further improving the models performance and extending their applicability to other organ segmentation tasks.

General Conclusion

General Conclusion

This thesis centers on advancing the preprocessing and semantic segmentation of medical images, with a focus on lung diseases, utilizing cutting-edge AI techniques. It starts by providing a deep dive into artificial vision, elucidating its pivotal role in medical imaging and paving the way for subsequent research endeavors. Building upon this foundation, the exploration extends into the broader realm of artificial intelligence, underscoring its transformative potential in automating and refining medical image analysis, particularly in diagnosing lung ailments. A critical review of the current state of the art highlights both the strengths and limitations of existing segmentation methodologies, guiding the thesis towards addressing key research gaps. The experimental phase showcases the efficacy of integrating advanced AI models, such as YOLOv8 and U-Net, for precise lung segmentation from CT scans. Furthermore, the development of Lung Vision, a user-friendly web application, underscores the practical implementation of AI in clinical workflows, promising enhanced patient care through more accurate diagnosis and treatment planning.

Key Contributions

This thesis has made several key contributions to the field of medical image analysis:

- **Advanced Preprocessing Techniques:** The research introduced effective preprocessing steps, including cropping, normalization, and grayscale conversion, which enhanced the quality and consistency of the CT images used for segmentation.
- **Innovative AI Models:** The development and integration of YOLOv8 with U-Net and Attention-UNet models provided a novel approach to lung segmentation, achieving high accuracy and efficiency.
- **Practical Application through Lung Vision:** The creation of Lung Vision demonstrated the practical applicability of the research, offering a comprehensive tool for medical professionals to manage and analyze lung images effectively.
- **Comprehensive State-of-the-Art Analysis:** The thorough review of existing methodologies provided valuable insights and identified gaps, guiding the research towards addressing specific challenges in lung image segmentation.

Implications and Future Work

The findings of this research have significant implications for the field of medical imaging and AI. The proposed methodologies can enhance early diagnosis and treatment of lung diseases, potentially improving patient outcomes. The integration of these techniques into clinical workflows through applications like Lung Vision can streamline processes and reduce the burden on healthcare professionals.

Future work could focus on further refining the AI models to handle more complex cases and expanding their application to other organs and diseases. Additionally, extensive clinical trials and collaborations with healthcare institutions will be essential to validate and improve the robustness and generalizability of the proposed solutions.

Conclusion

In conclusion, this thesis has successfully demonstrated the potential of AI techniques in preprocessing and semantic segmentation of medical images of lung diseases. By combining advanced AI models with practical applications, this research has paved the way for more accurate, efficient, and user-friendly tools in medical imaging. The comparative analysis of the proposed models (YOLOv8 with U-Net and YOLOv8 with Attention U-Net) against traditional segmentation models (U-Net and Attention U-Net) over 49 epochs demonstrated that the YOLOv8-based models significantly outperformed the traditional models across several key metrics. YOLOv8 U-Net achieved the highest Dice coefficient (0.990) and IoU (0.980), as well as the lowest loss (0.010), indicating superior segmentation performance. Both YOLOv8 models also demonstrated higher precision and recall, underscoring their robust performance and generalization capabilities. These contributions offer a solid foundation for future advancements in the field, promising significant benefits for both patients and healthcare provider.

Bibliography

1. Seemann, T.: Digital image processing using local segmentation. Doctoral dissertation, Monash University (2002)
2. MESSAOUDI, A.: Contribution à l'amélioration et la mise en oeuvre de nouveaux algorithmes pour la compression des signaux. Doctoral dissertation, University of Batna 2 (2017)
3. KARFOUF, M. A., BEDDIAF, A.: Reconnaissance des expressions faciales. Doctoral dissertation, University of Ouargla (2022)
4. Selsabile, L.: Compression of color image by DCT. Master's thesis, University of Biskra (2022)
5. Alabd, A. R. I: Medical video enhancement. Master's thesis, Fen Bilimleri Enstitüsü (2019)
6. Rafael, C. G., Richard, E. W.: Digital image processing. Pearson education (2018)
7. Introduction to Image Processing Engineers Garage, <https://www.engineersgarage.com/introduction-to-image-processing/>, last accessed 2024/05/04
8. Islam, S.M.S., Nasim, M.A.A., Hossain, I., Ullah, D.M.A., Gupta, D.K.D., Bhuiyan, M.M.H.: Introduction of Medical Imaging Modalities. In: Zheng, B., Andrei, S., Sarker, M.K., Gupta, K.D. (eds) Data Driven Approaches on Medical Imaging, pp. 1-25. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-47772-0_1
9. Larobina, M., Murino, L.: Medical image file formats. Journal of digital imaging, vol. 27, pp. 200-206 (2014). <https://doi.org/10.1007/s10278-013-9657-9>
10. What is Computer Vision? IBM. <https://www.ibm.com/topics/computer-vision#citation1>, last accessed 2024/05/06
11. Difference between Image Processing and Computer Vision GeeksforGeeks. <https://www.geeksforgeeks.org/difference-between-image-processing-and-computer-vision/>, last accessed 2024/05/07
12. IBM Data and AI Team. Types of Artificial Intelligence IBM. <https://www.ibm.com/think/topics/artificial-intelligence-types>, last accessed 2024/05/11
13. IBM. Artificial Intelligence in Medicine IBM. <https://www.ibm.com/topics/artificial-intelligence-medicine>, last accessed 2024/05/11
14. El Naqa, I., Murphy, M. J.: What is machine learning?. In: El Naqa, I., Li, R., Murphy, M. (eds) Machine Learning in Radiation Oncology. pp. 3-11. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18305-3_1
15. Sarker, I. H.: Machine learning: Algorithms, real-world applications and research directions. SN Computer Science, vol. 2, pp. 160 (2021). <https://doi.org/10.1007/s42979-021-00592-x>
16. Jason, B.: Master Machine Learning Algorithms: discover how they work and implement them from scratch. Machine Learning Mastery (2016).
17. Types of Machine Learning - Definition, Examples and Applications Medium. <https://medium.com/@growsolutions/types-of-machine-learning-definition-examples-and-applications-3f55ebd88251>, last accessed 2024/05/11

18. Almars, A.: Deepfakes detection techniques using deep learning: a survey. *Journal of Computer and Communications*, vol. 9, pp. 20-35 (2021).
<https://doi.org/10.4236/jcc.2021.95003>
19. Rechercher - éducol STI.
<https://eduscol.education.fr/sti/sites/eduscol.education.fr.sti/files/ressources/pedagogiques/14500/14500-brief-introduction-to-artificial-neural-networks-ensps.pdf>, last accessed 2024/05/11
20. Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L., Alves, S.: *Artificial neural networks: a practical course*. Springer (2016).
21. *Neural Networks Fundamentals* Medium. <https://towardsdatascience.com/neural-networks-fundamentals-1b4c46e7dbfe>, last accessed 2024/05/12
22. Zhang, A., Lipton, Z. C., Li, M., Smola, A. J.: *Dive into deep learning*. arXiv preprint (2021).
<https://doi.org/10.48550/arXiv.2106.11342>
23. *Activation Functions in Neural Networks* Medium. <https://bouzouitina-hamdi.medium.com/activation-functions-in-neural-networks-1c1de2c866a>, last accessed 2024/05/13
24. *Activation Functions in Neural Networks: With 15 examples* Encord.
<https://encord.com/blog/activation-functions-neural-networks/>, last accessed 2024/05/13
25. *Optimizers in deep learning* Medium. <https://musstafa0804.medium.com/optimizers-in-deep-learning-7bf81fed78a0>, last accessed 2024/05/13
26. Géron, A.: *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems*. 3rd edn. O'Reilly Media, Inc (2022).
27. Hassan, E., Shams, M. Y., Hikal, N. A., Elmougy, S.: *The effect of choosing optimizer algorithms to improve computer vision tasks: a comparative study*. *Multimedia Tools and Applications*, vol. 82, pp. 16591-16633 (2023). <https://doi.org/10.1007/s11042-022-13820-0>
28. Sehil, A. M.: *Image Classification using Deep Learning*. Doctoral dissertation, University of Ghardaia (2019).
29. *Deep learning series 1: Intro to deep learning* Medium <https://medium.com/intro-to-artificial-intelligence/deep-learning-series-1-intro-to-deep-learning-abb1780ee20>, last accessed 2024/05/14
30. *What's the difference between Machine Learning and Deep Learning?* Viso.ai.
<https://viso.ai/deep-learning/deep-learning-vs-machine-learning>, last accessed 2024/05/15
31. *Difference Between Machine Learning and Deep Learning* GeeksforGeeks.
<https://www.geeksforgeeks.org/difference-between-machine-learning-and-deep-learning/>, last accessed 2024/05/15
32. Goodfellow, I., Bengio, Y., Courville, A.: *Deep learning*. MIT press (2016).
33. Purwaningsih, T., Anjani, I. A., Utami, P. B.: *Convolutional neural networks implementation for chili classification*. *International Symposium on Advanced Intelligent Informatics (SAIN)*, pp. 190-194. IEEE, Indonesia (2018). <https://doi.org/10.1109/SAIN.2018.8673373>

34. 5 Common Architectures in Convolution Neural Networks (CNN) Analytics Steps.
<https://www.analyticssteps.com/blogs/common-architectures-convolution-neural-networks>,
last accessed 2024/05/15
35. Indolia, S., Goswami, A. K., Mishra, S. P., Asopa, P.: Conceptual understanding of convolutional neural network-a deep learning approach. *Procedia Computer Science*, vol. 132, pp. 679-688 (2018). <https://doi.org/10.1016/j.procs.2018.05.069>
36. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324 (1998).
<https://doi.org/10.1109/5.726791>
37. A Guide to Convolutional Neural Networks v7labs.
<https://www.v7labs.com/blog/convolutional-neural-networks-guide>, last accessed 2024/05/16
38. Convolutional Neural Networks (CNN) Overview Encord.
<https://encord.com/blog/convolutional-neural-networks-explained>, last accessed 2024/05/2024
39. Krizhevsky, A., Sutskever, I., Hinton, G. E.: ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, vol. 25, pp. 1097-1105 (2012).
40. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. *arXiv preprint* (2014). <https://doi.org/10.48550/arXiv:1409.1556>.
41. Zhang, A., Lipton, Z. C., Li, M., Smola, A. J.: *Dive into deep learning*. Cambridge University Press (2023).
42. *Transfer Learning for Machine Learning Seldon*. <https://www.seldon.io/transfer-learning>, last accessed 2024/05/15
43. Explain transfer learning in the context of deep learning.
<https://www.freetimelearning.com/software-interview-questions-and-answers.php?Explain-transfer-learning-in-the-context-of-deep-learning.&id=4184>, last accessed 2024/05/15
44. Bhattacharjee, A., Murugan, R., Goel, T., Soni, B.: Semantic segmentation of lungs using a modified U-Net architecture through limited Computed Tomography images. *2021 Advanced Communication Technologies and Signal Processing (ACTS)*, pp. 1-6. IEEE, India (2021).
<https://doi.org/10.1109/ACTS53447.2021.9708190>
45. Hu, Q., Souza, L. F. D. F., Holanda, G. B., Alves, S. S., Silva, F. H. D. S., Han, T., Reboucas Filho, P. P.: An effective approach for CT lung segmentation using mask region-based convolutional neural networks. *Artificial Intelligence in Medicine*, vol. 103, 101792 (2020).
<https://doi.org/10.1016/j.artmed.2020.101792>
46. Khanna, A., Londhe, N. D., Gupta, S., Semwal, A.: A deep Residual U-Net convolutional neural network for automated lung segmentation in computed tomography images. *Biocybernetics and Biomedical Engineering*, vol. 40, no. 3, pp. 1314-1327 (2020).
<https://doi.org/10.1016/j.bbe.2020.07.007>

47. The Python Tutorial - Python 3.7.4 documentation Python.org. <https://docs.python.org/3/tutorial/index.html>, last accessed 2024/06/05
48. Google Colab Research google. <https://research.google.com/colaboratory/faq.html#whats-colaboratory>, last accessed 2024/06/05
49. Anaconda Navigator - Anaconda documentation. <https://docs.anaconda.com/free/navigator/index.html>, last Accessed 2024/06/05
50. Jupyter Notebook Documentation - Jupyter Notebook 7.1.0a2 documentation. <https://jupyter-notebook.readthedocs.io/en/latest/>, last accessed 2024/06/05
51. TensorFlow. Introduction to TensorFlow TensorFlow. <https://www.tensorflow.org/learn>, last accessed 2024/06/05
52. Keras documentation : About Keras 3 Keras. <https://keras.io/about>, last accessed 2024/06/05
53. About OpenCV. <https://opencv.org/about>, last accessed 2024/06/05
54. Visual Studio Code. Documentation for Visual Studio Code Code visualstudio. <https://code.visualstudio.com/docs>, last accessed 2024/06/05
55. The Web framework for perfectionists with deadlines Django. <https://www.djangoproject.com/>, last accessed 2024/06/05
56. PostgreSQL: About Postgresql. <https://www.postgresql.org/about/>, last accessed 2024/06/05
57. Quick Start. React. <https://react.dev/learn>, last accessed 2024/06/05
58. Diagram Software and Flowchart Maker drawio. <https://www.drawio.com/>, last accessed 2024/06/05
59. Rister, B., Yi, D., Shivakumar, K., Nobashi, T., Rubin, D. L.: CT-ORG, a new dataset for multiple organ segmentation in computed tomography. Scientific Data, vol. 7, no. 1, 381 (2020). <https://doi.org/10.1038/s41597-020-00715-8>
60. Ronneberger, O., Fischer, P., Brox, T. U-net: Convolutional networks for biomedical image segmentation. Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, vol. 9351, pp. 234-241. Springer International Publishing (2015). https://doi.org/10.1007/978-3-319-24574-4_28
61. Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Rueckert, D. et al.: Attention u-net : Learning where to look for the pancreas. arXiv preprint (2018). <https://doi.org/10.48550/arXiv.1804.03999>
62. Terven, J., Cordova-Esparza, D.: A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. arXiv preprint (2023). <https://doi.org/10.48550/arXiv.2304.00501>